Fine-grained control of electric vehicle charging with policy gradient

Arno Moonens
Vrije Universiteit Brussel
Brussels, Belgium
arno.moonens@ai.vub.ac.be

Ann Nowé
Vrije Universiteit Brussel
Brussels, Belgium
ann.nowe@ai.vub.ac.be

ABSTRACT

In this work-in-progress paper, we consider how a fleet of electric vehicles can be charged with fine-grained control in order to balance the load on the electricity grid. We do this by using a policy gradient algorithm that uses the joint information of all the cars and outputs a continuous action for every agent. We observed that the reinforcement learning algorithm is able to learn a policy that reduces electricity consumption peaks and performs better than the business-as-usual case and when just using a backup controller.

KEYWORDS

reinforcement learning; policy gradient; electric vehicles; load balancing

1 INTRODUCTION

Electric vehicles (EVs) are becoming a popular alternative for traditional internal combustion engine vehicles. They are cost-competitive compared to a gasoline car [20] and the increase in charging stations can alleviate the range anxiety problem [3]. as more charging stations are installed, as well as being cost-competitive compared to a gasoline car [20]. Often, people are able to charge their car at work, where it is thus standing for a full work day [11]. In the naive case, these cars would just be charged when they arrive, which can lead to peaks in electricity consumption. These peaks can lead to voltage deviations and power losses [2]. However, as cars are connected longer than they are being charged, it is possible to shift the electricity consumption to other times of the day. In this work-in-progress paper, we apply reinforcement learning in order learn how much electricity to consume for every electric vehicle at each time step (i.e., a demand-response problem) as to minimize peaks in electricity consumption caused by the charging of multiple electric vehicles. We allow for fine-grained control, by processing the state information of every connected car and outputting the charging speed for each electric vehicle in the form of continuous actions. We do this by employing Proximal Policy Optimization [14]. We hypothesize that this can make it easier for a control algorithm to distribute the load of charging multiple electric vehicles.

2 RELATED WORK

Mets *et al.* [7] also aim at reducing peaks and flattening the load when charging multiple electric vehicles. To do this, they use quadratic programming, in which case the power can also be set as a continuous value. However, this technique assumes full knowledge of the model in order to compute the optimal charging schedule. This may not always be available in the real world or may be too complex to be used in model-known optimization. In contrast, a reinforcement learning algorithm only needs a reward as guidance towards the desired behavior.

Like our work, the algorithm described in [1] also has the objective of shaving peaks in electricity consumption. They use an aggregate-and-dispatch approach to deal with a large number of EVs. First, their states are aggregated in order to represent the power flexibility. They then use their control policy, which is learned using fitted Q-iteration, to output a single action. This action is then dispatched to a specific action for each EV using a bidding function based on the priority of each EV.

Vandael *et al.* [19] also apply the aggregate-and-dispatch approach. However, their algorithm contains 2 decision phases. In the first one, a schedule is learned for buying enough electricity in the day-ahead market. The second phase consists in deciding during a single day how much to charge the vehicles in order to satisfy the needs of the consumers and to fully use the bought electricity. The action of the latter is again dispatched to each vehicle based on priority.

Valogianni *et al.* [18] employ Q-learning for each individual EV, where there goal for each EV is to schedule the charging as to minimize the cost, while still providing enough electricity when needed. This reduces the peaks at typical hours of the day (e.g., after work) but introduces new, although less pronounced, peaks during the night. In [17] the authors aim at solving this problem by having a smart grid manager that learns to set, at each time step, the right price for every charging rate (i.e., speed of charging). This smart grid manager has the objective of having a specific aggregate demand vector (e.g., constant electricity consumption for a whole day), while the consumers still want to minimize the cost given their preferences. They do not apply reinforcement learning but instead use heuristics to find the right policy for both the smart grid manager and the consumers.

In [10], reinforcement learning is used in order to balance the electricity load when charging multiple electric vehicles. In order to deal with a large number of EV vehicles that have to be charged, they are binned based on their state. These bins are also used as input for their neural network, which also has an output (i.e., action) for every bin. The output for each bin is either to charge the cars in that bin with full power, half power or not at all. Fitted Q-iteration is used as the reinforcement learning algorithm. The input for this algorithm is a set of trajectories, which are randomly sampled. As they group cars both for input to the algorithms as when outputting actions, no fine-grained control (e.g., per car) is possible.

