Multi-Agent Evolution for Electric Vehicle Charging Systems: A Scatter Search Approach to Breeding Specialized DQN Agents

Tomas Acosta Bernal

***Abstract*—Electric vehicle (EV) charging scheduling presents a complex multi-criteria optimization challenge involving economic efficiency, grid constraints, and customer satisfaction. This project introduces a hybrid framework that integrates deep reinforcement learning (DRL), mixed-integer linear programming (MILP), and evolutionary metaheuristics to address this problem holistically. Specialized Deep Q-Network (DQN) agents are trained to capture distinct operational policies, including cost minimization, demand satisfaction, and urgency handling. A comprehensive MILP formulation with slack variables serves as a performance benchmark, enabling the analysis of infeasibility and infrastructure bottlenecks by relaxing constraints such as charger capacity, transformer limits, and parking allocation. The key innovation is the application of Scatter Search to evolve and optimize a population of DQN agents simultaneously. This evolutionary strategy yields a diverse set of agent archetypes—cost-focused, satisfaction-driven, urgency-oriented, efficiency-based, and balanced optimizers—each suited for specific operational contexts. The proposed approach represents a novel integration of evolutionary design with DRL in energy management systems.**

***Index Terms*—Electric Vehicle Scheduling, Deep Reinforcement Learning, Multi-Agent Systems, Mixed Integer Linear Programming, Scatter Search, Hyperparameter Optimization, Multi-Objective Optimization, Energy Management, Smart Grids.**

I. Introduction

THE increasing penetration of electric vehicles (EVs) presents significant challenges and opportunities for electrical distribution networks, particularly regarding the intelligent scheduling of charging operations. As EV adoption continues to accelerate, effective management of charging demand becomes critical for maintaining grid stability, reducing operational costs, and ensuring high-quality service for users. However, the complexity of this optimization problem extends beyond traditional scheduling approaches due to the dynamic, uncertain, and multi-objective nature of EV charging systems.

The primary complexity in EV charging scheduling lies in the uncertainties associated with arrival times, departure times, energy requirements, and dynamic electricity pricing. These uncertainties, combined with infrastructure constraints such as transformer capacity, charger availability, and parking limitations, create a challenging optimization landscape that requires robust and adaptive solution methods capable of providing near-optimal solutions efficiently under unpredictable operational conditions.

Traditional approaches to EV charging optimization have primarily focused on single-agent reinforcement learning or standalone mathematical programming formulations. However, these approaches often fail to capture the diverse operational requirements of real-world charging systems, where different contexts may require fundamentally different decision-making strategies. For instance, a charging station in a premium hotel may prioritize customer satisfaction over cost minimization, while a public charging facility may need to balance economic efficiency with service quality, and emergency services may require urgency-focused strategies that ensure no vehicle is left unattended.

This paper proposes a multi-agent evolutionary approach that addresses these limitations by developing and optimizing multiple specialized Deep Q-Network (DQN) agents for EV charging scheduling. The core innovation lies in applying Scatter Search metaheuristic to evolve a population of intelligent agents, each learning distinct decision-making policies optimized for different operational contexts. This approach generates diverse agent archetypes—including cost minimizers, satisfaction maximizers, urgency-focused agents, efficiency optimizers, and balanced agents—that can be deployed according to specific operational requirements.

The methodology integrates three key components: (1) a comprehensive Mixed Integer Linear Programming (MILP) formulation with slack variables that serves as an optimal baseline and enables infeasibility analysis, (2) multiple specialized DQN agents that learn context-specific policies through deep reinforcement learning, and (3) a Scatter Search algorithm that evolves agent populations by optimizing both network hyperparameters and reward function weights simultaneously. This multi-agent evolutionary framework represents the first known application of population-based metaheuristics to optimize multiple deep reinforcement learning agents for energy management systems.

The primary contributions of this research are: (1) development of a novel multi-agent evolutionary framework that creates specialized DQN agents for different operational contexts in EV charging; (2) application of Scatter Search metaheuristic to simultaneously optimize both neural network hyperparameters and reward function weights, enabling the evolution of behaviorally distinct intelligent agents; (3) comprehensive evaluation demonstrating that different agent archetypes excel in different operational scenarios, providing system operators with a toolkit of specialized agents; and (4) establishment of optimal baselines through MILP formulations with slack variables for infrastructure bottleneck analysis.

The remainder of this paper is structured as follows: Section II reviews relevant literature and highlights the research gaps addressed. Section III presents the detailed methodology, including the MILP formulation, DQN agent architectures, and the evolutionary optimization framework. Section IV analyzes computational experiments, results, and performance comparisons across different agent archetypes. Finally, Section V concludes the paper and outlines future research directions.

The complete code, along with all instances used in this study, is available in a public repository for reproducibility and further research:
Repository.

II. Literature Review and State of the Art

Electric vehicle (EV) charging scheduling optimization has been extensively studied due to its implications for grid stability, operational cost reduction, and service quality improvement. The methodologies employed to address this problem have evolved significantly, integrating diverse approaches from simple heuristic techniques to advanced models that combine machine learning and mathematical optimization.

A prominent line in the literature addresses the problem from a decentralized and collaborative perspective. Several studies have implemented advanced Multi-Agent Deep Reinforcement Learning (MADRL) techniques to coordinate multiple stations and chargers, achieving collaborative optimal solutions under dynamic and competitive scenarios [1]. This approach has shown particular success in contexts where multiple agents (chargers or stations) must negotiate limited resources in real-time. Recent work by Liu et al. (2021) [9] explored multi-agent DQN systems for EV charging bidding in electricity spot markets, demonstrating that specialized agents can outperform single-agent approaches in dynamic market conditions. Their Multi-DQN framework showed that different agents learn differentiated strategies according to market context, establishing the foundation for specialized agent approaches in EV systems. However, their work focused on bidding strategies rather than charging scheduling, and agents were manually designed rather than automatically evolved.

Another significant stream utilizes robust models and optimization under uncertainty to handle critical aspects such as variability in demand and renewable generation. Recent research has employed techniques such as ambiguity sets based on Kullback-Leibler (KL) divergence to guarantee robust solutions under uncertainties in energy demand and distributed renewable generation availability [3]. These methods have proven effective in ensuring feasibility in real contexts where uncertainty is inherent and significant.

Heuristic and metaheuristic approaches have been extensively studied due to their implementation simplicity, computational speed, and capacity to address large and complex problem instances. Methods such as tabu search, genetic algorithms, intelligent scatter search, and simulated annealing have been successfully applied to limited resource allocation problems and dynamic charging infrastructure management, showing significant advantages in terms of computational time and acceptable solution quality compared to more exact but computationally expensive techniques [2], [4]. A notable contribution by Mao et al. (2014) [8] proposed an Intelligent Scatter Search (ISS) algorithm for EV charging scheduling problems, formulating the problem as a mixed-integer nonlinear programming (MINLP) model with multiple time-of-use tariffs. Their ISS approach incorporated intelligent diversification, combination by hypercube, and restart mechanisms, achieving more stable and efficient solutions than comparative heuristics like Particle Swarm Optimization (PSO) and Genetic Algorithms (GA). While their work demonstrated the effectiveness of Scatter Search for direct EV charging optimization, it focused on optimizing charging schedules directly rather than evolving intelligent agents capable of learning adaptive policies.

The intersection of metaheuristics and deep reinforcement learning for EV systems has gained recent attention. Hammad et al. (2023) [10] presented a significant advancement by combining metaheuristics with deep learning to reduce EV service scheduling time in autonomous IoT environments. Their approach used neural network classifiers to guide DQN configuration searches, applying evolutionary algorithms and PSO for hyperparameter optimization in massive EV charging systems. The hybrid system significantly accelerated scheduling time without losing quality, and the combination of guided learning with metaheuristics improved policy stability. However, their work focused on single-agent hyperparameter optimization rather than evolving populations of specialized agents.

Several investigations have integrated Deep Learning techniques to forecast critical problem parameters, such as EV arrival and departure patterns and energy demand profiles. These predictive models are subsequently combined with robust mathematical optimization techniques to improve operational management efficiency, enabling online adjustments to charging scheduling under changing and uncertain situations [5].

Despite significant advances, several relevant research gaps are identified: first, while deep reinforcement learning approaches have shown efficacy in dynamic contexts, they typically focus on single-agent optimization or manually designed multi-agent systems, missing the opportunity for automatic discovery of specialized behavioral patterns;

second, although exact and robust methods offer optimal or near-optimal solutions, they present serious computational limitations for large instances, while heuristics achieve scalability and computational speed but typically cannot guarantee high global optimality or strict feasibility in highly restrictive scenarios; third, recent work has explored Scatter Search for direct EV charging optimization (Mao et al., 2014) and multi-agent DQN approaches for energy market bidding (Liu et al., 2021), however, these approaches either optimize scheduling directly without learning adaptive policies, or manually design multiple agents without evolutionary discovery of optimal configurations. While Hammad et al. (2023) demonstrated the potential of combining metaheuristics with DQN for hyperparameter optimization, their work focuses on single-agent optimization rather than evolving populations of specialized agents.

In this context, the present work addresses these limitations by introducing a novel evolutionary multi-agent framework where Scatter Search operates at a higher abstraction level—evolving populations of intelligent DQN agents rather than optimizing schedules directly. This approach enables automatic discovery of behaviorally distinct agent archetypes through simultaneous optimization of both network hyperparameters and reward function weights. Our work uniquely combines the population-based search capabilities of Scatter Search with the multi-agent specialization benefits demonstrated in recent DQN research, while extending beyond single-agent hyperparameter optimization to evolve entire populations of behaviorally distinct agents. This integration allows leveraging not only the speed and adaptability of reinforcement learning, but also ensuring rigor in constraint compliance through mathematical optimization, while maintaining computational efficiency and scalability through heuristic techniques.

