Optimized Scheduling for Electric Vehicle Charging: A Multi-Objective Approach to Grid Stability and User Satisfaction

Aimen Khiar[1,2][a], Mohamed-el-Amine Brahmia[1][b], Ammar Oulamara[3][c]
and Lhassane Idoumghar[2][d]

[1]*CESI Lineact UR 7527, Strasbourg, France*
[2]*IRIMAS UR 7499, University of Haute-Alsace, Mulhouse, France*
[3]*LORIA UMR 7503, University of Lorraine, Nancy, France*

Keywords: Electric Vehicle, Charging Station Allocation, Multi-Objective Optimization, Scheduling, Metaheuristics, NSGA-II, MOCS.

Abstract: The transition to electric mobility offers substantial environmental benefits but also introduces significant challenges, particularly in managing the high demand for electric vehicle (EV) charging. This demand creates the need for intelligent scheduling to optimize charging station resources and maintain grid stability. In order to address this purpose, we propose a multi-objective scheduling model designed to both minimize peak energy consumption and maximize user satisfaction by reducing waiting times at charging stations. Our model accurately represents real-world scenarios, including sequential charger usage, vehicle-to-charger compatibility, and limited availability of various charger types, each providing a constant power output. Given the complexity of the problem, we adapt and evaluate two metaheuristic algorithms: the Non-Dominated Sorting Genetic Algorithm-II (NSGA-II) and the Multi-Objective Cuckoo Search (MOCS), to approximate optimal solutions. The results show that the proposed MOCS adaptation surpasses that of NSGA-II in terms of dominance and achieving a well-distributed Pareto front approximation in a reasonable time frame. The proposed approach thus provides a powerful framework for efficient EV charging management, balancing user needs with grid stability and highlighting its strong potential for adoption in large-scale charging infrastructures.

1 INTRODUCTION

As global environmental concerns intensify due to factors like air pollution and climate change, Electric vehicles (EVs) have emerged as a promising alternative to combustion engine vehicles, as they significantly reduce a major part of our environmental impact (Electric Power Research Institute, 2023). This shift has led to a substantial rise in the number of EVs on the market. For instance, the International Energy Agency (IEA) predicts that EV sales will reach approximately 17 million units by 2024 (International Energy Agency (IEA), 2024).

However, while EVs show great promise, the main obstacle to their widespread adoption is the long time required to fully charge their battery. This delay results in several challenges, including electrical grid overload, fluctuations in electricity prices, and reduced customer satisfaction. To address these challenges, smart charging systems, also known as coordinated charging, have become critically important. Coordinated charging refers to finding optimal plans for charging vehicles in a way that maximizes resource efficiency, where schedules are generated by a control system, also known as an aggregator. In contrast, uncoordinated charging occurs when clients use chargers freely without controlled energy distribution. This is known as the Electric Vehicle Charging Scheduling Problem (EVCSP), which involves minimizing (or maximizing) one or more objectives while respecting imposed constraints.

The literature on this topic is considerably rich, with numerous studies conducted on the problem. For example, (Khiar et al., 2025) proposed a multiobjective optimization approach for grid stability, by minimizing the grid overload and maximizing the delivered energy to clients and proposed a MOCS vari-

[a] https://orcid.org/0009-0002-2724-223X
[b] https://orcid.org/0000-0003-0114-210X
[c] https://orcid.org/0000-0003-2357-0404
[d] https://orcid.org/0000-0001-8853-3968

Khiar, A., Brahmia, M.-E.-A., Oulamara, A. and Idoumghar, L.
Optimized Scheduling for Electric Vehicle Charging: A Multi-Objective Approach to Grid Stability and User Satisfaction.
DOI: 10.5220/0013236400003893
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 14th International Conference on Operations Research and Enterprise Systems (ICORES 2025)*, pages 146-155
ISBN: 978-989-758-732-0; ISSN: 2184-4372
Proceedings Copyright © 2025 by SCITEPRESS – Science and Technology Publications, Lda.

ant to approximate the optimal Pareto front. (Ren et al., 2024) proposed an Adaptive Robust Optimization model to address the uncertainty in EV charging demands, focusing on minimizing grid overload. Integer programming techniques were employed by (Zhu et al., 2014) and (An et al., 2023) to minimize the final charging time of vehicles. (Zaidi et al., 2024) developed a Mixed-Integer Linear Programming model to minimize electric grid capacity and used a hybrid iterative local search metaheuristic to approximate the optimal schedule. Additionally, (Mishra et al., 2023) introduced a multi-objective optimization framework for EV charge scheduling, aiming to maximize profit and customer satisfaction by utilizing a Satisfiability Modulo Theory solver for optimal solutions and the NSGA-II Algorithm to approximate the Pareto front. (Liu et al., 2019) improved a multi-objective self-adaptive differential evolution algorithm and applied it to optimize the fueling of hybrid EVs, aiming to minimize fuel consumption and emission load. (Chen et al., 2023) developed a multi-objective optimization scheduling model aimed at minimizing distribution network operating costs, reducing net load variance, and maximizing the photovoltaic consumption rate. They proposed an extension of the NSGA-II Algorithm called NSGA-II-NDAX to approximate the Pareto front.

This paper proposes a multi-objective scheduling model for EV charging where interruptions are not allowed during the charging process, aiming to minimize grid capacity usage and client service times under realistic scheduling conditions. The structure of this paper is as follows: In section 2, we describe the problem under study, while section 3 develops the mathematical formulation of our model. Section 4 describes the metaheuristic techniques employed to approximate the Pareto front. In section 5, we provide numerical experiments to compare the performance of the proposed methods, and finally, section 6 concludes the paper.

2 PROBLEM DESCRIPTION

In smart charging systems for EVs, aggregators or control systems play an important role in optimizing charging within the broader energy grid. These systems act as intermediaries between resources and EV owners, gathering data such as energy requirements (in kilowatt-hours, kWh) and expected arrival times. Aggregators use this information to design optimal charging schedules for each vehicle and to communicate charging or discharging instructions to the connected chargers.

