Comparison of metaheuristics for the problem of electric vehicle optimal charging and discharging in a smart parking lot

Vincent Roberge[1*], and *Mohammed* Tarbouchi[1], and *Omar* Bouattane[2]

[1]Royal Military College of Canada, Department of Electrical and Computer Engineering, 13 General Crerar Crescent, Kingston, Ontario, K7K 7B4, Canada
[2]École Normale Supérieure de l'Enseignement Technique Mohammedia (ENSET), Université Hassan II de Casablanca, Bd Hassan II, BP 159, Mohammadia, Morocco

Abstract. This paper presents a metaheuristic-based approach to the problem of charging, discharging, and scheduling for electric vehicles (EVs) in a smart parking lot. EVs are becoming more and more popular and create a significant load on the power grid. With variable electricity rates, it is possible to control and optimize the charging and discharging of EVs in order to minimize the burden on the grid and the overall cost. This paper proposes a new solution encoding and fitness function to be used with metaheuristics for this problem. It compares the efficiency of seven metaheuristics, namely the genetic algorithm (GA), the particle swarm optimization (PSO), the grey wolf optimizer (GWO), the Coyote Optimization Algorithm (COA), the Equilibrium Optimizer (EO), the War Strategy Optimization (WSO), and the Competition of tribes and cooperation of members algorithm (CTCM). Results show that the best metaheuristic for the problem is not the most recent one but the GA, which is the oldest of the algorithms used in the comparison.

1 Introduction

Electric Vehicles (EVs) are becoming more and more popular due to their environmental benefits. However, they represent a significant burden on the power grid. The International Energy Agency estimated that EVs used 100 TWh per year in 2022 and that this figure is likely to increase to 380 TWh by the year 2032 [1]. To support this increasing demand, modernization, automation, and optimization of the power grid must happen.

One way to optimize the power grid is to build smart parking lots with the ability to schedule the charging and discharging of EVs. Instead of recharging as soon as the vehicle is connected, the smart system would find an optimized schedule that reduces peak demand and overall recharge costs. This aspect of the problem is focusing on the grid to vehicles (G2V). There is also potential to further improve the system and allow vehicles to power the grid (vehicle-to-grid, or V2G) during peak demand and allow the user to benefit financially

* Corresponding author: vincent.roberge@rmc.ca

from this energy exchange. Developing a system that allows G2V and V2G is challenging. It is, in fact, a large-scale, multi-variable, non-convex optimization problem.

Various methods have been proposed in the literature to solve this problem. Most of them focuses on the G2V aspect. They can be classified into three groups: conventional optimization techniques, heuristics, and metaheuristics [2]. Conventional methods include linear programming, convex optimization and mixed integer nonlinear programming [3], [4]. They are effective for small problem sizes but may be challenged to solve larger problems. Heuristics include algorithms such as the graph search [5]. They rely on a set of predefined rules or trial and error. They are also quite efficient, but they may fall into local optima and lead to suboptimal solutions. The third category of methods is the metaheuristics. They are general optimization methods that can be applied to a wide range of problems. Such methods applied to the problem of vehicle recharge scheduling include the GA [6], the PSO [7], simulated annealing [8] and others. Each metaheuristics have their own advantages and disadvantages, and one may be a better fit for a specific problem. To build on the advantages of multiple metaheuristics, hybrid metaheuristics have been proposed for the problem of EV recharge scheduling, such as the hybrid GA-PSO in [9] and the hybrid GSA-PSO in [10]. In fact, metaheuristics are probably the most popular method for solving the EV recharge scheduling problem.

But which metaheuristic to choose? The *no free lunch* theorem [11] states that no one metaheuristic is better than the others in all problems. This means that to find the most performant metaheuristic for a given problem, one must compare several of them.

In this paper, we propose a metaheuristic-based approach for the problem of charging and discharging scheduling for EVs in a smart parking lot. A novel solution encoding and fitness function are defined, which allows the problem to be solved by a metaheuristic. Seven metaheuristics are implemented, tested and compared, namely the GA, the PSO, the GWO, the COA, the EO, the WSO, and the CTCM. The algorithms are tested on problem sizes ranging from 10 to 60 EVs. Intuitively, the most recent metaheuristic should be the most performant for the problem on hand. However, we find that the GA is the metaheuristic that performs the best for the problem of EV recharge scheduling.

The remainder of this paper is organized as follows. Section 2 outlines the problem definition, section 3 presents the methodology, which includes the solution encoding, the fitness function and the multiple metaheuristics compared in this paper. Section 4 presents and discusses the results. Finally, section 5 is a conclusion.

