Reinforcement Learning Based EV Charging Scheduling: A Novel Action Space Representation

Kun Qian, Rebecca Adam, Robert Brehm
Center for Industrial Electronics (CIE)
University of Southern Denmark
Sønderborg, Denmark
{kqian, rebeccaadam, brehm}@sdu.dk

2021 IEEE PES Innovative Smart Grid Technologies - Asia (ISGT Asia) | 978-1-6654-3339-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/ISGTASIA49270.2021.9715603

***Abstract*—In recent years, several optimization techniques have been proposed for electric vehicle (EV) charging scheduling. A common approach to intelligent scheduling is day-ahead planning, assuming full arrival time, departure time and energy demand knowledge or having them forecasted. However, the result from the day-ahead scheduling is limitedly applicable due to the uncertainties from the charging behaviors. With the deployment of the EV charging communication protocol defined in ISO 15118, it is realistic to assume that the EV will publish the departure time and the energy demand upon arrival. Thus, real-time scheduling, making decisions at each decision timeslot, can adapt to the new information and increase scheduling performance. Traditional model-based approaches like model predictive control (MPC) still require models, for example, for the future arrival times to solve the scheduling problem. Reinforcement learning (RL), a model-free approach, has also been successfully applied to real-time scheduling. RL can learn how to make decisions without relying on any system knowledge. This paper proposes a new action space construction method for an RL as proposed in a preceding work. The resulting action space size is significantly reduced compared to the original approach. Further, we compare the performance of a novel prioritized RL method to the original method. A publicly available charging session dataset is used for performance comparison in contrast to the original method. It is shown, that the prioritized RL performs better.**

***Index Terms*—electric vehicles, charging scheduling, reinforcement learning, action space**

I. Introduction

The rapidly developing transformation targeting vehicle electrification increase demands scalable and flexible intelligent charging solutions. Even though the total energy demands from EVs might be minimal to the grid, uncoordinated charging can significantly change the instantaneous demand shape, resulting in potential risks [1], such as overloading the local transformer. On the other hand, the smart grid can alleviate or preclude the risks introduced by the EV charging. EV smart grid interaction benefit the grid operators and EV owners. It can be step by step enabled at different EV technology stages and the involved infrastructure development [2]. To facilitate and accelerate the smart grid interaction, one focus area concerns scheduling algorithms development.

There is a substantial body of research falling on intelligent charging algorithms. The objectives for charging scheduling are typically flattening the grid load [3], reducing the charging cost [4], and maximizing the delivered energy to EVs [5]. Traditional computational scheduling methods include genetic programming [3], quadratic programming [6], and mixed-integer programming [7]. We refer to [8], [9] for a review of smart charging objectives and algorithms. These methods primarily focus on the static charging case, precomputing the charging schedules for EVs over a time horizon, usually over a day.

However, due to the charging behavior uncertainties, such as the EV arrival times, precomputed schedules have limited applicability in practice. ISO 15118 enables bi-directional communication between the EV and the charging station. Thus, the charging station can get critical information such as the departure time and the energy request upon the EV be applied to mitigate the weakness from static charging scheduling. Model-based methods such as model predictive control (MPC) require models, for example, for the future arrival times to solve the scheduling problem.

A data-driven or ”blind” learning approach - reinforcement learning (RL), on the other hand, has emerged to facilitate intelligent scheduling. Compared to model-based methods, RL can learn how to make a good decision without relying on any system knowledge. For example, Wan *et al.* [10] have applied a finite Markov decision process (MDP) with discrete-time steps to formulate the charging scheduling of an EV, to minimize the cost. Since the state transition depends on the randomness from the future electricity prices and the user's commuting behavior, they approximated the action-value with a deep neural network (DNN) and updated the value iteratively. Li *et al.* [4] have presented a similar approach. Though, instead of handling the constraints (for example, fully charge the EV before departing) as penalties, they modeled the problem as a constrained Markov Decision Process (CMDP) and solved it with an approach based on safe deep reinforcement learning (SDRL). Note that the RL-based solutions in [4], [10] apply only for an individual EV. In comparison, Sadeghianpourhamami *et al.* [11] propose a scalable MDP formulation applicable to a group of EV charging stations. They used a batch RL algorithm, fitted Q-iteration (FQI), to compute the charging scheduling for groups of EVs, to flatten the load.