In this study, we address the EVCSP using a bi-criteria optimization approach. Our aim is to schedule EV charging requests in a way that balances two key objectives:

1. **Minimizing service times** enhances customer satisfaction, fosters customer loyalty, and unlocks new business opportunities.
2. **Preventing grid overload** through efficient resource utilization and energy distribution by minimizing instantaneous energy demand, commonly referred to as grid capacity.

We consider a charging infrastructure consisting of a limited number of chargers, each delivering a constant power output and becoming available at predefined times. Charging durations are assumed to be linear and deterministic. The goal is to determine a feasible assignment of vehicles to chargers, while accounting for compatibility factors such as port compatibility, charging speed, and power delivery protocols.

Recall that in multi-objective optimization, a solution is said to dominate another if it performs better in at least one objective while being no worse in all others. A solution is considered Pareto optimal if no objective can be improved without degrading at least one other objective. In other words, a solution is Pareto optimal if it is not dominated by any other solution. The set of all such solutions forms the Pareto front, representing the best possible trade-offs between competing objectives. It is important to note that the objectives at hand are conflicting, that is, minimizing one tends to maximize the other. Therefore, the optimization model seeks to find the Pareto front.

For illustration purposes, figure 1 shows a feasible schedule for charging 30 vehicles on 7 chargers. The x-axis represents time, and the y-axis represents chargers with the power they deliver. The blue lines represent the chargers' availability, and at the bottom of the schedule, we indicate for each vehicle the expected arrival and the required energy expressed in kWh.

Summarizing, each client provides the aggregator with their expected arrival time and required energy. Using this information, along with data on chargers, the aggregator creates balanced charging schedules. Each schedule specifies the charging start time for each vehicle, and energy transmission continues uninterrupted until the required energy is fully delivered. Once a vehicle completes its charging session, its charger becomes available for another use. The next section presents our proposed mathematical model.

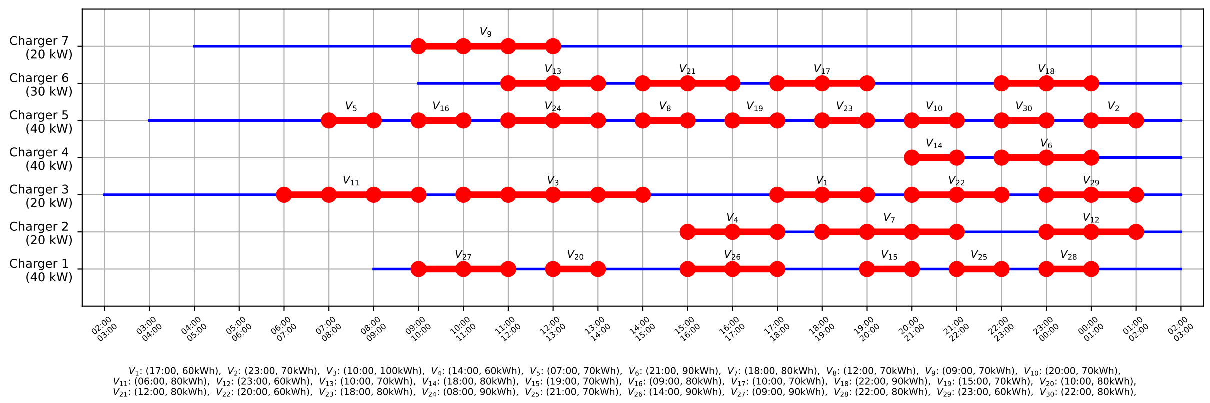

Figure 1: Example of a Schedule of 30 Vehicles on 7 Chargers.

3 MATHEMATICAL MODELING

The scheduling problem under study is defined on a discrete time horizon, subdivided into equal time units of length τ (in hours). We shall model this latter using the set of strictly positive integers $\mathbb{N}^* = \{1,2,3,....\}$.

It is important to note that small values of τ make the scheduling operation more flexible since they offer more options for the model in ordering the vehicles, and as τ becomes small, we approach continuous time scheduling.

An instance of the problem under study is characterized by a set of n charging requests. The charging operations of a vehicles are characterized by two main parameters, namely the expected arrival time $a_i^{(v)}$ and required energy amount e_i, expressed in kWh. The charging infrastructure comprises m chargers. The j^{th} charger provides a constant power of w_j and being available at time $a_i^{(c)}$. The assignability of a vehicle i to charger j as described in the preceding section is represented by binary constants $\delta_{i,j}$, where $\delta_{i,j} = 1$ if the i^{th} vehicle is assignable to the j^{th} charger, and 0 otherwise. Without loss of generality, we suppose that each vehicle is assignable to at least one charger, i.e

$$\forall i \in \{1,2,\cdots,n\} : \sum_{j=1}^{m} \delta_{i,j} \geq 1$$

Given the discrete scheduling horizon and under the linear deterministic charging time assumption, if vehicle i is assigned to charger j, it must deliver power for at least $\frac{e_i}{w_j}$ hours to meet the required energy. Consequently, the number of time units required to serve vehicle i by charger j is given by $p_{i,j} = \left\lceil \frac{e_i}{\tau w_j} \right\rceil$, where $\lceil .. \rceil$ denotes the ceiling function. These energy units are delivered consecutively. Referring back to figure 1, red dots represent these energy units.