2 Problem definition

This section defines in a mathematical way the objective function and the various constraints for the problem of charging and discharging scheduling for EVs in a parking lot. Before we start, we must first define two terms. An electricity cost profile is a chart that lists the electricity cost per kWh for each timeslot of the day. A parking lot profile includes the arrival time, the departure time, the power demand, and the current energy for each EV considered in the problem. With these terms defined, the problem of charging, discharging, and scheduling for EVs in a parking lot consists of:

minimizing

$$C = \sum_{i=1}^{numEV} \sum_{t=arr_i}^{dep_i} {x_d}_i^t * ElecPrice_t \tag{1}$$

subject to

$$\sum_{t=arr_i}^{dep_i} x_{d_i}^{t} = dem_i \ \forall i \in EV \tag{2}$$

$$-limChr \ \le \ x_{d_i}^{t} \le limChr \ \forall i \in EV, \forall t \in T \tag{3}$$

$$-limT \le \sum_{i=1}^{numEV} x_{d_i}^{t} \ \le limT \ \forall t \in T \tag{4}$$

$$x_{d_i}^{j} \le E_{arr\ i} + \sum_{t=arr_i}^{j-1} x_{d_i}^{t} \ \forall i \in EV, \forall t \in T, \forall x_{d_i}^{j} < 0 \tag{5}$$

$$E_{arr\ i} + \sum_{t=arr_i}^{j} x_{d_i}^{t} \le E_{i\ capacity} \ \forall i \in EV, \forall t \in T \tag{6}$$

In the optimization function, C is the recharging cost, $numEV$ is the number of electric vehicles, $x_{d_i}^{t}$ is the electricity charge assigned to EV_i for timeslot t and $ElecPrice_t$ is the electricity price for timeslot t. In the first constraint, dem_i is the energy demand for vehicle i. This constraint ensures that the sum of the charging and discharging is equal to the demand for energy. In the second constraint, $limChr$ is the charger limit. This constraint verifies that the charging and discharging rate never exceeds the charger rating. In the third constraint, $limT$ represents the transformer limit. This constraint ensures that the sum of the charge/discharge rate for a given timeslot never exceeds the parking lot transformer limit. The fourth constraint ensures that a negative power demand (i.e., a power discharge) never exceeds the current energy level of the EV. In the fifth constraint, $E_{i\ capacity}$ is the energy capacity of EV_i. This constraint ensures that the initial energy of the vehicle plus the sum of its charging/discharging values never exceed the energetic capacity of the vehicle.

3 Methodology

In this section, we present our methodology, which consists of the solution encoding, the fitness function, the multiple metaheuristics used in the comparison, and the multi-runs approach.

3.1 Solution encoding

The metaheuristics take as input an electricity cost profile and a parking lot profile. Examples are provided in tables 1 and 2. The parking lot profile includes the arrival time, the departure time, the power demand, and the current power of the vehicles. The candidate solutions are encoded as follows:

$$\boldsymbol{X} = \begin{Bmatrix} x_1^{arr_1}, \dots, x_1^{dep_1}, x_2^{arr_2}, \dots, x_2^{dep_2}, \dots, \\ x_i^{t}, \dots, x_{numEV}^{arr_{numEV}}, \dots, x_{numEV}^{dep_{numEV}} \end{Bmatrix} \tag{7}$$

The electricity charge assigned to EV_i during a given timeslot t is denoted by x_i^t. There is an x_i^t for each timeslot that EV_i is present in the parking lot. The value of x_i^t varies in the range of 0 to 1. To decode a candidate solution, we must first scale the solution in the range of -1 to +3. This is done to ensure that there are negative values (discharging) and positive values (charging) while favouring charging. For the negative values, we use the following equation to compute the discharge rate.

$$x_{d_i}^{\ t} = x_i^t * limChr \tag{8}$$

where $x_{d_i}^{\ t}$ is the decoded value associated with x_i^t and $limChr$ is the charger limit. For the positive values, we use:

$$x_{d_i}^{\ t} = \left(\frac{x_i^t}{\sum_{t=arr_i}^{dep_i} x_i^t}\right)(dem_i + dis_i) \tag{9}$$

where dem_i is the energy demand for the vehicle i and dis_i is the sum of the discharge energy calculated at equation (8). Using this decoding, we ensure that the constraint at equation (2) which stipulates that the sum of the charged and discharged rate times the duration must equal to the energy demand of the EV.

3.2 Fitness Function

The fitness function is defined in a way to integrate the optimization objective and the multiple constraints. For a candidate solution $\boldsymbol{X}$, the fitness function is defined as follows:

$$F(\boldsymbol{X}) = \begin{cases} 0 + \dfrac{1}{1+\rho}, & \rho \geq 0 \\ 1 + \dfrac{1}{1+C}, & \rho = 0 \end{cases} \tag{10}$$

where ρ is a penalty term and C if the charging/discharging cost as computed in equation (1). A penalty term is associated with infeasible solutions that violate the problem constraints. The definition of the fitness function ensures that unfeasible solutions have a fitness between 0 and 1 and feasible solutions, between 1 and 2.