However, the proposed RL method in [11] potentially has a vast action space; thus, it requires a large exploration dataset to achieve near-optimal scheduling. The follow-up paper [12] has reduced the action space size, but it is far from ideal.

This paper builds on the works of [11], [12]. We propose a priority matrix based on the aggregate demand matrix and construct the action space based on those two matrices. The resulting action space size is significantly reduced compared to the original methods. We further compare overall performance in simulation experiments.

II. Reinforcement Learning and Markov Decision Process

MDP is a mathematically idealized form for RL problems. Following the work of [11], a finite MDP with discrete-time steps is applied to formulate the real-time EV charging scheduling to flatten the load. An MDP is a 5-tuple $(\mathcal{S}, \mathcal{U}, \mathcal{P}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ is the set of system states; $\mathcal{U}$ is the set of actions; $\mathcal{P}$ is the state transition probability; $\mathcal{R}$ is the immediate reward; γ is a discount factor. Compared to [11], the main difference is the action space representation. We describe the details about the used MDP and the RL algorithm below.

A. State

From the connected EVs, we can extract the remaining time before departure, Δt^{depart}, and the required time to fully charge, Δt^{charge}. The aggregate demand, $\mathbf{X}_s$, is built based on each connected EV's Δt^{depart} and Δt^{charge}. The dimension of the $\mathbf{X}_s$ matrix, $S_{\text{max}} \times S_{\text{max}}$, depends on the scheduling window, H_{max}, and the decision timeslot, Δt^{slot}: $S_{\text{max}} \triangleq H_{\text{max}}/\Delta t^{\text{slot}}$. The element at position (j, i) of $\mathbf{X}_s$ counts the number of connected EVs, of which, $i = \lceil \Delta t^{\text{depart}}/\Delta t^{\text{slot}} \rceil$ and $j = \lceil \Delta t^{\text{charge}}/\Delta t^{\text{slot}} \rceil$. Further, all the elements are divided by the number of charging stations, N, to ensure scale-free, i.e., independent of N.

The system state at time step t is defined as $s = (t, \mathbf{X}_s)$, comprising two types of information: the timeslot, t, and the aggregate demand, $\mathbf{X}_s$.

B. Action

Any action in the action space $\mathcal{U}_s$ is to decide whether to charge a specific EV or not. And we assume the same charging rate, P, for all EVs. The *flexibility*, denoted as $\Delta t^{\text{flex}} = \Delta t^{\text{depart}} - \Delta t^{\text{charge}}$, indicates how long the charging can be delayed. We can easily infer that EVs binned on the same diagonal have the same flexibility. Ref. [11] and [12] use a vector $\mathbf{x}_s{}^{\text{total}}$ to denote the number of EVs binned on each diagonal of $\mathbf{X}_s$, where $\mathbf{x}_s{}^{\text{total}}[0]$ is the main diagonal, $\mathbf{x}_s{}^{\text{total}}[d]$ is the upper d^{th} diagonal, and $d = 0, ..., S_{\text{max}} - 1$.

Based on the vector $\mathbf{x}_s{}^{\text{total}}$, Sadeghianpourhamami *et al.* [11] define the available actions in the state s as to select a portion of each element in the vector $\mathbf{x}_s{}^{\text{total}}$. All the available actions build the action space $\mathcal{U}_s$. Thus, the resulting action space size for the state s is [11]:

$$|\mathcal{U}_s| = \prod_{d=0}^{S_{\text{max}}-1} \left(\mathbf{x}_s{}^{\text{total}}(d) + 1 \right). \tag{1}$$

On the basis of the work from [11], Lahariya *et al.* [12] find that, for EVs binned in the main diagonal of $\mathbf{X}_s$, it is reasonable to always select all to charge since they have zero flexibility for delayed charging. Thus, the resulting updated action space size for the state s is [12]:

$$|\mathcal{U}_s|_{\text{updated}} = 1 + \prod_{d=1}^{S_{\text{max}}-1} \left(\mathbf{x}_s{}^{\text{total}}(d) + 1 \right). \tag{2}$$