We define the decision variables $t_i^{(s)}$ for all $i \in \{1,2,\cdots,n\}$ representing the charging beginning time of vehicle i. We also define the binary decision variables $y_{i,j}$ for all $(i,j) \in \{1,2,\cdots,n\} \times \{1,2,\cdots,m\}$ that takes 1 if and only if vehicle i is assigned to charger j. Each vehicle must be plugged into a compatible charger, giving the constraints

$$\forall i \in \{1,2,\cdots,n\} : \sum_{j=1}^{m} y_{i,j} = 1 \tag{1}$$

and

$$\forall (i,j) \in \{1,2,\cdots,n\} \times \{1,2,\cdots,m\} : y_{i,j} \leq \delta_{i,j} \tag{2}$$

The final charging time of vehicle i denoted as $t_i^{(e)}$ is expressed by

$$t_i^{(e)} = t_i^{(s)} + \sum_{j=1}^{m} p_{i,j} y_{i,j} - 1$$

Thus, the number of vehicles that use charger j at time t is given by $\sum_{i=1}^{n} y_{i,j} \mathbf{1}_{\mathcal{T}_i}(t)$, where $\mathbf{1}_A$ represents the indicator function of the set A. If two vehicles are assigned to the same charger, then their usage periods of the charger cannot overlap; this is explained by the constraints

$$\forall (j,t) \in \{1,2,\cdots,m\} \times \mathbb{N}^* : \sum_{i=1}^{n} y_{i,j} \mathbf{1}_{\mathcal{T}_i}(t) \leq 1 \tag{3}$$

where $\mathcal{T}_i = \left\{ t_i^{(s)}, t_i^{(s)} + 1, \cdots, t_i^{(e)} \right\}$ denotes the charging period of vehicle i.

Additionally, the charging of vehicle i must begin after the vehicle's arrival date, which means $t_i^{(s)} \geq a_i^{(v)}$. Moreover, it cannot begin charging before the assigned charger becomes available, that is, $t_i^{(s)} \geq \sum_{j=1}^{m} a_j^{(c)} y_{i,j}$. This gives us the constraints

$$\forall i \in \{1,2,\cdots,n\} : t_i^{(s)} \geq \max\left(a_i^{(v)}, \sum_{j=1}^{m} a_j^{(c)} y_{i,j}\right) \quad (4)$$

Denote by $\mathbf{w}_G$ the maximum instantaneous energy consumption during the entire schedule, which is the grid capacity that we aim to minimize and represents the first objective function. The power consumed at time slot t by all vehicles is expressed by $\sum_{j=1}^{m}\sum_{i=1}^{n} w_j y_{i,j} \mathbf{1}_{\mathcal{T}_i}(t)$. Thus,

$$\mathbf{w}_G = \max_{t\in\mathbb{N}^*}\left(\sum_{j=1}^{m}\sum_{i=1}^{n} w_j y_{i,j} \mathbf{1}_{\mathcal{T}_i}(t)\right) \quad (5)$$

The second objective is simply the sum of the charging end times of the vehicles, which is given by

$$\mathbf{t}^{(e)} = \sum_{i=1}^{n} t_i^{(e)} \quad (6)$$

The multi-objective optimization problem is to minimize the objective functions (5) and (6) subject to the constraints (1), (2), (3) and (4).

An important consideration is the time complexity of the optimization process. By formulating a decision version of the optimization problem, one can show that identifying a feasible solution to this decision problem is strongly NP-Hard, even in the case of identical chargers, via a reduction to the bin-packing problem. Given this computational challenge, we are motivated to explore approximation techniques, such as nature-inspired metaheuristics, to approximate the Pareto front within a reasonable time frame. The next section will outline the specific adopted solving methods.

4 SOLVING METHODS

An adaptation of metaheuristics to combinatorial optimization problems, which are initially designed for real-valued function optimization, involves defining the algorithmic structure of a feasible solution and the operators needed to explore the feasibility space according to the chosen algorithm. This section presents the solving methods considered for our optimization problem.

Given that we are dealing with a bi-objective constrained optimization problem, the NSGA-II algorithm, proposed in (Deb et al., 2002), appears to be a suitable candidate approach. We also consider an adaptation of the MOCS algorithm proposed in (Yang and Deb, 2013) and compare both metaheuristics in the next section.

4.1 Non-Dominated Sorting Genetic Algorithm II

Genetic algorithms (GA) are metaheuristics inspired by the process of natural selection and are part of the larger class of evolutionary algorithms, first established by Holland's seminal work, (Holland, 1992). These algorithms address optimization problems by iteratively evolving a population of solutions using three main operators: selection, crossover, and mutation. The process starts with a random population and iteratively:

1. Selects promising individuals
2. Creates offspring through crossover and mutation
3. Updates the population based on fitness

This cycle continues until meeting specified stopping criteria.

In the realm of multi-objective optimization, (Deb et al., 2002) introduced the NSGA-II algorithm, a significant advancement in the field of evolutionary optimization algorithms. NSGA-II has gained popularity as one of the most widely applied nature-inspired metaheuristics for solving multi-objective problems.

The core mechanism of NSGA-II revolves around the principle of Pareto dominance, which serves as a fundamental comparison criterion between different solutions. Non-dominated sorting is a key process in this algorithm, where solutions are classified based on their dominance relationships. The objective is to categorize solutions into distinct "fronts" according to their level of dominance. The first front is composed of all non-dominated solutions, representing the optimal trade-offs among the multiple objectives. Subsequent fronts are formed by identifying solutions that are dominated only by those in the preceding fronts, thus creating a hierarchy of solutions that reflects their relative quality.

In addition to non-dominated sorting, NSGA-II incorporates a mechanism known as crowding distance. This measure quantifies the proximity of an individual solution to its neighbors within the objective space. Solutions with higher crowding distance values are considered superior because they contribute to a diverse set of solutions, which is important for exploring the solution space effectively. By promoting diversity, NSGA-II encourages the search process to identify a well-distributed approximation of the Pareto front.

