Scalable Attention-based Reinforcement Learning Method for Multi-asset Control

Seyed Soroush Karimi Madahi [1] **Giuseppe Gabriele** [1] **Bert Claessens** [1 2] **Chris Develder** [1]

Abstract

In this paper, we propose a scalable centralized reinforcement learning method to jointly optimize a fleet of flexible assets. The attention layer in our proposed architecture enables the agent to make decisions for each asset based on both its local and aggregated asset-specific information. As a proof-of-concept, we investigate the performance of the proposed method on an electric vehicle (EV) charging problem. The results show that the trained agent can effectively control multiple EVs to achieve a common objective (load flattening in our case).

1. Introduction

Buildings account for around 40% of global energy consumption and a quarter of CO2 emissions (Gao et al., 2024). To reduce their energy consumption, controlling Heating, Ventilation, and Air Conditioning (HVAC) systems and electric vehicles (EVs), as major energy consumers in buildings, plays a significant role. These flexible assets can be effectively managed to optimize energy costs and support the reliable operation of electricity grids, while also satisfying user comfort. However, managing these assets independently fails to capture the advantages of collective optimization, highlighting the importance of coordination among multiple assets to achieve a global objective or to satisfy coupling constraints among them.

Various approaches have been proposed to jointly control a fleet of assets, mostly based on model-based optimization methods, such as model predictive control (MPC) (Chen et al., 2018; Vandael et al., 2012; Afram & Janabi-Sharifi, 2014; Madahi et al., 2020). However, these model-based methods suffer from four major drawbacks: (i) they need an accurate system dynamics model, which can sometimes be challenging to develop, especially for building thermal models (Yu et al., 2020); (ii) since they need to solve an optimization problem on-the-fly, these methods can be time–consuming for large-scale problems (such as controlling a large number of EVs simultaneously), making them inefficient for real-time decision-making (Kamrani et al., 2025); (iii) due to their dependence on specific models, their generalizability is limited for problems involving multiple environments (such as multi-building scenarios) (Madahi et al., 2024); (iv) their performance is highly dependent on the accuracy of predictions for uncertain system states, such as future EV arrivals and required energy (Qiu et al., 2023).

Model-free reinforcement learning (RL) can address the aforementioned shortcomings of model-based methods, as it does not require any prior knowledge of system models. RL agents implicitly learn system dynamics by interacting with the environment(s), which leads to learning the optimal policy. The proposed RL-based solutions for the joint coordination of a fleet of flexible assets can be broadly categorized into two main groups: (i) *Centralized* (Orfanoudakis et al., 2025; Sadeghianpourhamami et al., 2019; Nagy et al., 2018; Claessens et al., 2016): a single-agent manages all assets based on complete knowledge of the system. Although such global oversight can lead to better decisions, most of these methods lack scalability due to the exponential growth of the input space with the number of agents, limiting their applicability to large-scale problems. (ii) *Distributed* (Savino et al., 2025; Yan et al., 2022; Li et al., 2024): Each asset is controlled by an individual agent that makes decisions based on local information. Typically, these agents are trained in a centralized manner to ensure effective coordination between them. However, distributed methods can result in sub-optimal decisions due to decision-making based on local states rather than the global state.

In this paper, we aim to address the gap in previous works by proposing a scalable centralized RL method for the joint optimization of a fleet of flexible assets. By aggregating fleet information using the attention mechanism, the decision for each asset is made based on its individual information and its priority relative to other assets. Since the lengths of individual information and priority vectors are fixed, increasing the number of assets does not affect the size of the network, making the proposed method scalable and suitable

[1]IDLab, Ghent University – imec, Ghent, Belgium [2]Beebop, Belgium. Correspondence to: Seyed Soroush Karimi Madahi <seyedsoroush.karimimadahi@ugent.be>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

for large-scale problems. As a proof-of-principle, and to assess the performance of our proposed method, we implement it for an EV charging problem, where a fleet of EVs at a residential parking lot is controlled to flatten the parking consumption profile.