We further analyze the example from [11], as shown in Fig. 1, where there are 2 EVs connected and the scheduling horizon has 3 decision timeslots. Their remaining time till departure and fully charged, as well as the resulting demand matrix, are also shown in Fig. 1. Provided that we need

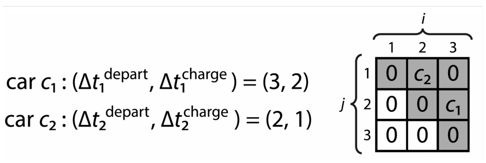
car $c_1 : (\Delta t_1^{\text{depart}}, \Delta t_1^{\text{charge}}) = (3, 2)$
car $c_2 : (\Delta t_2^{\text{depart}}, \Delta t_2^{\text{charge}}) = (2, 1)$
i
1 2 3
j
1 2 3
0 c_2 0
0 0 c_1
0 0 0

Fig. 1. Demand matrix for 2 connected EVs, and $S_{\text{max}} = 3$ as in [11].

to delay the charging for one EV, there are two potential load patterns formed by these two EVs between timeslot t and $t + 3 \cdot \Delta t^{\text{slot}}$: (a) $[W, W, W]$; and (b) $[W, 2 \cdot W, 0]$ $(W = P \cdot \Delta t^{\text{slot}})$. Together with other connected or to be connected EVs, the scheduling can achieve the desired load pattern over the scheduling horizon. For case (a), charging either c_1 or c_2 at timeslot t is applicable. But, for case (b), we must charge c_1 at timeslot t. Thus, to achieve the potential desired load pattern, we prefer selecting c_1 rather than c_2 to charge at timeslot t. Similarly, for those with the same Δt^{depart}, we prefer selecting EVs with longer Δt^{charge}; for those with the same Δt^{charge}, we prefer selecting EVs with longer Δt^{depart}. Consequently, we propose a priority matrix $\mathbf{X}_s{}^{\text{priority}}$, defined as:

$$[\mathbf{X}_s{}^{\text{priority}}]_{ij} = \begin{cases} 0, \text{ if } i < j \\ 1, \text{ if } i = j \\ S_{\text{max}} + j - 2i - 1, \text{ else,} \end{cases} \tag{3}$$

where $\mathbf{X}_s{}^{\text{priority}}$ has the same dimension as $\mathbf{X}_s$, i and j are row and column indexes.

Based on the proposed priority matrix, we define the action space $\mathcal{U}_s$ as: $\left\{ U_s(1), ..., U_s\left(\sum_{d=1}^{S_{\text{max}}-1} \mathbf{x}_s{}^{\text{total}}(d) + 1 \right) \right\}$, where each action means:

- $u_s = U_s(1)$: only select EVs binned on the main diagonal (if there are any);
- $u_s = U_s(2)$: besides EVs binned on the main diagonal, one extra EV having a higher priority according to the priority matrix will be selected;
- $u_s = U_s(3)$: besides EVs binned on the main diagonal, two extra EVs having higher priorities according to the priority matrix will be selected;

  ⋮
- $u_s = U_s\left(\sum_{d=1}^{S_{\text{max}}-1} \mathbf{x}_s{}^{\text{total}}(d) + 1 \right)$: select all EVs to charge.

Consequently, the resulting prioritized action space size is:

$$|\mathcal{U}_s|_{\text{prioritized}} = \sum_{d=1}^{S_{\max}-1} \mathbf{x}_s{}^{\text{total}}(d) + 1. \quad (4)$$

In the case when N charging stations are fully occupied and all EVs have $\Delta t^{\text{flex}} > 0$, $|\mathcal{U}_s|_{\text{prioritized}}$ reaches the maximum, $N + 1$.