3 BACKGROUND

Reinforcement learning considers a setting that consists of an infinite horizon discounted Markov Decision Process (MDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$. $\mathcal{S}$ is the finite set of all the possible states of the environment and $\mathcal{A}$ is the finite set of all the possible actions that can be taken in the environment. $\mathcal{P}$ is the transition function that defines a probability for ending up in a certain state given the current state and an action: $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$. The reward function $\mathcal{R}$ returns a scalar value given the current state, the action that was taken and the next state of the environment after executing the action: $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$. $\gamma \in [0, 1]$ is the discount factor for future rewards. For n timesteps $t_1, \ldots, t_n$, an agent observes the state of an environment s_t, takes an action a_t which is executed in that environment, and receives a scalar reward R_t for taking that action in that state of the environment. The environment also transitions to a new observable state s_{t+1}. The n-step return is then defined as $G_{t:t+n} = \sum_{i=1}^{n} \gamma^i R_{t+i}$.

To select an action given a state, we define the policy function π, which outputs a probability for taking an action in a certain state: $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$. The value function for a state is defined as the expected return for starting from state s and taking actions according to π afterwards: $V^\pi(s) = \mathbb{E}_\pi\{R_t | s_t = s\}$. The action value function for a state and action is the expected return for starting from state s, taking action a and following π afterwards: $Q^\pi(s, a) = \mathbb{E}_\pi\{R_t | s_t = s, a_t = a\}$.

3.1 Policy gradient methods

In policy gradient methods [16], the policy function is parametrized by θ: $\pi_\theta(s, a) = \mathbb{P}(a|s, \theta)$. θ can be, for example, the weights of an artificial neural network. Here, the goal is to find values for θ as to maximize the expected return from each state s. This can be done by using the gradient $\mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s, a)(Q^{\pi_\theta}(s, a) - V^{\pi_\theta}(s))]$ [15]. The second part of this formula is called the advantage function, i.e. $A^{\pi_\theta} = Q^{\pi_\theta}(s, a) - V^{\pi_\theta}(s)$. In practice, the value function and action-value function are approximated.

The Proximal Policy Optimization (PPO) algorithm [14] is a policy gradient algorithm that intends to improve data efficiency, scalability and robustness to hyperparameter values. It aims to achieve this by clipping the probability ratio of the old policy and the new policy. Intuitively, this means that the new policy cannot differ too much from the old one. By clipping, beyond a certain point the difference in policies is ignored and there is no incentive anymore to move the policy even further from the old one. The probability ratio is defined as $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$. The objective itself is the following:

$$L^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t \left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t) \right], \quad (1)$$

where the clip function $\text{clip}(u, \alpha, \beta) = \max(\min(u, \beta), \alpha)$ bounds the probability ratio $r_t(\theta)$ and where epsilon is a hyperparameter, typically $\epsilon = 0.2$. We take the minimum of the unclipped and clipped objective in order to have a pessimistic bound as the final objective. The algorithm starts by using possibly multiple actors that each use policy $\pi_{\theta_{\text{old}}}$ to collect a trajectory of T time steps. Afterwards, the data of all the actors is combined and used to learn θ for K epochs with minibatches of size $M \leq NT$. After updating θ, θ_{old} is set to θ and a new iteration can start.

4 ENVIRONMENT

In this setting, the goal is to charge cars at a charging station such that the load on the electricity grid is balanced and there are no peaks in electricity consumption. The environment consists of N charging stations at which cars can be charged. Cars that arrive at a charging station are characterized by the number of time steps that the car is still present at the charging station, $\Delta t_i^{\text{depart}}$, and the amount of energy that is required, Δe_i^{req}. The latter is expressed in terms of battery percentage that is required, with $0 \leq \Delta e_i^{\text{req}}, \leq 100$. It is assumed that all cars have the same battery capacity and can be charged at the same charging rates. The car only leaves after it has no time steps left (i.e., $\Delta t_i^{\text{depart}} = 0$), even if it was already charged by that time. At each time step, each car can be charged with an amount of power in the range $[0, 25]$, again expressed in terms of battery percentage. In case of cars with different charging speeds, the environment could be adapted as to directly use the power as action, bounded for each car by the maximum charging speed for that car.

Each charging station is equipped with a backup controller that can override the action of the reinforcement learner to ensure that a car was not charged more than needed and that it will always leave with all the required energy satisfied. Using a backup controller instead of incorporating this requirement into the reward gives us 2 advantages. First, we now have the guarantee of energy satisfaction. Second, we avoid having another parameter in case we would use a weighted sum as reward. In practice, the action a_i for each car i is clipped:

$$c_i = \text{clip}(a_i^{\min}, a_i^{\max}), \quad (2)$$

$$a_i^{\min} = \max(0, \Delta e_i^{\text{req}} - (\Delta t_i^{\text{depart}} - 1) * a_i^{\max}), \quad (3)$$

$$a_i^{\max} = \min(\Delta e_i^{\text{req}}, 25), \quad (4)$$

where clip is the same function as in Section 3.1.