The algorithm begins with an initial parent population consisting of pop_{size} solutions. In each iteration, an offspring population of the same size is generated from the parent population through the use of selection, crossover, and mutation operators. After the

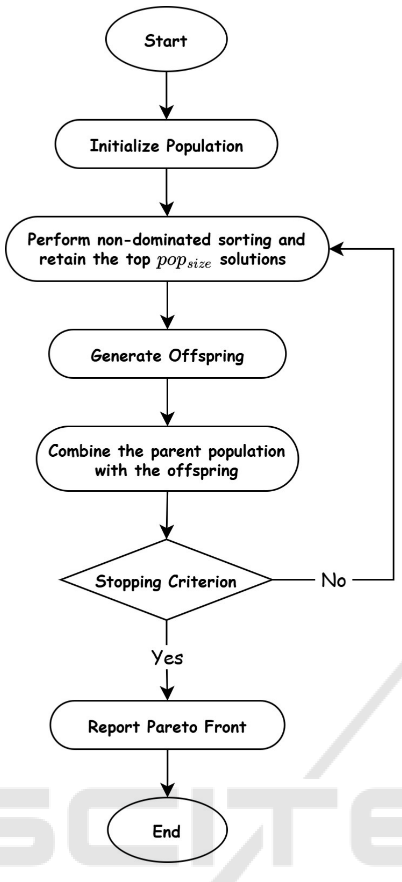
Start
Initialize Population
Perform non-dominated sorting and retain the top pop_{size} solutions
Generate Offspring
Combine the parent population with the offspring
Stopping Criterion
No
Yes
Report Pareto Front
End

Figure 2: Flowchart of NSGA-II Algorithm.

crossover operation, mutation is applied to each offspring with a probability $pm_1 \in (0,1)$. Subsequently, the parent and offspring populations are combined, followed by the application of non-dominated sorting. The top pop_{size} solutions from this combined population are then selected based on their dominance levels to form the next parent population. The flowchart illustrating the algorithm's process is presented in figure 2.

For a more comprehensive overview of the algorithm and the underlying concepts, we refer the reader to (Deb et al., 2002).

4.2 Multi-Objective Cuckoo Search Algorithm

The cuckoo search algorithm is a nature-inspired metaheuristic, developed by (Yang and Deb, 2009) for mono-objective optimization. It is inspired by the breeding behavior of a type of bird called cuckoos, which lay their eggs in the nests of other birds. These cuckoos search for nests of host birds that have eggs similar to cuckoo eggs, maximizing the chance that their eggs are not discovered by the host bird. Each egg in a nest represents a solution, and a cuckoo egg represents a new solution. The aim is to use these new and potentially better solutions (cuckoos) to replace suboptimal solutions in the nests, resulting in a better population of eggs that can survive over more generations. The main assumptions of the algorithm are:

1. Each cuckoo lays one egg at a time and drops it in a randomly chosen nest.
2. The best nests with high-quality eggs will carry over to the next generations.
3. The number of available host nests is fixed.
4. The egg laid by a cuckoo is discovered by the host bird with a probability $p_a \in (0,1)$. In this case, the host bird can either throw the egg away or abandon the nest and build a completely new nest.

Later, they proposed a generalization of the algorithm to tackle multi-objective optimization problems in (Yang and Deb, 2013), where the first and last assumptions are modified respectively as follows:

1. Each cuckoo lays K eggs at a time and drops its eggs in randomly chosen nests.
2. Newly generated nests contain K eggs, representing the new solutions.

Here, K represents the number of objective functions (in our case, $K = 2$). For simplicity, the last assumption of the algorithm can be approximated by replacing a fraction p_a of the n nests with new nests (i.e., new random solutions).

Additionally, various studies have shown that the flight behavior of many animals and insects exhibits characteristics of Lévy flights, which are random walks governed by the Lévy distribution. Yang and Deb incorporated these Lévy flights to generate new solutions (cuckoo eggs) in their continuous optimization algorithm. For a more comprehensive overview of these algorithms and the underlying concepts, we refer the reader to (Yang and Deb, 2009) and (Yang and Deb, 2013).

Since Lévy flights are suitable only for continuous domain optimization, we adapted the algorithm for combinatorial optimization by defining a specialized operator to find new solutions. The algorithm's flowchart is illustrated in figure 3.

4.3 Generating a Random Feasible Schedule

Each feasible schedule for our optimization model is represented by a $n \times 3$ matrix, where each row i contains respectively: the assigned charger for vehicle i,

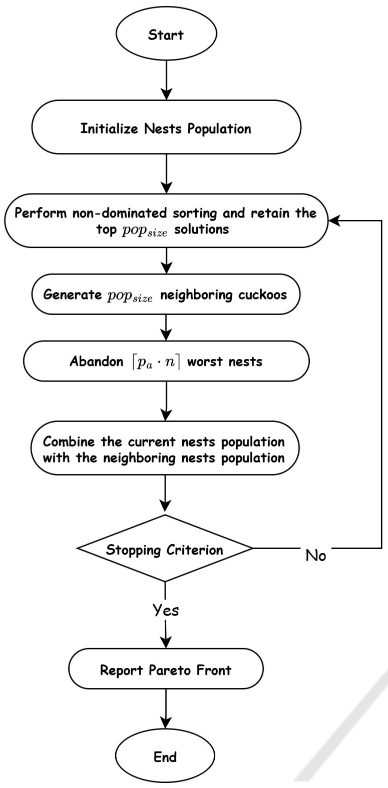
Start
Initialize Nests Population
Perform non-dominated sorting and retain the top pop_{size} solutions
Generate pop_{size} neighboring cuckoos
Abandon $\lceil p_a \cdot n \rceil$ worst nests
Combine the current nests population with the neighboring nests population
Stopping Criterion
No
Yes
Report Pareto Front
End

Figure 3: Flowchart of MOCS Algorithm.

the charging start time $t_i^{(s)}$, and the charging end time $t_i^{(e)}$. Although $t_i^{(e)}$ can be derived from the first two elements, it is included separately to optimize calculations.