Here is another example, where there are 4 EVs connected and $S_{\max} = 3$:

$$\begin{matrix} \text{car } c_1 : (\Delta t_1^{\text{depart}}, \Delta t_1^{\text{charge}}) = (2, 2) \\ \text{car } c_2 : (\Delta t_2^{\text{depart}}, \Delta t_2^{\text{charge}}) = (2, 2) \\ \text{car } c_3 : (\Delta t_3^{\text{depart}}, \Delta t_3^{\text{charge}}) = (2, 1) \\ \text{car } c_4 : (\Delta t_4^{\text{depart}}, \Delta t_4^{\text{charge}}) = (3, 1) \end{matrix} \Rightarrow \begin{bmatrix} 0 & c_3 & c_4 \\ 0 & c_1, c_2 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

and the resulting $\mathbf{x}_s{}^{\text{total}}$ is $[2, 1, 1]$. The $\mathbf{X}_s{}^{\text{priority}}$ is:

$$\mathbf{X}_s{}^{\text{priority}} = \begin{bmatrix} 1 & 3 & 4 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix},$$

Thus, the prioritized action space has three available actions:

- $u_s = U_s(1)$: select c_1 and c_2 to charge;
- $u_s = U_s(2)$: besides c_1 and c_2, select one extra EV to charge. According to the priority matrix, the selected EV is c_3;
- $u_s = U_s(3)$: select all EVs to charge.

As a comparison, we have the different action space sizes as:

$$|\mathcal{U}_s| = 12,$$
$$|\mathcal{U}_s|_{\text{updated}} = 5,$$
$$|\mathcal{U}_s|_{\text{prioritized}} = 3.$$

We will further demonstrate how significantly we can reduce the action space size by constructing the action space with the proposed priority matrix in the simulation.

C. State Transition

In an MDP, the state transition probability for going from one state s to the next state s' is denoted as:

$$s' = p(s, u_s, \omega_s), \quad (5)$$

where the action u_s will cause a non-deterministic result due to the randomness ω_s, which comes from the stochasticity of the future arrival EVs and their energy demands. Modeling the randomness ω_s is difficult since it is subjected to many factors. Instead, we can estimate the transition probabilities by interaction and observe the resulting s' and its associated reward from taking action u_s at the state s.

D. Reward Function

Instead of using the term reward, Sadeghianpourhamami *et al.* [11] refer to it as a cost. Consequently, the objective is to minimize the cost. Like [12], we have eliminated the penalty part since all the action options cover charging those EVs binned on the main diagonal. So, the immediate cost when transitioning to state s' from taking action u_s at the state s is defined as [12]:

$$C(s, u_s, s') \triangleq C^{\text{demand}}(\mathbf{X}_s, u_s), \quad (6)$$

where $C^{\text{demand}}(\mathbf{X}_s, u_s)$ is the quadratic power consumption resulting from taking action u_s at the state s.

E. Action-Value Function

The quality of the action u_s at the state s at a given t is assessed by the expected K time steps cost return:

$$Q^{\pi}(s, u_s) = \mathbb{E}_{\pi}\left[\sum_{k=0}^{K} \gamma^k \cdot C_{t+k} \middle| s_t = s, u_t = u_s\right], \quad (7)$$

where $Q^{\pi}(s, u_s)$ is the value action function, π is the scheduling policy deciding whether to charge a specific EV or not at a state s, and γ is the discount factor encoding the long-term return desirability. The objective is to find the optimal policy to minimize the action-value function:

$$Q^*(s, u_s) = \min_{u \in \mathcal{U}_{\text{prioritized}}} \mathbb{E}\left[C(s, u_s, s') + \gamma Q^*(s', u)\right]. \quad (8)$$

If the transition probability is available, we can analytically determine the optimal policy using dynamic programming. Due to the lack of transition probability, we apply the learning algorithm, FQI, to approximate the action-value function.

F. Fitted Q-iteration

Same as [11], we use FQI to learn a control policy from the past experience buffer. The experience buffer is in the form $(s, u_s, s', C(s, u_s, s'))$, and we generate it by following a non-optimal control policy. We refer readers to [11] for a detailed description of FQI.

