A comparative study of multi-objective and neuroevolutionary-based reinforcement learning algorithms for optimizing electric vehicle charging and load management

Neele Kemper, Michael Heider, Dirk Pietruschka, Jörg Hähner

Angaben zur Veröffentlichung / Publication details:

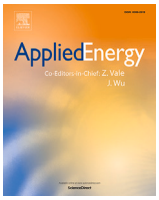
Kemper, Neele, Michael Heider, Dirk Pietruschka, and Jörg Hähner. 2025. "A comparative study of multi-objective and neuroevolutionary-based reinforcement learning algorithms for optimizing electric vehicle charging and load management." Applied Energy 391: 125890. https://doi.org/10.1016/j.apenergy.2025.125890.

Applied Energy 391 (2025) 125890

Contents lists available at ScienceDirect

Applied Energy

journal homepage: www.elsevier.com/locate/apen



A comparative study of multi-objective and neuroevolutionary-based reinforcement learning algorithms for optimizing electric vehicle charging and load management

Neele Kemper[a,*], Michael Heider[a], Dirk Pietruschka[b], Jörg Hähner[a]

[a] Organic Computing Group, University of Augsburg, Augsburg, 86159, Bavaria, Germany
[b] Technology Transfer Center for Sustainable Energies, Aschaffenburg University of Applied Science, Aschaffenburg, 63743, Bavaria, Germany

HIGHLIGHTS

- Propose neuroevolution for optimizing EV charging and grid stability.
- Adapt existing reinforcement learning approaches to support multiple objectives.
- Outperform gradient-based methods in multi-objective constraint optimization.
- Demonstrate strong generalization, adaptability, and robust performance.

ARTICLE INFO

Keywords:
Charging management
Electric vehicle charging
Evolutionary reinforcement learning
Multi-objective reinforcement learning
Neuroevolution
Smart grid

ABSTRACT

The electrification of transportation requires the development of smart charging management systems for electric vehicles to optimize grid performance and enhance user satisfaction. However, existing methods often reduce multi-objective problems to single-objective formulations, limiting their ability to balance conflicting objectives and requiring iterative runs for diverse solutions.

In this study, we propose a Multi-Objective Evolutionary Reinforcement Learning (MOEvoRL) framework to optimize electric vehicle charging strategies and discover multiple policies within a single training run. Our approach focuses on maximizing the batteries' state of charge, increasing photovoltaic power consumption, reducing peak loads, and smoothing the overall load on the grid. Simultaneously, it adheres to essential grid constraints, such as load balancing and grid connection node limits, to ensure grid stability, efficiency, and real-world applicability.

MOEvoRL utilizes the exploratory power of Evolutionary Algorithms and the sequential decision-making strengths of Reinforcement Learning. By employing neuroevolution, we optimize the weights and topologies of policy networks.

Our approach employs the Non-dominated Sorting Genetic Algorithm II, Strength Pareto Evolutionary Algorithm 2, and a modified NeuroEvolution of Augmenting Topologies as optimizers and benchmarks their performance against the gradient-based Multi-Objective Deep Deterministic Policy Gradient (MODDPG) and a single-objective DDPG that simplifies multiple objectives into a single objective using a linear scalarization function.

The results show that MOEvoRL approaches are superior to MODDPG in terms of generalization, robustness, constraint compliance, and multi-objective optimization capabilities. In contrast, DDPG exhibits poor and unstable performance. This positions MOEvoRL as a robust strategy for managing electric vehicle charging while optimizing local grid loads.

* Corresponding author.
Email address: neele.kemper@uni-a.de (N. Kemper).

https://doi.org/10.1016/j.apenergy.2025.125890
Received 19 December 2024; Received in revised form 7 March 2025; Accepted 5 April 2025
Available online 19 April 2025

1. Introduction

In light of climate change, it is becoming increasingly urgent to transition transportation to a more sustainable and energy-efficient sector. The shift from combustion engines to environmentally friendlier Electric Vehicles (EVs) is a multifaceted challenge, particularly due to the impact on energy demand and grid infrastructure. Smart Charging Management (CM) plays a crucial role in this transition by managing increasing electricity demand, optimizing load balancing on the local distribution grid, integrating renewable energy sources, and ensuring user satisfaction while addressing unbalanced load issues.

The CM we propose aims to optimize three objectives for efficient and sustainable EV charging, while considering load management and system resilience. First, maximizing the battery State of Charge (SoC) ensures that EVs are charged reliably and on demand, which is crucial for user acceptance of CM strategies. Second, maximizing photovoltaic (PV) power consumption reduces reliance on external energy sources, which are often costly and polluting, leading to more cost-efficient and sustainable energy use. Third, by reducing and smoothing peak loads, the system directly improves load management and minimizes strain on the local distribution grid infrastructure. In addition, practical considerations are incorporated by simulating a three-phase grid following German regulations,[1] an important factor for maintaining grid integrity. The regulations include avoiding unbalanced loads and limiting external grid power draw, thereby preventing infrastructure damage and ensuring efficient and reliable grid operation.

Given the complexity of these objectives and constraints, as well as the sequential nature of decision-making, the CM is modelled as a Multi-Objective Markov Decision Process (MOMDP). This approach enables the balancing of multiple competing objectives and is well suited to address the temporal and stochastic challenges of EV charging in a dynamic real-world environment, optimizing both short-term gains and long-term sustainability.

While Deep Reinforcement Learning (DRL) has shown strong performance in decision-making tasks and has been widely explored for EV-CM, it faces challenges such as partial observability, early convergence, sparse rewards, and difficulty in balancing conflicting objectives. Existing Multi-Objective RL (MORL) methods often use scalarization techniques to simplify multi-objective problems into single-objective formulations. However, these approaches are computationally expensive, as they require training separate models for each scalarization function, rely on predefined scalarization assumptions, and struggle to generalize across diverse objectives.

To overcome these limitations, we propose a neuroevolution-based Multi-Objective Evolutionary Reinforcement Learning (MOEvoRL) framework that leverages population-based search strategies. By integrating Evolutionary Algorithms (EAs) into the RL process, MOEvoRL natively optimizes multiple objectives without relying on repeated scalarizations or predefined utility assumptions, enhancing computational efficiency and real-world applicability. EvoRL's global search capabilities and genetic diversity enhance exploration, prevent premature convergence, and improve resilience to sparse rewards and incomplete information. Multi-Objective EAs (MOEAs) are inherently well-suited for optimizing conflicting objectives simultaneously. Furthermore, EvoRL's scalable evolutionary strategies and low sensitivity to environmental noise further strengthen its robustness and efficiency.

Despite its potential, EvoRL remains underexplored in MORL, even though it has been widely studied and successfully applied to single-objective RL [1]. Unlike gradient-based methods, the proposed MOEvoRL framework simultaneously discovers diverse policies for conflicting objectives in a single training execution, without the need for predefined scalarization assumptions. This capability, combined with its robust exploration and adaptability, makes it an effective and practical solution for complex, partially observable environments.

Our contributions are as follows:

- We introduce a neuroevolution-based method that optimizes multiple objectives within a single training process, eliminating the need for iteratively solving single-objective problems.
- We present lightweight neuroevolution-based MOEvoRL methods with a simpler algorithmic structure compared to complex, resource-intensive gradient-based DRL techniques, making our approach more suitable for real-time and resource-constrained scenarios.
- We benchmark *Deep Deterministic Policy Gradient (DDPG)*, which addresses multi-objective problems by reducing them to single-objective formulations using linear scalarization, and *Multi-Objective Deep Deterministic Policy Gradient (MODDPG)*, a gradient-based extension of DDPG.
- The approach is evaluated in a realistic simulation environment with practical regulatory constraints, such as power grid connection limits and unbalanced load avoidance. In this RL setting, the deterministic rewards ensure consistent evaluation of the central agent, which optimizes charging currents for 5–15 Charging Stations (CSs) at one-minute intervals. This small-scale workplace scenario provides a relevant testbed to assess the effectiveness and adaptability of MOEvoRL in multi-objective CM.
- By optimizing load stability, maximizing PV energy utilization, and ensuring reliable EV charging, our framework enables the sustainable and cost-effective integration of EV fleets into existing electrical grids, contributing to a cleaner and more resilient transportation infrastructure.

The paper is structured as follows: Section 2 provides an overview of recent work in the field of MORL within EV-CM. Section 3 introduces the main theoretical concepts of the analysed MORL algorithms. The simulation setup and its RL representation are described in detail in Section 4. Section 5 outlines the experimental setup and evaluation. The results are presented in Section 6 and discussed in Section 7. Section 8 concludes with a summary and future research directions.

2. Related work

The literature on EV-CM has increasingly explored RL and DRL approaches to tackle dynamic and complex decision-making tasks [2,3]. Early studies often simplified the inherently multi-objective nature of EV-CM by scalarizing multiple objectives into a single one [4–6]. While this approach offers computational simplicity, it limits the ability to effectively balance trade-offs. More advanced MORL methods have emerged, but they often rely on gradient-based DRL techniques, which struggle to manage conflicts among objectives.

He et al. [7] present an entropy-constrained Proximal Policy Optimization algorithm that utilizes policy entropy and multi-objective rewards to train five distinct agents. They later propose a multi-objective Actor-Critic method which uses a user-customizable utility function to track travel efficiency and energy savings [8]. Zhang et al. [9] describe the Hamlet method, a hierarchical MORL strategy that dynamically manages EV charging performance, where a leader sets targets based on current energy conditions and followers cooperatively optimize charging strategies to maximize financial and operational outcomes. However, these methods rely on solving multiple scalarized single-objective formulations, requiring assumptions about the scalarization functions and substantially increasing computational overhead.

Multi-agent approaches have also been applied to address multi-objective problems in EV-CM. A multi-agent selfish-collaborative architecture developed by Silva et al. [10] balances energy cost, battery SoC, and overload prevention, enabling agents to adapt to user preferences for self-optimization or collaboration, optimizing both individual and collective outcomes. Yang et al. [11] use the Nash-Q learning method

[1] Germany is selected for this study as it is the largest EU member state and has the largest EV fleet within the EU.

for multi-agent optimization in the energy management of hybrid vehicles, which targets economic and battery efficiency in a stochastic game. Abid et al. [12] improve the exploration and generate diverse experiences for optimizing EV charging scheduling and resource allocation by integrating MOEAs with multi-agent Deep Deterministic Policy Gradient (DDPG). Similarly, Adetunji et al. [13] address the multi-objective EV charging problem using a two-tailed incentive pricing scheme within a MOMDP framework. They employ DDPG to obtain Pareto-optimal solutions, while leveraging population-based EA exploration to ensure diverse experience generation. These multi-agent RL approaches assign individual objectives to separate agents, with each agent optimizing a single objective. While this enables handling multiple objectives within a single-objective framework, it often leads to challenges such as poor coordination among agents, difficulty in balancing trade-offs, and increased communication overhead, which can hinder scalability and overall system performance.

The integration of evolutionary techniques with DRL has shown promise in overcoming the limitations of gradient-based methods; however, their application has so far been limited to single-objective RL. An Evolutionary Curriculum Learning multi-agent Twin Delayed DDPG, developed by Li et al. [14], enhances adaptability to complex challenges like EV charging by combining incremental learning with evolutionary computation, resulting in improved outcomes and faster convergence. To address stochastic EV charging in a connected power–transport network, Qian et al. [15] design an RL framework that combines gradient-based DDPG with gradient-free Deep Genetic Policy (DGP) algorithms, with DGP outperforming DDPG by evolving policy parameters through selection and mutation. Zhou et al. [16] improve the energy management of fuel cell EVs by integrating Particle Swarm Optimization with DDPG to optimize hybridization parameters under unknown driving conditions. Finally, Fan et al. [17] introduce an evolutionary-DRL algorithm that enhances convergence and robustness in multi-microgrid systems by combining DDPG with EAs and a novelty search mechanism. DDPG policy network parameters are treated as individuals within the EA.

Despite these advancements, existing methods often simplify multi-objective problems by decomposing them into multiple single-objective formulations, which hinders their ability to optimize trade-offs among conflicting objectives and increases computational costs. This work extends the current literature by introducing a neuroevolution-based MORL framework that inherently handles multiple objectives. The framework enables efficient and direct discovery of robust EV charging strategies, leveraging key strengths of EAs, such as global search capabilities and diverse exploration. By utilizing the natural multi-objective optimization capabilities of MOEAs, the proposed method eliminates the need for repeated scalarizations, offering a computationally efficient and scalable solution for complex, real-world scenarios.

3. Background

The multi-objective decision problem is formalized as MOMDP [18], represented by the tuple $\langle S, A, T, \gamma, \mu, \mathbf{R}\rangle$. Here, S and A are finite sets of states and actions; $T : S \times A \times S \to [0, 1]$ is the probabilistic transition function; $\gamma \in [0, 1)$ is the discount factor; $\mu : S \to [0, 1]$ is the initial state distribution; and $\mathbf{R} : S \times A \times S \to \mathbb{R}^d$ is the vector-valued reward function that defines the expected immediate rewards for each combination of state, action and subsequent state for $d \geq 2$ objectives. The main difference between a single-target MDP and a MOMDP lies in the vector-valued reward function $\mathbf{R}$, which provides a numerical feedback signal for each considered objective.

In MOMDPs, an agent's actions follow a policy π from the feasible set Π, where π defines a probabilistic relationship $\pi : S \times A \to [0, 1]$ between states and actions. The expected value, or the value function, of adhering to a policy π in a MOMDP is defined as:

$$\mathbf{V}^{\pi} = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k \mathbf{r}_{k+1} | \pi, \mu\right], \tag{1}$$

where $\mathbf{r}_{k+1} = \mathbf{R}(s_k, a_k, s_{k+1})$ denotes the reward obtained at the $k + 1$th timestep. For MOMDPs the value function $\mathbf{V}^{\pi}$ is a vector, with each component representing a different objective, thus $\mathbf{V}^{\pi} \in \mathbb{R}^d$.

Using a utility function (or scalarization function) $u : \mathbb{R}^d \to \mathbb{R}$ simplifies the MOMDP to a single-objective MDP by mapping multi-objective value of a policy to a scalar value:

$$\mathbf{V}_u^{\pi} = u(\mathbf{V}^{\pi}). \tag{2}$$

However, applying such a utility function may not always be feasible and multiple policies have to be learned, known as multi-policy MORL [18].