The state at each step of the environment is defined as

$$s = (t, \Delta t_1^{\text{depart}}, \Delta e_1^{\text{req}}, \Delta t_2^{\text{depart}}, \Delta e_2^{\text{req}}, \ldots, \Delta t_N^{\text{depart}}, \Delta e_N^{\text{req}}), \quad (5)$$

Where t is the current timestep. Intuitively, besides the timestep, the state consists of the amount of time steps that the car at each charging station is still there and the amount of energy that it has to be charged with by the time it leaves. When there is no car at a charging station i, $\Delta t_i^{\text{depart}} = 0$ and $\Delta e_i^{\text{req}} = 0$. As the order of the charging stations does not influence any part of the environment, it is chosen randomly at the start of the experiment. The action at each time step is the amount to charge the car at every charging station with:

$$a = (a_1, a_2, \ldots, a_N). \quad (6)$$

The reward given at every time step is the negative of the amount by which all the cars at the charging stations are charged:

$$R = -\left(\sum_{i=1}^{N} c_i\right)^2. \quad (7)$$

Note that the reward includes the actual consumption c_i and not the actions a_i of the reinforcement learner. For example, it will get a negative reward if the reinforcement learner does not "want" to charge the car(s) but the backup controller overrides it in order to satisfy the required energy. Thus, because of this way the backup controller works and because every car that arrives needs to be charged (i.e., cars never have a full battery upon arrival), the total reward of an episode will always be negative.

One episode consists of 168 time steps, i.e., a time step for each hour of a week. Before the episode starts, we sample using a normal distribution for each charging station and for each day at what time the car will arrive. At each time step t, first the amount of required energy and time steps until departure are updated for each car:

$$\begin{aligned} \Delta t_i^{\text{depart}} &\leftarrow \Delta t_i^{\text{depart}} - 1, \\ \Delta e_i^{\text{req}} &\leftarrow \Delta e_i^{\text{req}} - c_i. \end{aligned} \tag{8}$$

Afterwards, cars for which $\Delta t_i^{\text{depart}} = 0$ are removed and new cars are added. Then, we add cars according to the arrival times we sampled before. The amount of time steps that the car will stay, $\Delta t_i^{\text{depart}}$, and the energy that is required, Δe_i^{req}, are also sampled using a normal distribution. The means and standard deviations for the normal distributions used for sampling when cars arrive and for sampling Δe_i^{req} and $\Delta t_i^{\text{depart}}$ are specified in Appendix A. These values are loosely based on data regarding the *Charge near work* cluster described in [12], which is hypothesized to represent employees arriving at work in the morning and leaving after a work day.

5 ALGORITHM DETAILS

As reinforcement learning algorithm, the Proximal Policy Optimization algorithm, described in Section 3.1, was used. It contains a separate fully connected network for the actor and for the critic. Both networks contain 2 hidden layers with each 256 hidden units, with a tanh non-linearity applied after each layer. The network of the critic then has a single output, which is the predicted value. The network of the actor outputs for each car the mean of a gaussian distribution. There is also a parameter for the standard deviation of the gaussian distribution of each charging station. As it is not part of the actor network itself, it does not depend on the state. In order to get an action (i.e., the amount of power for the car at each charging station) to be executed in the environment, the gaussian distribution of each charging station is first sampled, then clipped between 0 and 1, and then scaled to the actual range of possible power output $[0, 25]$. This clipping and scaling is done because, due to the initialization, the neural network of the actor at first outputs values around 0. If we would just use the samples directly as actions, we would only slowly move towards the maximum action if this would be optimal. This is due to the learning rate $\alpha \ll 1$. Before adapting parameters, the chance that a high action is tried would also be low.

The other way around, the state and reward are normalized before using them in the action selection or learning process of the algorithm. This is done for stability purposes.

The critic $V_\theta(s_t)$ is learned using the squared error loss $L_t^V = \left(V_\theta(s_t) - V_t^{\text{targ}}\right)^2$, where $V_t^{\text{targ}} = G_t$. The critic is used for computing the advantage $\hat{A}_t$, which is computed by applying Generalized Advantage Estimation [13]. The values for the hyperparameters used to execute the experiments, chosen empirically, are listed in Appendix B.

6 RESULTS

Here, we show the results of our algorithm applied for a total of 10000 episodes to a setting of our environment with $N = 10$ charging stations. We compare our algorithm to 2 other policies. The business-as-usual (BAU) policy just charges the cars with full power as soon as they arrive. The backup policy does the opposite and charges the car at the last moment possible. The backup policy is the situation where we just give action 0 for every charging station to the environment, thus letting the backup controller described in Section 4 charge each car.