III. Simulation and Analysis

A. Data Preparation

For the analysis, we use the publicly available charging sessions collected by Caltech ACN [13]. Specifically, we take 10 weeks of collected sessions starting from 1 Feb, 2021 at the site JPL. JPL is a national research lab located in La Canada, CA, and currently, it has 50 charging ports and is only open to employees.

Same as [11], we set the maximum connection duration to $H_{\max} = 24$h, and set the decision timeslot to $\Delta t^{\text{slot}} = 2$h. Consequently, $S_{\max} = H_{\max}/\Delta t^{\text{slot}} = 12$.

To define the starting time of an episodic day, we plotted the connection status during a day over the eight weeks, as shown in Fig. 2. Obviously, at 9 am, there are fewer connected EVs. Thus, we define 9 am as the start of a day, and the day ends 24h later.

Then, we reorganized the charging sessions. Firstly, we only keep those sessions in which the stay duration is longer than 0.5h. Second, for those span over two days, we truncate and assign them to the day where the EVs stayed longer, and we adjust the energy demands accordingly to ensure the complete charging of the EVs. Note that we have set the fixed and maximum charging power to 7 kW.

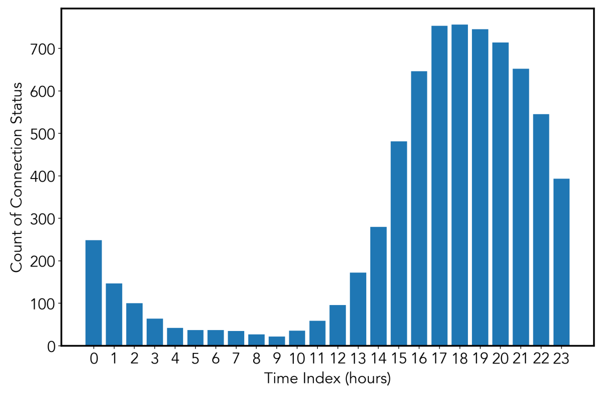

Fig. 2. Connection status over eight weeks. The bar heights indicate how many charging stations are occupied at the corresponding time index over eight weeks.

B. Algorithm Settings

The creation of the experience buffer and the neural network architecture is similar to [11]. For details, please refer to [11]. The main difference is that the input layer for the prioritized RL is in the form of a vector of length $S_{\text{max}}^2 + 1 + 1$ instead of $S_{\text{max}}^2 + S_{\text{max}} + 1$, as the action is a scalar value. Besides, when generating the available actions of the next state, since the prioritized action space has a maximum of 51 options, we thus randomly generate 51 actions based on the $\mathbf{x}_s{}^{\text{total}}$ of the next state for the updated RL policy.

C. Performance Evaluation

Among the 10 weeks of charging sessions from ACN-data, we use the first 6 weeks as the training set and the last 4 weeks as the test set. We compare the prioritized RL policy with the updated RL policy, park-to-charge policy, and optimal policy to evaluate the performance. We do not include the policy from [11] as it is concluded in [12] that the updated policy has a similar performance. Park-to-charge policy means that the EVs get charged immediately once connected. For the optimal policy, we provide all the necessary information (time of arrivals, time of departures, and energy demands) for a day and use an integer quadratic programming solver to calculate the optimal scheduling for that day.

To compare the performance, let e_i be the i^{th} day from the test set, $C_\pi^{e_i}$ denotes the cost for the i^{th} day when following either the optimal, park-to-charge, updated RL, or prioritized RL policy, and $C_{\text{opt}}^{e_i}$ denotes the cost for the i^{th} day when following the optimal policy. We notice that the charging behavior within those 50 charging stations varies significantly from day to day, especially between weekdays and weekends. Thus, we choose to compare the weighted mean of the normalized daily cost to evaluate the overall performance:

$$C_\pi = \sum_{e \in D} \frac{C_{\text{opt}}^e}{\sum_{e \in D} C_{\text{opt}}^e} \frac{C_\pi^e}{C_{\text{opt}}^e}, \tag{9}$$

where $D = \{e_i | i = 1, ..., 28\}$, which is the 4-week test set.