Multi-policy MORL can be categorized into outer loop and inner loop approaches. Outer loop methods iterate through different utility functions, re-running a single policy method for each function. Inner loop methods, used in this paper, modify the underlying RL algorithm to find multiple policies simultaneously without needing explicit assumptions about the utility function.

3.1. Multi-objective deep deterministic policy gradient

DDPG [19] is a widely used DRL algorithm for single-objective CM [4,5]. We extend DDPG to handle MOMDPs through the MODDPG algorithm (see Algorithm 1 in Appendix A), which incorporates a weight vector $\mathbf{w}_t \in \mathbb{R}^d$ at each timestep for scalarizing multiple objectives.

In MODDPG, the actor $\mu(s, \mathbf{w}|\theta^{\mu})$ generates actions based on the state s and the weight vector $\mathbf{w}_t$, while the critic $Q(s, a, \mathbf{w}|\theta^{Q})$ evaluates these actions using the same priorities. This dual integration ensures that policy generation and evaluation consistently align with multiple objectives.

The weight vector $\mathbf{w}_t$ dynamically scalarizes the multi-objective reward into a single scalar value, optimizing trade-offs and allowing the algorithm to adapt to changing priorities. It is sampled from a Dirichlet distribution at each timestep to ensure uniform exploration across the objective space. The weight vector is stored in the experience replay buffer alongside the state, action, next state, and reward to preserve context during training.

During training, the actor and critic networks are iteratively updated using the weight vector to focus on different objectives. Target networks are smoothly updated to stabilize learning, and experience replay is used to ensure convergence by breaking sample correlations.

The MODDPG algorithm can theoretically find the optimal Pareto Front (PF) by leveraging systematic exploration and optimization. Weight vectors $\mathbf{w}$, sampled from a Dirichlet distribution, ensure uniform and dense coverage of the PF, while scalarizing the multi-objective reward as $\mathbf{w}^{\top}\mathbf{r}$ enables the optimization of diverse trade-offs. Although weighted sum methods are limited to capturing convex regions of the PF, MODDPG can approximate non-convex regions under suitable conditions. This is achieved through its iterative learning process, which enables the discovery of complex, non-linear trade-offs. The actor network $\mu(s, \mathbf{w})$ learns deterministic policies for different weight vectors $\mathbf{w}$, allowing diverse solutions across the PF. In theory, with expressive function approximators and sufficient exploration, MODDPG can approximate the PF by converging to optimal policies for each $\mathbf{w}$. However, this relies on ideal conditions, including infinite training time, a large enough replay buffer, and optimal hyperparameter tuning.

3.2. Evolutionary reinforcement learning

In EvoRL [20], RL components – such as agents, actions, or parameters – are treated as individuals within a population. These individuals evolve over successive generations through evolutionary operations like crossover, mutation, and selection, enabling diversification and improvement in policy search.

Neuroevolution focuses on optimizing the neural networks that represent RL policies, which determine the agent's actions in response to environmental states – the core of decision-making in RL. Unlike gradient-based DRL, which uses backpropagation to adjust network weights, neuroevolution employs gradient-free EAs to optimize both

network weights and architectures. This approach is particularly advantageous in environments with sparse rewards, non-differentiable objectives, or when gradient computation is infeasible or inefficient.

The connection between neuroevolution and RL lies in the evaluation and improvement of policies. In neuroevolution, the cumulative reward obtained over an episode serves as the fitness value for each policy network. This fitness value is used to rank and select policies for evolutionary operations. This process aligns directly with RL's objective of maximizing cumulative rewards, as iterative evolutionary updates drive the discovery of increasingly effective strategies that progressively approach optimal or near-optimal strategies.

In MOEvoRL, integrating neuroevolution MOEAs enables the efficient handling of multiple conflicting objectives by identifying solutions that converge to the PF without the need for scaling objectives. This enables the development of diverse policy networks that can balance trade-offs between objectives. Two widely used MOEAs are Non-dominated Sorting Genetic Algorithm II (NSGA-II) [21] and Strength Pareto Evolutionary Algorithm 2 (SPEA2) [22]. NeuroEvolution of Augmenting Topologies (NEAT) [23] is a prominent method for evolving neural network architectures and weights.

3.2.1. NSGA-II

NSGA-II is a genetic algorithm for the optimization of multi-objective problems. It ranks individuals using non-dominated sorting to find Pareto-optimal solutions and maintains diversity with crowding distance. The algorithm uses binary tournament selection, simulated binary crossover and polynomial mutation to explore the search space and prevent early convergence. Elitism ensures that the best solutions are retained, which enhances convergence speed and prevents the loss of optimal solutions. Survival selection combines parent and offspring populations to identify the best individuals for the next generation.

3.2.2. SPEA2

SPEA2 is an elitist MOEA that uses density estimation and clustering to ensure diverse solution sets and prevent premature convergence. It assigns fitness based on dominance and population density, using the kth nearest neighbour method and balancing exploration and exploitation. An external archive preserves non-dominated solutions and discards dominated solutions or those in dense regions. The offspring are generated by binary tournament selection, recombination and polynomial mutation. SPEA2 combines the population with the archive for the next generation to maintain diversity and ensure that the best solutions persist.

3.2.3. NEAT

NEAT evolves neural networks by incrementally adding nodes and connections. It uses a genetic algorithm to encode networks as genomes, with speciation protecting innovative topologies and preserving diversity. Mutations introduce variability by altering connection weights and network structures. Crossover aligns the genes of the parent genomes, preserving and combining the characteristics of both. Speciation allows genomes to compete in niches defined by topological similarities, which helps maintain and protect innovative characteristics. In multi-objective optimization, NEAT can incorporate NSGA-II to evaluate solutions based on non-domination and crowding distance, facilitating the development of solutions for multiple objectives.

4. Electric vehicle charging environment

The simulation of the EV charging process is a key component of the RL environment. It models alternating current CSs with a maximum power of 11 kW and a current limit of 16 A. The energy supply is simulated from a PV system and a public grid connection, with no other energy consumers considered. The PV system, based on real data from southwest Germany, has a peak output of 45 kW and is connected in three phases so that it does not contribute to unbalanced loads. The external grid's connection node is capped at 40 kW, with the unbalanced load limited to 20 A. A simplified load flow calculation and battery SoC estimates are used, with the CM specifying the charging current for each CS every minute. The simulation framework and the integration of the CM system are depicted in Fig. B.6.

Twelve distinct EV models are simulated, varying in battery capacity, maximum charging power, and phase connections. The specific EV configurations are detailed in Table 1.

Table 1
Summary of 12 distinct EV configurations. EVs with one connected phase use phase a, those with two connected phases are connected to phases a and b, and EVs with three connected phases utilize all three phases of the power grid.

EV model	Battery capacity (kWh)	Maximal charging power (kW)	No. of connected phases
Audi Q8 e-tron 50 quattro	89.0	11.0	3
BMW i4 eDrive40	80.7	11.0	3
Porsche Taycan	71.0	11.0	3
Mercedes EQV 250 Long	60.0	11.0	3
Fiat 600e	50.8	11.0	3
Renault Megane E-Tech EV40	40.0	11.0	3
Volkswagen ID.3 Pro	58.0	7.4	2
Mercedes EQT 200 Standard	45.0	7.4	2
Mini Cooper E	37.0	3.7	1
Dacia Spring Electric 45	25.0	3.7	1
Smart EQ fortwo	16.7	3.7	1
Audi A3 Sportback e-tron	8.8	3.7	1

4.1. Generation of charging events

Charging sessions are modelled with a Gaussian Mixture Model (GMM) to analyze the distribution of arrival time, session duration and delivered energy. GMMs capture complex, multi-modal distributions by fitting multiple Gaussian distributions using the Expectation-Maximization algorithm. The charging behaviour should be modelled as joint distributions for a more comprehensive analysis as shown by [24,25]. To determine the appropriate number of components, the elbow curve of the Bayesian information criterion is used, which balances model complexity against goodness of fit. The output is a probability distribution over arrival time, duration, and delivered energy, allowing not only an estimation of how likely each session type is, but also generation of new sessions that closely match the observed data.

The GMMs are trained on the Adaptive Charging Network (ACN) data [24], a publicly available dataset[2] of over 54,956 workplace EV charging sessions in California. Each session's arrival time, charging duration, and total energy delivered are recorded. The dataset spans 3 years (2018–2021) and shows consistent weekday usage patterns of higher morning arrival times and relatively high driver laxity. Because real-world driver behavior often clusters into a finite set of profiles, each charging session can be treated as a variation of these profiles perturbed by Gaussian noise. Thus, a GMM naturally reproduces predictable patterns and random variations in driver activity. In addition, experts have verified that the resulting sessions also correspond to typical workplace charging sessions in Europe.

Fig. 1 compares arrival times, charging durations, and energy consumption between ACN data and GMM-simulated sessions. While the GMM-generated data occasionally show slightly higher average energy consumption and a smoother distribution, the overall agreement with actual sessions is strong. Minor discrepancies in peak shapes are to be expected, given that the smoothing effect of Gaussian components naturally mitigates some of the inherent irregularities found in real data.

To validate that the GMM correctly captures the charging patterns, we apply the Energy Test [26], a non-parametric method for comparing multivariate distributions. With $\alpha = 0.05$, the test fails to reject the null

[2] See https://ev.caltech.edu/dataset.

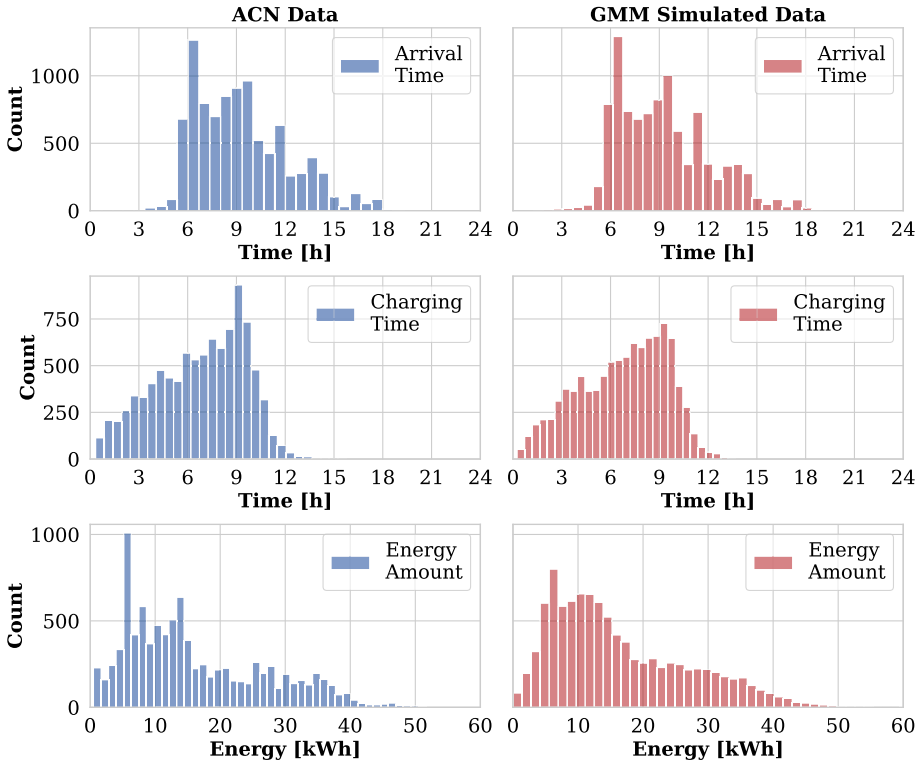


Fig. 1. Histogram comparing EV charging patterns at a workplace during weekdays. The plot shows the distribution of arrival times, charging durations, and energy consumption for both the ACN dataset and GMM-simulated data.

hypothesis ($P = 0.10$), hinting that there is no clear statistically significant difference between the GMM-generated data and the original ACN dataset. This suggests that the GMM adequately captures the underlying charging behavior. Although small biases might arise from smoothing effects in our GMM-generated charging events, these are unlikely to give any optimization method an advantage as all algorithms work with the same synthetic distribution.

Based on the energy demand, the battery's SoC upon arrival ($\mathrm{SoC_{arr}}$) and a suitable EV configuration are derived, assuming a linear relationship between energy demand and $\mathrm{SoC_{arr}}$. The skewed distribution of $\mathrm{SoC_{arr}}$ spans from 23 % to 70 %, with a mean of approximately 54 % and a standard deviation of 0.8 %.

4.2. SoC estimation and charging power calculation

The SoC at time t for the jth EV is estimated using the Coulomb Counting method, which integrates the charging power over time:

$$\mathrm{SoC}_j(t) = \mathrm{SoC}_j(t-1) + \frac{P_{\mathrm{charge},j}(t)}{C_{\mathrm{bat},j}} \cdot \Delta t, \tag{3}$$

where $C_{\mathrm{bat},j}$ is the total battery capacity of the jth EV. The charging power $P_{\mathrm{charge},j}(t)$ accounts for the efficiency η, the maximum charging power of the jth EV $P_{\mathrm{EV},j}$, and the optimal SoC $\mathrm{SoC_{opt}}$:

$$P_{\mathrm{charge},j}(t) = \begin{cases} \eta \cdot P_{\mathrm{avail},j}(t), & \text{if } \mathrm{SoC}_j(t-1) < \mathrm{SoC_{opt}}, \\ \eta \cdot \min\left(P_{\mathrm{avail},j}(t), P_{\mathrm{decay},j}(t)\right), & \text{otherwise.} \end{cases} \tag{4}$$

where $\eta = 0.9$ and $\mathrm{SoC_{opt}} = 85$ %. The decayed charging power $P_{\mathrm{decay},j}(t)$ is calculated as:

$$P_{\mathrm{decay},j}(t) = -\frac{P_{\mathrm{EV},j}}{1-\mathrm{SoC_{opt}}} \cdot \mathrm{SoC}_j(t-1) + \frac{P_{\mathrm{EV},j}}{1-\mathrm{SoC_{opt}}}. \tag{5}$$

The available charging power $P_{\mathrm{avail},j}(t)$ is determined as:

$$P_{\mathrm{avail},j}(t) = \min(P_{\mathrm{EV},j}, I_{\mathrm{CM},i}(t) \cdot V_{\mathrm{charge},j}), \tag{6}$$

where $I_{\mathrm{CM},i}(t)$ is the current provided to the ith CS at time t. The charging voltage $V_{\mathrm{charge},j}$ of the jth EV depends on the number of charging phases: 230 V for single-phase, 460 V for two-phase, and approximately 690 V for three-phase charging ($\sqrt{3} \cdot 400$ V for delta connection or $3 \cdot 230$ V for star connection). The simulation uses $\sqrt{3} \cdot 400$ V. The load on each phase $P_{\mathrm{CS,a/b/c},i}(t)$ for the ith CS is calculated as:

$$P_{\mathrm{CS,a/b/c},i}(t) = c_{\mathrm{EV,a/b/c},j} \cdot \frac{1}{n_{\mathrm{phase},j}} \cdot \frac{P_{\mathrm{charge},j}(t)}{\eta}, \tag{7}$$

where $c_{\mathrm{EV,a/b/c},j}$ indicates phase connection (1 if connected, otherwise 0). The total power $P_{\mathrm{CS},i}(t)$ drawn by the ith CS is the sum of the power on each phase:

$$P_{\mathrm{CS},i}(t) = P_{\mathrm{CS,a},i}(t) + P_{\mathrm{CS,b},i}(t) + P_{\mathrm{CS,c},i}(t). \tag{8}$$

The current of the i-CS for each phase $I_{\mathrm{CS,a/b/c},i}(t)$ is:

$$I_{\mathrm{a/b/c},i}(t) = \frac{P_{\mathrm{CS,a/b/c},i}(t)}{230\,\mathrm{V}}, \tag{9}$$

The total power and current drawn by all CSs are:

$$P_{\mathrm{CS}}(t) = \sum_{i=1}^{n} P_{\mathrm{CS},i}(t), \tag{10}$$

$$I_{\mathrm{a/b/c}}(t) = \sum_{i=1}^{n} I_{\mathrm{a/b/c},i}(t). \tag{11}$$

4.3. State space

The continuous state space is divided into *system* and *station* states. The system state captures the overall conditions of the charging simulation, while station states describe the specific conditions of each CS.