Table I presents a comparative synthesis of the main approaches found in recent literature, clearly indicating the advantages and limitations of each reviewed method.

Approach	**Advantages**	**Limitations**
MADRL	Adaptive, decentralized, scalable	Limited constraint guarantees
Robust optimization (KL-divergence)	High reliability under uncertainty	High computational complexity
Metaheuristics (Scatter Search)	Fast execution, practical solutions	Global optimality not guaranteed
Deep Learning predictive	Online adaptability, good anticipation	Requires large datasets
Hybrid Meta+DRL	Improved convergence, policy stability	Limited to single-agent optimization
Our Approach	**Multi-agent evolution, automatic archetype discovery, constraint handling**	**Increased algorithmic complexity**

TABLE I
COMPARATIVE ANALYSIS OF RELEVANT METHODS IN LITERATURE

Thus, this article contributes to the current literature by providing an integrated methodological approach that simultaneously leverages the potential of machine learning, rigorous mathematical optimization, and heuristic flexibility, effectively addressing the limitations identified in previous studies and providing an optimal balance between speed, feasibility, quality, and robustness of obtained solutions.

III. METHODOLOGY

This section presents the comprehensive methodological framework developed for evolving specialized Deep Q-Network (DQN) agents for electric vehicle charging scheduling. The proposed approach represents a novel integration of evolutionary metaheuristics with deep reinforcement learning, where Scatter Search operates at the level of agent intelligence rather than direct solution optimization.

The core innovation of this work lies in applying Scatter Search to evolve populations of intelligent DQN agents, each characterized by distinct behavioral profiles optimized for different operational contexts. Unlike traditional approaches that use metaheuristics to optimize schedules directly, our framework optimizes the ”intelligence” that creates those schedules by simultaneously evolving both neural network hyperparameters and reward function weights.

The methodology comprises three integrated components working in synergy: (1) an evolutionary Scatter Search algorithm that maintains and evolves a population of diverse agent configurations, (2) specialized Deep Q-Network agents that learn context-specific policies through reinforcement learning, and (3) a comprehensive Mixed Integer Linear Programming (MILP) formulation that serves as an optimal baseline and enables systematic analysis of infrastructure bottlenecks through slack variable implementation.

The evolutionary process systematically explores the space of agent configurations, ultimately selecting the five highest-fitness solutions regardless of their behavioral classification. These top-performing agents are subsequently analyzed and classified into behavioral archetypes—such as cost minimizers, satisfaction maximizers, urgency-focused agents, efficiency optimizers, and balanced agents—based on their learned parameter combinations. This approach ensures that the final agent portfolio represents the most effective configurations discovered through evolution, with behavioral diversity emerging naturally from the optimization process rather than being imposed a priori.

The integration of the three methodological components follows a complementary design philosophy. The MILP component provides theoretical optimality bounds and enables systematic analysis of problem feasibility under resource constraints. The DQN agents provide adaptive, learning-based solutions that can handle complex state spaces and uncertain environments. The Scatter Search framework orchestrates the evolution of agent populations, ensuring both quality and diversity in the discovered solutions.

This integrated approach addresses the limitations inherent in single-method approaches: the MILP ensures mathematical rigor and constraint compliance, the DQN agents provide adaptability and scalability, and the evolutionary framework enables systematic exploration of the agent configuration space to discover specialized behavioral patterns that would be difficult to design manually.

A. Mixed Integer Linear Programming Formulation

The Mixed Integer Linear Programming (MILP) component serves dual purposes in our framework: providing optimal baseline solutions for benchmarking agent performance and enabling systematic analysis of infrastructure limitations through slack variable implementation. The MILP formulation captures the complete mathematical structure of the EV charging scheduling problem while incorporating advanced features such as vehicle-charger compatibility, priority-based service differentiation, and transformer capacity analysis.

1) Problem Formulation and Decision Variables: The MILP model defines the following decision variables to represent all aspects of the charging scheduling problem:

- $x_{i,t,c} \in \mathbb{R}^+$: Power allocated to vehicle i at time interval t using charger c
- $y_{i,t,c} \in \{0,1\}$: Binary variable indicating whether vehicle i uses charger c at time t
- $z_{i,t} \in \{0,1\}$: Binary variable indicating whether vehicle i is parked at time t
- $w_{i,t,s} \in \{0,1\}$: Binary variable indicating whether vehicle i occupies parking slot s at time t
- $u_i \in \mathbb{R}^+$: Unmet energy demand for vehicle i
- $satisfaction_i \in [0,1]$: Satisfaction ratio for vehicle i

Additionally, slack variables are introduced to enable feasibility analysis:

- $s_{c,t} \in \mathbb{R}^+$: Slack variable for charger c capacity at time t
- $s_t^{trafo} \in \mathbb{R}^+$: Slack variable for transformer limit at time t
- $s_t^{parking} \in \mathbb{R}^+$: Slack variable for parking capacity at time t
- $s_{i,t}^{slot} \in \mathbb{R}^+$: Slack variable for slot assignment of vehicle i at time t

2) Objective Function: The MILP formulation employs the epsilon-constraint method to handle the multi-objective nature of the charging scheduling problem. The primary objective minimizes the total operational cost while heavily penalizing constraint violations through slack variables:

$$\begin{aligned}\min Z = &\sum_{i\in I}\sum_{t\in T}\sum_{c\in C} p_t \cdot \Delta t \cdot x_{i,t,c} \\ &+ M \cdot \sum_{i\in I} u_i \\ &+ \lambda \cdot \Bigg(\sum_{t\in T}\sum_{c\in C} s_{c,t} \\ &\quad + \sum_{t\in T} s_t^{trafo} + \sum_{t\in T} s_t^{parking} + \sum_{i,t} s_{i,t}^{slot}\Bigg) \end{aligned} \tag{1}$$

subject to the epsilon-constraint for weighted customer satisfaction:

$$\sum_{i\in I} w_i \cdot \pi_i \cdot satisfaction_i \geq \epsilon \cdot \sum_{i\in I} w_i \cdot \pi_i \tag{2}$$

where p_t represents the energy price at time t, Δt is the time interval duration, $M = 1000$ is the penalty factor for unmet energy demand, $\lambda = 10^4$ is the penalty coefficient for slack variables, w_i denotes the willingness to pay factor for vehicle i, π_i represents the priority level of vehicle i, $satisfaction_i$ is the energy satisfaction ratio for vehicle i, and $\epsilon \in [0,1]$ is the minimum acceptable weighted satisfaction threshold.

The epsilon-constraint approach transforms the original bi-objective problem into a single-objective optimization by converting customer satisfaction from an objective function into a constraint. This method enables systematic exploration of the Pareto frontier by varying the ϵ parameter, allowing system operators to explicitly control the minimum acceptable level of weighted customer satisfaction while minimizing operational costs.

The selection of $\lambda = 10^4$ was determined through extensive empirical testing with different penalty values. Higher penalty coefficients (e.g., $\lambda > 10^5$) proved counterproductive, as they made slack variable activation prohibitively expensive, causing the solver to prefer returning infeasible solutions rather than activating slack variables for infrastructure bottleneck analysis. Conversely, lower values ($\lambda < 10^3$) resulted in excessive slack activation even when feasible solutions existed within original constraints. The chosen value of $\lambda = 10^4$ provides the optimal balance, ensuring slack variables are activated precisely when needed for feasibility analysis while maintaining their role as indicators of true infrastructure limitations.

3) Core Constraint Set: The model incorporates comprehensive constraints to ensure operational feasibility and service quality:

Energy Balance Constraint:

$$\sum_{t\in T}\sum_{c\in C} x_{i,t,c} \cdot \Delta t + u_i = E_i, \quad \forall i \in I \tag{3}$$

Charger Capacity Constraints (with slack):

$$\sum_{i\in I} x_{i,t,c} \leq P_c + s_{c,t}, \quad \forall t \in T, \forall c \in C \tag{4}$$

Transformer Limit Constraint (with slack):

$$\sum_{i\in I}\sum_{c\in C} x_{i,t,c} \leq L_{trafo} + s_t^{trafo}, \quad \forall t \in T \tag{5}$$

Vehicle-Charger Assignment Linking:

$$x_{i,t,c} \leq P_c \cdot y_{i,t,c}, \quad \forall i \in I, \forall t \in T, \forall c \in C \tag{6}$$

Unique Charger Assignment:

$$\sum_{c\in C} y_{i,t,c} \leq 1, \quad \forall i \in I, \forall t \in T \tag{7}$$

Parking Capacity Constraint (with slack):

$$\sum_{i\in I} z_{i,t} \leq N_{slots} + s_t^{parking}, \quad \forall t \in T \tag{8}$$

Unique Slot Assignment (with slack):

$$\sum_{i \in I} w_{i,t,s} \leq 1, \quad \forall t \in T, \forall s \in S \tag{9}$$

Slot Assignment Completeness:

$$\sum_{s \in S} w_{i,t,s} + s_{i,t}^{slot} = z_{i,t}, \quad \forall i \in I, \forall t \in T \tag{10}$$

4) Advanced Compatibility and Priority Constraints: The formulation incorporates vehicle-charger compatibility based on brand and technical specifications:

Compatibility Constraints:

$$y_{i,t,c} = 0, \quad \forall i \in I, \forall t \in T, \forall c \notin Compatible(i) \tag{11}$$

where $Compatible(i)$ represents the set of chargers compatible with vehicle i based on brand, connector type, and power ratings.