The penalty term ρ requires some additional explanations. It is composed of four terms, which are summed together as follows:

$$\rho = d_{limChr} + d_{limTf} + d_{curr.\ E.} + d_{capacity} \tag{11}$$

First is d_{limChr} which is a penalty associated with exceeding the charger limit. This constraint is always respected for negative values of $x_{d_i}^{\ t}$ due to the solution encoding, but must be verified for positive values. Second is d_{limTf}, which is a penalty associated with exceeding the transformer limit. This must be verified for every timeslot considered. Third is $d_{curr.\ E.}$, which is a penalty associated with exceeding the current energy of the EV. The current energy is calculated for every time slot, and the discharge rate must never exceed the current energy. Lastly, $d_{capacity}$ is a penalty associated with exceeding the maximum capacity of the battery. In other words, the vehicle must not charge more than its capacity.

3.3 Metaheuristics

Once the solution encoding and the fitness function have been defined, a metaheuristic is used to find quasi-optimal solutions that maximize the fitness function. All metaheuristics work similarly in the sense that they propose a strategy to improve candidate solutions for a given fitness function. Their difference lies in the strategy they use to perform this improvement. The *no free lunch* theorem [11] states that no one metaheuristic is better than the others in all problems. This means that for a given problem, multiple metaheuristics must be tested in order to find the most performant one. In this work, we test seven metaheuristics for the problem of optimal charging and discharging of EVs in a smart parking lot. The metaheuristics were chosen due to their popularity and are presented here chronologically based on the year they were first published.

3.3.1 Genetic algorithm (GA)

The GA is a population-based metaheuristic that was first proposed in 1975 by John Holland [12]. It is inspired by the natural evolution of species, where organisms adapt to their environment through mutation and crossover over several generations. In the algorithm, the candidate solutions adapt to the fitness function also through mutation and crossover. The GA is probably the most cited metaheuristic, with more than 60,000 citations since its first publication [13]. It is used in a wide range of engineering problems and has passed the test of time. Compared to the other metaheuristics, it is old but well established.

The GA executes following the steps below:

a) Initialize the candidate solutions.
b) Compute the fitness.
c) Select parents for crossover.
d) Generate new solutions using the crossover operator.
e) Modify the new solutions using the mutation operator.
f) Compute the fitness of the new solutions.
g) Replace the old solutions with the new solutions.
h) Return to c) until the termination criterion is met.

In our implementation, the candidate solutions are initialized randomly over the entire search space. Parents are selected through tournament selection in [14] where n solutions are randomly picked and the best one is selected. The process is repeated to select the second parent. In our implementation we set n to 3 to offer a good balance between exploration and convergence. Once two parents are selected, the child solution is generated using the blend crossover operator [14] where each element y_i of the new solution, or child, is calculated using:

$$y_i = rand\left(x_i^1 - \alpha(x_i^2 - x_i^1),\ x_i^2 + \alpha(x_i^2 - x_i^1)\right) \tag{12}$$

where x_i^1 is element i of parent 1, x_i^2 is element i of parent 2, y_i is element i of the child and α is a constant that controls the expansion of the blend crossover. In our implementation, this constant is set to 0.5 to offer good exploration. The selection and crossover operators are repeated until there are as many child solutions as there are parents. This ensures that the population of candidate solutions remains constant throughout the evolution process. Once all children are created, they are subject to a mutation with a given probability, set to 0.1 in our implementation. The mutation operator selected here is the uniform mutation as described in [14]. Finally, parent solutions are replaced by the child solutions with elitism where the best 10% of the parents survive to the new generation, replacing the worst 10% of the children. The process is repeated until the termination criterion is met, which in our implementation is a set number of iterations.

3.3.2 Particle Swarm Optimization (PSO)

The PSO is another very popular metaheuristic with more than 50,000 citations since it first appeared [13]. It was proposed in 1995 by Kennedy and Eberhart [15] and mimics the movement of a flock of birds or a school of fish. In the PSO, the candidate solutions are represented by particles that move in the search space. Particles are attracted by their previous best position and the best position of the entire swarm. Particles are typically initiated randomly in the search space. At each iteration t of the algorithm, the velocity $\vec{v}_t$ and position $\vec{x}_t$ are updated using:

$$\vec{v}_{t+1} = \omega\vec{v}_t + c_1\vec{r}_1.*\left(\vec{b}_t - \vec{x}_t\right) + c_2\vec{r}_2.*\left(\vec{g}_t - \vec{x}_t\right) \tag{13}$$

$$\vec{x}_{t+1} = \vec{x}_t + \vec{v}_{t+1} \tag{14}$$

where $\vec{b}_t$ is the particle's previous best position, $\vec{g}_t$ is the swarm's best position, $\vec{r}_1$ and $\vec{r}_2$ are vectors of random values between 0 and 1, ω is the inertia weight, c_1 is the personal influence weight, and c_2 is the social influence weight. The attraction towards the particle's previous best position and the swarm's best position ensures the convergence of the algorithm, while the random vectors encourage exploration of the search space. The beauty of the PSO is that it is a simple algorithm, yet very powerful.