The system state includes time-dependent parameters such as PV power $P_{\mathrm{PV}}(t)$, total CS power consumption $P_{\mathrm{CS}}(t)$, and the current minute of the day t_{min}. Operational constraints include the phase currents $I_a(t)$, $I_b(t)$, and $I_c(t)$; phase current differences $\Delta I_{a,b}(t)$, $\Delta I_{a,c}(t)$, $\Delta I_{b,c}(t)$; and external grid power $P_{\mathrm{ext}}(t)$. The system state vector at time t is:

$$S_{\mathrm{system}}(t) = [P_{\mathrm{PV}}(t), P_{\mathrm{CS}}(t), t_{\mathrm{min}}, I_a(t), I_b(t), I_c(t), \Delta I_{a,b}(t), \Delta I_{a,c}(t), \Delta I_{b,c}(t), P_{\mathrm{ext}}(t)]. \tag{12}$$

Each CS and its connected EV are described by station states. The binary indicator $s_{\mathrm{CS},i}(t)$ shows the activity of the ith CS (1 if active, 0 if inactive), with inactive CS elements defaulting to zero. The SoC of the jth EV is $\mathrm{SoC}_j(t)$, with $C_{\mathrm{bat},j}$ as battery capacity, $P_{\mathrm{EV},j}$ as charging power, and $t_{\mathrm{remain},j}$ as remaining charging time. The phase connection of the jth EV to the ith CS is $p_{\mathrm{CS}_i,\mathrm{EV}_j}$. The station state vector for the ith CS and jth EV at time t is:

$$S_{\mathrm{CS}_i,\mathrm{EV}_j}(t) = \left[s_{\mathrm{CS},i}(t), \mathrm{SoC}_j(t), C_{\mathrm{bat},j}, P_{\mathrm{EV},j}, t_{\mathrm{remain},j}, p_{\mathrm{CS}_i,\mathrm{EV}_j} \right]. \tag{13}$$

The total state of the environment at time t, combining system and station states for n CSs and m EVs, is given by:

$$S(t) = S_{\mathrm{system}}(t) \oplus S_{\mathrm{CS}_1,\mathrm{EV}_1}(t) \oplus \cdots \oplus S_{\mathrm{CS}_n,\mathrm{EV}_m}(t), \tag{14}$$

where $\oplus$ denotes vector concatenation. The number of CSs n is constant, while the number of charging EVs m varies over time.

4.4. Action space

At each time step t, the agent determines the current for each CS using a continuous action vector $a(t)$, which consists of scaling factors for the charging current of each of the n CSs:

$$a(t) = [a_{\mathrm{CS},1}(t), a_{\mathrm{CS},2}(t), \ldots, a_{\mathrm{CS},n}(t)]^\top, \tag{15}$$

where each $a_{\mathrm{CS},i}(t) \in [0, 1]$ adjusts the current for the corresponding CS. The actual current supplied, $I_{\mathrm{CM}}(t)$, is calculated as:

$$I_{\mathrm{CM}}(t) = a(t) \cdot 16\mathrm{A}, \tag{16}$$

translating the normalized actions into real-time current values.

4.5. Reward structure

The reward vector is designed to maximize the SoC of each EV, increase PV power consumption, and smooth peak loads. The rewards are modelled as sparse and are returned at the end of each episode.

4.5.1. SoC reward

The SoC reward encourages maximizing EV battery capacity at departure, with a target SoC of 80 %. The reward for the jth EV, $R_{\mathrm{SoC},j}$, is calculated as:

$$R_{\mathrm{SoC},j} = \begin{cases} \left(\frac{\mathrm{SoC}_{\mathrm{dep},j}}{\mathrm{SoC}_{\mathrm{target}}} \right)^2, & \text{if } \mathrm{SoC}_{\mathrm{dep},j} \leq \mathrm{SoC}_{\mathrm{target}}, \\ 1, & \text{otherwise}, \end{cases} \tag{17}$$

where $\mathrm{SoC}_{\mathrm{dep},j}$ is the SoC of the jth EV at departure, and $\mathrm{SoC}_{\mathrm{target}}$ is set to 80. The overall SoC reward is the average of individual rewards:

$$R_{\mathrm{SoC}} = \frac{1}{m} \sum_{j=1}^{m} R_{\mathrm{SoC},j}, \tag{18}$$

where m is the number of departed EVs.

This reward dimension promotes fairness and consistent charging by penalizing undercharging below the 80 % SoC-target while rewarding EVs that meet or exceed this level, ensuring optimal and collective charging performance.

4.5.2. PV power consumption reward

The reward R_{PV} aligns PV power with CS consumption by minimizing the normalized difference between the generated PV power and the amount of power consumed over a time period T:

$$\Delta P_{\mathrm{PV,\,CS}} = \frac{\sum_{t=1}^{T} |P_{\mathrm{CS}}(t) - P_{\mathrm{PV}}(t)|}{\sum_{t=1}^{T} \max(P_{\mathrm{CS}}(t), P_{\mathrm{PV}}(t))}, \tag{19}$$

where $P_{\mathrm{CS}}(t)$ is power drawn by all CSs and $P_{\mathrm{PV}}(t)$ is PV power at time t The PV reward is calculated as:

$$R_{\mathrm{PV}} = 1 - \Delta P_{\mathrm{PV,\,CS}}. \tag{20}$$

This reward dimension promotes efficient use of PV energy by balancing generation and consumption, reducing grid dependence, and aligning with sustainable energy objectives.

4.5.3. Power usage efficiency reward

The reward $R_{P_{\mathrm{CS}}}$ evaluates efficiency and stability by combining the normalized standard deviation of power consumption and the Peak-to-Average Power Ratio (PAPR).

Power consumption is normalized as:

$$P_{\mathrm{norm}}(t) = \frac{P_{\mathrm{CS}}(t) - P_{\mathrm{min,\,T}}}{P_{\mathrm{max,\,T}} - P_{\mathrm{min,\,T}}}, \tag{21}$$

where $P_{\mathrm{min,\,T}} = \min_{1 \leq t \leq T} P_{\mathrm{CS}}(t)$ is the minimum power consumption over time period T, respectivley $P_{\mathrm{max,\,T}} = \max_{1 \leq t \leq T} P_{\mathrm{CS}}(t)$ is the maximum power consumption over the same period.

The normalized standard deviation over a period T is defined as:

$$P_{\sigma,\mathrm{norm}} = \sqrt{\frac{1}{T-1} \sum_{t=1}^{T} \left(P_{\mathrm{norm}}(t) - \frac{1}{T} \sum_{t=1}^{T} P_{\mathrm{norm}}(t) \right)^2}, \tag{22}$$

PAPR measures the peak load relative to the average load and is calculated as:

$$P_{\mathrm{PAPR}} = \frac{P_{\mathrm{max,\,T}}}{\frac{1}{T} \sum_{t=1}^{T} P_{CS}(t)}. \tag{23}$$

The power usage reward is:

$$R_{\mathrm{smooth}} = 0.5 \cdot (1 - 2 \cdot P_{\sigma,\mathrm{norm}}) + 0.5 \cdot \frac{1}{P_{\mathrm{PAPR}}}, \tag{24}$$

with P_{PAPR} normalized, and $P_{\sigma,\mathrm{norm}}$ weighted more heavily to emphasize stability.

This reward dimension promotes balanced charging by encouraging stable power usage through standard deviation and managing peak loads in the local grid.

4.5.4. Reward vector

The composite reward vector R integrates the three individual rewards, each scaled by α, as defined:

$$R = \alpha \cdot [R_{\mathrm{SoC}}, R_{\mathrm{smooth}}, R_{\mathrm{PV}}]^\top, \tag{25}$$

where $R_{\mathrm{SoC}}, R_{\mathrm{smooth}}, R_{\mathrm{PV}} \in [0, 1]$. During training, the scaling factor α is set to 100 to ensure that gradient-based algorithms (e.g. MODDPG) receive sufficiently strong feedback signals. In preliminary experiments, we observed that lower α values sometimes led to weaker gradients and slower learning for gradient-based methods, while higher values did not bring any clear improvement. However, the neuroevolutionary methods (MOEvoRL) are generally less sensitive to the magnitude of reward signals and could likely train as effectively with the raw, non-scaled rewards. During the evaluation, of which we report the results in the following sections, α is reduced to 1 to keep the cumulative reward values within a practical range, ensuring the resulting performance metrics remain clear, interpretable, and easy to analyze.

4.5.5. Constraint violation penalties

The agent must adhere to strict constraints, with any violation resulting in immediate termination of the environment and a negative reward vector. This mechanism applies uniformly to all MORL approaches examined.

The unbalanced load must not exceed the maximum allowable limit of 20 A. The phase difference for each phase pair at time t is:

$$\Delta I_{\text{phase}}(t) = \sum_{i,j \in a,b,c, i \neq j} \max(0, \Delta I_{\text{i, j}}(t) - 20\,\text{A}), \tag{26}$$

where $\Delta I_{\text{i, j}}(t)$ is the phase current difference between phases i and j of CSs at time t.

The power drawn from the external grid must not exceed the grid connection node limit of 40 kW. The grid power violation at any time t is:

$$\Delta P_{\text{ext}}(t) = \max(0, P_{\text{ext}}(t) - 40\,\text{kW}), \tag{27}$$

where $P_{\text{ext}}(t)$ is the external grid power at time t.

Both $\Delta I_{\text{phase}}(t)$ and $\Delta P_{\text{ext}}(t)$ are clipped and normalized. The penalty reward, R_{penalty}, is calculated based on the normalized values:

$$R_{\text{penalty}} = -1 - 0.5 \cdot \Delta I_{\text{phase}}(t) - 0.5 \cdot \Delta P_{\text{ext}}(t), \tag{28}$$

resulting in $R_{\text{penalty}} \in (-1, -2]$. The total reward vector R in case of a constraint violation is:

$$R = [R_{\text{penalty}}, R_{\text{penalty}}, R_{\text{penalty}},]^{\top}. \tag{29}$$

The penalty rewards ensure strict adherence to safety constraints by applying a base penalty, fine-tuned based on normalized constraint violations, with equal weight given to all constraints to promote balanced, reliable, and efficient system performance.

5. Experimental design and evaluation

This section outlines the experimental framework used to evaluate six MORL models for multi-objective charging processes. Five of these models are MOEvoRL models, which leverage neuroevolution to optimize the weights and architectures of policy networks. The models evaluated are:

- *FF-NEAT*: Combines NEAT with NSGA-II to evolve the architecture and weights of a simple feedforward network.
- *RNN-NEAT*: Uses NEAT with NSGA-II to develop a Recurrent Neural Network (RNN), enhancing the model's ability to capture temporal dependencies.
- *FF-NSGA-II*: Applies NSGA-II to optimize the weights of a fixed feedforward network.
- *FF-SPEA2*: Employs SPEA2 to optimize the weights of a fixed feedforward network.
- *LSTM-NSGA-II*: Utilizes NSGA-II to optimize the weights of a fixed Long Short-Term Memory (LSTM) network, to effectively capture long-range dependencies.
- *DDPG* [19]: A classical single-objective, gradient-based DRL model that uses prioritized experience replay [27]. To address multi-objective RL within a single-objective framework, a linear scalarization approach is applied, calculating the overall reward as the equally weighted average of individual objectives. This model serves as a baseline to highlight the potential advantages of multi-objective approaches over single-objective methods. While it is theoretically possible to achieve each solution with this algorithm, the design of the reward function and objective weighting is highly non-trivial, requiring extensive a-priori knowledge and engineering. These challenges are avoided in our approach, which leverages Pareto Front (PF) optimization to handle multiple objectives effectively.
- *MODDPG*: A gradient-based DRL model modified from DDPG for multi-objective optimization that serves as a benchmark, utilizing prioritized experience replay to select experiences based on their time difference error for prioritization.

5.1. Model training and evaluation

The study defines three main scenarios based on the number of Charging Stations (CSs): CS05 uses 5 stations, CS10 uses 10, and CS15 uses 15. In each scenario, charging events are generated over a continuous period of 12 months, resulting in approximately 370 RL environments. The GMM generates these charging events, as presented in Section 4.1, using a fixed seed to ensure reproducibility. Each environment represents a time period with at least one active charging event. The raw PV data is combined with the charging events – accounting for daily and seasonal fluctuations – to provide the agents with realistic energy availability patterns throughout the year.