Generating a random feasible schedule can be achieved by repeatedly selecting a vehicle, a compatible charger, and a possible charging period at random until all vehicles are scheduled. Suppose that we need to schedule vehicle i on charger j. Assume that there is at least one vehicle already scheduled on charger j, and let $n_j \geq 1$ represent the number of vehicles already scheduled to charge on that charger, ordered according to their charging order as $i_1, i_2, ..., i_{n_j}$. Then, there are at most $n_j + 1$ possible charging sessions denoted $\mathcal{S}_k$ for $k \in \{1, 2, \cdots, n_j + 1\}$, for vehicle i on that charger. A "charging session" refers to a time frame during which the charger is being free, and charging start time for vehicle i can be chosen which determines its charging period. The possible charging sessions can be one of the three cases

1. **Session $\mathcal{S}_1$:** between $t_{i_0} = \max\left(a_j^{(c)}, a_i^{(v)}\right)$ and the charging start time of vehicle i_1,i.e
$$\mathcal{S}_1 := \left\{t_{i_0}, t_{i_0} + 1, \cdots, t_{i_1}^{(s)} - 1\right\}.$$
2. **Session $\mathcal{S}_k$ for $k \in \{2, 3, \cdots, n_j\}$:** between the charging periods of vehicle i_{k-1} and i_k,i.e
$$\mathcal{S}_k := \left\{t_{i_{k-1}}^{(e)} + 1, t_{i_{k-1}}^{(e)} + 2, \cdots, t_{i_k}^{(s)} - 1\right\}.$$
3. **Session $\mathcal{S}_{n_j+1}$:** after the charging period of the last vehicle, i.e
$$\mathcal{S}_{n_j+1} := \left\{t_{i_{n_j}}^{(e)} + 1, t_{i_{n_j}}^{(e)} + 2, \cdots\right\}.$$

Now, if $n_j = 0$ (i.e., no vehicle is scheduled to charge on charger j), there is only one candidate charging session, namely $\{t_{i_0}, t_{i_0} + 1, \cdots\}$. With this in mind, the scheduling of vehicle i on charger j is performed using algorithm 1.

```
Data: A schedule, a vehicle $i \in \{1, 2, \ldots, n\}$, a
      compatible charger $j \in \{1, 2, \ldots, m\}$, and
      a positive number $\sigma$
Result: Start charging time $t_i^{(s)}$ of vehicle $i$ on
        charger $j$
1. Calculate $n_j$;
if $n_j > 0$ then
    a. Set the list of candidate charging sessions
       as:
           $H := \{n_j + 1\} \cup \{k \in \{1, 2, \ldots, n_j\} \mid |S_k| \geq p_{i,j}\}$;
    b. Choose a random index $k$ uniformly from
       $H$;
    if $k = 1$ then
        Choose $t_i^{(s)}$ uniformly from the range
          $\{t_{i_0}, t_{i_0} + 1, \ldots, t_{i_1}^{(s)} - p_{i,j}\}$;
    end
    else if $k \in \{2, 3, \ldots, n_j\}$ then
        Choose $t_i^{(s)}$ uniformly from the range
          $\{t_{i_{k-1}}^{(e)} + 1, t_{i_{k-1}}^{(e)} + 2, \ldots, t_{i_k}^{(s)} - p_{i,j}\}$;
    end
    else
        a. Generate a random number $\lambda$ from
           the normal distribution $\mathcal{N}(0, \sigma^2)$;
        b. Set $t_i^{(s)} = t_{i_{n_j}}^{(e)} + \lceil|\lambda|\rceil + 1$;
    end
end
else
    a. Generate a random number $\lambda$ from the
       normal distribution $\mathcal{N}(0, \sigma^2)$;
    b. Set $t_i^{(s)} = t_{i_0} + \lceil|\lambda|\rceil$;
end
return $t_i^{(s)}$;
```

Algorithm 1: Schedule a Vehicle on a Charger.

Notice that if $n_j > 0$ and the last session (i.e., $\mathcal{S}_{n_j+1}$) is chosen, which contains an infinite number of choices for $t_i^{(s)}$, then we set $t_i^{(s)} := t_{i_{n_j}}^{(e)} + \lceil|\lambda|\rceil + 1$. If $n_j = 0$, then the unique charging session is also theoretically infinite, and we set $t_i^{(s)} := t_{i_0} + \lceil|\lambda|\rceil$, where λ is drawn from $\mathcal{N}\left(0, \sigma^2\right)$. Here, as the standard deviation σ increases, the probability of $t_i^{(s)}$ being large also increases, which encourages schedules that are less favorable to the second objective function (6), thereby improving the exploration of the search space.

4.4 Selection Operator

We define the selection operator which is the mechanism that chooses individuals from the population for reproduction phase based on their ranks and crowding distances. Given a population where non-dominated sorting algorithm is already applied, we calculate crowding distances for each solution. We rank each solution by the order of the front they belong to, solutions with rank 1 are the non-dominated solutions, hence the smaller the rank, the better the solution is.

Next, we select randomly without replacement two schedules from 1 to $N = \left\lceil \frac{pop_{size}}{4} \right\rceil$ with probability $p_k = \frac{2(N-k+1)}{N(N+1)}$ for $k \in \{1, 2, \cdots, N\}$, where pop_{size} is the population size. We then retain the schedule with smaller rank. If both individuals belong to the same rank, the individual with the greater crowding distance is preferred; if their crowding distances are equal, one of the individuals is selected arbitrarily.

4.5 Crossover Operator

Given two selected parent solutions saying $\mathbf{S}_1$ and $\mathbf{S}_2$, the crossover operator is done as follows

1. Construct the set of vehicles in schedule $\mathbf{S}_1$ where their assigned charger is free for use in schedule $\mathbf{S}_2$ on the same charging period in $\mathbf{S}_1$, denote this set as $\mathcal{M}$.
2. Create a new schedule $\mathbf{S}_2'$ by choosing randomly without replacement $\left|\mathcal{M}\right\lceil/3$ vehicles from the set $\mathcal{M}$ and copy their charging plan from $\mathbf{S}_1$ to $\mathbf{S}_2$.
3. Apply steps 1 and 2 where now you interchange the order of $\mathbf{S}_1$ and $\mathbf{S}_2$ and derive a schedule $\mathbf{S}_1'$

Steps 1 and 2 ensure that when exchanging vehicles between schedules $\mathbf{S}_1$ and $\mathbf{S}_2$, we do not get overlapping charging periods while combining the information contained in each schedules.