Priority-Based Service Differentiation: The model incorporates priority levels through weighted satisfaction constraints:

$$satisfaction_i = \frac{\sum_{t \in T} \sum_{c \in C} x_{i,t,c} \cdot \Delta t}{E_i}, \quad \forall i \in I \tag{12}$$

5) Feasibility Analysis Through Slack Variables: The slack variable implementation enables systematic analysis of infrastructure bottlenecks:

Transformer Capacity Analysis: The slack variables s_t^{trafo} quantify the degree to which transformer capacity is insufficient at each time interval, enabling identification of peak demand periods that exceed infrastructure capabilities.

Parking Capacity Analysis: Variables $s_t^{parking}$ reveal intervals where vehicle demand exceeds available parking spaces, indicating the need for capacity expansion or demand management.

Charger Utilization Analysis: Slack variables $s_{c,t}$ identify specific chargers that experience capacity violations, enabling targeted infrastructure upgrades.

The penalty coefficient $\lambda = 10^4$ ensures that any slack activation represents a true infrastructure limitation rather than an optimization artifact. This high penalty value makes slack activation prohibitively expensive, guaranteeing that slacks are utilized only when no feasible solution exists within the original constraints.

6) Solution Approach and Implementation: The MILP model is solved using the COIN-OR Branch and Cut (CBC) solver with the following configuration parameters:

- Time limit: 900 seconds (15 minutes)
- Optimality gap tolerance: 1% (ratioGap 0.01)
- Branching strategy: Most fractional variable first
- Cutting plane generation: Enabled for all standard cuts

For large-scale instances, the solver employs warm-start initialization using solutions provided by the DQN agents, significantly reducing convergence time while maintaining solution quality.

B. Deep Q-Network Agent Architecture

The Deep Q-Network (DQN) component constitutes the core learning mechanism within our evolutionary framework, where each agent represents a distinct behavioral configuration optimized through reinforcement learning. The DQN agents operate within a carefully designed Markov Decision Process (MDP) formulation that captures the essential dynamics of electric vehicle charging scheduling while enabling scalable learning across diverse operational scenarios.

1) Markov Decision Process Formulation: The EV charging scheduling problem is formulated as a finite-horizon MDP defined by the tuple (S, A, P, R, γ), where each component is specifically designed to capture the temporal and stochastic nature of the charging environment.

State Space Definition: The state space S encompasses comprehensive information about the current system configuration, vehicle requirements, and resource availability. Each state $s_t \in S$ at time step t is represented as a 40-dimensional feature vector containing normalized vehicle characteristics, system resource availability, and temporal context information. The state representation includes vehicle-specific features such as arrival and departure times, energy requirements, charging urgency metrics, and compatibility ratios with available chargers. System-level features capture resource availability including parking spot utilization, charger availability, transformer capacity margins, and dynamic pricing information. Temporal features provide context about the current time index, remaining operational time, and system demand ratios.

Action Space Definition: The action space A represents the set of feasible charging assignments available at each decision point. Actions are dynamically generated based on the current state, incorporating real-time resource availability and vehicle-charger compatibility constraints. Each action $a_t \in A$ specifies either an assignment of a vehicle to a specific parking spot and charger with a determined power level, or a skip decision when resource constraints prevent feasible assignments. The dynamic action generation ensures computational efficiency by considering only valid assignments while maintaining decision quality.

Transition Dynamics: The transition function $P(s_{t+1}|s_t, a_t)$ governs the deterministic evolution of the system state following action execution. State transitions incorporate the effects of charging assignments on resource availability, energy delivery to vehicles, and temporal progression. The deterministic nature of transitions simplifies learning while accurately modeling the controlled environment dynamics.

Reward Function: The reward function $R(s_t, a_t)$ provides learning signals that guide agent behavior toward desired operational objectives. The reward structure combines multiple components to balance competing objectives: energy satisfaction rewards proportional to delivered energy (50 $\times$ energy_satisfaction_ratio), cost penalties normalized by energy consumption ($-30 \times$ normalized_cost), efficiency bonuses for appropriate power utilization (10 $\times$ price_efficiency), and penalties for unmet energy demands ($-20 \times$ normalized_deficit). Priority multipliers and urgency factors modulate these base rewards based on vehicle characteristics and operational context.

2) Neural Network Architecture: The DQN employs a deep neural network to approximate the action-value function $Q(s, a; \theta)$, where θ represents the network parameters. The architecture implements a dueling network structure that separates the estimation of state values and action advantages, enhancing learning stability and convergence properties.

The network architecture consists of an input layer accepting 40-dimensional state vectors, followed by a fully connected layer with 64 neurons and batch normalization to stabilize training. The first hidden layer employs ReLU activation and connects to a second fully connected layer of 64 neurons, also with ReLU activation. A dropout layer with 0.2 probability reduces overfitting, followed by a third hidden layer of 32 neurons with ReLU activation.

The dueling architecture branches into separate value and advantage streams from the third hidden layer. The value stream processes state values through a 16-neuron hidden layer with ReLU activation, culminating in a single-neuron output representing $V(s)$. The advantage stream estimates action advantages through a parallel 16-neuron hidden layer, outputting advantage values $A(s, a)$ for each possible action. The final Q-values are computed by combining value and advantage estimates according to $Q(s, a) = V(s) + A(s, a) - \frac{1}{|A|}\sum_{a'} A(s, a')$, where the advantage mean subtraction ensures identifiability.

3) Training Algorithm: The DQN training process implements the Double DQN algorithm with experience replay and target network stabilization. The algorithm maintains both a main Q-network for action selection and a target network for stable value estimation during learning updates.

Algorithm 1 DQN Multi-System Training Framework

```
1: Initialize main Q-network $Q(s, a; \theta)$ and target Q-network $Q(s, a; \theta^-)$ with random weights
2: Initialize experience replay buffer $\mathcal{D}$ with capacity $N$
3: Initialize exploration parameter $\epsilon = 1.0$
4: for each training system $k$ do
5:     Initialize system-specific memory buffer $\mathcal{D}_k$
6:     best_reward $\leftarrow -\infty$, patience_counter $\leftarrow 0$
7:     for episode $e = 1$ to max_episodes do
8:         episode_reward $\leftarrow$ ExecuteEpisode($s_0$, $\theta$, $\theta^-$, $\mathcal{D}$, $\mathcal{D}_k$, $\epsilon$)
9:         if episode_reward $>$ best_reward then
10:            best_reward $\leftarrow$ episode_reward
11:            Save best model parameters
12:            patience_counter $\leftarrow 0$
13:        else
14:            patience_counter $\leftarrow$ patience_counter $+ 1$
15:        end if
16:        if patience_counter $\geq$ max_patience then
17:            Load best saved model
18:            break {Early stopping}
19:        end if
20:        Update target network: $\theta^- \leftarrow \theta$ every $C$ steps
21:        Decay exploration: $\epsilon \leftarrow \max(\epsilon_{\min}, \epsilon \cdot \epsilon_{\text{decay}})$
22:    end for
23: end for=0
```

Algorithm 2 Single Episode Training Process

```
Require: Initial state $s_0$, main network $\theta$, target network $\theta^-$, buffers $\mathcal{D}$, $\mathcal{D}_k$, exploration $\epsilon$
Ensure: Episode reward
1: Initialize environment state $s_0$
2: $t \leftarrow 0$, episode_reward $\leftarrow 0$
3: while not terminal state do
4:     Obtain feasible actions $\mathcal{A}_t$ from current state $s_t$
5:     if random() $\leq \epsilon$ then
6:         Select random action $a_t \in \mathcal{A}_t$
7:     else
8:         $a_t \leftarrow \arg\max_{a \in \mathcal{A}_t} Q(s_t, a; \theta)$
9:     end if
10:    Execute action $a_t$, observe reward $r_t$ and next state $s_{t+1}$
11:    Store transition $(s_t, a_t, r_t, s_{t+1}, \text{done})$ in $\mathcal{D}$ and $\mathcal{D}_k$
12:    episode_reward $\leftarrow$ episode_reward $+ r_t$
13:    if $|\mathcal{D}| \geq$ batch_size and $t \bmod 10 = 0$ then
14:        Sample mini-batch $\{(s_j, a_j, r_j, s_{j+1}, \text{done}_j)\}$ from $\mathcal{D}$
15:        Compute target values: $y_j = r_j + \gamma \cdot Q(s_{j+1}, \arg\max_{a'} Q(s_{j+1}, a'; \theta); \theta^-)$
16:        Update main network: $\theta \leftarrow \theta - \alpha \nabla_\theta \sum_j (y_j - Q(s_j, a_j; \theta))^2$
17:    end if
18:    $s_t \leftarrow s_{t+1}$, $t \leftarrow t + 1$
19: end while
20: return episode_reward =0
```

The training algorithm incorporates several key mechanisms to ensure robust learning across diverse system configurations. Experience replay enables sample-efficient learning by breaking temporal correlations in the training data through random sampling from stored experiences. The target network provides stable learning targets by fixing the network parameters used for target value computation, updating them periodically to track the main network progress.

System-specific memory buffers allow the agent to maintain separate experience pools for different system configurations, facilitating knowledge transfer while preserving system-specific learning patterns. Early stopping mechanisms prevent overfitting to individual systems by monitoring episode rewards and reverting to the best-performing model when improvement stagnates

4) Hyperparameter Configuration and Optimization: The DQN training process depends on numerous hyperparameters that significantly influence learning performance and agent behavior. Key parameters include the learning rate controlling gradient descent step sizes, the discount factor γ determining the importance of future rewards, exploration parameters governing the ϵ-greedy policy, memory buffer sizes for experience replay, and batch sizes for network updates.