Within these scenarios, we define three sub-scenarios that differ in the occupancy rate, the PV data, and the random seed used to synthesize the charging events. The first sub-scenario, CSxx-Norm-42, reflects a normal occupancy distribution derived by the GMM from the real ACN data with seed 42 and uses raw PV measurements from southwest Germany. This sub-scenario is used exclusively for training and internal evaluation: The environments are split into 80 % training, 10 % validation, and 10 % test. For each number of CSs (5, 10 or 15), 30 agents per MORL approach are trained with random seeds 42–71.[3] This setup shows how effectively each algorithm can learn under conditions closely matching the original data. During training, we apply an early-stopping criterion based on the hypervolume indicator to avoid overfitting. Prior to training, the hyperparameters of the model are optimized using Bayesian optimization, maximizing the hypervolume on the validation set with the weights and biases framework [28]. The agent and event generation for hyperparameter tuning is initialized with seed 1. During this process, 80 % of the RL environments are used for tuning and 20 % for validation. Post-training, the performance of each model is evaluated using the 30 trained agents on a hold-out test set of environments not seen during training.

The second sub-scenario, CSxx-Norm-71, similarly uses the normal distribution of charging sessions but changes the random value to 71 and adds a 5 % normal variation to the measured PV data, creating previously unknown conditions that do not appear during training. Since no further training is performed here, it is only used to test how well the guidelines learned in the first sub-scenario can be generalized to new charging sessions and PV profiles.

The third sub-scenario, CSxx-High-71, retains the seed 71 and PV noise of the second sub-scenario but enforces a higher occupation level of the CSs while respecting the underlying charging session distributions. This creates a more challenging environment that imposes higher loads on the local grid and shows how robust each CM strategy is when station utilization deviates further from the training conditions. As in the second sub-scenario, no additional training takes place, so the already trained agents have to adapt to these challenging circumstances. Finally, we conduct a failure analysis across the CSxx-Norm-71 and CSxx-High-71 scenarios to identify precisely when and why each algorithm might violate system constraints (e.g. exceed grid limits or cause phase imbalances).

[3] We have chosen seeds 42 and 71 to create the RL environments. Seed 42 is a standard reference in computer science, inspired by *The Hitchhiker's Guide to the Galaxy*, while Seed 71 complements the previous set of 30 agents generated from seed 42 to 71.

5.2. Evaluation metrics

To compare the algorithms with each other, different metrics are considered.

The *number of generations to convergence* or *number of episodes to convergence* quantifies learning speed by counting the episodes or generations required for the policy to stabilize, indicating the efficiency of the learning process. For MOEvoRL algorithms, convergence is measured in generations, while for MODDPG, it is measured in episodes.

The *success rate* measures how frequently the agent operates within the defined constraints

The *hypervolume* [29] measures the volume of the objective space dominated by the PF approximation $P \in \mathbb{R}^d$, bounded by a reference point $r \in \mathbb{R}^d$, where $\forall p \in P : p \leq r$. For a k-objective minimization problem, the hypervolume is given by:

$$HV(P) = \text{Volume}\left(\bigcup_{p \in P} \left[f_1(p), r_1\right] \times \cdots \times \left[f_k(p), r_k\right]\right) \tag{30}$$

A larger hypervolume value indicates better coverage of the objective space.

The *spread* metric evaluates the distribution and uniformity of solutions along the approximated PF by assessing the Euclidean distances between consecutive solutions. For problems with more than two objectives (k), the spread metric [30] is defined as:

$$\text{spread}(P) = \frac{\sum_{j=1}^{k} d(e_j, P) + \sum_{i=1}^{|P|} |d_i - \bar{d}|}{\sum_{j=1}^{k} d(e_j, P) + |P|\bar{d}}, \tag{31}$$

where $e_j \in P$ are reference points for the jth objective, $d(e_j, P)$ is the minimum distance from e_j to P, d_i is the distance between adjacent solutions, and $\bar{d}$ is the average distance. A lower spread value indicates a more uniform and well-distributed set of solutions, effectively capturing distribution uniformity and extreme area coverage.

The *spacing* [31] measures uniformity in solution distribution on the PF approximation:

$$\text{spacing}(P) = \sqrt{\frac{1}{|P| - 1} \sum_{i=1}^{|P|} (d_i - \bar{d})^2} \tag{32}$$

where d_i is the distance between adjacent solutions, and $\bar{d}$ is the average distance. A lower spacing value indicates more evenly spaced solutions, emphasizing local distribution uniformity rather than entire PF coverage.

6. Results

The analysis indicates that the differences between the CSxx-Norm-42 and CSxx-Norm-71 scenarios are minimal, with only slight variations across the metrics. As a result, we will primarily discuss the results of the CSxx-Norm-71 scenarios, which have been validated across several test environments. The results for CSxx-Norm-42 are provided in Tables C.5–C.7 in Appendix C for reference. As calculating multi-objective metrics for a single-objective solution is not meaningful, DDPG is excluded initially. The first analysis primarily focuses on multi-objective approaches, with DDPG being addressed separately in Section 6.2.

Tables 2–4 present the performance of the algorithms in environments generated with seed 71 and additional PV noise. The tables also include the number of generations/episodes and the hypervolume calculated on the validation set, which serves as the stopping criterion for the early stopping mechanism.

Table 2
Performance metrics for **CS05-Norm-71** on environments generated with seed 71, including additional PV noise. The table compares the hypervolume, success rate, spread, and spacing metrics for each algorithm. The number of generations/episodes and the hypervolume calculated on the validation set are also presented, which serve as the criteria for the early stopping mechanism. The values represent the mean and standard deviation across 30 agents, rounded to three decimal places.

Metric	FF-NSGA-II	FF-SPEA2	LSTM-NSGA2	FF-NEAT	RNN-NEAT	MODDPG
No. Gen./ No. Ep.	407 ± 269	457 ± 285	420 ± 261	858 ± 146	888 ± 140	1506 ± 637
HV (validation set)	19.284 ± 0.065	19.299 ± 0.043	19.647 ± 0.065	19.785 ± 0.126	19.720 ± 0.144	17.168 ± 0.488
Hypervolume	18.918 ± 0.098	18.944 ± 0.075	18.953 ± 0.064	19.379 ± 0.107	19.311 ± 0.139	15.189 ± 1.133
Success rate	0.989 ± 0.005	0.990 ± 0.007	0.970 ± 0.011	0.968 ± 0.012	0.977 ± 0.007	0.901 ± 0.053
Spread	0.729 ± 0.151	0.731 ± 0.126	0.753 ± 0.129	0.668 ± 0.139	0.667 ± 0.134	0.153 ± 0.323
Spacing	0.032 ± 0.018	0.028 ± 0.012	0.018 ± 0.010	0.040 ± 0.017	0.045 ± 0.020	0.001 ± 0.002

Table 3
Evaluation of the performance on the **CS10-Norm-71** scenario with metrics as in Table 2.

Metric	FF-NSGA-II	FF-SPEA2	LSTM-NSGA2	FF-NEAT	RNN-NEAT	MODDPG
No. Gen./ No. Ep.	363 ± 229	462 ± 249	425 ± 274	292 ± 169	666 ± 326	2329 ± 835
HV (validation set)	18.500 ± 0.052	18.234 ± 0.058	18.603 ± 0.056	17.416 ± 0.273	18.046 ± 0.323	15.957 ± 0.566
Hypervolume	18.010 ± 0.063	18.025 ± 0.078	18.057 ± 0.042	17.375 ± 0.305	17.624 ± 0.329	14.107 ± 0.904
Success rate	0.981 ± 0.009	0.981 ± 0.010	0.961 ± 0.018	0.928 ± 0.045	0.983 ± 0.019	0.906 ± 0.038
Spread	0.760 ± 0.162	0.766 ± 0.234	0.733 ± 0.182	0.503 ± 0.280	0.622 ± 0.202	0.170 ± 0.326
Spacing	0.025 ± 0.021	0.022 ± 0.013	0.015 ± 0.007	0.044 ± 0.038	0.039 ± 0.030	0.001 ± 0.002

Table 4
Performance on the **CS15-Norm-71** scenario with metrics as detailed in Table 2.

Metric	FF-NSGA-II	FF-SPEA2	LSTM-NSGA2	FF-NEAT	RNN-NEAT	MODDPG
No.Gen./No.Ep.	405 ± 290	410 ± 239	403 ± 293	176 ± 141	650 ± 340	2915 ± 1042
HV (validation set)	17.681 ± 0.044	17.679 ± 0.040	17.794 ± 0.035	16.968 ± 0.166	17.230 ± 0.139	14.622 ± 1.695
Hypervolume	17.437 ± 0.071	17.445 ± 0.070	17.512 ± 0.025	16.751 ± 0.166	17.027 ± 0.182	12.200 ± 2.386
Success rate	0.958 ± 0.014	0.951 ± 0.013	0.945 ± 0.026	0.955 ± 0.041	0.990 ± 0.007	0.839 ± 0.115
Spread	0.570 ± 0.344	0.657 ± 0.372	0.737 ± 0.204	0.779 ± 0.322	0.645 ± 0.234	0.107 ± 0.235
Spacing	0.027 ± 0.048	0.031 ± 0.044	0.009 ± 0.006	0.098 ± 0.042	0.041 ± 0.035	0.000 ± 0.000

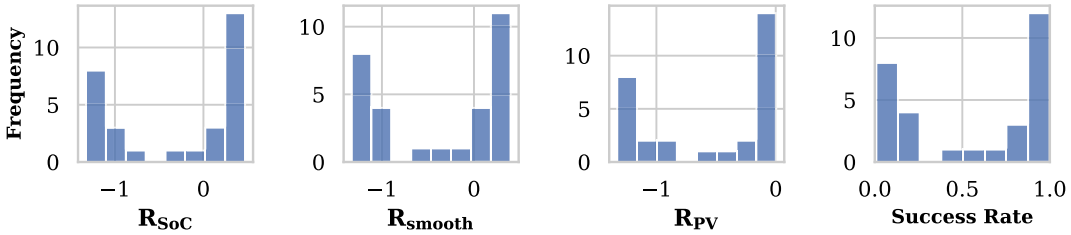


Fig. 2. CS10-Norm-71: average reward values for R_{SoC}, R_{smooth}, R_{PV}, and success rate across the charging events of the 30 DDPG agents. The rewards R_{SoC}, R_{smooth}, and R_{PV} are designed to maximize EV battery SoC, smooth peak loads, and increase PV power consumption. A reward becomes negative if constraints such as unbalanced load limits (20 A) or grid power limits (40 kW) are violated, reflecting penalties that encourage compliance with safety and operational standards.

As an initial observation, in the absence of a CM, where all CSs operate at full power, only 20 % of charging events succeed without violating the unbalanced load constraint when 5 CSs are connected. This success rate drops to 1 % with 10 CSs and declines further as the number of CSs increases to 15.

6.1. Description

As scenario complexity increases from CS05 to CS15, all algorithms exhibit a consistent decline in hypervolume, reflecting the growing challenge of maintaining optimal performance. Distributing the same amount of energy across more EVs makes it harder to fully charge each one. Additionally, the increase in single-phase or two-phase EV charging exacerbates phase imbalances, resulting in slower overall charge rates.

LSTM-NSGA-II and MODDPG show a significant drop in performance from the validation set to the test set, indicating a tendency toward overfitting and difficulties in adapting to unseen situations. In contrast, the other MOEvoRL algorithms are generally less prone to overfitting.

FF-NSGA-II and FF-SPEA2 demonstrate comparable performance across all metrics. They maintain stable convergence times across scenarios, showing robustness. Their hypervolume and success rates remain strong, although neither the highest nor the lowest, with only minor fluctuations and a slight decrease in the most complex scenario (CS15), demonstrating their adaptability. However, the increasing variability in spread and spacing with rising complexity indicates challenges in maintaining a well-distributed and uniform solution distribution.

LSTM-NSGA-II shows consistent convergence times and performs well in hypervolume, achieving the highest or near-highest values as complexity increases. However, its success rate declines in more complex scenarios, suggesting difficulty in handling more intricate challenges. Despite this, LSTM-NSGA-II excels in spread, particularly in CS15, with low spacing values indicating both broad and uniform solution distribution.

FF-NEAT converges quickly, especially in complex scenarios such as CS10 and CS15, which is probably due to parameter tuning, while RNN-NEAT converges the slowest among all MOEvoRLs. Both algorithms perform well in the simpler CS05 scenario achieving the highest hypervolumes, but their performance declines with increasing complexity. FF-NEAT's success rate drops over the scenarios, while RNN-NEAT improves, becoming the top performer in CS10 and CS15, demonstrating strong adaptability. RNN-NEAT is the only algorithm that maintains a stable success rate as complexity increases. Both algorithms consistently maintain well-distributed solutions, as seen in their spread and spacing values. However, MOEvoRL algorithms with fixed structures tend to produce more stable results, reflected in lower standard deviations compared to NEAT-based approaches.

MODDPG struggles with increasing scenario complexity, showing longer and more variable convergence times. Compared to the MOEvoRL algorithms, it consistently performs worse in terms of hypervolume and success rate, particularly in complex scenarios with high variability. While it achieves seemingly favourable spread values, its high standard deviations indicate inconsistency. Its near-zero spacing values suggest a lack of diversity, making MODDPG the least reliable algorithm, especially in challenging environments.

6.2. Single-objective DDPG

To evaluate and show that a multi-objective approach is necessary and that equal or better performance could not be achieved by a classical single-objective DRL method, we trained a DDPG model that simplified the multi-objective problem into a single-objective setting using an equally weighted scalarized reward function. Extensive hyperparameter tuning was performed for the DDPG model, consistent with the MORL algorithms, by training 30 agents with ascending random seeds. However, no stable parameter configurations could be identified. The results for the CS10-Norm-71 scenario are presented as a representative example, as similar patterns were observed across other scenarios. These results are publicly available online.[4]

As shown by the negative rewards and low success rate in Fig. 2, several DDPG agents fail to meet constraints, particularly regarding unbalanced load management. Among agents that partially comply, many converge to local maxima, resulting in suboptimal performance compared to the MORL algorithms (see Fig. 3). Furthermore, the bimodal distribution of DDPG results makes metrics like the mean or standard deviation uninformative and unrepresentative of actual performance.

6.3. Pareto front analysis

To analyze the trade-offs between the individual objectives and evaluate the performance of the algorithms, the PF of the MORL algorithms for the CS10-Norm-71 scenario is examined, as shown in Fig. 3.[5]

The 3D plot highlights the clear performance differences among the algorithms. DDPG, operating under a single-objective framework, becomes trapped in a local optimum and fails to locate diverse, high-quality solutions. Although one might expect its single-objective solution to lie on the true PF, it lags far behind the multi-objective approaches that better approximate the unknown PF. In contrast, MOEvoRL algorithms – such as FF-SPEA2, LSTM-NSGA-II, RNN-NEAT, and FF-NSGA-II

[4] See https://github.com/NeeleKemper/moevorl-ev-cm.