4.6 Mutation Operator

Notice that the crossover operator maintain the same charger for a vehicle if it is exchanged between two schedules, hence to be sure that any feasible schedule could be found by the algorithm, given a schedule, we defined the mutation operator to select randomly $\lceil pm_2 * pop_{size} \rceil$ vehicles and than change their charger randomly using algorithm 1, where $pm_2 \in (0, 1)$.

4.7 Finding a Neighbor Solution Operator (for MOCS)

For a feasible schedule, the neighbor solution operator used in the MOCS algorithm is defined by randomly selecting $\lceil p_c * n \rceil$ vehicles and a compatible charger for each vehicle (which can be the current one), and scheduling the vehicle to charge on it using algorithm 1, where $p_c \in (0, 1)$. Large values of p_c tend to perturb more the schedule at hand, which we also observed through experiments; thus, one needs to carefully choose this latter to enhance the quality of the exploitation of the search space.

5 EMPIRICAL EXPERIMENTS

In this section, we present numerical experiments to compare the proposed approaches. The experiments were carried out on a computer featuring an Intel® Core™ i5-8350U CPU running at 1.70 GHz (with a turbo boost of up to 1.90 GHz) and equipped with 8.00 GB of RAM.

We considered instances of varying sizes, specifically 50, 100, 150 and 200 charging requests. For each instance size, five random instances were generated, ordered from 1 to 20. The instances were created with the following parameters:

- The time slots length τ was set to 10 minutes.
- Vehicle arrivals were selected randomly starting from 00 : 00, and the required energy of each vehicle was sampled randomly from the set $\{20, 30, 40, ..., 300\}$.
- The number of available chargers was set to $m = \left\lceil \frac{n}{4} \right\rceil$, and the delivered energy was chosen randomly from the set $\{10, 20, 30, 40, 50\}$. Availability times were selected randomly starting from 00 : 00.
- For experimentation purposes, we assumed that vehicles are assignable to all chargers, i.e: $\delta_{i,j} = 1$ for all $(i, j) \in \{1, 2, \cdots, n\} \times \{1, 2, \cdots, m\}$.

The parameters of NSGA-II and MOCS were chosen based on preliminary experimentation. We compared each algorithm with itself by running it several times on different instances of varying sizes, and we

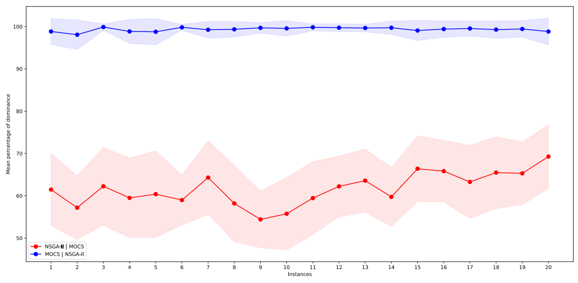

Figure 4: Mean Percentage of Dominance Comparison.

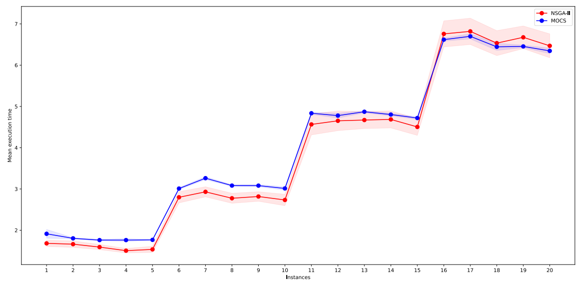

Figure 5: Mean Execution Time Comparison.

selected the parameters that seemed to yield the best results in terms of balancing a highly dominant approximation of the Pareto front with a well-distributed spread of the best solutions.

For both algorithms, we used 300 generations per instance with a population size of 200 schedules, which proved sufficient to achieve significant convergence for the instance sizes considered. As previously mentioned, higher values of the parameter σ tend to increase service times, a trend we observed throughout our experiments. Setting σ to 1 yielded satisfactory results for both algorithms, likely because smaller steps improved exploitation quality. Thus, we used $\sigma = 1$ in our experiments. For NSGA-II, we set the mutation probability $pm_1 = 0.2$ and the proportion $pm_2 = 0.05$ for vehicle charger changes, as higher values for pm_2 tend to excessively perturbate the schedule at hand, an effect also observed experimentally. For MOCS, we found a proportion of abandoned nests $p_a = 0.25$ to work well across different instance sizes, with the proportion $p_c = 0.05$ for changes in vehicle charging sessions.

In order to compare between NSGA-II and MOCS, we calculated the mean percentage of dominance between both methods for each instance. The dominance percentage of NSGA-II over MOCS, for example, is defined as the number of solutions on the approximated Pareto front found by MOCS that are dominated by NSGA-II, divided by the total number of non-dominated solutions found by MOCS, and then multiplied by 100.

If we focus on the solutions within the section of the Pareto front approximation found by the first method that lies close to the approximation from the second method and vice versa the mean dominance percentage can serve as an indicator of solution quality. Moreover, if we consider the opposite sense, an important metric to consider is the percentage of non-dominance. This percentage represents the portion of solutions on the Pareto front found by the first method that remain non-dominated by any solution from the second method. This metric can be interpreted as a measure of how well the first method explores the true Pareto front: a higher non-dominated percentage indicates better performance compared to the second method, as it signifies that the first method is finding unique trade-offs not matched by the second method. The percentage of non-dominance of Method 1 against Method 2 is calculated as 100 minus the percentage of dominance of Method 2 against Method 1. A lower dominance percentage of Method 1 compared to Method 2 indicates better performance of the first method relative to the second.