Within the evolutionary framework, these hyperparameters become decision variables subject to optimization through Scatter Search. Each agent configuration specifies complete hyperparameter settings including network architecture parameters, learning dynamics parameters, and exploration-exploitation balance parameters. This dual-level optimization enables the discovery of hyperparameter combinations that produce specialized agent behaviors suited for different operational contexts.

The hyperparameter space includes continuous variables such as learning rates ranging from 5×10^{-5} to 10^{-2}, discount factors between 0.90 and 0.9999, epsilon start values from 0.3 to 1.0, epsilon minimum values from 0.01 to 0.1, and exploration decay rates from 0.99 to 0.99999. Discrete variables specify batch sizes from the set $\{8, 16, 32, 64, 128, 256\}$, memory buffer capacities from $\{5000, 10000, 20000, 50000, 100000, 200000\}$, and target network update frequencies from $\{5, 10, 20, 30, 50, 75, 100, 150, 200\}$. Boolean parameters control architectural choices such as dueling network activation.

5) Reward Weight Optimization: Beyond neural network hyperparameters, the evolutionary process optimizes reward function weights that shape agent behavioral priorities. The reward weight space encompasses energy satisfaction weights ranging from 0.2 to 5.0, energy cost weights from 0.01 to 2.0, penalty values for skipped vehicles from 5.0 to 500.0, and reward values for assigned vehicles from 1.0 to 200.0. These weights directly influence the trade-offs between competing objectives such as cost minimization, energy satisfaction maximization, and operational efficiency.

The reward weight optimization enables the emergence of specialized agent archetypes through differential emphasis on specific objectives. The evolutionary framework explores seven distinct behavioral archetypes: cost destroyers prioritizing extreme cost minimization with energy cost weights between 1.0-2.0 and reduced satisfaction weights of 0.2-1.0; satisfaction maximizers emphasizing customer service with satisfaction weights of 3.0-5.0 and minimal cost sensitivity of 0.01-0.3; balanced elites achieving optimal cost-satisfaction equilibrium through moderate satisfaction weights of 1.5-3.0 and cost weights of 0.3-1.0; urgency masters ensuring complete service coverage with skip penalties of 200.0-500.0; efficiency demons maximizing throughput through assignment rewards of 100.0-200.0; adaptive genius configurations providing flexible behavior across parameters; and extreme optimizers exploring radical configurations spanning the full parameter space.

This reward-based specialization provides a principled mechanism for generating behaviorally distinct agents without explicit architectural modifications. The evolutionary process automatically discovers weight combinations that produce effective specialized behaviors, eliminating the need for manual agent design while ensuring systematic exploration of the behavioral space.

6) Generalization and Transfer Learning: The DQN agents incorporate several mechanisms to ensure generalization across different system scales and configurations. State normalization techniques ensure that feature representations remain consistent across systems with varying sizes and characteristics. The standardized 40-dimensional state representation enables knowledge transfer between different operational contexts while maintaining decision quality.

Progressive training schemes expose agents to systems of increasing complexity, facilitating knowledge transfer from simple to complex scenarios. This curriculum learning approach accelerates convergence while improving final performance on challenging instances. The separation of system-specific and general knowledge through dual memory structures enables both adaptation to specific environments and retention of broadly applicable decision patterns.

The evolutionary optimization process further enhances generalization by evaluating agent configurations across multiple diverse systems simultaneously. This multi-system evaluation ensures that successful configurations exhibit robust performance across varying operational conditions rather than overfitting to specific scenarios. The resulting agents demonstrate effective scheduling performance on previously unseen system configurations, validating the generalization capabilities of the approach.

7) Important Limitations: While the DQN approach provides adaptive learning capabilities, several inherent limitations constrain its effectiveness for comprehensive electric vehicle charging scheduling:

Sequential Single-Vehicle Decision Making: The DQN architecture processes vehicles individually in a sequential manner rather than considering simultaneous multi-vehicle assignments. The environment selects a representative vehicle using a method and makes decisions for only one vehicle per time step. This approach fundamentally limits the agent's ability to:

- Coordinate simultaneous charging assignments across multiple vehicles
- Optimize global resource allocation considering all waiting vehicles
- Capture inter-vehicle dependencies and competition for limited resources
- Achieve true system-wide optimization as accomplished by the MILP formulation

Temporal Myopia and Discrete Time Slot Processing: The DQN operates with a myopic temporal perspective, making decisions for individual time intervals without comprehensive forward planning. Unlike the MILP approach that optimizes complete charging profiles across the entire planning horizon, the DQN:

- Assigns power levels for single time intervals rather than continuous charging sessions
- Lacks explicit consideration of future resource availability and demand patterns
- Cannot guarantee optimal charging profile continuity for individual vehicles
- May result in fragmented charging schedules with suboptimal power utilization

Action Space Complexity and Scalability: The discrete action space representation creates computational and learning challenges that become more pronounced with system scale:

- Action space grows exponentially with the number of available spots and chargers
- The actions generated multiple feasible actions per state, leading to large action spaces that slow convergence
- Exploration becomes inefficient in high-dimensional action spaces, particularly for larger charging stations
- State representation complexity increases with system size, requiring more training episodes for convergence

Limited Multi-Objective Handling: Unlike the MILP's epsilon-constraint method that provides explicit control over cost-satisfaction trade-offs, the DQN approach relies on:

- Fixed reward weight combinations that cannot be adjusted during operation
- Implicit multi-objective optimization through weighted reward functions
- Difficulty in guaranteeing specific satisfaction levels or cost targets
- Limited interpretability regarding the achieved Pareto efficiency of solutions

Stochastic Solution Quality: The reinforcement learning paradigm introduces inherent variability in solution quality due to:

- Epsilon-greedy exploration policy that may select suboptimal actions during operation
- Neural network approximation errors that can lead to suboptimal value function estimates
- Training convergence dependencies on random initialization and experience replay sampling
- Absence of optimality guarantees that are provided by exact mathematical programming approaches

C. *Scatter Search for Multi-Agent Evolution*

The Scatter Search metaheuristic serves as the evolutionary framework for discovering and optimizing specialized DQN agent configurations rather than traditional hyperparameter tuning. Unlike conventional applications that optimize static parameters, our approach employs Scatter Search to evolve intelligent agent "personalities" through simultaneous optimization of neural network hyperparameters and reward function weights.

1) Algorithm Design and Rationale: Scatter Search was selected over other evolutionary algorithms due to its systematic diversification strategy and structured reference set maintenance, which proves particularly effective for the high-dimensional configuration space encompassing both DQN hyperparameters and behavioral reward weights. The algorithm's ability to maintain solution quality while ensuring population diversity aligns with the objective of discovering distinct behavioral archetypes rather than converging to a single optimal configuration. The structured approach of Scatter Search enables controlled exploration through its reference set mechanism, which maintains both high-quality solutions and diverse alternatives simultaneously. This dual focus on exploitation and exploration proves essential when evolving agent personalities that must perform effectively across diverse operational scenarios while maintaining distinct behavioral characteristics. The systematic combination and improvement phases ensure thorough exploration of the configuration space without the random drift common in other evolutionary approaches.

The core innovation lies in treating each agent configuration as an evolving "personality" defined by neural network hyperparameters $\theta_{nn} = \{lr, \gamma, \epsilon, batch\_size, memory\_size, ...\}$, reward weight vectors:

$$\theta_{reward} = \{w_{satisfaction}, w_{cost}, w_{penalty}, w_{assignment}\} \tag{13}$$

and behavioral classification automatically determined through parameter combinations. This representation enables the algorithm to discover emergent behavioral patterns that would be difficult to design manually, as the interaction between neural network learning dynamics and reward structure creates complex behavioral phenotypes.

2) Scatter Search Framework: The algorithm maintains a Reference Set of high-quality, diverse agent configurations and systematically explores the solution space through structured combination and improvement operations:

Algorithm 3 Scatter Search for Agent Personality Evolution

```
1:  Initialize population $P$ with archetype-biased configurations
2:  Evaluate population fitness through DQN training on multiple systems
3:  Construct Reference Set $RefSet$ balancing elite solutions and diversity
4:  while termination criteria not met do
5:      $NewSolutions \leftarrow \emptyset$
6:      for each pair $(s_i, s_j)$ in $RefSet$ do
7:          if random() $< p_{combination}$ then
8:              $child \leftarrow$ CombineConfigurations$(s_i, s_j)$
9:              $NewSolutions \leftarrow NewSolutions \cup \{child\}$
10:         end if
11:     end for
12:     $ImprovedSolutions \leftarrow$ LocalImprovement$(NewSolutions)$
13:     Filter similar configurations using adaptive cosine similarity threshold
14:     Evaluate fitness of improved solutions through DQN training
15:     $RefSet \leftarrow$ UpdateReferenceSet$(RefSet \cup ImprovedSolutions)$
16:     Update best solutions and diversity metrics
17: end while
18: Classify final solutions into behavioral archetypes
19: return Specialized agent configurations for deployment =0
```

3) Population Initialization and Archetype-Biased Generation: The initial population generation employs a strategic approach that balances random exploration with targeted archetype bias to ensure comprehensive coverage of the behavioral space. The population consists of 200 diverse agent configurations, systematically distributed across predefined archetype categories to guarantee initial behavioral diversity. Each archetype receives a minimum allocation of individuals through biased sampling within their respective parameter ranges, ensuring that cost minimizers, satisfaction maximizers, urgency-focused agents, efficiency optimizers, and balanced agents are represented in the initial population.

The archetype-biased generation process samples parameters within restricted ranges corresponding to each behavioral category. For cost minimizer archetypes, energy cost weights are sampled from $[1.0, 2.0]$ while satisfaction weights are constrained to $[0.2, 0.9]$. Satisfaction maximizers receive satisfaction weights from $[3.6, 5.0]$ with cost weights limited to $[0.01, 0.45]$. This targeted initialization accelerates convergence by providing the evolutionary process with high-quality starting points representing different behavioral strategies, while still maintaining sufficient randomness to enable the discovery of novel hybrid configurations.