[5] The analysis of additional seeds shows that the resulting PFs are largely consistent, which proves the representativeness of the displayed front. Results are presented for RL agents initialized with seed 67, corresponding to the best-performing DDPG agent in this scenario. All MORL algorithms shown are also initialized with seed 67 to ensure consistency and comparability with the DDPG results. Additional results and analyses are available online. The goal for all objectives is maximization. In multi-objective optimization, each algorithm discovers a unique set of non-dominated solutions that may differ in both size and distribution, explaining any variation in the number of points that appear on the PF.

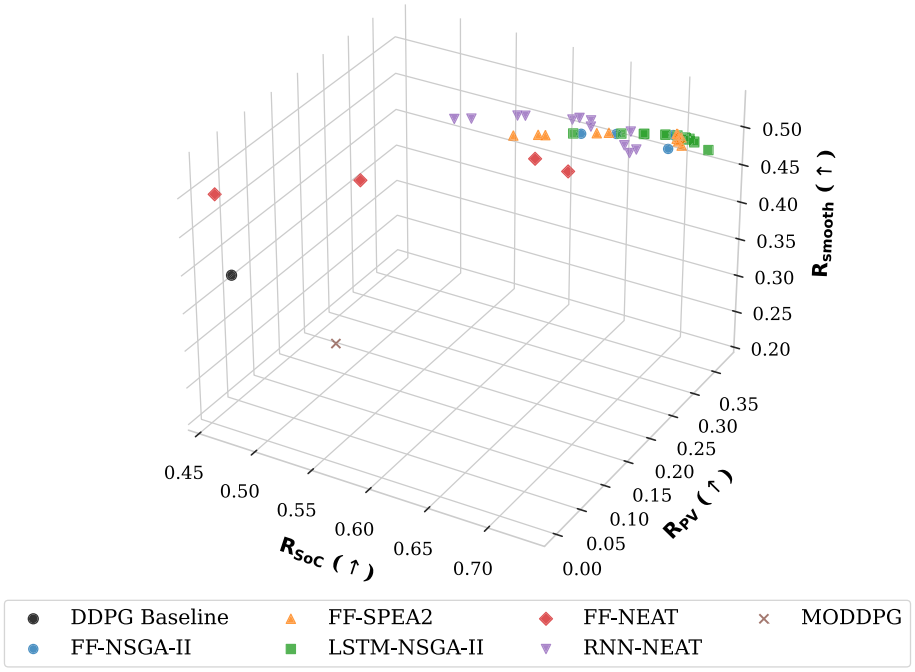


Fig. 3. CS10-Norm-71: comparison of PF for the various MORL approaches (each producing multiple Pareto-optimal policies) and the single DDPG solution, all trained with seed 67. Each point represents a strategy in the three-dimensional target space defined by R_{SoC}, R_{smooth} and R_{PV}, all of which must be maximized. We have also provided an interactive 3D Pareto plot online for further inspection (https://neelekemper.github.io/moevorl-ev-cm/).

– show stronger and more balanced results overall, clustering in regions where all objectives are maximized simultaneously. In particular, LSTM-NSGA-II and RNN-NEAT consistently produce high-quality and diverse strategies and demonstrate superior PF coverage across multiple targets. Meanwhile, FF-NSGA-II and FF-SPEA2 also show solid performance, even if their coverage is somewhat lower and concentrated in certain regions. FF-NEAT, while showing broad exploration and occasionally approaching the top performers, generally remains below the higher-ranked clusters of LSTM-NSGA-II and RNN-NEAT, suggesting that although it finds diverse solutions, it struggles to consistently achieve the same level of performance as the stronger MOEvoRL methods. MODDPG suffers from low diversity and identifies only a single Pareto-optimal policy in its solution set, as reflected by its low spread and spacing measures.

6.4. Statistical analysis

Bayesian data analysis is employed to assess the significance of the observed differences between the MOEvoRL algorithms, offering a more robust alternative to traditional null-hypothesis significance testing (NHST) [32]. Unlike NHST, Bayesian methods offer clear interpretability by providing probabilistic insights, such as the probability of one algorithm outperforming another and credible intervals that directly quantify parameter uncertainty. They eliminate the reliance on arbitrary significance thresholds such as $P < 0.05$ and are particularly effective for small sample sizes as they use prior information to draw robust and meaningful conclusions. Due to its consistently poorer performance, MODDPG and DDPG are excluded from further analysis as its ranking is very clear already.

Using the method by Calvo et al. [33,34], the posterior distribution of the probability that each MOEvoRL algorithm will perform best is determined. The model is applied to both hypervolume and success rate observations for the CSxx-Norm-71 scenario. The results for spread and spacing, as detailed in Figs. C.7 and C.8 in Appendix C, reveal significant uncertainty regarding which model performs best, though there is a bias favoring NEAT-based MOEvoRL for spread in CS05 and CS10, FF-NSGA-II in CS15 and LSTM-NSGA-II for spacing. However, these results will not be discussed further here.

For hypervolume (e.g. Fig. 4a–c), FF-NEAT and RNN-NEAT outperform other algorithms in the simplest scenario CS05, while LSTM-NSGA-II gains an edge as scenario complexity increases. In contrast, FF-NSGA-II and FF-SPEA2 consistently show lower probabilities of being the top performers. Regarding success rate (e.g. Fig. 5a–c), FF-NSGA-II and FF-SPEA2 lead in simpler scenarios, with RNN-NEAT becoming more competitive in complex ones, emerging as the most likely best model in CS15. LSTM-NSGA-II and FF-NEAT generally have low probabilities of being the best in this metric.

While these models show varying strengths, none consistently surpasses the 80 % threshold required to be considered clearly preferable [32]. Further analysis is required. If the data pairs follow a normal distribution, Kruschke's model [35] will be applied. For non-normally distributed data, the model proposed by Benavoli et al. [36] will be used. A Region of Practical Equivalence (ROPE) of 0.01 is used for the success rate due to its binary nature, where even small differences are significant. For hypervolume, a broader ROPE of 0.01 to 0.05 is applied to account for the inherent variability in multi-objective optimization, allowing for a more nuanced assessment.

Further analysis confirms the ranking of Calvo's model and supports the observation that FF-SPEA2 and FF-NSGA-II are practically equivalent in terms of hypervolume and success rate. The top performers identified by Calvo's model stand out with a high degree of certainty, while for probably equally ranked models, there is either practical equivalence or considerable uncertainty regarding which algorithm performs better.

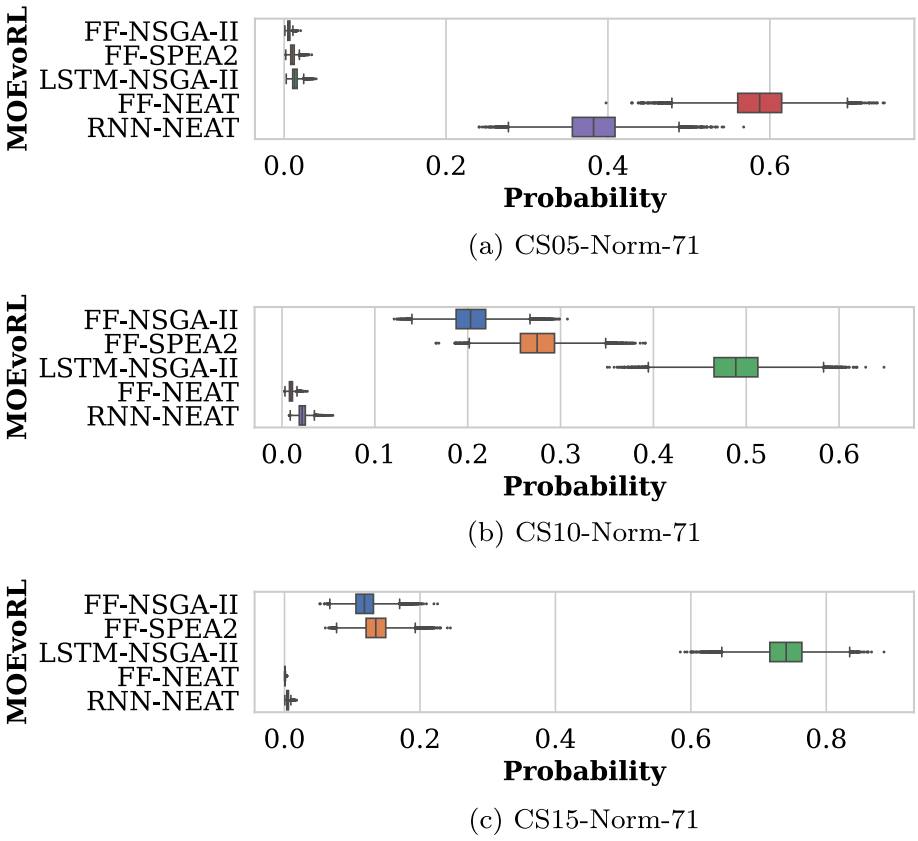


Fig. 4. Box plots of the posterior distributions obtained from the ranking model by Calvo et al. [33,34] applied to the **hypervolume** data of the scenarios **CS05-Norm-71**, **CS10-Norm-71**, and **CS15-Norm-71**. The Markov Chain Monte Carlo process of the model sampled 50,000 times to estimate the distribution, converged and is well-formed. The plots display standard 1.5 IQR whiskers and outliers of these. The probability value p for an MOEvoRL method indicates the likelihood of it performing the best with respect to hypervolume.

6.5. High utilization level

The high-utilization scenarios are designed to evaluate how well the MORL approaches handle situations where the distribution of charging events deviates from the learned patterns and the CSs have a higher charging volume, making the multi-objective problem more challenging to solve. The results for CSxx-High-71 are detailed in Tables C.8–C.10 in Appendix C, with key observations for each algorithm summarized below.

All MORLs experience a decrease in both success rate and hypervolume, with MODDPG showing the most pronounced decline – dropping to a success rate of 0.687 and a hypervolume of 9.006 in CS15-High. In contrast, the worst-performing MOEvoRL in CS15-High, LSTM-NSGA-II, still achieves a success rate of 0.924, and FF-NEAT records a hypervolume of 16.599. MODDPG also exhibits a significant increase in standard deviation, indicating instability and unreliability, with spacing values suggesting that its solutions are overly clustered.

The drop in performance for the MOEvoRLs is minimal, and their rankings and performance patterns remain largely consistent with those observed in the CSxx-Norm-71 scenario. Additionally, the MOEvoRLs exhibit stability, showing no significant increase in standard deviations across the metrics. FF-NSGA-II and FF-SPEA2 show only minor deviations from their performance in the CSxx-Norm scenarios. FF-NEAT shows the most significant decrease in success rate in the CSxx-High scenarios, while RNN-NEAT remains the most stable, achieving a success rate of 0.983 in CS15-High. In scenario CS05-High, FF-NEAT and RNN-NEAT experience the largest drop in the hypervolume, indicating some difficulty adapting to more challenging scenarios, although they still achieve the highest hypervolume. However, in other scenarios, their performance decline is comparable to that of other MOEvoRLs. LSTM-NSGA-II leads in hypervolume across all other CSxx-High scenarios. Regarding solution space coverage, MOEvoRLs show no apparent degradation, although the spacing values suggest that MOEvoRLs with fixed topologies tend to produce overly clustered solutions in scenario CS15-High.

6.6. Failure analysis

The failure analysis of the CSxx-Norm/High-71 scenarios reveals distinct patterns in the MOEvoRL algorithms and emphasizes specific challenges they encounter. In comparison, MODDPG shows less pronounced failure patterns, indicating that it struggle to handle a wider range of situations effectively. A comprehensive failure analysis is presented in Table C.11 in Appendix C.

For all MORL algorithms, phase imbalance, especially between phases a and c, proves to be the most significant source of failures. It is noteworthy that FF-NEAT and MODDPG are the only algorithms that occasionally fail due to exceeding node connectivity limits, although these cases are rare. LSTM-NSGA-II exhibits the lowest phase limit violations. However, FF-NEAT and MODDPG have significant difficulties

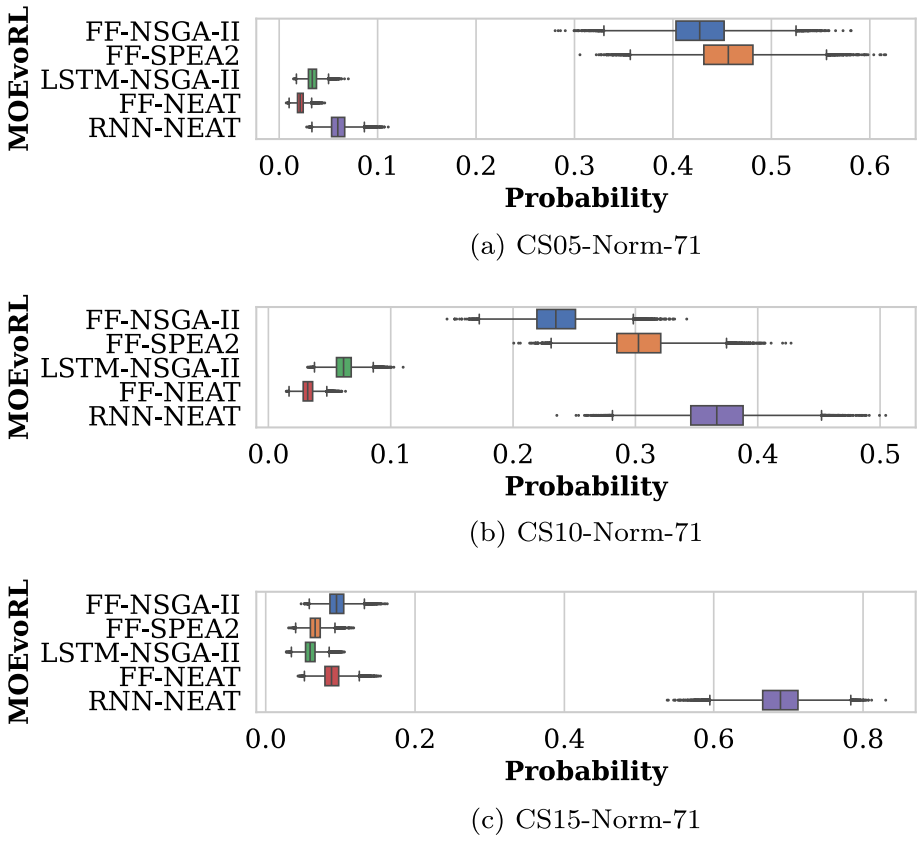


Fig. 5. Box plots illustrating the posterior distributions from the ranking model as described in Fig. 4 but for the **success rate** data of the **CS05-Norm-71**, **CS10-Norm-71**, and **CS15-Norm-71** scenarios.

and exhibit the highest violations, suggesting that their performance is more context-dependent and less reliable.