Another important metric for comparing optimization algorithms is execution time. We present in table 1 the mean dominance percentage and the mean execution time, calculated over the 50 runs, with the standard deviation in parentheses. The results are illustrated in figure 4 and figure 5, respectively, with the shaded area representing confidence intervals.

Table 1: Numerical Results for Mean Percentage of Dominance and Execution Time.

	Mean percentage of dominance		Mean execution time	
Instances	NSGA-II |MOCS	MOCS |NSGA-II	NSGA-II	MOCS
1	3.62 (5.85)	95.6 (6.62)	1.68 (0.08)	1.91 (0.11)
2	3.01 (4.96)	95.52 (6.68)	1.66 (0.08)	1.8 (0.01)
3	4.79 (6.34)	93.57 (9.13)	1.59 (0.06)	1.76 (0.02)
4	2.96 (5.31)	95.39 (7.03)	1.5 (0.05)	1.76 (0.04)
5	8.43 (11.41)	88.25 (16.48)	1.54 (0.07)	1.77 (0.01)
6	9.25 (7.52)	86.96 (10.51)	2.8 (0.14)	3.01 (0.03)
7	8.12 (7.22)	89.12 (10.53)	2.93 (0.12)	3.26 (0.04)
8	12.53 (7.93)	82.35 (11.32)	2.78 (0.12)	3.08 (0.02)
9	9.79 (7.96)	85.6 (12.01)	2.82 (0.12)	3.08 (0.03)
10	10.17 (7.05)	86.16 (9.4)	2.73 (0.14)	3.01 (0.04)
11	15.93 (9.82)	76.31 (14.37)	4.56 (0.25)	4.84 (0.03)
12	21.04 (10.79)	68.86 (16.47)	4.65 (0.24)	4.78 (0.06)
13	17.54 (8.49)	74.71 (12.45)	4.67 (0.21)	4.87 (0.03)
14	18.55 (10.36)	71.55 (16.45)	4.69 (0.21)	4.8 (0.03)
15	18.3 (9.89)	75.19 (12.46)	4.5 (0.2)	4.72 (0.03)
16	25.73 (10.42)	64.0 (16.8)	6.76 (0.32)	6.62 (0.04)
17	24.34 (13.53)	62.8 (21.68)	6.82 (0.32)	6.7 (0.06)
18	23.41 (8.87)	66.2 (12.49)	6.53 (0.3)	6.44 (0.09)
19	22.37 (7.73)	69.16 (12.43)	6.68 (0.27)	6.45 (0.03)
20	25.77 (9.77)	63.93 (12.95)	6.47 (0.29)	6.35 (0.06)

We also illustrated in figures 6,7,8 and 9 the Pareto front found by both methods on instances 5, 10, 15, and 20, which are respectively of size 50, 100, 150, and 200 requests.

The findings indicate that MOCS outperforms NSGA-II in dominance metrics. Specifically, the mean dominance percentage of MOCS over NSGA-II is markedly higher than that of NSGA-II over

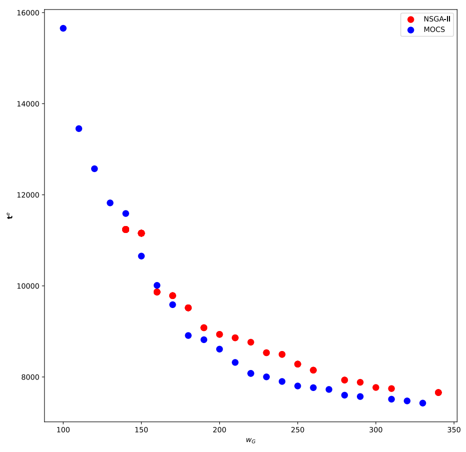

Figure 6: Comparison of Pareto Fronts Generated by Both Methods for Instance 5.

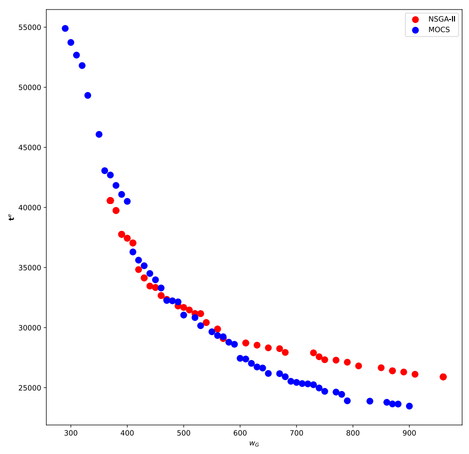

Figure 8: Comparison of Pareto Fronts Generated by Both Methods for Instance 15.

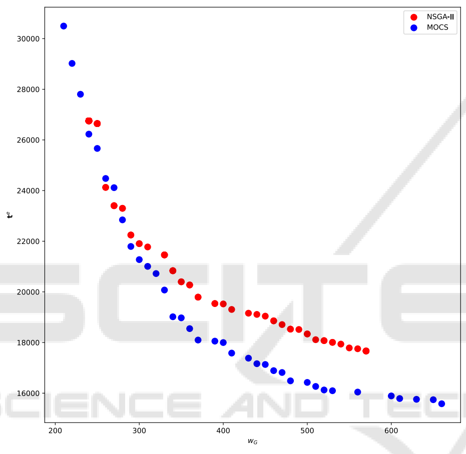

Figure 7: Comparison of Pareto Fronts Generated by Both Methods for Instance 10.

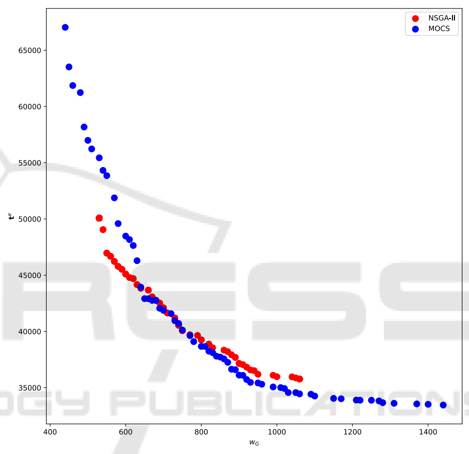

Figure 9: Comparison of Pareto Fronts Generated by Both Methods for Instance 20.