2. Proposed Architecture

Figure 1 shows an overview of the proposed architecture. $x_i^l \in \mathbb{R}^{1\times d_l}$ denotes features related to asset i. First, these vectors are fed into a fully connected neural network, FC$(\cdot)$, which outputs embedding vectors $e_i \in \mathbb{R}^{1\times d_e}$ as follows:

$$e_i = \text{FC}\left(x_i^l\right). \tag{1}$$

We use a self-attention mechanism, introduced in (Vaswani et al., 2017), to obtain a fleet-aware representation for each asset as follows:

$$\mathbf{h} = \text{softmax}\left(\frac{(W_q\mathbf{e})(W_k\mathbf{e})^T}{\sqrt{d_e}}\right) W_v\,\mathbf{e}, \tag{2}$$

where $\mathbf{e} = \bigoplus_{i=1}^{N} e_i$ is the concatenation of embeddings, N is the total number of assets, and $\mathbf{h} \in \mathbb{R}^{N\times d_e}$ denotes our proposed fleet-aware representation. $W_q, W_k, W_v \in \mathbb{R}^{d_e\times d_e}$ are projection matrices that transform embeddings into query, key, and value matrices, respectively. A fleet-aware representation for each asset encapsulates the relation of that asset to the entire fleet. In this way, each asset has access to aggregated information about the entire fleet, processed into its specific representation h_i.

As a part of the final asset representation, we also represent asset-independent global information as a single vector $x^g \in \mathbb{R}^{1\times d_g}$. We simply concatenate $\mathbf{x^g} = \bigoplus_{i=1}^{N} x^g$ with the original asset representations and the fleet-aware representations $\mathbf{h}$. Finally, the output of our model, $\mathbf{o} \in \mathbb{R}^{N\times d_o}$, is calculated using a fully connected layer as follows:

$$\mathbf{o} = \bigoplus_{i=1}^{N} \text{FC}(x_i^l, x^g, h_i). \tag{3}$$

Before the fully connected layer, we pass its inputs through a normalization layer to stabilize and speed up training (Ba et al., 2016).

Since the sizes of the fully connected networks are determined by the length of the information and embedding vectors for an individual asset, it remains independent of the number of assets, thereby making the proposed architecture scalable to large asset pools. Furthermore, the inputs to the fully connected network can be parallelized to improve the computational efficiency of the architecture. Note that using an attention layer provides two main benefits. First, it computes a vector for each asset that reflects its situation relative to the other assets. For decision-making, the final fully connected layer thus can rely on both aggregated asset-specific and local information to assess the priority or urgency of each asset in relation to the rest of the fleet. Second, the attention layer makes the architecture dynamic, enabling it to handle a variable number of assets at different decision time steps. This is especially important when flexible assets are not always available for control (such as when controlling multiple HVAC systems, some of which may become uncontrollable due to communication loss) or when their number changes dynamically over time (as in the case of the EV charging problem).

3. Problem Formulation

While our proposed architecture can be applied for controlling any aggregate of flexible assets (e.g., heat pumps, batteries, EV charging stations), as a proof-of-principle, we will apply it here for the joint control of multiple EV chargers at a parking site. Specifically, we aim to control the concurrent charging of multiple EVs to flatten the load of the whole parking, while satisfying the needs of EV users. Since the objective concerns the joint load of EV chargers together, they must collaborate and communicate effectively to achieve it, highlighting the importance of the attention layer and fleet-aware representation in the proposed architecture for decision-making.

3.1. MDP Formulation

We formulate the EV problem as a Markov decision process (MDP). This MDP is represented by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ represent the state and action spaces, respectively, $\mathcal{R} : \mathcal{S}\times\mathcal{A} \to \mathbb{R}$ denotes the instantaneous reward function, $\mathcal{P} : \mathcal{S}\times\mathcal{A}\times\mathcal{S} \to [0,1]$ defines the unknown state transition probability distribution (where $\mathcal{P}(s_1, a, s_2)$ represents the probability of transitioning to state s_2 when applying action a on state s_1), and $\gamma \in (0,1]$ is the discount factor (Sutton & Barto, 2018).