3.3.3 Grey Wolf Optimizer (GWO)

Moving up in time, the GWO is another population-based metaheuristic that was proposed by Mirjalili et al. in 2014 [16] and has been cited more than 19,700 times since publication, according to Science Direct [17]. The algorithm is inspired by the hierarchy of wolves in a wolf pack and by their behaviour when surrounding and attacking preys. In the GWO, the wolves' positions represent the candidate solutions and are usually initiated randomly over the search space. There is an alpha α wolf which is the leader of the pack. There is a beta β wolf which acts as the second in command. There is a delta δ wolf which is the third in importance. Finally, the remainder of the wolves are the omega ω wolves. At any given iteration, the alpha α wolf is the best solution of the population. The beta β wolf is the second-best solution. The delta δ wolf is the third-best solution and the ω wolves are the rest of the solutions.

The algorithm works as follows. At each iteration, for every wolf, the distance $\vec{D}_\alpha$ between the wolf and the α wolf, the distance $\vec{D}_\beta$ between the wolf and the β wolf and the distance $\vec{D}_\delta$ between the wolf and the δ wolf are calculated using:

$$\vec{D}_\alpha = \left\|\vec{C}_1.*\vec{X}_\alpha(t) - \vec{X}(t)\right\| \tag{15}$$

$$\vec{D}_\beta = \left\|\vec{C}_2.*\vec{X}_\beta(t) - \vec{X}(t)\right\| \tag{16}$$

$$\vec{D}_\delta = \left\|\vec{C}_3.*\vec{X}_\delta(t) - \vec{X}(t)\right\| \tag{17}$$

where $\vec{X}$ is the position of the wolf, $\vec{X}_\alpha$ is the position of the alpha wolf, $\vec{X}_\beta$ is the position of the beta wolf, $\vec{X}_\delta$ is the position of the delta wolf and $\vec{C}_1$, $\vec{C}_2$ and $\vec{C}_3$ are random vectors calculated using:

$$\vec{C} = 2\,\vec{r}_1 \tag{18}$$

where $\vec{r}_1$ is a vector of random number between 0 and 1. Following the calculation of the distances between the wolf and the three best wolves, the new position $\vec{X}(t+1)$ of the wolf is computed using:

$$\vec{X}_{p,1}(t) = \vec{X}_{\alpha}(t) - \vec{A}_1 .* \vec{D}_{\alpha} \tag{19}$$

$$\vec{X}_{p,2}(t) = \vec{X}_{\beta}(t) - \vec{A}_2 .* \vec{D}_{\beta} \tag{20}$$

$$\vec{X}_{p,3}(t) = \vec{X}_{\delta}(t) - \vec{A}_3 .* \vec{D}_{\delta} \tag{21}$$

$$\vec{X}(t+1) = \frac{\vec{X}_{p,1}(t) + \vec{X}_{p,2}(t) + \vec{X}_{p,3}(t)}{3} \tag{22}$$

where

$$\vec{A} = 2\, a\, \vec{r}_2 - a \tag{23}$$

where $\vec{r}_2$ is a random vector with values between 0 and 1 and a is reducing from a_0 to a_f during the optimization process using:

$$a = a_0 + {}^{t}/_{T}\,(a_f - a_0) \tag{24}$$

where t is the iteration number and T is the total number of iterations. In the original implementation and in our implementation, the value of a_0 is 2 and the value of a_f is 0. This ensures good exploration during the initial part of the metaheuristic when a is larger than 1 and good exploitation or convergence during the later part of the algorithm when a is smaller than 1.

3.3.4 Coyote Optimization Algorithm (COA)

The COA is a more recent population-based metaheuristic that was published by Pierezan and Dos Santos Coelho in 2018 [18]. It has since been cited more than 680 times according to IEEE Xplore [19] demonstrating the high popularity and success of the algorithm. Compared to the GWO, the COA has a different structure and relies on the experience exchange happening in the *Canis latrans* species instead of relying on the hierarchy and attack behaviour of the *Canis lupus* species. One key characteristic of the COA algorithm is that the population of candidate solutions, or coyotes, is divided into multiple packs. The improvement of the coyotes occurs within each pack, but there is a mechanism to exchange coyotes between packs. Working with multiple packs favours exploration, while the exchange of coyotes favours convergence. The overall working of the COA algorithm is as follows:

a) Initialize the packs with their coyotes.
b) Compute the fitness.
c) For each pack.
    i. Define the alpha coyote
    ii. Compute the social tendency of the pack
    iii. For each coyote in the pack
        a. Update the social condition
        b. Compute the new social condition
        c. Adaptation
    iv. Birth of new coyotes and death of old coyotes
d) Exchange of coyote between packs
e) Update of coyotes' age.
f) Return to c) until the termination criterion is met.
g) Return the best adapted coyote

The detailed description of each step with the associated equations used in the various computations are provided in the original paper [18].