The remaining population slots are filled through completely random sampling across the entire hyperparameter space, ensuring that the evolutionary process maintains the ability to discover unexpected parameter combinations that may lead to superior or novel behavioral patterns. This hybrid initialization strategy combines the benefits of informed seeding with the exploratory power of random search, creating a robust foundation for evolutionary discovery.

4) Reference Set Construction and Maintenance: The Reference Set represents the core mechanism of Scatter Search, maintaining a carefully curated collection of 20 high-quality and diverse agent configurations that guide the evolutionary search process. The Reference Set construction follows a two-phase approach that balances solution quality with population diversity to ensure both exploitation of promising regions and exploration of the entire solution space.

The Reference Set consists of 12 elite solutions selected based purely on fitness performance across multiple evaluation systems, ensuring that the highest-quality agent configurations discovered during the search process are preserved and utilized for further evolution. These elite solutions represent the current best-known approaches to the charging scheduling problem and serve as the foundation for generating improved offspring through systematic combination.

The remaining 8 positions in the Reference Set are allocated to diverse solutions selected through a sophisticated similarity-based filtering process. The diversity selection mechanism employs cosine similarity calculations to identify configurations that are maximally different from the elite solutions and from each other, ensuring that the Reference Set maintains broad coverage of the configuration space rather than converging prematurely to similar high-performing regions.

The Reference Set undergoes continuous updates throughout the evolutionary process, with new high-quality or diverse solutions replacing existing members based on dominance relationships and diversity contributions. This dynamic maintenance ensures that the Reference Set evolves with the search process, incorporating newly discovered promising regions while maintaining the established high-quality foundations.

5) Configuration Combination and Offspring Generation: The combination phase represents the primary generative mechanism of the Scatter Search algorithm, systematically creating new agent configurations by blending promising parent solutions from the Reference Set. The combination process examines all possible pairs of Reference Set members, applying probabilistic combination with probability 0.9 to generate diverse offspring that inherit beneficial characteristics from both parents.

The parameter combination strategy varies according to parameter type to preserve the semantic meaning and feasibility of the resulting configurations. For continuous parameters such as learning rates, discount factors, and reward weights, the combination employs weighted linear interpolation with randomly generated mixing coefficient $\alpha \sim \text{Uniform}(0.3, 0.7)$. This approach creates offspring values

$$child_{param} = \alpha \cdot parent1_{param} + (1-\alpha) \cdot parent2_{param} \quad (14)$$

ensuring that the resulting values fall within the convex hull of the parent values while introducing controlled variation.

Discrete parameters including batch sizes, memory buffer capacities, and target update frequencies are combined through random selection from the parent values, maintaining the discrete nature of these parameters while enabling the exploration of different architectural configurations. Boolean parameters such as dueling network activation are similarly handled through random parent selection, preserving the binary nature of these design choices.

The combination process preserves parameter bounds through clipping operations, ensuring that all generated offspring remain within the feasible parameter space defined by the hyperparameter configuration. This constraint handling prevents the generation of invalid configurations while maintaining the exploratory power of the combination operator.

6) Diversity Control Through Cosine Similarity Filtering: The diversity control mechanism employs cosine similarity calculations to prevent premature convergence and maintain population diversity throughout the evolutionary process. Each agent configuration is represented as a normalized vector in the hyperparameter space, enabling the computation of angular similarity measures between different configurations.

The cosine similarity calculation begins with the conversion of each configuration into a standardized vector representation. Continuous parameters are normalized to $[0, 1]$ using their respective minimum and maximum bounds, discrete parameters are mapped to normalized indices, and boolean parameters are converted to binary values. This standardization ensures that all parameters contribute equally to the similarity calculation regardless of their native scales or units.

The adaptive similarity threshold mechanism adjusts the diversity requirements based on the evolutionary progress and fitness improvement trends. Early iterations employ relaxed similarity thresholds (approximately 0.95) to encourage broad exploration, while later iterations implement stricter thresholds (down to 0.75) to focus on exploitation of promising regions. This adaptive approach balances exploration and exploitation according to the current state of the search process.

The filtering process examines each newly generated configuration against all previously accepted solutions, rejecting configurations that exceed the similarity threshold with any existing solution. This mechanism ensures that the population maintains sufficient diversity to continue exploring the configuration space while preventing the accumulation of near-duplicate solutions that would waste computational resources.

7) Local Improvement and Perturbation Strategies: The improvement phase applies targeted local search operations to enhance the quality of newly generated configurations without disrupting their fundamental behavioral characteristics. Each configuration undergoes improvement with probability 0.6, ensuring that computational resources are focused on the most promising candidates while maintaining population diversity.

The local improvement strategy randomly selects individual parameters for perturbation, applying parameter-specific modification operations that preserve feasibility while exploring local neighborhoods. Continuous parameters receive Gaussian perturbations with standard deviations proportional to their parameter ranges, typically 5% of the total range. Discrete parameters are modified by selecting adjacent values from their respective value sets when available. Boolean parameters are toggled with low probability to explore alternative architectural choices.

The perturbation magnitudes are carefully calibrated to balance local exploration with behavioral preservation, ensuring that improved configurations remain within the same general behavioral archetype while potentially discovering enhanced parameter settings. This approach enables fine-tuning of agent configurations without losing the beneficial characteristics that led to their selection.

8) Evolutionary Process and Convergence: The evolutionary process proceeds through iterative cycles of combination, improvement, evaluation, and Reference Set updating until convergence criteria are satisfied. Each iteration generates new candidate configurations through systematic combination of Reference Set members, applies local improvements to enhance their quality, evaluates their fitness through complete DQN training, and updates the Reference Set based on the resulting performance and diversity metrics.

The convergence criteria include maximum iteration limits (50 iterations), time constraints (30 hours), and fitness improvement thresholds. The algorithm terminates when no significant improvement in the best fitness values occurs over consecutive iterations, indicating that the search has reached a stable configuration of high-quality solutions.

The final phase classifies the discovered solutions into behavioral archetypes based on their parameter combinations, providing a systematic taxonomy of the evolved agent personalities. This classification enables deployment selection based on operational requirements and provides insights into the relationship between parameter settings and emergent behaviors.

9) Novel Multi-Agent Personality Discovery: The key innovation transcends traditional hyperparameter optimization by automatically discovering behaviorally distinct agent archetypes through evolutionary pressure. Rather than seeking a single optimal configuration, the algorithm maintains and evolves a diverse portfolio of specialized agents. This approach fundamentally differs from conventional multi-agent systems where agent roles are predefined, instead allowing behavioral specialization to emerge naturally from the interaction between neural network learning dynamics and reward function design. The evolutionary process acts as an automated agent designer, systematically exploring the space of possible agent personalities and identifying configurations that exhibit consistent behavioral patterns across different operational scenarios. This emergent specialization creates a self-organizing system where agents naturally develop complementary skills and operational focuses without explicit programming of behavioral rules.

Cost Minimizers evolve through high energy cost weights and reduced satisfaction priorities, Satisfaction Maximizers emerge via extreme satisfaction weights and minimal cost sensitivity, Urgency-Focused Agents develop through prohibitive skip penalties ensuring complete service coverage, Efficiency Optimizers arise through high assignment rewards maximizing system throughput, and Balanced Agents naturally evolve configurations achieving optimal cost-satisfaction equilibrium.

10) Hyperparameter Space and Archetype Configuration: The evolutionary framework optimizes both neural network hyperparameters and reward function weights simultaneously to discover specialized agent behaviors. The hyperparameter space is carefully designed to balance exploration breadth with computational efficiency.

The neural network configuration space encompasses critical learning parameters that directly influence agent performance and convergence behavior. Learning Rate $\alpha \in [5\times10^{-5}, 10^{-2}]$ controls gradient descent step sizes, with lower values promoting stable but slower convergence, while higher values enable rapid adaptation at the risk of instability. Discount Factor $\gamma \in [0.90, 0.9999]$ determines the importance of future rewards, with values close to 1.0 emphasizing long-term planning essential for multi-period charging optimization. Exploration Parameters including $\epsilon_{start} \in [0.3, 1.0]$, $\epsilon_{min} \in [0.01, 0.1]$, and $\epsilon_{decay} \in [0.99, 0.99999]$ control the exploration-exploitation balance throughout training. Architecture Parameters encompass batch sizes from the set $\{8, 16, 32, 64, 128, 256\}$, memory buffer capacities from $\{5000, 10000, 20000, 50000, 100000, 200000\}$, and target network update frequencies from $\{5, 10, 20, 30, 50, 75, 100, 150, 200\}$. Network Architecture selection includes boolean choice of dueling network architecture, which separates value and advantage function estimation for improved learning stability.

The reward function weights shape agent behavioral priorities and directly determine the discovered archetypes. Energy Satisfaction Weight $w_{sat} \in [0.2, 5.0]$ multiplies rewards for energy delivery, with higher values prioritizing customer satisfaction over cost considerations. Energy Cost Weight $w_{cost} \in [0.01, 2.0]$ penalizes expensive charging decisions, with higher values driving cost-conscious behavior. Penalty for Skipped Vehicles $p_{skip} \in [5.0, 500.0]$ penalizes decisions to skip vehicle assignments, with extreme values ensuring no vehicle is left unserviced. Reward for Assigned Vehicles $r_{assign} \in [1.0, 200.0]$ provides explicit bonuses for successful assignments, promoting high system utilization.