At each time step t, the state $\mathbf{s_t}$ consists of global and local inputs from all EVs currently present in the parking lot:

$$\mathbf{s_t} = (\mathbf{x_t^l}, \mathbf{x_t^g}) \tag{4}$$

$$x_{i,t}^l = (t_{i,t}^{\text{arr}}, t_{i,t}^{\text{dep}}, E_{i,t}^{\text{req}}) \tag{5}$$

$$x_t^g = (t, N_t), \tag{6}$$

where $t_{i,t}^{\text{arr}}$ denotes the time elapsed since the arrival of the i^{th} EV, $t_{i,t}^{\text{dep}}$ is the time remaining until the departure of the i^{th} EV, $E_{i,t}^{\text{req}}$ represents the energy required by the i^{th} EV, and N_t indicates the number of currently connected EVs.

We consider a discrete action space with two possible actions for each EV, defined as follows:

$$a_{i,t} \in \mathcal{A}, \quad \mathcal{A} = \{0, P_i^{\max}\}, \tag{7}$$

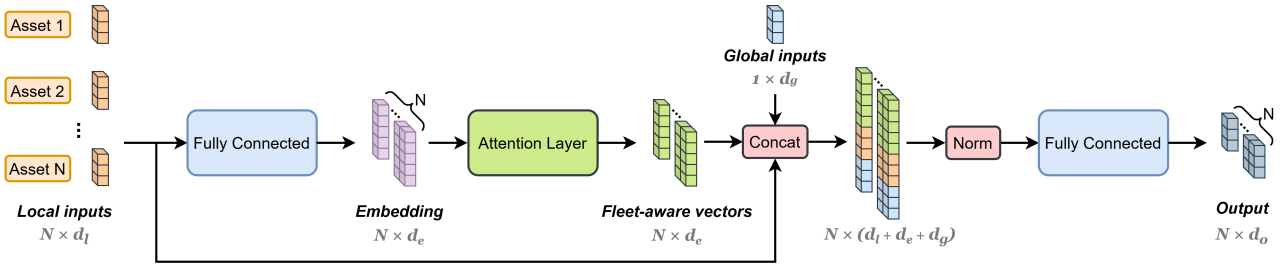


Figure 1: Our proposed architecture for centralized control of a fleet of assets.

where $P_i^{\max}$ is the maximum charging power of the i^{th} EV. In this paper, the decision-making time resolution is 1 hour.

The goal of the centralized agent is to flatten the parking load while meeting energy required by EVs. Thus, the reward function is formulated as:[1]

$$r_t = -\left(\sum_{i=1}^{N_t} a_{i,t}\right)^2 - \alpha \sum_{i=1}^{N_t} \mathbb{1}\left\{E_{i,t}^{\text{req}} > t_{i,t}^{\text{dep}} \cdot P_i^{\max}\right\}. \quad (8)$$

Since the RL agent learns a policy to maximize the reward, we use the negative of the squared sum of all EVs' actions in the reward formulation. The second term in Equation (8) corresponds to the penalty for unfinished charging. It counts the number of EVs that cannot be fully charged in their remaining connection time t_i^{dep} and penalizes the agent proportionally, using a penalty factor of α. In this paper, α is set to $2\ N_{\max}\ (\max_i P_i^{\max})^2$, following the suggestion in (Sadeghianpourhamami et al., 2019).

A state transition probability function $\mathcal{P}$ describes system dynamics. Even though part of the system dynamics in the EV problem — specifically the state-of-charge updates — can be explicitly formulated, the overall transition function remains unknown due to uncertainty on future EV arrivals. Through interaction with the environment, this transition function is implicitly learned by the RL agent.

3.2. RL Algorithm

In this paper, we use soft actor-critic (SAC), because of its superior performance and sample efficiency (Haarnoja et al., 2018). This off-policy method learns the policy using an actor network, while the Q-function is estimated by a critic network. The actor aims to maximize both the Q-values and the entropy, with the latter encouraging further exploration of the environment. To stabilize learning and mitigate Q-value overestimation, we implement a distributional variant of SAC (which we refer to as DistSAC) in which the critic network learns a *distribution* over returns instead of a single-value return (Bellemare et al., 2017; Ma et al., 2020). More specifically, the critic network estimates the quantiles of return distributions and uses the quantile regression loss, rather than the mean squared error loss, to update its weights (Dabney et al., 2018).