3.3.5 Equilibrium Optimizer (EO)

The EO algorithm was published by Faramarzi et al. in 2020 [20] and has been cited more than 2300 times since then, according to Science Direct [17]. It is inspired by control volume mass balance models used to estimate both dynamic and equilibrium states in chemistry. Each particle with its concentration works as a candidate solution. The particles stochastically update their concentration using the best particles to eventually reach an equilibrium state. The details of the algorithm are given in the original paper [20], but essentially it consists of taking the four best-so-far particles and computing their average concentration. Then, the concentration of every particle is updates using one of the four best-so-far particles or the average. The process is repeated until the termination criterion is meet. In our implementation, this termination criterion is a fixed number of iterations determined experimentally, as for the other metaheuristics.

3.3.6 War Strategy Optimization (WSO)

The next metaheuristic is the WSO algorithm [21]. Published by Ayyarao et al. in 2022, this metaheuristic has been cited more than 235 times according to IEEE Xplore [19] which demonstrates its popularity. The algorithm is based on ancient war strategy where the movement of troops (the candidate solutions) is influenced by the position of the king (best-so-far solution) and the commander (second best-so-far solution). The algorithm implements attack and defense strategies, which offers a balance between exploration and exploitation. This algorithm has been demonstrated to be more performance than a wide range of metaheuristics on several benchmark problems [21]. The details of the algorithm with equations are provided in the original paper [21].

3.3.7 Competition of tribes and cooperation of members algorithm (CTCM)

The last metaheuristic if the CTCM algorithm, which is a brand new metaheuristic that was just published in March 2025 by Chen et al. [22] and has been cited 15 times since then according to Science Direct [17]. The algorithm is inspired by tribes from the early time of civilization, where there is competition between tribes and cooperation between members of a tribe.

Initially, the candidate solutions are randomly generated and divided into tribes. The cooperation behavior uses the same equations as the PSO (eq. (13) and (14)), but the best solution is selected within the tribe and not within the entire population. This behaves like multiple PSOs executed independently. This represents the exploitation (or convergence) phase of the algorithm.

In the competition phase, a rival tribe is selected. If the fitness of the rival tribe (i.e., the fitness of its best member) is worse than the fitness of the current tribe, the member escapes the rival tribe. This rivalry implements the information sharing between the tribes. The detailed pseudo-code of the metaheuristic with equations is provided in the original paper [22].

Because the CTCM algorithm simulates the movement of multiple tribes and uses the same equations as the PSO in the cooperative phase, the author of this paper would like to point out that there are similarities between the CTCM and a multi-island PSO as described in [23]. The novelty of the CTCM lies in the communication mechanism between the tribes or islands.

3.4 Multi-Run Implementation

Because metaheuristics are non-deterministic algorithms, they can produce different results each time they are run with some runs yielding inferior results. To avoid inferior results, it is recommended to run the metaheuristic multiple times and take the best result. In our implementation, we use multiple runs and assign the runs to different OpenMP [24] threads on our multicore experimental setup. As described in the next section, our test workstation has 24 cores, which means that 24 threads executing 24 runs is possible without significantly increasing the runtime of the algorithms. This greatly increases the quality of the final solution found by each metaheuristic.

4 Results

The experimental tests consist of two numerical simulations. The simulations are run on a Dell 7920 workstation equipped with dual Intel Xeon Silver 4214R CPU with 12 cores and running at a base frequency of 2.40 GHz. The system has 128 GB of DDR4 RAM and runs Linux Ubuntu 24.04. The programs were written in C++ and compiled using GCC 13.3.0.

The first test consists of using the GA to compute an optimized charging/discharging schedule for a 20 EV smart parking lot. The variable rate electricity cost profile used is listed in Table 1 and the parking lot profile in Table 2. The parameters used to configure the GA are listed in Table 3. These parameters were determined experimentally to give the best results. The charging/discharging schedule computed by the GA is listed in Tables 4 and 5. The overall cost is $87.10. From Tables 4 and 5, we can note that the charger limit of 9.6 kW is never exceeded. The same is observed for the transformer limit of 60 kW. The energy demand is met exactly for all EVs. We can also observe that the EVs tend to charge when the electricity cost is low and discharge when the electricity cost is high. This test demonstrates the good functioning of the GA algorithm for the program of optimal charging and discharging of EVs in a smart parking lot.

The second test consists of running all implemented metaheuristics to optimize schedules for parking lots of sizes varying from 10 to 60 EVs. The aim is to see which metaheuristic performs the best. The metaheuristics were configured using the parameters listed in Table 3. These parameters were either selected experimentally to give the best performance or taken from the original papers describing the metaheuristic. Each test was repeated 10 times, and the average recharge costs are listed in Table 6. In this table, we also list the dimension of the problem, i.e., the number of independent variables being optimized. We note that the dimension varies from 64 for the 10 EV parking lot to 501 for the 60 EV lot. We also list the baseline cost. This is the cost obtained using a deterministic heuristic that prioritizes EV by EV number and allocates the recharge rate so that the constraints are respected. The baseline is a feasible schedule that only includes charging and not discharging. From the results obtained, we can note that the GA is the best-performing algorithm. This is somewhat counterintuitive, as the GA is the oldest of the metaheuristics presented in this paper, and newer metaheuristics are expected to be more performant. This demonstrates the fact that one should always test multiple metaheuristics for a given problem in order to find the most performant one and not rely on intuition. The PSO comes in second for small problem sizes but is overperformed by the COA for larger problem sizes. The CTCM comes in third, followed by the GWO, the EO, and the WSO in last.