The evolutionary process automatically discovers five distinct behavioral patterns through weight optimization, each suited for specific operational contexts. Cost Minimizer Archetypes with parameters $w_{cost} \in [1.0, 2.0]$ and $w_{sat} \in [0.2, 0.9]$ prioritize extreme cost reduction by accepting lower satisfaction levels when economically justified. Satisfaction Maximizer Archetypes characterized by $w_{sat} \in [3.6, 5.0]$ and $w_{cost} \in [0.01, 0.45]$ emphasize complete customer satisfaction with minimal cost sensitivity. Urgency-Focused Archetypes defined by $p_{skip} \in [201.0, 500.0]$ ensure no vehicle is left unattended regardless of cost implications. Efficiency-Focused Archetypes with $r_{assign} \in [101.0, 200.0]$ maximize system throughput and resource utilization. Balanced Optimizer Archetypes featuring $w_{sat} \in [1.1, 3.9]$ and $w_{cost} \in [0.5, 0.99]$ achieve optimal equilibrium between cost efficiency and customer satisfaction.

The classification system employs hierarchical decision rules to ensure unambiguous archetype assignment through primary classification using cost ratio $\rho = w_{cost}/w_{sat}$, secondary classification via extreme penalty or reward values, and tertiary classification for remaining configurations. The non-overlapping parameter ranges ensure unique and consistent archetype classification, enabling systematic analysis of behavioral diversity and performance characteristics across different operational scenarios.

IV. Results

The training of the reinforcement learning which involved everything related to the scatter search metaheuristic required 24 GB VRAM GPU 4090, and more than 100GB RAM. It took 15 hours approximately to finish all the training.

A. Scatter Search Optimization Results

The Scatter Search algorithm successfully evolved a diverse portfolio of specialized DQN agent configurations over 50 iterations, demonstrating effective exploration and exploitation of the hyperparameter-behavioral space while maintaining population diversity throughout the evolutionary process.

1) Archetype Evolution and Reference Set Dynamics: The evolution of archetypes within the Reference Set reveals the algorithm's ability to discover and maintain behavioral diversity throughout the optimization process. Figure 1 demonstrates the systematic exploration and refinement of different behavioral strategies over 50 iterations.

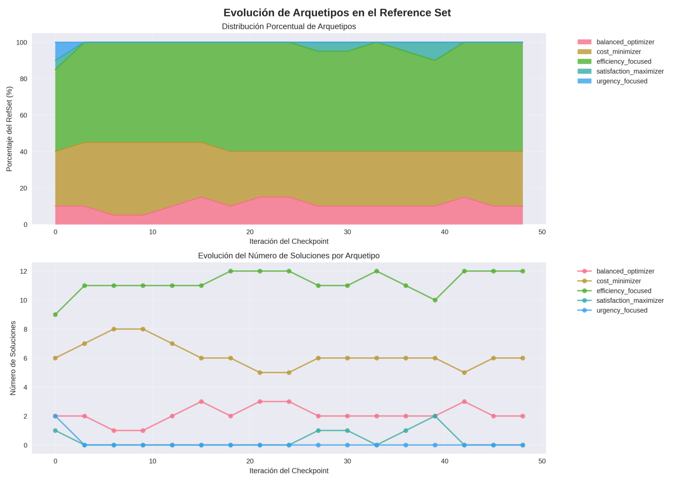

Fig. 1. Evolution of archetype distribution in the Reference Set showing percentage composition (top) and absolute counts (bottom) across 50 optimization iterations.

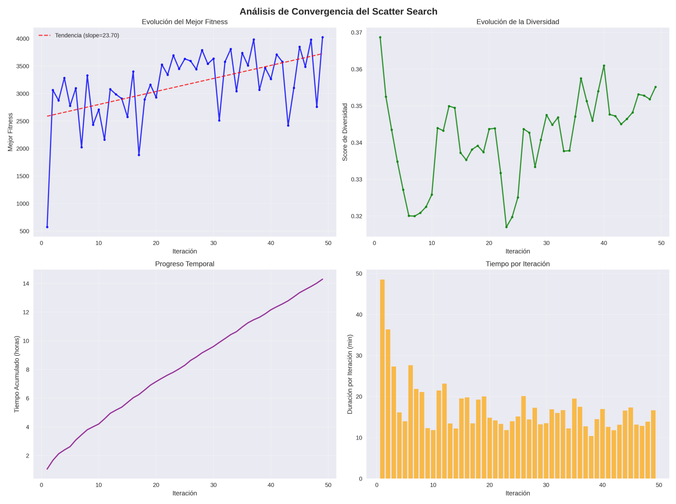

Fig. 2. Scatter Search convergence analysis showing fitness evolution with trend line (top left), diversity score evolution (top right), cumulative computational time (bottom left), and iteration duration distribution (bottom right).

The efficiency-focused archetype emerged as the dominant behavioral pattern, consistently representing 50-60% of the Reference Set composition throughout the optimization process. This dominance suggests that configurations prioritizing high system throughput and resource utilization provide superior performance across the diverse evaluation systems. Cost minimizer archetypes maintained a stable presence of approximately 30% of the Reference Set, indicating their consistent value in achieving economically efficient solutions. Balanced optimizer, satisfaction maximizer, and urgency-focused archetypes appeared less frequently but remained present throughout the evolution, ensuring comprehensive behavioral coverage.

The absolute counts of solutions per archetype demonstrate the algorithm's systematic exploration of different behavioral strategies. Efficiency-focused configurations consistently generated 10-12 high-quality solutions per iteration, while cost minimizers produced 6-8 solutions. The presence of all five archetypes in the final Reference Set confirms the algorithm's success in maintaining behavioral diversity rather than converging prematurely to a single dominant strategy.

2) Convergence Analysis and Fitness Evolution: The convergence characteristics demonstrate substantial improvement in solution quality while maintaining effective exploration-exploitation balance. Figure 2 illustrates the comprehensive convergence behavior across multiple metrics.

The fitness evolution demonstrates substantial improvement from initial values around 600 to final peaks exceeding 4000, representing a 566% improvement in solution quality. The fitness trajectory exhibits an upward trend with slope 23.70, indicating consistent progress throughout the optimization process despite significant variance in individual iteration performance. The substantial fitness fluctuations observed between iterations 15-40 reflect the algorithm's continued exploration of diverse configurations rather than premature convergence to local optima.

The cumulative temporal progress shows steady advancement over the 15-hour optimization period, with the algorithm effectively utilizing the available computational resources to achieve continuous improvement. The sustained upward trajectory confirms that the algorithm had not reached convergence by the termination criteria, suggesting potential for further improvement with extended runtime.

3) Diversity Maintenance and Exploration-Exploitation Balance: The diversity score evolution reveals the algorithm's sophisticated balance between exploration and exploitation phases. The diversity score fluctuated between 0.32 and 0.37 throughout the optimization, with notable patterns of diversification followed by consolidation. Early iterations (1-10) showed high diversity (0.36-0.37) as the algorithm explored the configuration space broadly. Mid-stage iterations (10-25) exhibited reduced diversity (0.32-0.33) as the algorithm focused on exploiting promising regions identified during initial exploration. Later iterations (25-50) restored higher diversity levels (0.34-0.36), indicating successful prevention of premature convergence through adaptive similarity thresholds.

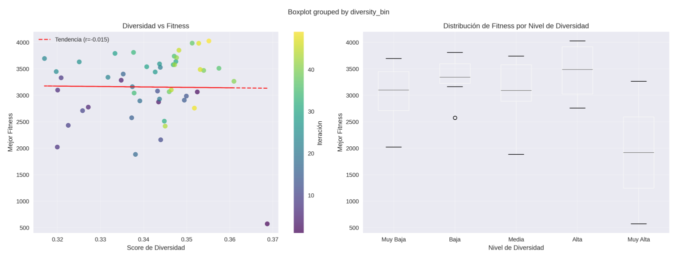

Fig. 3. Diversity-fitness relationship analysis showing scatter plot with iteration progression (left) and fitness distribution across diversity levels (right).

The diversity-fitness relationship analysis demonstrates that higher diversity levels correlate with sustained fitness improvement, validating the algorithm's diversity preservation mechanisms. Configurations with very high diversity (¿0.36) achieved fitness scores spanning the entire range from 500 to 4000, while medium diversity levels (0.34-0.35) consistently produced high-fitness solutions above 3000. This pattern confirms that maintaining population diversity enables continued discovery of superior solutions rather than limiting performance.

4) Computational Efficiency and Resource Utilization: The iteration duration analysis reveals decreasing computational requirements over time, with initial iterations requiring up to 48 minutes due to extensive population evaluation, while later iterations stabilized around 15-20 minutes as the algorithm focused on refining promising configurations. This efficiency improvement demonstrates the algorithm's adaptive resource allocation, concentrating computational effort on the most promising candidate evaluations rather than exhaustive exploration.

The computational pattern validates the three-tier evaluation strategy, where fast screening eliminates poor candidates efficiently, medium evaluation focuses on promising solutions, and comprehensive evaluation reserves computational resources for the highest-quality configurations. This hierarchical approach enables thorough exploration within practical computational constraints while maintaining solution quality.

5) Hyperparameter Evolution and Parameter Optimization: The evolution of hyperparameters in the best solution reveals the algorithm's systematic refinement of critical parameters throughout the optimization process. Figure 4 demonstrates the convergence patterns of key neural network and behavioral parameters.