Most failures in MOEvoRL occur immediately after the arrival of an EV, suggesting that these algorithms are more responsive to immediate changes, with far fewer errors occurring after departure or in later timesteps of the charging process. LSTM-NSGA-II stands out with a higher portion of its errors occurring much later or without a clear triggering event such as arrival or departure, pointing to potential challenges in maintaining long-term system stability or handling less predictable scenarios. MODDPG encounters errors both after the arrival and departure of EVs.

7. Discussion

The strengths and weaknesses of the MORL approaches are discussed in the following section.

The multi-objective approaches, particularly MOEvoRL, consistently outperform the single-objective DDPG by achieving superior trade-offs across all objectives. The DDPG agent, trained with an equally weighted scalarized reward, produces a suboptimal solution that fails to align with the PF approximation proposed by MORL methods. This limitation arises from scalarization, which oversimplifies the multi-objective problem and biases optimization toward a fixed direction, restricting the exploration of balanced trade-offs. Furthermore, DDPG's reliance on gradient-based updates exacerbates this issue, emphasizing local search and exploitation rather than the global exploration required for diverse, Pareto-optimal solutions.

MODDPG consistently performs the worst among all MORL methods in all scenarios, especially with increasing complexity. Its low hypervolumes, success rates, and near-zero spacing values indicate difficulty in managing trade-offs and maintaining diversity in multi-objective problems, likely due to its reliance on local search and incremental adjustments that lead to clustered, suboptimal solutions. High standard deviations across metrics further highlight its inconsistency and unreliability. MODDPG's long convergence time is due to its incremental approach, which often causes it to get stuck in suboptimal regions. Being a gradient-based algorithm, it is also prone to overfitting, especially when the validation patterns do not represent the broader problem space, leading to local optima that do not generalize well, and its narrow focus on local exploitation further limits exploration, increasing the risk of overfitting, especially in sparse-reward environments.

In contrast to MODDPG, MOEAs are characterized by the use of global search strategies and a population-based approach in multi-objective optimization. This allows them to explore a wider range of solutions through mutation and crossover, avoid local optima, maintain diversity, and effectively approximate the PF. MOEvoRL algorithms benefit from these evolutionary strategies and enable faster and more robust convergence, even in complex scenarios. The population-based nature of MOEAs reduces the risk of getting stuck in local optima and improves generalization across different scenarios, making MOEvoRL algorithms more robust than the narrowly focused MODDPG.

FF-NSGA-II and FF-SPEA2 show similar performance across all scenarios, effectively balancing multiple objectives. This suggests that it makes no significant difference whether NSGA-II or SPEA2 is used to optimize the network weights in the given use case. Their fixed

feedforward architecture, coupled with the strong optimization capabilities of MOEAs, ensures stable and reliable performance, especially in less complex scenarios. However, in the most complex scenario (CS15), both algorithms show a slight performance degradation with issues in maintaining a uniform solution distribution and some solution clustering, indicating that their fixed architecture may lack the flexibility required for more sophisticated environments – a limitation also observed for the fixed LSTM topology.

The LSTM architecture effectively captures temporal patterns, resulting in good PF coverage and a well-distributed diverse set of solutions, demonstrating robustness and adaptability in most scenarios. However, this strong performance comes with a higher risk of overfitting and a lower success rate compared to FF-NSGA-II, indicating that the LSTM's complex recurrent architecture, while powerful, is harder to optimize than a simpler feedforward network. The decline in success rates indicates that further tuning and regularization of the network is required to achieve high performance even under the most difficult conditions.

NEAT-based MOEvoRL excels in multi-objective optimization for simpler tasks, consistently finding optimal and uniformly distributed solutions. However, as scenario complexity increases, NEAT's performance declines and becomes more variable, reflecting the challenge of optimizing both structure and weights in more complex environments, as indicated by the drop in hypervolume and increased standard deviation compared to other MOEvoRLs. Nevertheless, NEAT-based MOEvoRLs avoid solution clustering by dynamically evolving both network structure and weights, enabling exploration of diverse architectures. While FF-NEAT struggles with a decreasing success rate as complexity rises, RNN-NEAT maintains the highest and most stable success rate among the MOEvoRLs, thanks to its recurrent architecture, which effectively learns sequential data and long-term dependencies. In contrast, FF-NEAT is more prone to failures in complex scenarios due to its lack of memory. RNN-NEAT's longer convergence time reflects the added complexity of evolving both topology and weights in recurrent networks, which manage temporal dependencies. A critical trade-off is observed between RNN-NEAT and LSTM-NSGA-II: LSTM-NSGA-II excels at generating a diverse solution set, as evidenced by its high hypervolume, yet it struggles with consistency in achieving top-quality outcomes, reflected in its lower success rate. In contrast, RNN-NEAT consistently delivers high-quality results, indicated by its high success rate, but tends to explore a more limited portion of the solution space, resulting in a lower hypervolume.

In addition to their robustness to local optima and their ability to explore different solutions, MOEvoRL algorithms are particularly well-suited for real-world applications in EV-CM. They optimize charging strategies while effectively adhering to grid constraints, such as maintaining load balance and minimizing strain on grid infrastructure. This not only ensures efficient energy management but also grid resilience, which are critical factors for the successful integration of EVs into power grids.

Real-world application

MOEA-based approaches inherently handle multiple objectives without the extensive architectural modifications often required by multi-objective DRL (MODRL). By evolving entire populations of candidate solutions, MOEvoRL can adapt to dynamic environments more easily and often requires fewer data samples for training. Its population-based nature also enables straightforward parallelization, making it well-suited for distributed systems, unlike MODRL, which relies on sequential updates based on gradient descent. In addition, MODRL typically relies on extensive GPU-based backpropagation and more extensive data collection. Because MOEvoRL avoids these requirements and runs efficiently on CPUs, its overall computational cost and hardware demands are significantly lower.

Our experiments focus on 5–15 charging stations, primarily to illustrate the basic feasibility of the proposed MOEvoRL approach. We acknowledge that real-world applications often involve hundreds of CSs. Nevertheless, many workplaces and public lots still operate with 5–15 CSs, making small-scale scenarios crucial for practical deployment and evaluation. As the number of CSs increases, both the state and action spaces scale linearly, which remains feasible for MOEvoRL because population-based EAs can evaluate multiple candidate solutions in parallel and effectively manage increasing dimensionality. Consequently, we assume that MOEvoRL will continue to deliver functional solutions with moderate computational effort even in scenarios with up to 100 CSs. In cases where MOEvoRL becomes trapped in a local optimum within a larger state space, additional recovery mechanisms can be introduced to guide the algorithm away from suboptimal solutions. Although our current system is centralized, future implementations could adopt a hierarchical or multi-agent framework where multiple local controllers manage subsets of stations. This reduces communication bottlenecks by limiting detailed data exchange to local clusters while also decomposing the overarching problem into more manageable subproblems. Such decentralized architectures are compatible with the neuroevolutionary method, though communication overhead has not yet posed a significant challenge for the relatively small scenarios explored in this paper.

The failure analysis reveals that MOEvoRL exhibits consistent failure patterns, whereas MOMDDPG has variable error sources, making it less reliable and harder to prevent. Despite MOEvoRL algorithms achieving a high success rate of over 90 %, practical implementation requires integrating safety constraints to ensure operational reliability and safety. This includes setting conservative limits and establishing emergency protocols, which are essential for managing high-risk scenarios or communication delays. As is common in many CM systems, mission-critical safety conditions are hard-coded at a low level. If a hardware safety limit or constraint is reached, the CSs immediately lower the power, regardless of the RL policy.

For implementation in a real-time environment, the RL agent functions as a meta-controller and is pre-trained offline before deployment. Offline training allows the agent to extensively explore the solution space using historical or simulated data and optimize its strategy without the real-time constraints of iterative network updates, ensuring robust performance. Additionally, the computational power required for this process is much greater than what would be practical in a deployed real-world system and is therefore performed in a cloud environment. During deployment, the RL agent operates at a higher-level control layer, issuing target setpoints (e.g. charging currents) on a minute-by-minute schedule. Although dynamic load variations can arise on sub-minute timescales, local controllers or hardware safeguards promptly intervene when hardware limits (e.g. fuse ratings) are approached. This layered architecture aligns with standard practice in power systems, where minute-scale decisions suffice for overarching objectives (such as smoothing peak loads or maximizing PV usage), while lower-level protections handle rapid fluctuations. Because the control loop updates at discrete intervals, communication overhead remains tractable – data need only be exchanged once per minute (or at a similarly chosen frequency). Local safety mechanisms temporarily curtail charging until the next RL update if faster changes occur. An observer component continuously monitors system states and environmental conditions, such as CS usage, energy supply from the network, and vehicle demand patterns. These observations are fed into the RL agent, which uses its pre-trained policy to make high-level decisions on load distribution and scheduling. The real-time inference process is computationally lightweight, requiring only a few milliseconds per decision step, allowing the system to meet strict latency requirements for high-level control. During operation, an additional sandbox environment can be run in parallel to the live system. By feeding real or simulated data into this sandbox, one can

continue refining the RL agent's strategy without disrupting ongoing operations. As soon as a new policy is shown to outperform the current one under a variety of simulated or shadowed conditions, it can be seamlessly deployed. This process ensures continual adaptation to evolving conditions, such as changing load profiles or the introduction of new vehicle types. Overall, the combination of offline pre-training, periodic high-level control, and local fallback protections yields a flexible, scalable solution that retains safety, adapts to real-world uncertainties, and optimizes multiple performance objectives over time.

8. Conclusion

We introduced neuroevolution-based MOEvoRL algorithms for multi-objective, centralized control of EV-CM, benchmarking their performance against the gradient-based MODDPG and classical single-objective DDPG. These MORL algorithms focus on optimizing three critical objectives: maximizing batteries' SoC, maximizing PV power usage, and minimizing peak loads while simultaneously smoothing the overall load on the local distribution grid of the CS network. A key consideration of these algorithms is the integration of essential local grid constraints, ensuring balanced loads and compliance with grid connection limits. This approach enables efficient EV integration into the local distribution grid by managing CS loads, optimizing energy usage, and ensuring practical real-world applicability.

The single-objective DDPG, which applies scalarization to combine multiple objectives into a single reward via a linear combination, exhibits poor and unstable performance in this setting. It struggles to balance competing objectives and converges to a single, suboptimal solution. MOEvoRL algorithms outperform MODDPG, especially as complexity increases, due to MODDPG's reliance on local search and incremental adjustments, leading to clustered, suboptimal solutions. MODDPG's low performance metrics, high variability and tendency to get stuck in local minimal optima, combined with a high risk of overfitting, make it less reliable for multi-objective optimization. In contrast, MOEAs excel by leveraging global search strategies and a population-based approach that promotes diversity and effective exploration of the solution space. These strategies enable MOEvoRL algorithms to achieve robust convergence and better generalization across complex scenarios, making them significantly more effective and stable than MODDPG. Recurrent policy networks that capture temporal dependencies are particularly noteworthy, with LSTM-NSGA-II excelling in generating a diverse set of solutions and RNN-NEAT most effectively adhering to critical grid constraints. Additionally, MOEvoRL offers several practical advantages over MODRL, including simpler implementation, inherent multi-objective design, adaptability to dynamic environments, reduced reliance on large datasets, ease of parallelization and distributed computing, lower memory overhead, and minimal dependence on computationally intensive backpropagation. All in all, we demonstrated, that lightweight neuroevolutionary-based MOEvoRL models are a viable alternative to DRL for the specific use case of multi-objective EV-CM.

Future research should build on the success of the investigated MOEvoRL algorithms. Exploring advanced MOEAs and deeper investigations into neuroevolutionary algorithms can enhance performance and reveal new capabilities in adaptive networks. The development of hybrid algorithms that combine the global search of evolutionary methods with the precision of gradient-based approaches could optimize the balance between exploration and exploitation. The centralized CM approach may reach the limits of scalability and robustness when the number of CS to be controlled increases. Future work should focus on evaluating the MOEvoRL framework in large-scale scenarios involving hundreds of charging stations to assess its performance and identify potential bottlenecks. Decentralized CM can address this by distributing tasks across the network, improving adaptability and robustness, and preventing single points of failure. The integration of hierarchical and multi-agent policies can further improve decentralized systems.

CRediT authorship contribution statement

Neele Kemper: Writing – original draft, visualization, validation, software, methodology, formal analysis, and conceptualization. **Michael Heider:** Writing – review & editing, methodology, and conceptualization. **Dirk Pietruschka:** Supervision, project administration, funding acquisition, and conceptualization. **Jörg Hähner:** Supervision, project administration, and funding acquisition.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Neele Kemper reports financial support was provided by enisyst GmbH. Dirk Pietruschka reports a relationship with enisyst GmbH that includes: employment. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We gratefully acknowledge the support of Enisyst[6] for their partial financial contribution and valuable expertise as our industrial partner. We also gratefully acknowledge the resources provided on the LiCCA HPC cluster of the University of Augsburg, co-funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-ID 499211671.

Appendix A. Deep deterministic policy gradient algorithm

See Algorithm 1.

Algorithm 1 MODDPG algorithm.

Randomly initialize critic network $Q(s, a, \mathbf{w}|\theta^Q)$ and actor $\mu(s, \mathbf{w}|\theta^\mu)$ with weights θ^Q and θ^μ. ▷ Initialization of networks

Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$. ▷ Target networks are clones of the main networks

Initialize replay buffer $\mathcal{R}$. ▷ Storage for experience replay

for episode = 1 to M **do** ▷ Run for M episodes

  Initialize a random process $\mathcal{N}$ for action exploration. ▷ OU process for exploration

  Receive initial observation State s_1.

  for t = 1 to T **do** ▷ For each step of the episode

    Sample weight vector $\mathbf{w}_t \sim$ Dirichlet Distribution(α). ▷ Dirichlet distribution generates uniform distributions over the simplex

    Select action $a_t = \mu(s_t, \mathbf{w}_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise.