D. Result Analysis

Before evaluating the performance, we demonstrate how much the prioritized action space size is reduced compared to the original methods. We have picked an example day, 3rd March 2021, from the test set. Table I shows the action space sizes derived from the corresponding aggregate demand for each decision timeslot (we randomly select an action at each timeslot from the prioritized action space). Obviously, our action space construction method with priority matrix is better. For example, at timeslot index 5, there are 576 and 97 options for the original methods, while there are only 10 options if we construct the action space based on the priority matrix. Further, from an analytic point of view, with 50 charging stations, the action space sizes from the original methods can potentially go beyond millions scale while the prioritized method can maximum have 51 options.

TABLE I
DIFFERENT ACTION SPACE SIZES FOR 3RD MARCH 2021

Action space type	**timeslot index**											
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
$\lvert\mathbf{U}_s\rvert$	2	4	9	72	576	96	60	10	1	2	2	2
$\lvert\mathbf{U}_s\rvert_{\text{updated}}$	3	3	4	25	97	25	13	3	2	2	3	2
$\lvert\mathbf{U}_s\rvert_{\text{prioritized}}$	2	2	3	6	10	7	7	2	1	1	2	1

To compare the overall performance, we have chosen 0, 100, 200, 300, 400, and 500 as the experimental number of sampled trajectories per day, where 0 means we do not train the RL models after the random initialization. To count for the randomness introduced by the experience buffer generation and training, we decided to have 12 runs for each point and calculate the average cost. Fig. 3 shows the overall performance comparison. As shown, the untrained models

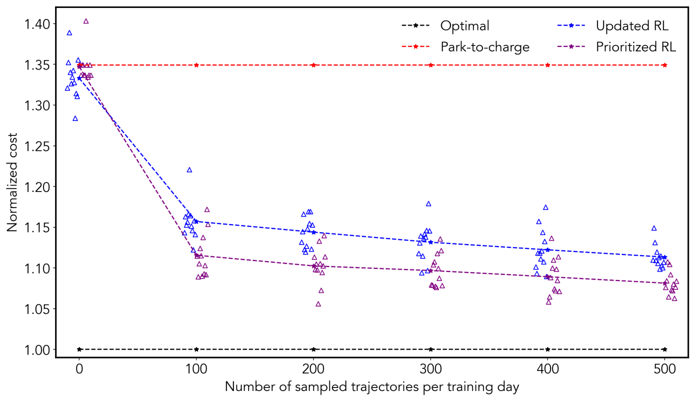

Fig. 3. Normalized total cost for the test set with different policies and different number of sampled trajectories per day. There are 12 runs for the RL policies at each point, and the results are presented as △. The averaged result from the 12 runs is denoted as $*$.

have similar performance to the park-to-charge policy. After training, both updated and prioritized RL policies can reduce the cost.

As shown in Fig. 4, the t-tests[1] show that the RL with the prioritized action space has significantly improved the performance compared to the original method, and the improvements are all around 4%. One possible reason for the improvements

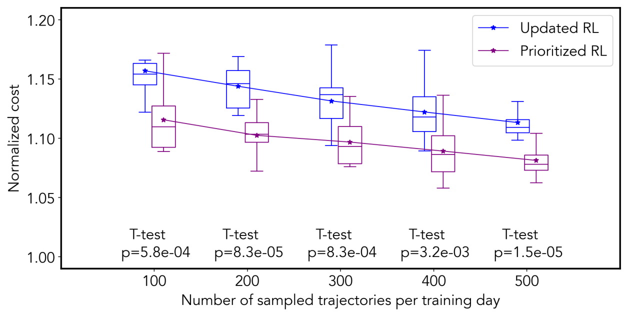

Fig. 4. Normalized total cost for the test set with different policies and different number of sampled trajectories per day. The average results are denoted as $*$.

is that the maximum size for the prioritized action space (in this case, 51) has limited the number of randomly sampled actions from the next state for the updated RL policy. As can be seen from Table I, the value 51 is sometimes much smaller compared to the action space size for the updated RL policy, which leads to the situation that the optimal action is not evaluated and cannot be selected.