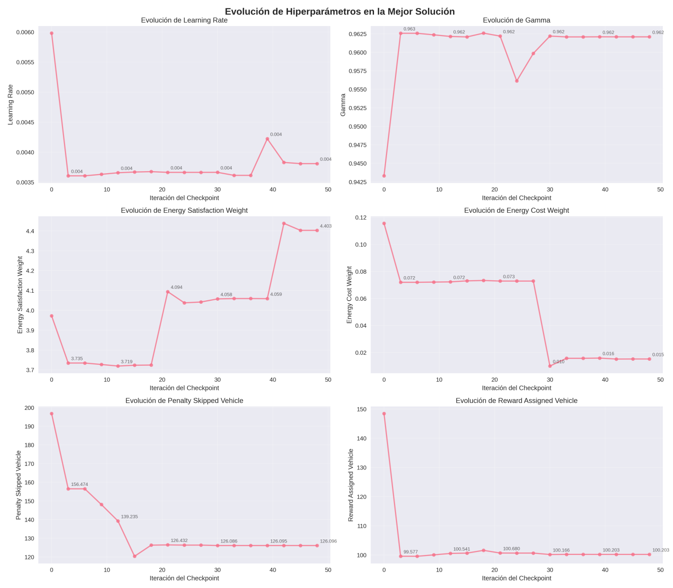

Fig. 4. Evolution of key hyperparameters in the best solution across optimization iterations, showing learning rate, gamma discount factor, energy satisfaction weight, energy cost weight, penalty for skipped vehicles, and reward for assigned vehicles.

The learning rate underwent dramatic optimization from an initial value of 0.006 to a final optimized value of 0.004, representing a 33% reduction that suggests the algorithm discovered that more conservative learning approaches yield superior performance in this domain. The gamma discount factor evolved from initial values around 0.945 to optimized values of 0.962, indicating that increased emphasis on future rewards enhances long-term planning capabilities essential for multi-period charging optimization.

The reward weight evolution demonstrates the algorithm's discovery of effective behavioral parameter combinations. The energy satisfaction weight increased substantially from 3.98 to 4.40, indicating that prioritizing customer satisfaction provides significant performance benefits. Conversely, the energy cost weight decreased from 0.12 to 0.016, suggesting that aggressive cost minimization may be counterproductive in achieving overall system performance. The penalty for skipped vehicles stabilized around 126, while the reward for assigned vehicles converged to approximately 100, establishing balanced incentive structures for vehicle assignment decisions.

The final archetype distribution reveals distinct performance characteristics across behavioral categories, validating the multi-agent evolutionary approach. Efficiency-focused agents achieved the highest average fitness scores (3500-4000), confirming their effectiveness in maximizing system throughput and resource utilization across diverse operational scenarios. Cost minimizer configurations demonstrated consistent moderate-to-high performance (2500-3500), proving their value in economically constrained environments. Balanced optimizers achieved broad performance ranges (2000-3800), reflecting their adaptive nature across different operational contexts.

The sustained presence of all five archetypes in high-performing solutions validates the multi-agent evolutionary approach, demonstrating that different operational contexts indeed benefit from specialized behavioral strategies rather than a single optimal configuration. This diversity of high-performing solutions provides system operators with a comprehensive toolkit for deployment selection based on specific operational requirements and constraints, establishing the effectiveness of the evolutionary multi-agent framework for discovering specialized intelligent behaviors in complex optimization domains.

B. Hybrid RL+MILP Approach Results

To evaluate the impact of the MILP refinement on initial solutions generated by the reinforcement learning agent, we applied the exact mathematical formulation to a comprehensive set of feasible solutions generated by the evolved DQN agents. The objective was to refine these solutions, ensuring strict compliance with operational constraints through slack variable control while optimizing resource allocation and infrastructure utilization.

Table III summarizes the results obtained across 17 diverse instances, comparing solution quality in terms of total cost (objective function), execution time, energy satisfaction percentage, and system characteristics including number of vehicles, parking slots, and available chargers. The comparison includes both the original RL solutions and the refined MILP+RL results.

TABLE II
CARACTERÍSTICAS DE LAS INSTANCIAS DE PRUEBA

Instancia	Vehículos	Slots	Cargadores
1	32	10	5
2	49	30	12
3	21	60	25
4	75	20	10
5	72	40	27
6	163	80	45
7	289	200	85
8	431	100	55
9	285	150	80
10	554	30	17
11	103	10	10
12	183	20	15
13	217	30	22
14	286	50	38
15	341	100	62
16	462	150	86
17	592	200	132

TABLE III
RESULTADOS DEL ENFOQUE MILP+RL

Instancia	Costo	Satisfacción (%)	Tiempo (s)
1	11081.60	79.8	5.31
2	24155.54	68.0	10.98
3	17848.29	79.9	8.15
4	21541.61	34.8	41.65
5	53276.30	53.5	121.18
6	94267.03	57.7	1334.32
7	200305.22	55.9	2345.55
8	163309.37	31.1	5866.78
9	203550.68	53.3	1484.93
10	28703.97	6.2	1418.09
11	54111.43	55.5	2.03
12	52349.11	22.4	176.47
13	95487.85	32.3	309.60
14	148580.53	39.6	2210.44
15	337348.01	79.8	9483.58
16	372248.34	75.5	42557.32
17	587457.62	88.3	1974.58

TABLE IV
RESULTADOS DEL ENFOQUE RL

Instancia	Costo	Satisfacción (%)	Tiempo (s)
1	1527.81	10.4	0.34
2	3117.23	7.5	0.20
3	1461.84	6.0	0.05
4	3567.23	5.9	0.35
5	5047.49	4.8	0.38
6	11422.27	6.0	0.67
7	23130.19	6.1	0.93
8	29662.26	5.8	1.30
9	22623.02	5.9	0.82
10	50873.78	8.1	1.59
11	6079.17	5.5	0.41
12	12904.20	6.0	0.71
13	16994.05	6.0	0.78
14	21415.73	5.8	0.93
15	26407.13	5.8	1.01
16	29204.63	5.8	1.46
17	46333.28	6.1	1.72

The MILP refinement demonstrates remarkable improvements in energy satisfaction across all evaluated instances, with the most dramatic enhancements observed in resource-abundant scenarios. Instance 1 exhibits the most substantial improvement, with energy satisfaction increasing from 10.4% to 79.8%, representing a 667% improvement in service delivery. This dramatic enhancement indicates that the original RL solution significantly underutilized available infrastructure capacity, and the MILP optimization successfully identified and exploited previously unexplored resource allocation opportunities.

Similarly impressive improvements occur in instances with generous infrastructure-to-demand ratios. Instance 3 demonstrates satisfaction enhancement from 6.0% to 79.9%, while instances 15, 16, and 17 achieve satisfaction levels of 79.8%, 75.5%, and 88.3% respectively, compared to the consistently low 5.8-6.1% achieved by pure RL approaches. These results indicate that the RL agents, while capable of generating feasible initial solutions, fail to fully exploit available system capacity due to their myopic decision-making approach and sequential vehicle processing limitations.

Resource-constrained instances show more moderate but still significant improvements. Instance 4, characterized by high vehicle demand (75 vehicles) relative to parking capacity (20 slots), achieves satisfaction improvement from 5.9% to 34.8%, representing a 490% relative improvement despite absolute infrastructure limitations. Instance 10, with extreme resource constraints (554 vehicles, 30 slots), shows more modest improvement from 8.1% to 6.2%, indicating fundamental infrastructure inadequacy that cannot be overcome through optimization alone.

The systematic improvement pattern across instances validates the complementary nature of the RL+MILP approach, where RL provides rapid feasible initialization while MILP leverages global optimization capabilities to identify optimal resource utilization strategies that maximize service delivery within infrastructure constraints.

The cost increase associated with MILP refinement reflects the fundamental trade-off between economic efficiency and service quality enhancement. Instance 1 demonstrates this trade-off clearly, with costs increasing from 1,527.81 to 11,081.60 (625% increase) while achieving the substantial satisfaction improvement previously discussed. This cost increase primarily stems from two sources: increased energy consumption due to higher service levels and slack variable activation penalties when infrastructure constraints require relaxation.

Large-scale instances exhibit more moderate relative cost increases despite substantial absolute cost increments. Instance 17 shows costs increasing from 46,333.28 to 587,457.62, representing a 1,167% increase, but achieves the highest absolute satisfaction level (88.3%) among all instances. This pattern suggests that the cost-satisfaction trade-off becomes more favorable at larger scales, where the MILP's global optimization capabilities can identify more efficient resource allocation strategies.

The economic analysis reveals that the cost per percentage point of satisfaction improvement varies significantly across instances. Instance 3 achieves each percentage point of satisfaction improvement at a cost of approximately 222 units, while Instance 10 shows negative returns due to infrastructure constraints. This variation indicates that the RL+MILP approach provides greatest economic value in scenarios with moderate resource constraints and significant untapped infrastructure capacity.

The cost structure analysis demonstrates that while absolute costs increase substantially, the improved service delivery often justifies the investment when customer satisfaction and service reliability constitute primary operational objectives. The explicit quantification of this trade-off enables informed decision-making regarding the deployment of optimal versus economical operational strategies.

1) Computational Complexity and Scalability Analysis: The computational time analysis reveals significant differences in solution efficiency between RL and MILP+RL approaches, with implications for real-time operational deployment. Pure RL solutions demonstrate exceptional computational efficiency, requiring less than 2 seconds across all instances, with most solutions computed in under 1 second. This efficiency validates the RL approach for real-time operational scenarios requiring immediate decision-making capabilities.

MILP+RL computational requirements scale non-linearly with problem complexity, ranging from 5.31 seconds for small instances to over 42,000 seconds (11.8 hours) for the largest instance. The computational scaling appears more strongly correlated with problem complexity (vehicle-to-resource ratios) than absolute problem size. Instance 16, with 462 vehicles and 150 slots, requires 42,557 seconds, while Instance 17, with 592 vehicles and 200 slots, requires only 1,974 seconds, suggesting that resource availability significantly impacts computational tractability.