    Execute action a_t and observe reward vector $\mathbf{r}_t$ and new State s_{t+1}.

    Store transition $(s_t, a_t, \mathbf{w}_t, \mathbf{r}_t, s_{t+1})$ in $\mathcal{R}$. ▷ Store the experience

    Sample a random minibatch of N transitions $(s_i, a_i, \mathbf{w}_i, \mathbf{r}_i, s_{i+1})$ from $\mathcal{R}$.

    Set $y_i = \mathbf{r}_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}, \mathbf{w}_i|\theta^{\mu'}), \mathbf{w}_i|\theta^{Q'})$. ▷ Compute target for current transitions

    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i, \mathbf{w}_i|\theta^Q))^2$.

    Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q(s, a, \mathbf{w}|\theta^Q)|_{s=s_i, \mathbf{w}=\mathbf{w}_i, a=\mu(s_i, \mathbf{w}_i)} \cdot \nabla_{\theta^\mu} \mu(s, \mathbf{w}|\theta^\mu)|_{s=s_i, \mathbf{w}_i}$$

    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$

$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

    ▷ Softly update the target networks

  end for

end for

[6] See https://www.enisyst.de/en/home/.

Appendix B. Charging management schema

See Fig. B.6.

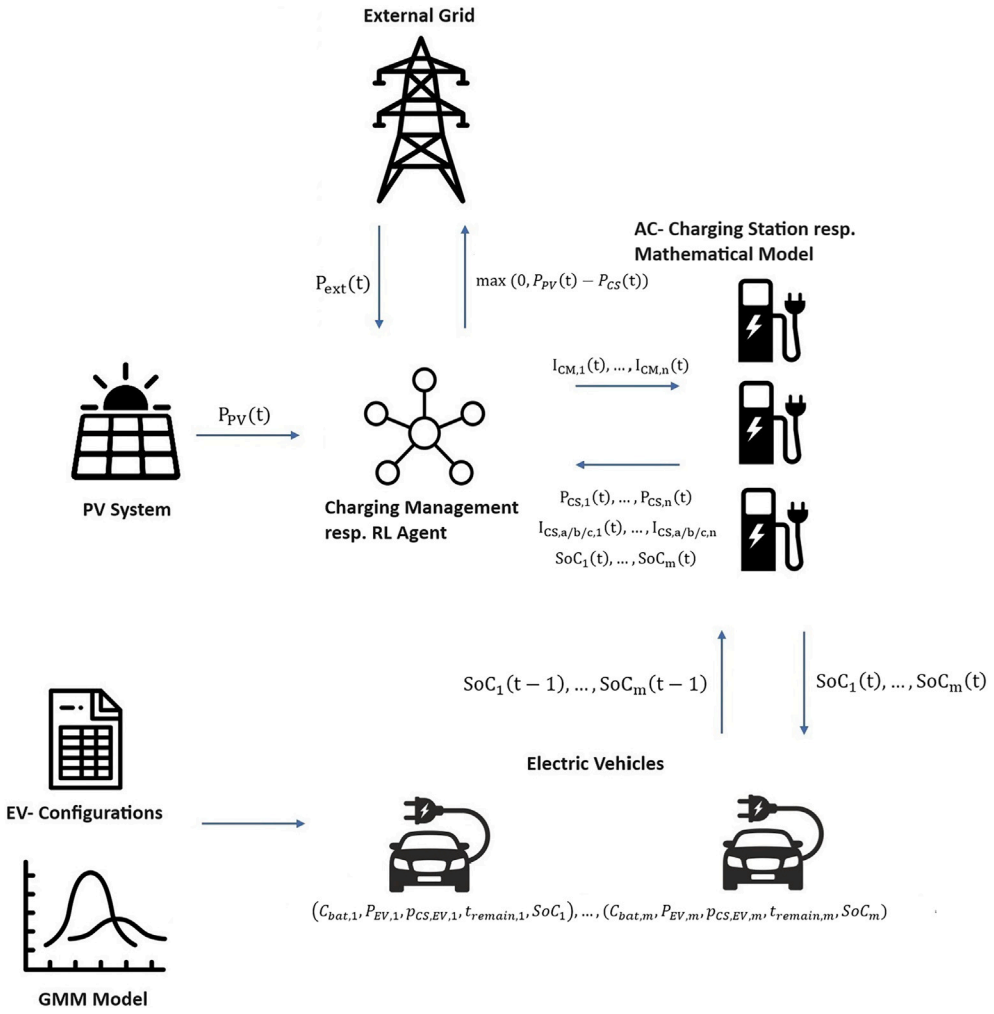


Fig. B.6. The schema illustrates the simulation framework for managing n CSs and m EVs at a specific time step t. At the start of the simulation, EV objects are generated using predefined configurations (cf. Table 1) and a GMM, which determines variable arrival times, charging durations, and energy amount. From this, EV-specific arrival times, departure times, remaining charging times (t_{remain}), and initial SoC are derived. Each EV configuration includes battery capacity (C_{bat} in kWh), maximum charging power (P_{EV} in kW), and the number of connected phases ($p_{\text{CS,EV}}$). The central component of the schema is the CM system, represented by an RL agent. At each time step t, the agent receives observations, including PV power output ($P_{\text{PV}}(t)$), external power requirements ($P_{\text{ext}}(t)$), power consumption of each CS ($P_{\text{CS,1}}(t), \ldots, P_{\text{CS,n}}(t)$), phase currents ($I_{\text{CS,a/b/c,1}}(t), \ldots, I_{\text{CS,a/b/c,n}}(t)$), and EV-specific data such as battery capacities, SoC, charging power, remaining charging times, and the number of connected phases. A detailed description of the observation space is provided in Section 4.3. Based on these observations, the RL agent calculates the charging currents for each station ($I_{\text{CM,1}}(t), \ldots, I_{\text{CM,n}}(t)$), which are used to update the EVs' SoC, grid power requirements, station power consumption, and phase currents. These updated values serve as inputs for the next observation cycle. A description of the RL agent's actions is provided in Section 4.4. Additionally, the schema accounts for surplus PV power not consumed by the CSs at time t. This surplus, calculated as the difference between PV power output and total CS consumption, is fed back into the external grid. Further details on SoC estimation and charging power calculations can be found in Section 4.2.

Appendix C. Additional results

See Tables C.5–C.11 and Figs. C.7 and C.8.

Table C.5
Performance metrics for **CS05-Norm-42** scenario on environments generated with seed 41, which were not used during training. This table compares the hypervolume, success rate, spread, and spacing. The values represent the mean and standard deviation performance over the 30 agents, rounded to three decimal places.

Metric	FF-NSGA-II	FF-SPEA2	LSTM-NSGA-II	FF-NEAT	RNN-NEAT	MODDPG
Hypervolume	18.951 ± 0.118	18.978 ± 0.085	18.763 ± 0.066	19.465 ± 0.113	19.411 ± 0.149	15.606 ± 1.375
Success rate	0.990 ± 0.005	0.990 ± 0.007	0.948 ± 0.018	0.967 ± 0.012	0.975 ± 0.007	0.882 ± 0.075
Spread	0.711 ± 0.118	0.700 ± 0.130	0.736 ± 0.280	0.628 ± 0.125	0.592 ± 0.136	0.315 ± 0.369
Spacing	0.033 ± 0.014	0.029 ± 0.012	0.026 ± 0.019	0.039 ± 0.015	0.041 ± 0.016	0.001 ± 0.002

Table C.6
Evaluation of the performance on the **CS10-Norm-42** scenario with metrics as in Table C.5.

Metric	FF-NSGA-II	FF-SPEA2	LSTM-NSGA-II	FF-NEAT	RNN-NEAT	MODDPG
Hypervolume	17.962 ± 0.097	18.328 ± 0.107	18.048 ± 0.070	17.584 ± 0.336	17.903 ± 0.389	14.703 ± 1.160
Success rate	0.973 ± 0.013	0.974 ± 0.013	0.946 ± 0.023	0.926 ± 0.045	0.981 ± 0.021	0.902 ± 0.049
Spread	0.668 ± 0.287	0.748 ± 0.279	0.597 ± 0.312	0.555 ± 0.288	0.588 ± 0.231	0.255 ± 0.331
Spacing	0.024 ± 0.029	0.029 ± 0.026	0.011 ± 0.009	0.049 ± 0.044	0.042 ± 0.032	0.001 ± 0.002

Table C.7
Performance on the **CS15-Norm-42** scenario with metrics as detailed in Table C.5.

Metric	FF-NSGA-II	FF-SPEA2	LSTM-NSGA-II	FF-NEAT	RNN-NEAT	MODDPG
Hypervolume	17.760 ± 0.124	17.771 ± 0.124	18.055 ± 0.044	16.834 ± 0.215	17.299 ± 0.245	11.913 ± 2.811
Success rate	0.960 ± 0.013	0.954 ± 0.012	0.939 ± 0.028	0.945 ± 0.049	0.986 ± 0.011	0.814 ± 0.134
Spread	0.625 ± 0.430	0.733 ± 0.310	0.498 ± 0.244	0.579 ± 0.353	0.575 ± 0.236	0.221 ± 0.293
Spacing	0.039 ± 0.044	0.041 ± 0.039	0.006 ± 0.005	0.068 ± 0.051	0.028 ± 0.025	0.000 ± 0.000

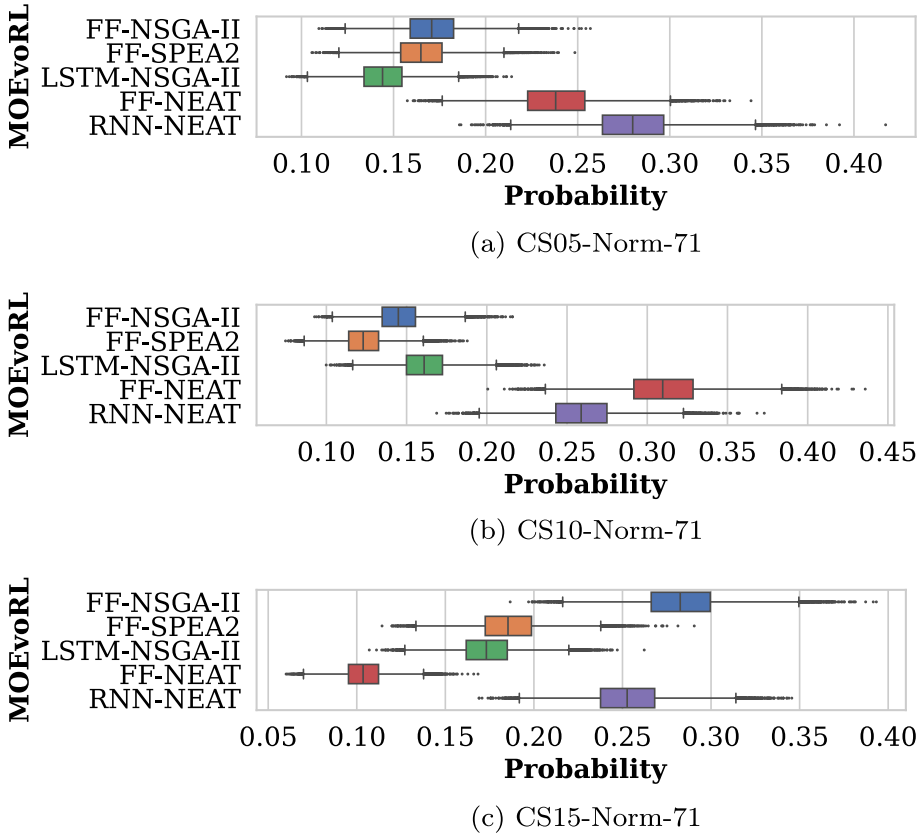
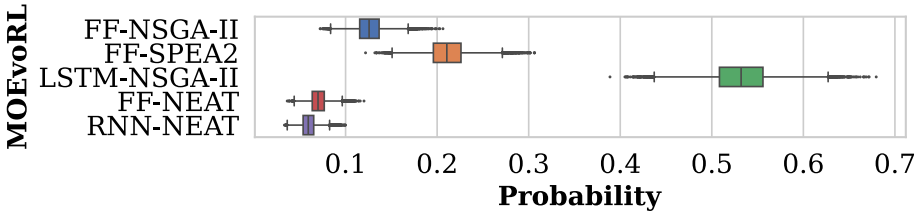
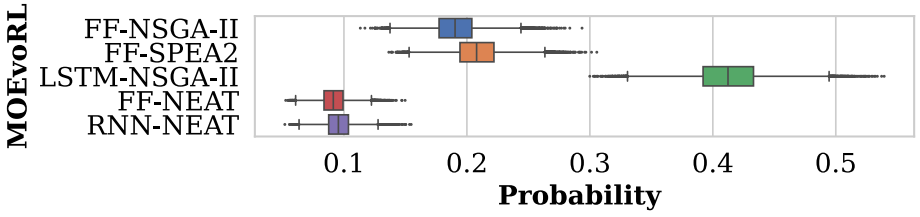


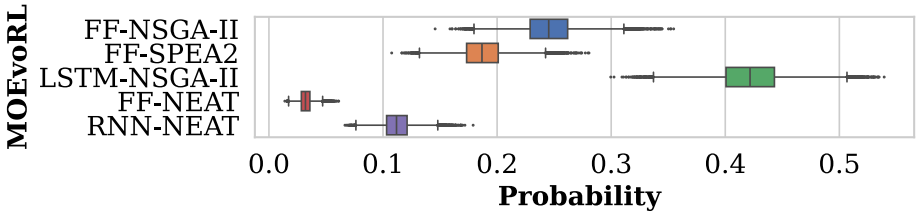
Fig. C.7. Box plots of the posterior distributions obtained from the ranking model as described in Fig. 4 applied to the **spread** data of the scenarios **CS05-Norm-71**, **CS10-Norm-71**, and **CS15-Norm-71**.



(a) CS05-Norm-71



(b) CS10-Norm-71



(c) CS15-Norm-71

Fig. C.8. Box plots illustrating the posterior distributions from the ranking model as described in Fig. 4 but for the **spacing** data of the **CS05-Norm-71**, **CS10-Norm-71**, and **CS15-Norm-71** scenarios.

Table C.8
Performance evaluation for the **CS05-High-71** scenario using the metrics outlined in Table C.5.