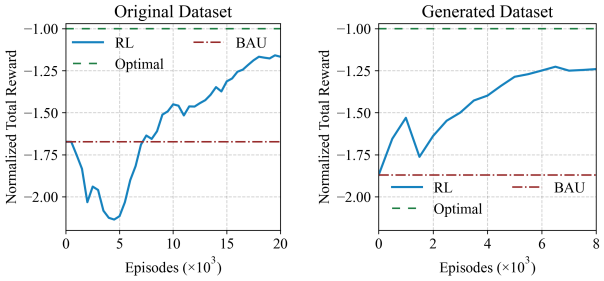
We use our proposed architecture from Figure 1 for both the actor and critic networks in aforementioned DistSAC variant — but please note that the proposed architecture could equally be integrated in other RL variants. In our DistSAC, at each time step t, $\mathbf{s_t}$ is provided as input to the actor network, which outputs a probability distribution over the actions for each EV. It is important to note that the proposed method is applicable not only to discrete actions but also to continuous ones. The critic network receives both the state $\mathbf{s_t}$ and the joint action probability vectors $\mathbf{o_t}$ as inputs. Action probability vectors are transformed into one-hot vectors using Gumbel softmax before inputting to the model. These one-hot vectors are concatenated with the local information vectors to create inputs for the embedding layer. The critic network outputs quantile values of the return distribution for each EV.

4. Results

We use two datasets to assess the performance of the proposed architecture for the EV problem: a historical dataset of Belgian residential charging sessions collected from 20 EVs, and a generated, scaled dataset of 100 EVs based on that historical data, used to evaluate the scalability of the proposed architecture. Appendix A lists further details of these datasets. RL hyperparameters are provided in detail in Appendix B. Furthermore, we benchmark our trained RL agent against (i) a business-as-usual *(BAU)* controller, where EVs start charging upon arrival, as well as (ii) the *Optimal* policy, which is obtained by solving a perfect foresight optimization problem with knowledge of all future data.

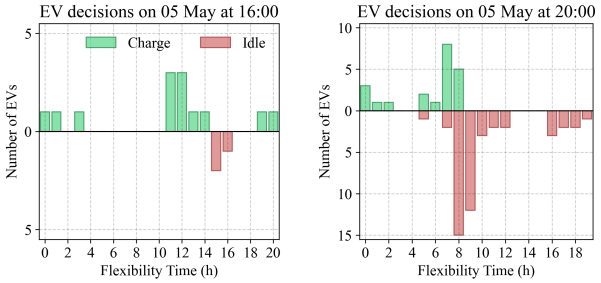
The learning curves in Figure 2 show the evolution of the RL agent's performance over the course of training on their validation sets. The trained RL agents on the original and scaled datasets outperform the BAU controller by 28% and 35.3% on their test sets, respectively. Both agents learned the policies that are only about 20% worse than the optimal

[1] Here $\mathbb{1}\{x\}$ represents the indicator function, i.e., $\mathbb{1}\{x\} = 1$ **if** x **else** 0.



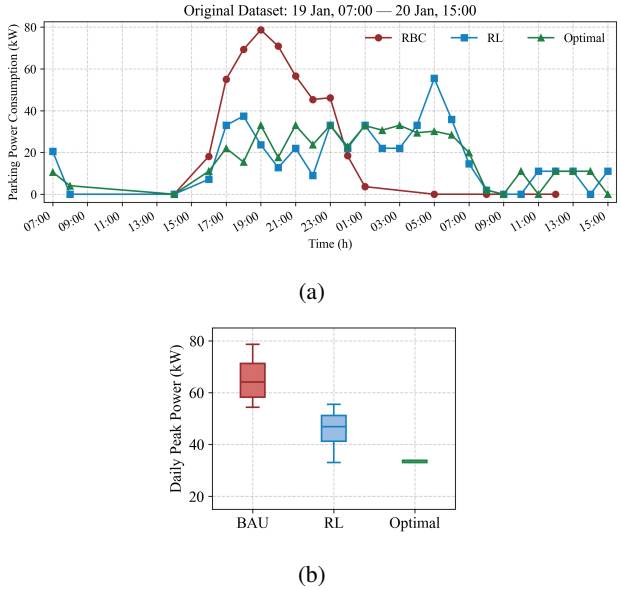
(a) (b)

Figure 2: The learning process for (a) the original dataset, and (b) the generated dataset.