 https://doi.org/10.1051/e3sconf/202568000048

Table 1. Electricity cost ($/kWh)

Time	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Electricity Cost	0.1	0.1	0.2	0.2	0.2	0.4	0.4	0.4	0.2	0.2	0.1	0.1

Time	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
Electricity Cost	0.1	0.1	0.2	0.2	0.2	0.4	0.4	0.4	0.2	0.2	0.1	0.1

Table 2. EV parking lot profile

EV	**Arrival time**	**Departure time**	**Demand (kWh)**	**Current charge (kWh)**	**EV**	**Arrival time**	**Departure time**	**Demand (kWh)**	**Current charge (kWh)**
1	2	7	36	24	11	6	16	52	8
2	6	10	30	30	12	4	12	34	26
3	2	10	50	10	13	10	18	30	30
4	4	9	36	24	14	8	16	32	28
5	4	7	30	30	15	6	14	28	32
6	12	23	44	16	16	10	17	26	34
7	14	17	28	32	17	8	14	32	28
8	16	22	32	28	18	10	18	38	22
9	14	17	20	40	19	2	24	56	4
10	12	18	40	20	20	2	10	32	28

Table 3. Metaheuristic parameters

Parameters	**Value**
Common parameters	
Number of candidate solutions	1000
Number of iterations	10,000
GA parameters	
Tournament size	3
Crossover operator	Blend crossover
Crossover α constant	0.5
Mutation operator	Uniform mutation
Mutation rate	0.1
Number of elitism solutions	50
PSO parameters	
ω	0.7298
c_1	1.4960
c_2	1.4960
GWO parameters	
a_0	2.0
a_f	0.0

https://doi.org/10.1051/e3sconf/202568000048

COA parameters	
Number of coyotes per pack	14
EO parameters	
a_1	2.0
a_2	1.0
GP	0.5
V	1.0
WSO parameters	
R	0.2
a	2.0
CTCM parameters	
ω	0.5
c_1	2.0
c_2	1.0
c_3	0.1
Number of solutions per group	100

Table 4. Schedule for 20 EVs as calculated by the GA (hour 1 to 12 of a day)

EV	**T1**	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**T8**	**T9**	**T10**	**T11**	**T12**
EV1	-	9.50	9.50	7.78	9.22	-0.78	0.78	-	-	-	-	-
EV2	-	-	-	-	-	4.68	4.17	5.86	6.28	9.02	-	-
EV3	-	9.60	9.60	9.48	7.08	3.10	5.75	-8.90	7.54	6.75	-	-
EV4	-	-	-	9.15	9.58	-2.13	2.40	7.51	9.50	-	-	-
EV5	-	-	-	9.55	8.91	6.58	4.97	-	-	-	-	-
EV6	-	-	-	-	-	-	-	-	-	-	-	7.14
EV7	-	-	-	-	-	-	-	-	-	-	-	-
EV8	-	-	-	-	-	-	-	-	-	-	-	-
EV9	-	-	-	-	-	-	-	-	-	-	-	-
EV10	-	-	-	-	-	-	-	-	-	-	-	6.95
EV11	-	-	-	-	-	1.49	1.66	-9.55	8.05	3.75	6.06	7.58
EV12	-	-	-	9.39	9.47	-8.64	2.97	-8.47	5.78	7.54	9.58	6.38
EV13	-	-	-	-	-	-	-	-	-	2.68	3.26	3.72
EV14	-	-	-	-	-	-	-	-9.39	5.37	4.62	5.68	6.42
EV15	-	-	-	-	-	-9.46	-7.37	0.14	2.61	9.23	7.71	8.40
EV16	-	-	-	-	-	-	-	-	-	1.08	5.59	2.73
EV17	-	-	-	-	-	-	-	-8.55	7.56	5.23	8.55	3.47
EV18	-	-	-	-	-	-	-	-	-	1.10	4.77	2.02
EV19	-	9.60	9.60	9.35	8.21	-9.60	-9.60	-9.60	1.56	-0.42	8.81	5.20
EV20	-	9.60	9.57	5.31	7.52	-7.96	-9.45	2.24	5.74	9.43	-	-
Total	**0.00**	**38.30**	**38.27**	**60.00**	**60.00**	**-22.72**	**-3.74**	**-38.71**	**60.00**	**60.00**	**60.00**	**60.00**

Table 5. Schedule for 20 EVs as calculated by the GA (hour 13 to 24 of a day)