Metric	FF-NSGA-II	FF-SPEA2	LSTM-NSGA-II	FF-NEAT	RNN-NEAT	MODDPG
Hypervolume	18.688 ± 0.088	18.707 ± 0.064	18.712 ± 0.058	19.043 ± 0.106	18.975 ± 0.095	13.386 ± 1.487
Success Rate	0.983 ± 0.008	0.983 ± 0.011	0.954 ± 0.017	0.946 ± 0.018	0.961 ± 0.011	0.825 ± 0.073
Spread	0.795 ± 0.191	0.742 ± 0.177	0.801 ± 0.191	0.648 ± 0.185	0.680 ± 0.140	0.040 ± 0.122
Spacing	0.035 ± 0.019	0.032 ± 0.016	0.020 ± 0.010	0.037 ± 0.020	0.043 ± 0.018	0.000 ± 0.000

Table C.9
Analysis of the performance metrics for the **CS10-High-71** scenario, as listed in Table C.5.

Metric	FF-NSGA-II	FF-SPEA2	LSTM-NSGA-II	FF-NEAT	RNN-NEAT	MODDPG
Hypervolume	17.770 ± 0.048	17.780 ± 0.044	17.822 ± 0.047	17.151 ± 0.324	17.381 ± 0.293	11.857 ± 1.801
Success Rate	0.970 ± 0.013	0.970 ± 0.016	0.941 ± 0.024	0.903 ± 0.062	0.974 ± 0.026	0.811 ± 0.084
Spread	0.765 ± 0.302	0.817 ± 0.173	0.771 ± 0.311	0.497 ± 0.381	0.545 ± 0.242	0.034 ± 0.140
Spacing	0.023 ± 0.024	0.020 ± 0.014	0.015 ± 0.011	0.043 ± 0.049	0.029 ± 0.031	0.000 ± 0.001

Table C.10
Evaluation of **CS15-High-71** scenario performance based on the metrics from Table C.5.

Metric	FF-NSGA-II	FF-SPEA2	LSTM-NSGA-II	FF-NEAT	RNN-NEAT	MODDPG
Hypervolume	17.421 ± 0.060	17.419 ± 0.057	17.494 ± 0.026	16.599 ± 0.205	16.970 ± 0.226	9.006 ± 2.515
Success Rate	0.941 ± 0.017	0.933 ± 0.018	0.924 ± 0.034	0.942 ± 0.051	0.983 ± 0.012	0.687 ± 0.146
Spread	0.365 ± 0.359	0.353 ± 0.324	0.588 ± 0.329	0.666 ± 0.376	0.553 ± 0.289	0.025 ± 0.135
Spacing	0.009 ± 0.017	0.006 ± 0.008	0.006 ± 0.005	0.084 ± 0.048	0.028 ± 0.028	0.000 ± 0.000

Table C.11
Summary of the different sources of failure for the MORL algorithms. $\Delta I_{a,b}$, $\Delta I_{a,c}$, and $\Delta I_{b,c}$ represent the percentage of phase imbalance failures between phase pairs. $\Delta P_{\text{ext},\%}$ indicates the percentage of errors due to exceeding the node connection limit. ΔI_{phase} and ΔP_{ext} show the average and standard deviation of exceeding the maximum permitted phase imbalance (20 A) and grid connection node capacity (40 kW), respectively. The columns Arrival, Departure and Unknown indicate the percentage of failures attributable to an EV connection event. Finally, TS 1, TS 2, TS 3, and TS > 3 denote the percentage of time steps after which an failure occurred following an event such as Arrival or Departure.

Metric	FF-NSGA-II	FF-SPEA2	LSTM-NSGA-II	FF-NEAT	RNN-NEAT	MODDPG
$\Delta I_{a,b}$ [%]	13.42	13.40	10.78	24.47	18.61	9.12
$\Delta I_{a,c}$ [%]	85.41	85.41	88.25	69.69	78.17	65.76
$\Delta I_{b,c}$ [%]	1.18	1.19	0.97	5.84	3.22	25.03
$\Delta P_{\text{ext},\%}$ [%]	0.00	0.00	0.00	0.01	0.00	0.08
ΔI_{phase} [A]	1.73 ± 2.15	1.74 ± 1.97	1.19 ± 1.28	4.63 ± 3.51	1.78 ± 1.78	4.20 ± 3.62
ΔP_{ext} [kW]	–	–	–	1.30 ± 1.09	–	1.66 ± 1.30
Arrival [%]	95.99	97.16	85.03	94.50	89.67	41.14
Departure [%]	3.33	2.33	2.01	4.14	7.78	41.68
Unknown [%]	0.69	0.50	12.95	1.36	2.55	17.18
TS 1 [%]	98.06	98.61	55.19	86.31	73.71	74.32
TS 2 [%]	0.93	0.70	11.12	8.79	15.83	13.95
TS 3 [%]	0.45	0.29	7.15	2.74	5.34	3.76
TS > 3 [%]	0.57	0.40	26.55	2.16	5.12	7.97

Data availability

The code used in this paper is available on GitHub at https://github.com/NeeleKemper/moevorl-ev-cm.

References

[1] Zhu Q, Wu X, Lin Q, Ma L, Li J, Ming Z, et al. A survey on evolutionary reinforcement learning algorithms. Neurocomputing 2023;556:126628. doi:https://doi.org/10.1016/j.neucom.2023.126628.

[2] Qiu D, Wang Y, Hua W, Strbac G. Reinforcement learning for electric vehicle applications in power systems: a critical review. Renew Sustain Energy Rev 2023;173:113052. doi:https://doi.org/10.1016/j.rser.2022.113052.

[3] He H, Meng X, Wang Y, Khajepour A, An X, Wang R, et al. Deep reinforcement learning based energy management strategies for electrified vehicles: recent advances and perspectives. Renew Sustain Energy Rev 2024;192:114248. doi:https://doi.org/10.1016/j.rser.2023.114248.

[4] Guo J, Wang J, Xu Q, Wang B, Li K. Deep reinforcement learning-based hierarchical energy control strategy of a platoon of connected hybrid electric vehicles through cloud platform. IEEE Trans Transp Electrif 2024;10(1):305–15. doi:https://doi.org/10.1109/TTE.2023.3260824.

[5] Liu D, Zeng P, Cui S, Song C. Deep reinforcement learning for charging scheduling of electric vehicles considering distribution network voltage stability. Sensors (Switzerland) 2023;23(3):1618. doi:https://doi.org/10.3390/s23031618.

[6] Sultanuddin S, Vibin R, Kumar AR, Behera NR, Pasha MJ, Baseer K. Development of improved reinforcement learning smart charging strategy for electric vehicle fleet. J Energy Storage 2023;64:106987. doi:https://doi.org/10.1016/j.est.2023.106987.

[7] He X, Fei C, Liu Y, Yang K, Ji X. Multi-objective longitudinal decision-making for autonomous electric vehicle: a entropy-constrained reinforcement learning approach. In: 2020 IEEE 23rd international conference on intelligent transportation systems (ITSC); 2020. p. 1–6. doi:https://doi.org/10.1109/ITSC45102.2020.9294736.

[8] He X, Lv C. Towards energy-efficient autonomous driving: a multi-objective reinforcement learning approach. IEEE/CAA J Autom Sinica 2023;10(5):1329–31. doi:https://doi.org/10.1109/JAS.2023.123378.

[9] Zhang F, Yang Q, An D. Hamlet: hierarchical multi-objective reinforcement learning method for charging power control of electric vehicles with dynamic quantity. IEEE Trans Smart Grid 2024;15(1):783–94. doi:https://doi.org/10.1109/TSG.2023.3288277.

[10] Silva FLD, Nishida CEH, Roijers DM, Costa AHR. Coordination of electric vehicle charging through multiagent reinforcement learning. IEEE Trans Smart Grid 2020;11(3):2347–56. doi:https://doi.org/10.1109/TSG.2019.2952331.

[11] Yang N, Han L, Liu R, Wei Z, Liu H, Xiang C. Multiobjective intelligent energy management for hybrid electric vehicles based on multiagent reinforcement learning. IEEE Trans Transp Electrif 2023;9(3):4294–305. doi:https://doi.org/10.1109/TTE.2023.3236324.

[12] Abid MS, Apon HJ, Hossain S, Ahmed A, Ahshan R, Lipu MH. A novel multi-objective optimization based multi-agent deep reinforcement learning approach for microgrid resources planning. Appl Energy 2024;353:122029. doi:https://doi.org/10.1016/j.apenergy.2023.122029. https://www.sciencedirect.com/science/article/pii/S0306261923013934.

[13] Adetunji KE, Hofsajer IW, Abu-Mahfouz AM, Cheng L. A two-tailed pricing scheme for optimal EV charging scheduling using multiobjective reinforcement learning. IEEE Trans Ind Inf 2024;20(3):3361–70. doi:https://doi.org/10.1109/TII.2023.3305682.

[14] Li S, Hu W, Cao D, Zhang Z, Huang Q, Chen Z, et al. EV charging strategy considering transformer lifetime via evolutionary curriculum learning-based multiagent deep reinforcement learning. IEEE Trans Smart Grid 2022;13(4):2774–87. doi:https://doi.org/10.1109/TSG.2022.3167021.

[15] Qian T, Shao C, Li X, Wang X, Shahidehpour M. Enhanced coordinated operations of electric power and transportation networks via EV charging services. IEEE Trans Smart Grid 2020;11(4):3019–30. doi:https://doi.org/10.1109/TSG.2020.2969650.

[16] Zhou J, Feng C, Su Q, Jiang S, Fan Z, Ruan J, et al. The multi-objective optimization of powertrain design and energy management strategy for fuel cell–battery electric vehicle. Sustainability 2022;14(10):6320. doi:https://doi.org/10.3390/su14106320.

[17] Fan P, Ke S, Yang J, Wen Y, Xie L, Li Y, et al. A frequency cooperative control strategy for multimicrogrids with EVS based on improved evolutionary-deep reinforcement learning. Int J Electr Power Energy Syst 2024;159:109991. doi:https://doi.org/10.1016/j.ijepes.2024.109991.

[18] Hayes CF, Rădulescu R, Bargiacchi E, Källström J, Macfarlane M, Reymond M, et al. A practical guide to multi-objective reinforcement learning and planning. Auton Agent Multi Agent Syst 2022;36(1):26. doi:https://doi.org/10.1007/s10458-022-09552-y.

[19] Lillicrap TP, Hunt JJ, Pritzel A, Heess N, Erez T, Tassa Y, et al. Continuous control with deep reinforcement learning; 2015. arXiv preprint arXiv:1509.02971. doi:https://doi.org/10.48550/arXiv.1509.02971.

[20] Bai H, Cheng R, Jin Y. Evolutionary reinforcement learning: a survey. Intell Comput 2023;2:0025. https://spj.science.org/doi/pdf/10.34133/icomputing.0025.

[21] Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 2002;6(2):182–97. doi:https://doi.org/10.1109/4235.996017.

[22] Zitzler E, Laumanns M, Thiele L. SPEA2: improving the strength pareto evolutionary algorithm. TIK Report 2001;103. doi:https://doi.org/10.3929/ethz-a-004284029.

[23] Stanley KO, Miikkulainen R. Evolving neural networks through augmenting topologies. Evol Comput 2002;10(2):99–127. doi:https://doi.org/10.1162/106365602320169811.

[24] Lee ZJ, Li T, Low SH. ACN-Data: analysis and applications of an open EV charging dataset. In: Proceedings of the tenth ACM international conference on future energy systems. New York, NY, USA: Association for Computing Machinery; 2019. p. 139–49. doi:https://doi.org/10.1145/3307772.3328313.

[25] Powell S, Vianna Cezar G, Apostolaki-Iosifidou E, Rajagopal R. Large-scale scenarios of electric vehicle charging with a data-driven model of control. Energy 2022;248:123592. doi:https://doi.org/10.1016/j.energy.2022.123592.

[26] Székely GJ, Rizzo ML. Energy statistics: a class of statistics based on distances. J Stat Plan Inference 2013;143(8):1249–72. doi:https://doi.org/10.1016/j.jspi.2013.03.018. https://www.sciencedirect.com/science/article/pii/S0378375813000633.

[27] Schaul T, Quan J, Antonoglou I, Silver D. Prioritized experience replay; 2015. arXiv preprint arXiv:1511.05952. doi:https://doi.org/10.48550/arXiv.1511.05952.

[28] Biewald L. Experiment tracking with weights and biases; 2020. Software available from https://www.wandb.com/.

[29] Zitzler E. Evolutionary algorithms for multiobjective optimization: methods and applications, vol. 63. Shaker Ithaca; 1999.

[30] Zhou A, Jin Y, Zhang Q, Sendhoff B, Tsang E. Combining model-based and genetics-based offspring generation for multi-objective optimization using a convergence criterion. In: 2006 IEEE international conference on evolutionary computation; 2006. p. 892–9. doi:https://doi.org/10.1109/CEC.2006.1688406.

[31] Schott JR. Fault tolerant design using single and multicriteria genetic algorithm optimization [Ph.D. thesis]. Massachusetts Institute of Technology; 1995.

[32] Benavoli A, Corani G, Demšar J, Zaffalon M. Time for a change: a tutorial for comparing multiple classifiers through Bayesian analysis. J Mach Learn Res 2017;18(77):1–36. http://jmlr.org/papers/v18/16-305.html.

[33] Calvo B, Ceberio J, Lozano JA. Bayesian inference for algorithm ranking analysis. In: Proceedings of the genetic and evolutionary computation conference companion, GECCO '18. New York, NY, USA: Association for Computing Machinery; 2018. p. 324–5. doi:https://doi.org/10.1145/3205651.3205658.

[34] Calvo B, Shir OM, Ceberio J, Doerr C, Wang H, Bäck T, et al. In: Bayesian performance analysis for black-box optimization benchmarking, GECCO '19. New York, NY, USA: Association for Computing Machinery; 2019. p. 1789–97. doi:https://doi.org/10.1145/3319619.3326888.

[35] Kruschke JK. Bayesian estimation supersedes the t test. J Exp Psychol Gen 2013;142(2):573. doi:https://doi.org/10.1037/a0029146.

[36] Benavoli A, Mangili F, Corani G, Zaffalon M, Ruggeri F. A Bayesian wilcoxon signed-rank test based on the dirichlet process. In: Proceedings of the 31st international conference on international conference on machine learning, vol. 32, ICML'14, JMLR.org, 2014. p. II-1026–34.