(a) (b)

Figure 4: The control decisions of EVs for the scaled dataset at (a) 16:00 (b) 20:00.



(a)

(b)

Figure 3: The original dataset parking lot(a) power consumption (b) daily peak power

policies, calculated with perfect foresight. During testing, both agents were able to satisfy 100% of users, similar to the other benchmarks. The inference runtime for agents using the original and generated datasets per day is 186 ms and 199 ms, respectively. These values for the optimization method are 4.46 s and 5.11 s, respectively. After scaling, the execution time of the proposed method increased by 7%, compared to 14.6% for the optimization method. The experiments were conducted on a machine with an Intel Core i5 CPU (2.90 GHz) and 32 GB of RAM.

Figure 3a illustrates the impact of different controllers on the power consumption of the parking lot based on the original 20 EVs dataset. The RL agent, similar to the optimal policy, spreads the charging of EVs throughout their stay time instead of charging them immediately upon arrival, resulting in the desired flattened power consumption profile. According to Figure 3b, the trained RL agent reduces peak power by 26.9% on average compared to the BAU policy during the test days.

The results above demonstrate the effectiveness of the learned policy. Finally, we further investigate whether it learns to prioritize charging the right EVs, i.e., focuses on charging those with limited flexibility. Here, we define time flexibility as the duration by which EV charging can be delayed: it reflects the urgency of EV charging, with greater flexibility indicating a lower charging priority (Lahariya et al., 2022). Figure 4b confirms that our agent appropriately learns this concept of time flexibility. Furthermore, we also note that the agent considers time as a decisive feature to anticipate future events (implicitly learned from historic observations): in Figure 4a, the agent decides to charge almost all EVs in the parking lot even though most of them have more than 10 hours of flexibility. The reason is that the agent expects a large number of EV arrivals in the near future, and since charging the current EVs does not cause a peak, it chooses to charge them now to prevent a future peak.

5. Conclusion and Future Work

In this paper, we propose a scalable centralized RL method to simultaneously control a fleet of flexible assets. We use the attention layer in our proposed architecture to aggregate asset-specific information for each EV, enabling the agent to make decisions based not only on local information, but also on each EV's relative situation compared to others. The size of our proposed architecture is independent of the number of assets, enabling scalability to large asset pools. We implemented the proposed method for load flattening in an EV parking lot to demonstrate its performance. The results revealed that the RL agent learned a policy to jointly control multiple EVs toward a common goal, outperforming the BAU controller by 28%.

As discussed earlier, applying the proposed method to other multi-agent problems in the energy domain, such as multi-heat pump control, is part of our future work. Another possible extension of this work is to adapt the proposed method to the joint control of a fleet of heterogeneous flexible assets. Exploring transfer learning to improve the scalability of the proposed method across diverse and unseen fleets is another direction for future work.

Acknowledgements

This research was partially funded by the Flemish Government under the "Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen" programme and the Horizon Europe project BlueBird (https://bluebird-project.eu/ - grant agreement no. 101192452).

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. One of the main potential societal benefits of this work is the reduction of carbon emissions through more efficient energy usage. At the same time, real-world implementation of our proposed method may raise privacy concerns, as asset data must be shared with a centralized decision-maker.

References

Afram, A. and Janabi-Sharifi, F. Theory and applications of hvac control systems–a review of model predictive control (mpc). *Building and environment*, 72:343–355, 2014.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Bellemare, M. G., Dabney, W., and Munos, R. A distributional perspective on reinforcement learning. In *International conference on machine learning*, pp. 449–458. PMLR, 2017.

Chen, J., Augenbroe, G., and Song, X. Lighted-weighted model predictive control for hybrid ventilation operation based on clusters of neural network models. *Automation in Construction*, 89:250–265, 2018.

Claessens, B. J., Vrancx, P., and Ruelens, F. Convolutional neural networks for automatic state-time feature extraction in reinforcement learning applied to residential load control. *IEEE Transactions on Smart Grid*, 9(4):3259–3269, 2016.

Dabney, W., Rowland, M., Bellemare, M., and Munos, R. Distributional reinforcement learning with quantile regression. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Gao, Y., Shi, S., Miyata, S., and Akashi, Y. Successful application of predictive information in deep reinforcement learning control: A case study based on an office building hvac system. *Energy*, 291:130344, 2024.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. Pmlr, 2018.