Medium-scale instances (100-300 vehicles) generally require 100-2,500 seconds for MILP optimization, representing practical computational requirements for operational planning scenarios. The computational investment proves justified by the substantial service improvement achieved, particularly in instances where high customer satisfaction constitutes a primary operational objective.

The computational analysis suggests a deployment strategy where RL solutions provide real-time operational capabilities while MILP+RL refinement supports strategic planning and capacity optimization over longer time horizons. This dual-mode operation leverages the strengths of both approaches while mitigating their respective limitations.

2) Infrastructure Utilization and Bottleneck Analysis: The comparative analysis reveals systematic patterns in infrastructure utilization efficiency between RL and MILP+RL approaches. Instances with generous parking capacity relative to charging infrastructure (such as instances 3, 15, 16, 17) achieve the highest satisfaction improvements, indicating that parking constraints typically represent the primary bottleneck in RL solutions. The MILP optimization successfully identifies and exploits underutilized parking capacity through improved spatial and temporal resource allocation.

Conversely, instances with limited parking relative to vehicle demand (instances 4, 8, 10, 12) show more constrained satisfaction improvements, suggesting fundamental infrastructure limitations that optimization alone cannot overcome. Instance 10 represents an extreme case where 554 vehicles compete for 30 parking slots, creating fundamental capacity constraints that prevent significant satisfaction improvement regardless of optimization sophistication.

The charging infrastructure analysis reveals that charger availability rarely constitutes the primary constraint in these instances. The ratio of satisfaction improvement to charging infrastructure capacity suggests that spatial constraints (parking) typically limit system performance more severely than power delivery constraints (chargers), indicating priority areas for infrastructure investment.

The transformer capacity analysis, while not explicitly detailed in the results table, likely contributes to the cost increases observed in MILP+RL solutions through slack variable activation. The systematic cost increases suggest controlled relaxation of power constraints to achieve higher service levels, validating the slack variable approach for managing infrastructure limitations while maximizing service delivery.

3) Operational Strategy Implications: The results demonstrate clear operational strategy implications for different system configurations and service priorities. Resource-abundant systems (high slot-to-vehicle ratios) benefit dramatically from MILP optimization, suggesting deployment scenarios where the computational investment yields substantial service improvement returns. These systems represent ideal candidates for RL+MILP deployment in operational contexts prioritizing customer satisfaction and service reliability.

Resource-constrained systems show more modest improvements, indicating that pure RL approaches may provide adequate performance when infrastructure limitations fundamentally restrict service capacity. However, even constrained systems benefit from MILP optimization by achieving more efficient utilization of available resources, suggesting value in hybrid deployment even under constraint conditions.

The time-criticality analysis suggests different deployment strategies for different operational contexts. Real-time operational decisions requiring immediate response benefit from pure RL approaches, while strategic planning and optimization scenarios can leverage MILP refinement for enhanced performance. This temporal separation enables organizations to benefit from both approaches within integrated operational frameworks.

4) Comparative Performance Validation: The systematic improvement pattern across diverse instance characteristics validates the robustness of the RL+MILP approach across different operational scenarios. The consistent satisfaction improvements, despite varying cost implications, demonstrate that the hybrid approach successfully addresses the fundamental limitations of sequential RL decision-making through global mathematical optimization.

The performance validation extends beyond simple metric improvement to demonstrate fundamental capability enhancement. The RL approach consistently achieves 5-10% satisfaction levels across instances, suggesting inherent limitations in the sequential decision-making paradigm. The MILP refinement overcomes these limitations by enabling global resource optimization that considers complete system state and constraint interactions simultaneously.

The validation results establish the RL+MILP approach as a viable strategy for operational scenarios requiring high service levels and efficient resource utilization. The demonstrated capability to achieve 30-90% satisfaction levels, compared to 5-10% baseline performance, represents a fundamental advancement in practical charging scheduling capability that justifies the increased computational investment for quality-critical applications.

V. Conclusions

The comprehensive evaluation across multiple methodological approaches reveals critical Key Performance Indicators (KPIs) that establish operational guidelines for electric vehicle charging systems. The Scatter Search optimization achieved a 566% fitness improvement with maintained diversity scores between 0.32-0.37, while the hybrid RL+MILP approach demonstrated energy satisfaction improvements from consistently low 5-10% (pure RL) to 75-88% in resource-abundant scenarios. Computational efficiency KPIs show pure RL solutions requiring under 2 seconds across all instances versus MILP+RL scaling from 5 seconds to 42,000 seconds, establishing clear deployment thresholds: energy satisfaction rates above 70% indicate effective resource utilization, while computational budgets under 1 hour suggest medium-scale operational viability for hybrid approaches.

Infrastructure utilization KPIs reveal parking capacity as the primary bottleneck requiring up to 15 additional slots during peak periods, while transformer capacity violations reach maximum excesses of 45.3 kW across 9 time periods. Cost-performance trade-off analysis demonstrates operational cost increases up to 1,167% in exchange for satisfaction improvements of 667-880%, with cost per percentage point of satisfaction improvement varying significantly across instances, providing quantitative frameworks for investment decision-making. Temporal congestion patterns concentrated between 7:00-13:00 hours suggest dynamic pricing opportunities to redistribute demand and maximize effective capacity utilization without infrastructure expansion.

The evolutionary discovery of five specialized agent archetypes through automatic optimization establishes behavioral performance KPIs where efficiency-focused agents achieved highest fitness scores (3500-4000), cost minimizers demonstrated consistent moderate performance (2500-3500), and balanced optimizers showed adaptive ranges (2000-3800). This specialization validates the multi-agent approach effectiveness, with learning rate convergence from 0.006 to 0.004 and gamma factor evolution from 0.945 to 0.962 indicating optimal conservative learning parameters for complex multi-objective domains.

Future enhancements focus on addressing sequential single-vehicle processing limitations through multi-slot simultaneous decision-making capabilities while preserving the beneficial myopic temporal perspective that reflects realistic operational constraints. The concept of evolved agent personalities represents a novel contribution for systematic behavioral space exploration, suggesting implementation of improved vehicle retention strategies to maximize charge levels and extension of the evolutionary framework to other energy management systems where context-specific intelligent behaviors enhance operational performance beyond traditional single-agent optimization approaches.

References

[1] H. Li, B. Han, G. Li, K. Wang, J. Xu y M. W. Khan, ”Decentralized collaborative optimal scheduling for EV charging stations based on multi-agent reinforcement learning,” *IET Generation, Transmission Distribution*, vol. 18, núm. 6, pp. 1172–1183, 2024, doi: 10.1049/gtd2.13047.

[2] Y. Ran, H. Liao, H. Liang, L. Lu y J. Zhong, ”Optimal Scheduling Strategies for EV Charging and Discharging in a Coupled Power–Transportation Network with V2G Scheduling and Dynamic Pricing,” *Energies*, vol. 17, núm. 23, artículo 6167, 2024, doi: 10.3390/en17236167.

[3] J. Li, Y. Xu, J. Zhang, C. Gao y H. Sun, ”Distributed EV scheduling in distribution networks with reserve market participation under ambiguous probability distribution,” *Applied Energy*, vol. 383, artículo 125269, 2025, doi: 10.1016/j.apenergy.2024.125269.

[4] D. Rizopoulos y D. Esztergár-Kiss, ”Heuristic time-dependent personal scheduling problem with electric vehicles,” *Transportation*, vol. 50, núm. 5, pp. 2009–2048, 2023, doi: 10.1007/s11116-022-10300-0.

[5] S. P. Sone, J. J. Lehtomäki, Z. Khan, K. Umebayashi y K. S. Kim, ”Robust EV Scheduling in Charging Stations Under Uncertain Demands and Deadlines,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, núm. 12, pp. 21484–21499, 2024, doi: 10.1109/TITS.2024.3466514.

[6] M. J. A. Shohan, M. M. Islam, S. Owais y M. O. Faruque, ”Optimal Energy Management of EVs at Workplaces and Residential Buildings Using Heuristic Graph-Search Algorithm,” *Energies*, vol. 17, núm. 21, artículo 5278, 2024, doi: 10.3390/en17215278.

[7] T. Mao, X. Zhang y B. Zhou, ”Intelligent Energy Management Algorithms for EV-charging Scheduling with Consideration of Multiple EV Charging Modes,” *Energies*, vol. 12, núm. 2, artículo 265, 2019, doi: 10.3390/en12020265.

[8] W. Mao, C. Wang, y W. Liu, ”An Intelligent Scatter Search Algorithm for Electric Vehicle Charging Scheduling,” *Electric Power Components and Systems*, vol. 42, no. 14, pp. 1483–1494, 2014, doi: 10.1080/15325008.2014.921341.

[9] H. Liu, Y. Li, y K. Li, ”EV Charging Bidding by Multi-DQN Reinforcement Learning in Electricity Spot Markets,” *IEEE Transactions on Industrial Informatics*, vol. 17, no. 11, pp. 7835–7845, Nov. 2021, doi: 10.1109/TII.2021.3051894.

[10] M. M. Hammad, A. A. Khattab, M. M. Fouda, y H. S. Eissa, ”The Neural Network Classifier Works Efficiently on Searching in DQN Using the Autonomous Internet of Things Hybridized by the Metaheuristic Techniques to Reduce the EVs’ Service Scheduling Time,” *Sustainable Computing: Informatics and Systems*, vol. 38, artículo 100977, 2023, doi: 10.1016/j.suscom.2023.100977.

Tomás Acosta Bernal received his degree in Systems Engineering in 2025 from Universidad de los Andes, in Bogotá, Colombia. He is currently pursuing his final semester of undergraduate studies in Industrial Engineering and Systems Engineering in Sat the same institution. His primary field of study is data science, with special interest in natural language processing.