EV	T13	T14	T15	T16	T17	T18	T19	T20	T21	T22	T23	T24	Total
EV1	-	-	-	-	-	-	-	-	-	-	-	-	36.00
EV2	-	-	-	-	-	-	-	-	-	-	-	-	30.00
EV3	-	-	-	-	-	-	-	-	-	-	-	-	50.00
EV4	-	-	-	-	-	-	-	-	-	-	-	-	36.00
EV5	-	-	-	-	-	-	-	-	-	-	-	-	30.00
EV6	8.17	4.20	7.45	5.25	8.51	-9.59	-6.35	-9.57	9.60	9.60	9.60	-	44.00
EV7	-	5.39	7.28	6.93	8.40	-	-	-	-	-	-	-	28.00
EV8	-	-	-	9.56	8.54	3.02	1.02	-9.35	9.60	9.60	-	-	32.00
EV9	-	5.31	5.87	3.18	5.64	-	-	-	-	-	-	-	20.00
EV10	6.63	2.76	6.47	6.54	6.65	4.00	-	-	-	-	-	-	40.00
EV11	8.45	8.45	8.77	7.28	-	-	-	-	-	-	-	-	52.00
EV12	-	-	-	-	-	-	-	-	-	-	-	-	34.00
EV13	4.58	0.45	4.01	4.84	4.25	2.21	-	-	-	-	-	-	30.00
EV14	5.83	5.83	2.93	4.72	-	-	-	-	-	-	-	-	32.00
EV15	8.16	8.58	-	-	-	-	-	-	-	-	-	-	28.00
EV16	2.93	2.46	3.82	2.19	5.20	-	-	-	-	-	-	-	26.00
EV17	8.46	7.27	-	-	-	-	-	-	-	-	-	-	32.00
EV18	6.57	5.70	5.44	6.17	4.62	1.61	-	-	-	-	-	-	38.00
EV19	0.22	3.59	7.96	3.32	8.19	-9.60	-9.60	-9.60	9.60	9.60	9.60	9.60	56.00
EV20	-	-	-	-	-	-	-	-	-	-	-	-	32.00
Total	**60.00**	**60.00**	**60.00**	**60.00**	**60.00**	**-8.35**	**-14.93**	**-28.52**	**28.80**	**28.80**	**19.20**	**9.60**	**706.00**

Table 6. Comparison of the charging cost ($) for the solutions calculated by the various metaheuristics (best solutions are identified in bold)

Num of EVs	Dimen-sion	Baseline	GA	PSO	GWO	COA	EO	WSO	CTCM
10	64	74.44	**59.56**	**59.56**	59.91	**59.56**	59.67	59.65	59.58
20	167	159.80	**87.15**	87.19	92.25	88.05	117.00	116.28	88.16
30	231	232.92	**131.18**	141.09	166.92	136.72	212.62	208.94	141.01
40	334	318.32	**174.75**	187.79	237.70	181.41	288.14	300.78	185.56
50	398	392.00	**214.98**	259.77	336.65	229.79	396.27	401.80	241.09
60	501	476.52	**262.60**	313.14	421.84	280.09	470.98	475.12	294.03

5 Conclusion

In conclusion, this paper presented and tested seven metaheuristics for the problem of optimal charging and discharging scheduling for EVs in a smart parking lot. The metaheuristics selected consist of the GA, the PSO, the GWO, the COA, the EO, the WSO, and the CTCM. All these metaheuristics are very popular in the literature. The first one dates from 1975 and the last one from 2025. The algorithms were tested and compared with six different problem sizes ranging from parking lots of 10 EVs to parking lots of 60 EVs. Intuitively, one would expect newer metaheuristics to outperform older ones. However, the numerical simulation clearly shows the superiority of the GA for this specific problem. This paper has made two important contributions. First, it proposed a solution encoding and fitness function for the problem of charging and discharging of EVs in a smart parking lot. Second, it compares the performance of seven popular metaheuristics for this problem. Our future work includes parallelizing these algorithms on graphics processing units to reduce the computation time and allow for a faster calculation.

We would like to acknowledge the Canadian Defence Academy who funded this research.