Kamrani, A. S., Dini, A., Dagdougui, H., and Sheshyekani, K. Multi-agent deep reinforcement learning with online and fair optimal dispatch of ev aggregators. *Machine Learning with Applications*, 19:100620, 2025.

Lahariya, M., Sadeghianpourhamami, N., and Develder, C. Computationally efficient joint coordination of multiple electric vehicle charging points using reinforcement learning. *arXiv preprint arXiv:2203.14078*, 2022.

Li, H., Han, B., Li, G., Wang, K., Xu, J., and Khan, M. W. Decentralized collaborative optimal scheduling for ev charging stations based on multi-agent reinforcement learning. *IET Generation, Transmission & Distribution*, 18(6):1172–1183, 2024.

Ma, X., Xia, L., Zhou, Z., Yang, J., and Zhao, Q. Dsac: Distributional soft actor critic for risk-sensitive reinforcement learning. *arXiv preprint arXiv:2004.14547*, 2020.

Madahi, S. S. K., Nafisi, H., Abyaneh, H. A., and Marzband, M. Co-optimization of energy losses and transformer operating costs based on smart charging algorithm for plug-in electric vehicle parking lots. *IEEE Transactions on Transportation Electrification*, 7(2):527–541, 2020.

Madahi, S. S. K., Van Puyvelde, T., Gokhale, G., Claessens, B., and Develder, C. Multi-source transfer learning in reinforcement learning-based home battery controller. In *Proceedings of the 11th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, pp. 341–345, 2024.

Nagy, A., Kazmi, H., Cheaib, F., and Driesen, J. Deep reinforcement learning for optimal control of space heating. *arXiv preprint arXiv:1805.03777*, 2018.

Orfanoudakis, S., Robu, V., Salazar, E. M., Palensky, P., and Vergara, P. P. Scalable reinforcement learning for large-scale coordination of electric vehicles using graph neural networks. *Communications Engineering*, 4(1):118, 2025.

Qiu, D., Wang, Y., Hua, W., and Strbac, G. Reinforcement learning for electric vehicle applications in power systems: A critical review. *Renewable and Sustainable Energy Reviews*, 173:113052, 2023.

Sadeghianpourhamami, N., Deleu, J., and Develder, C. Definition and evaluation of model-free coordination of electrical vehicle charging with reinforcement learning. *IEEE Transactions on Smart Grid*, 11(1):203–214, 2019.

Savino, S., Minella, T., Nagy, Z., and Capozzoli, A. A scalable demand-side energy management control strategy for large residential districts based on an attention-driven multi-agent drl approach. *Applied Energy*, 393:125993, 2025.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT Press, 2018.

Vandael, S., Claessens, B., Hommelberg, M., Holvoet, T., and Deconinck, G. A scalable three-step approach for demand side management of plug-in hybrid vehicles. *IEEE Transactions on Smart Grid*, 4(2):720–728, 2012.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Yan, L., Chen, X., Chen, Y., and Wen, J. A cooperative charging control strategy for electric vehicles based on multiagent deep reinforcement learning. *IEEE Transactions on Industrial Informatics*, 18(12):8765–8775, 2022.

Yu, L., Sun, Y., Xu, Z., Shen, C., Yue, D., Jiang, T., and Guan, X. Multi-agent deep reinforcement learning for hvac control in commercial buildings. *IEEE Transactions on Smart Grid*, 12(1):407–419, 2020.

A. EV Data

We used historical EV session data collected from 20 residential EVs in Belgium between November 15, 2021, and January 21, 2022. The dataset was preprocessed to remove invalid sessions, such as those with a stay time of less than an hour or sessions without energy requirement data. Moreover, we modified sessions where the EV stay time exceeded 24 hours to ensure that EVs did not remain parked for more than 24 hours. Since EV user behavior differs between weekdays and weekends, we only considered weekday sessions. Sessions from November 15, 2021, to January 7, 2022, were used as the training set, sessions from January 10 to January 14 as the validation set, and the remaining sessions as the test set.