MOCS. This suggests that MOCS consistently finds more solutions that dominate those produced by NSGA-II, with many MOCS solutions remaining non-dominated by NSGA-II. Across all instances, the total mean dominance percentage of NSGA-II over MOCS is 14.28%, compared to 79.56% for MOCS over NSGA-II, with notably lower standard deviations, reflecting consistency in performance.

Regarding execution time, NSGA-II exhibits slightly faster performance than MOCS; however, this difference is marginal, remaining under 0.33 seconds across all instances, including larger cases. The standard deviation remains similarly low, indicating stability in computational efficiency.

Additionally, the figures 6,7,8 and 9 clearly illustrate that the Pareto front approximations generated by MOCS are more extensive than those produced by NSGA-II. This trend was consistently observed across all executions and instances, suggesting that MOCS provides greater diversity in trade-off solutions along the Pareto fronts. Consequently, MOCS provides a broader range of options for decision-makers.

The results clearly indicate that the proposed adaptation of MOCS is more effective than NSGA-II in identifying satisfactory solutions within a reasonable time frame for this specific problem. There is potential to enhance both methods further by exploring alternative operator definitions, employing refined versions of each algorithm, as suggested by various studies in the literature, or by considering additional population-based metaheuristics within the multi-objective optimization framework. These approaches may yield even more promising outcomes, which could be a focus of future research. Nonetheless, the proposed MOCS demonstrates significant promise and appears well-suited to the requirements of the proposed model.

To support reproducibility and further research, a GitHub repository [1] provides the Python implementations of both algorithms, optimized with the Numba library. This repository includes instance simulations and test cases in Jupyter Notebook format. Additionally, it offers Pareto front approximations for both methods across all instances and executions, alongside numerical data on dominance and execution times.

6 CONCLUSION

In summary, this study presents an innovative multi-objective optimization model for EV charging scheduling, aiming to minimize peak energy consumption and reduce charging times. By incorporating real-world factors such as sequential charger usage, compatibility, and operational constraints, the model provides a robust framework for optimization. It allows decision makers to select from various schedules on the Pareto Front, balancing grid stability with client service times based on actual grid capabilities.

We have adapted the NSGA-II and MOCS algorithms to our model, with comparative analysis demonstrating that MOCS is the more effective solution in this context. MOCS achieves broader Pareto front coverage within a reasonable time frame of less than 7 seconds, with a dominance percentage averaging 79.56% for MOCS over NSGA-II, compared to only 14.28% for NSGA-II over MOCS.

The proposed model and MOCS adaptation have significant implications for real-world EV infrastructure, providing a scalable and efficient solution to meet the growing demands of electric mobility. Future work may include exploring other metaheuristics and solving methods, as well as investigating variable charging powers and preemptive charging modes to enhance scheduling flexibility.

REFERENCES

An, Y., Gao, Y., Wu, N., Zhu, J., Li, H., and Yang, J. (2023). Optimal scheduling of electric vehicle charging operations considering real-time traffic condition and travel distance. *Expert Systems with Applications*, 213:118941.

Chen, J., Mao, L., Liu, Y., Wang, J., and Sun, X. (2023). Multi-objective optimization scheduling of active distribution network considering large-scale electric vehicles based on nsgaii-ndax algorithm. *IEEE Access*, 11:97259–97273.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

Electric Power Research Institute (2023). Environmental assessment of a full electric transportation portfolio: Executive summary. Retrieved April 29, 2024, from https://www.epri.com/research/products/3002006881.

Holland, J. H. (1992). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. The MIT Press.

International Energy Agency (IEA) (2024). Global ev outlook 2024. IEA, Paris. Retrieved from https://www.iea.org/reports/global-ev-outlook-2024.

Khiar, A., Brahmia, M.-e.-A., Oulamara, A., and Idoumghar, L. (2025). Multi-objective approach for efficient grid resources allocation in electric vehicle charging schedules. In *Congrès de la Société Française de Recherche Opérationelle et d'Aide à la Décision (ROADEF'25)*, Paris.

Liu, M., Wang, X., Sheng, Y., and Wang, L. (2019). Improvement of multi-objective differential evolutionary algorithm and its application for hybrid electric vehicles. In *2019 Chinese Control And Decision Conference (CCDC)*, pages 553–558.

Mishra, S., Mondal, A., and Mondal, S. (2023). A multi-objective optimization framework for electric vehicle charge scheduling with adaptable charging ports. *IEEE Transactions on Vehicular Technology*, 72(5):5702–5714.

Ren, Y., Tan, M., Su, Y., Wang, R., and Wang, L. (2024). Two-stage adaptive robust charging scheduling of electric vehicle station based on hybrid demand response. *IEEE Transactions on Transportation Electrification*, pages 1–1.

Yang, X.-S. and Deb, S. (2009). Cuckoo search via lévy flights. In *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)*, pages 210–214.

Yang, X.-S. and Deb, S. (2013). Multiobjective cuckoo search for design optimization. *Computers & Operations Research*, 40(6):1616–1624. Emergent Nature Inspired Algorithms for Multi-Objective Optimization.

Zaidi, I., Oulamara, A., Idoumghar, L., and Basset, M. (2024). Minimizing grid capacity in preemptive electric vehicle charging orchestration: Complexity, exact and heuristic approaches. *European Journal of Operational Research*, 312(1):22–37.

Zhu, M., Liu, X.-Y., Kong, L., Shen, R., Shu, W., and Wu, M.-Y. (2014). The charging-scheduling problem for electric vehicle networks. In *2014 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 3178–3183.

[1] https://github.com/aikhiar/Minimizing-grid-capacity-and-service-times