References

1. "World Energy Outlook 2022," International Energy Agency (IEA). Accessed: Mar. 28, 2024. [Online]. Available: https://www.iea.org/reports/world-energy-outlook-2022
2. A. Amin *et al.*, "A Review of Optimal Charging Strategy for Electric Vehicles under Dynamic Pricing Schemes in the Distribution Charging Network," *Sustainability*, vol. 12, no. 23, Art. no. 23, Jan. 2020, doi: 10.3390/su122310160.
3. O. Sassi and A. Oulamara, "Electric vehicle scheduling and optimal charging problem: complexity, exact and heuristic approaches," *Int. J. Prod. Res.*, vol. 55, no. 2, pp. 519–535, Jan. 2017, doi: 10.1080/00207543.2016.1192695.
4. K. Khalili *et al.*, "Optimal Allocation of Electric Vehicles Charging Stations in Commercial Parking Lots: A Mixed-Integer Nonlinear Programming Approach," *Sustainability*, vol. 17, no. 23, Dec. 2025, doi: 10.3390/su172310862.
5. Z. Wang *et al.*, "Research on Electric Vehicle Charging Scheduling Strategy Based on Graph Model," *J. Phys. Conf. Ser.*, vol. 1673, no. 1, p. 012063, Nov. 2020, doi: 10.1088/1742-6596/1673/1/012063.
6. S. Singh, S. Kachhap, A. Kumar, and D. Kumar, "Optimizing electric vehicle charging schedules using genetic algorithm under time and power constraints," *Life Cycle Reliab. Saf. Eng.*, Nov. 2025, doi: 10.1007/s41872-025-00369-8.
7. X. Pang, X. Wang, Y. Yu, H. Jiang, and Z. Zheng, "Optimal scheduling for electric vehicle charging and discharging via double Q-learning-based multi-objective particle swarm optimization," *Energy*, vol. 339, p. 139133, Dec. 2025, doi: 10.1016/j.energy.2025.139133.
8. A. Brown, G. Droge, A. Brown, and G. Droge, "A Simulated Annealing Approach to the Scheduling of Battery-Electric Bus Charging," *Future Transp.*, vol. 4, no. 3, pp. 1022–1045, Sept. 2024, doi: 10.3390/futuretransp4030049.
9. S. S. Amiri and S. Jadid, "Optimal charging schedule of electric vehicles at battery swapping stations in a smart distribution network," in *2017 Smart Grid Conference (SGC)*, Dec. 2017, pp. 1–8. doi: 10.1109/SGC.2017.8308875.
10. K. Pan, C.-D. Liang, and M. Lu, "Optimal scheduling of electric vehicle ordered charging and discharging based on improved gravitational search and particle swarm optimization algorithm," *Int. J. Electr. Power Energy Syst.*, vol. 157, p. 109766, June 2024, doi: 10.1016/j.ijepes.2023.109766.
11. D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, Apr. 1997, doi: 10.1109/4235.585893.
12. J. Holland, *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.
13. M. A. Lones, "Mitigating Metaphors: A Comprehensible Guide to Recent Nature-Inspired Algorithms," *SN Comput. Sci.*, vol. 1, no. 1, p. 49, Nov. 2019, doi: 10.1007/s42979-019-0050-8.
14. X. Yu and M. Gen, *Introduction to Evolutionary Algorithms*. London: Springer-Verlag, 2010.

15. J. Kennedy and R. Eberhart, “Particle swarm optimization,” in *, IEEE International Conference on Neural Networks, 1995. Proceedings*, Nov. 1995, pp. 1942–1948 vol.4. doi: 10.1109/ICNN.1995.488968.
16. S. Mirjalili, S. M. Mirjalili, and A. Lewis, “Grey Wolf Optimizer,” *Adv. Eng. Softw.*, vol. 69, pp. 46–61, Mar. 2014, doi: 10.1016/j.advengsoft.2013.12.007.
17. “Science Direct.” Accessed: Aug. 03, 2025. [Online]. Available: https://www.sciencedirect.com/
18. J. Pierezan and L. Dos Santos Coelho, “Coyote Optimization Algorithm: A New Metaheuristic for Global Optimization Problems,” in *2018 IEEE Congress on Evolutionary Computation (CEC)*, July 2018, pp. 1–8. doi: 10.1109/CEC.2018.8477769.
19. “IEEE Xplore.” Accessed: Aug. 03, 2025. [Online]. Available: https://ieeexplore-ieee-org.journal.rmc.ca/Xplore/home.jsp
20. A. Faramarzi, M. Heidarinejad, B. Stephens, and S. Mirjalili, “Equilibrium optimizer: A novel optimization algorithm,” *Knowl.-Based Syst.*, vol. 191, p. 105190, Mar. 2020, doi: 10.1016/j.knosys.2019.105190.
21. Tummala. S. L. V. Ayyarao *et al.*, “War Strategy Optimization Algorithm: A New Effective Metaheuristic Algorithm for Global Optimization,” *IEEE Access*, vol. 10, pp. 25073–25105, 2022, doi: 10.1109/ACCESS.2022.3153493.
22. Z. Chen, S. Li, A. T. Khan, and S. Mirjalili, “Competition of tribes and cooperation of members algorithm: An evolutionary computation approach for model free optimization,” *Expert Syst. Appl.*, vol. 265, p. 125908, Mar. 2025, doi: 10.1016/j.eswa.2024.125908.
23. D. Izzo, M. Ruciński, and F. Biscani, “The Generalized Island Model,” in *Parallel Architectures and Bioinspired Algorithms*, F. Fernández de Vega, J. I. Hidalgo Pérez, and J. Lanchares, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 151–169. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-28789-3_7
24. Tim Lewis, “OpenMP,” OpenMP. Accessed: Aug. 05, 2025. [Online]. Available: https://www.openmp.org/