To evaluate the scalability of the proposed architecture, we fit distributions to the original dataset to generate the desired number of sessions. To capture the correlation between arrival and departure times, a mixture Gaussian model is employed to model the arrival and stay time distributions. The energy required distribution is modeled using a normal distribution. Figure 5 illustrates the distribution of arrival and departure times for the original and 100 generated sessions. We use 30 days of generated sessions for training the agent, 7 days for evaluation, and 7 days for testing.

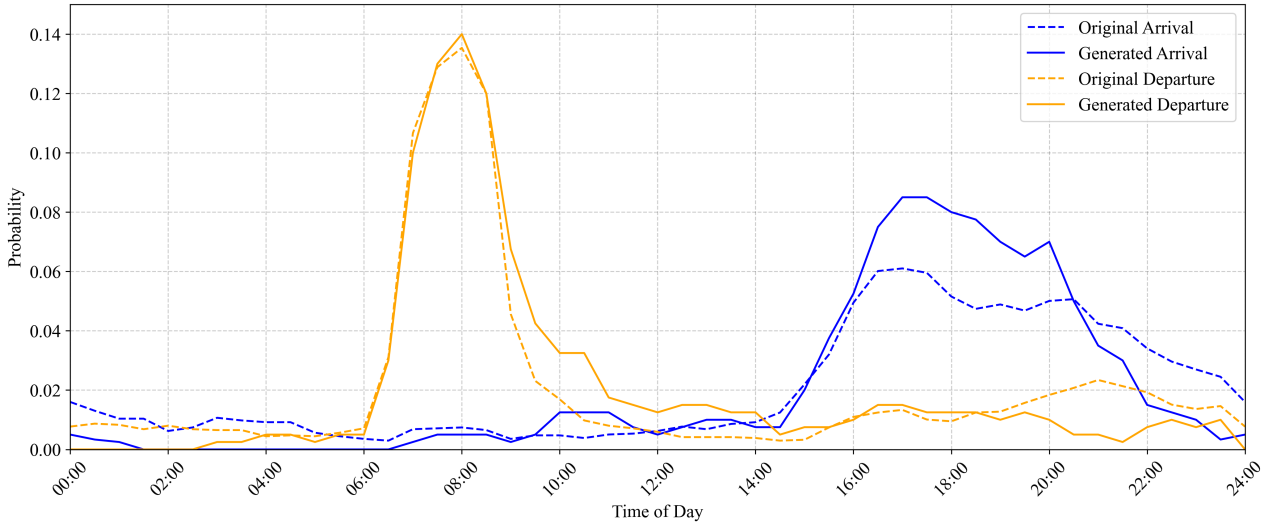


Figure 5: Historical and generated arrival and departure times.

B. Simulation Setup

Each training episode consists of a single day. The discount factor γ, the soft update factor for the target critic network τ, the experience replay buffer size, and the mini-batch size are set to 0.995, 0.1, $16\,384$, and 256, respectively. Both the actor and critic networks use the proposed architecture, in which the embedding layer and the final layer each contain two hidden layers with 256 and 128 neurons, respectively. In the actor network, the local information dimension (d_l), embedding dimension (d_e), global dimension (d_g), and output dimension (d_o) are set to 3, 16, 2, and 2, respectively. The critic network uses the same embedding and global dimensions, while $d_l = 5$ and $d_o = 20$. The learning rates of the actor and critic networks are 5×10^{-5} and 5×10^{-4}, respectively.