IV. Conclusion

In this paper, we formulate the charging scheduling problem as a finite MDP with unknown system dynamics or transition probability. By analyzing the demand matrix and the potential load patterns from different action options, we find that prioritized EV selection is preferable than random selection. Thus, we propose a priority matrix for the aggregate demand matrix and construct the action space based on those two matrices. The resulting prioritized action space is significantly reduced compared to the original methods. We then apply the FQI to learn the control policy with a real-world dataset and compare it to the park-to-charge policy, a theoretically optimal policy assuming full charging behavior knowledge, and the policy from the previous literature. The result shows that both the prioritized policy and the original policy in the prior literature can achieve a good performance. Yet, our prioritized policy performs better.

References

[1] M. Muratori, “Impact of uncoordinated plug-in electric vehicle charging on residential power demand,” *Nature Energy*, vol. 3, no. 3, pp. 193–201, 2018.

[2] D. P. Tuttle and R. Baldick, “The evolution of plug-in electric vehicle-grid interactions,” *IEEE Transactions on Smart Grid*, vol. 3, no. 1, pp. 500–505, 2012.

[3] M. Alonso, H. Amaris, J. G. Germain, and J. M. Galan, “Optimal charging scheduling of electric vehicles in smart grids by heuristic algorithms,” *Energies*, vol. 7, no. 4, pp. 2449–2475, 2014.

[1]t-test tells if there is a significant difference between the means of two groups, and we use the conventional cutoff of 0.05 for the p-value.

[4] H. Li, Z. Wan, and H. He, “Constrained EV charging scheduling based on safe deep reinforcement learning,” *IEEE Transactions on Smart Grid*, vol. 11, no. 3, pp. 2427–2439, 2020.

[5] Z. Darabi, P. Fajri, and M. Ferdowsi, “Intelligent Charge Rate Optimization of PHEVs Incorporating Driver Satisfaction and Grid Constraints,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 5, pp. 1325–1332, 2017.

[6] Y. He, B. Venkatesh, and L. Guan, “Optimal scheduling for charging and discharging of electric vehicles,” *IEEE Transactions on Smart Grid*, vol. 3, no. 3, pp. 1095–1105, 2012.

[7] O. Frendo, N. Gaertner, and H. Stuckenschmidt, “Real-Time Smart Charging Based on Precomputed Schedules,” *IEEE Transactions on Smart Grid*, 2019.

[8] Q. Wang, X. Liu, J. Du, and F. Kong, “Smart charging for electric vehicles: A survey from the algorithmic perspective,” *IEEE Communications Surveys Tutorials*, vol. 18, no. 2, pp. 1500–1517, 2016.

[9] Z. Yang, K. Li, and A. Foley, “Computational scheduling methods for integrating plug-in electric vehicles with power systems: A review,” *Renewable and Sustainable Energy Reviews*, vol. 51, pp. 396–416, 2015.

[10] Z. Wan, H. Li, H. He, and D. Prokhorov, “Model-free real-time EV charging scheduling based on deep reinforcement learning,” *IEEE Transactions on Smart Grid*, vol. 10, no. 5, pp. 5246–5257, 2019.

[11] N. Sadeghianpourhamami, J. Deleu, and C. Develder, “Definition and evaluation of model-free coordination of electrical vehicle charging with reinforcement learning,” *IEEE Transactions on Smart Grid*, vol. 11, no. 1, pp. 203–214, 2020.

[12] M. Lahariya, N. Sadeghianpourhamami, and C. Develder, “Reduced state space and cost function in reinforcement learning for demand response control of multiple EV charging stations,” in *Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, BuildSys ’19, (New York, NY, USA), p. 344–345, Association for Computing Machinery, 2019.

[13] Z. J. Lee, T. Li, and S. H. Low, “ACN-data: Analysis and applications of an open EV charging dataset,” in *Proceedings of the Tenth ACM International Conference on Future Energy Systems*, e-Energy ’19, (New York, NY, USA), p. 139–149, Association for Computing Machinery, 2019.