C. Further Results

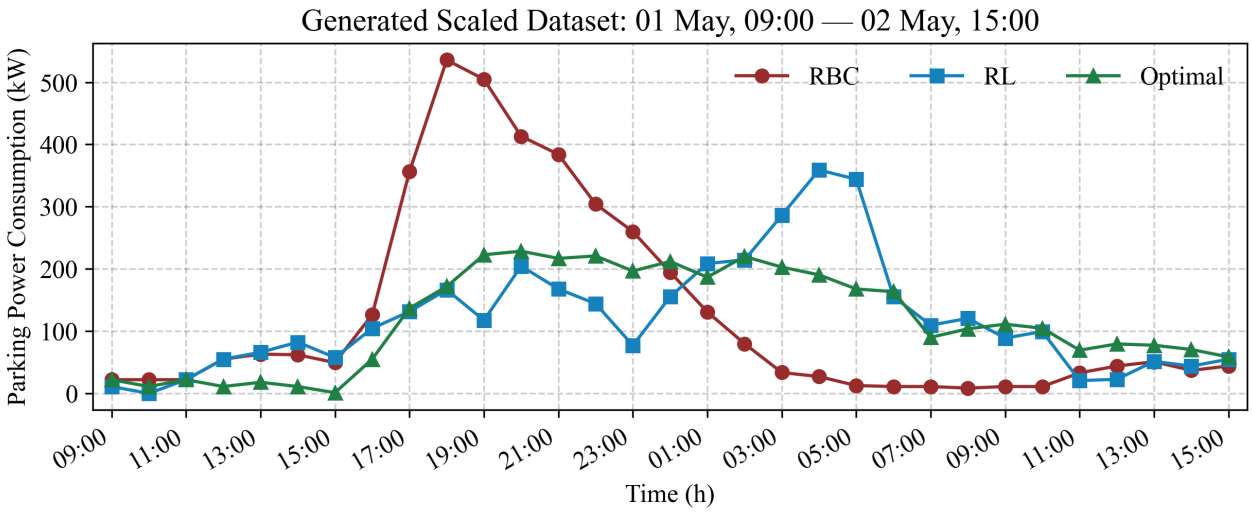


Figure 6: Parking lot power consumption for 100 EVs.

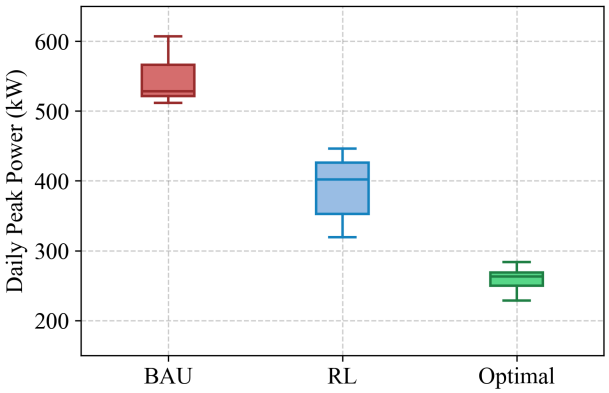
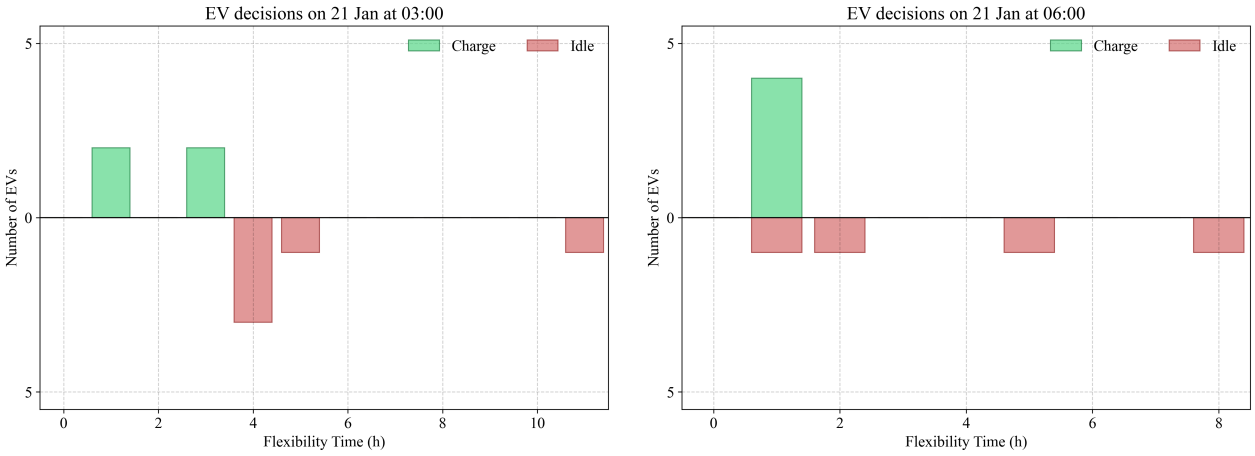


Figure 7: Parking lot daily power power for the generated dataset.



(a) (b)

Figure 8: The control decisions of EVs for the original dataset at (a) 3:00 (b) 6:00.