The graph of the total reward per episode can be seen in Figure 1. As can be seen, the algorithm converges after around 3000 episodes. The variation in reward is due to the sampling from gaussian distributions when deciding the arrival time and time until leaving. The reinforcement learning already starts with a higher reward than the BAU policy. This can be explained by fact that the reinforcement learner first does random actions and that it uses the backup controller to make sure that the required energy is satisfied. We can also see that, after 3000 episodes, the reward of the RL agent is still not always better than if we would just use the backup controller. This is caused by reward function. The backup policy only consumes at the last time steps possible for each car, while the reinforcement learner spread its consumption better but thus also accumulates penalties in more time steps. The backup policy performs better than the BAU policy because there is more variance in the time until leaving than in the arrival time.

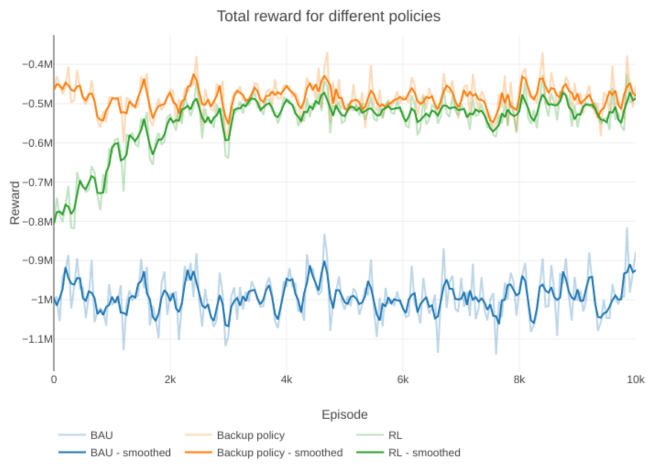

Figure 1: Total reward per episode for different policies. The values were smoothed using exponential smoothing with weight 0.4.

Figure 2 shows the amount of peak violations per episode. This means that, per episode, we count the amount of time steps in which the sum of the consumption of the charging stations exceeds a certain threshold. Here, this threshold τ is defined as being the total consumption if half of the agents would be given their maximal power. So, when there are 10 charging stations in use and the

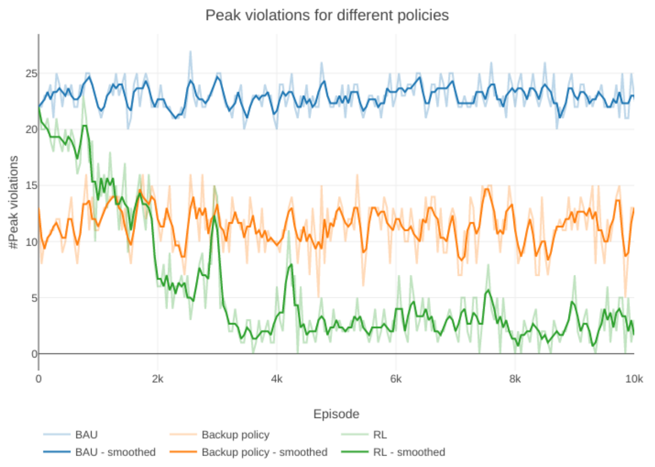

Figure 2: Amount of peak violations per episode for different policies. The values were smoothed using exponential smoothing with weight 0.4.

maximum consumption percentage is 25, $\tau = 125$. We can see that the BAU policy and the RL agent start with the same amount of peak violations. However, after some time, the RL agents seems better at avoiding peaks and performs better than the backup policy. Some peak violations may still occur due to randomness in arrival times and time until leaving and due to the RL agent sampling actions from normal distributions. We tried using the negative of the amount of peak violations directly as the reward, but this did not give as favorable results.

In Figure 3, the consumption during the last episode (10000) can be seen. Interestingly, different learned behaviors where observed for different charging stations. For some charging stations, the cars there were always immediately charged upon arrival. The cars of another part of the charging stations were just charged randomly every time step until they were fully charged. For the last group of charging stations, the reinforcement learner did not command them to be charged at all. Thus, the backup controller had to make sure that they where charged and they were charged at the last moment possible.

7 CONCLUSION

In this work-in-progress paper, we saw how we can control the charging of electric vehicles in a fine-grained manner in order to balance the load on the electricity grid. We do this by employing a policy gradient algorithm that gets information about all the cars as input and uses an artificial neural network to output an action for every car. The algorithm outperforms the business-as-usual case and is on par in terms of reward with a backup policy which charges the car at the last moment possible. However, our algorithm leads to less peaks in electricity consumption than this backup policy.

8 FUTURE WORK

In future work, we would like to gather data in order to build a more accurate simulator of the setting we want to learn in. We will also experiment with variants of the environment with more agents to see what the effect is for the reinforcement learner. Without modifications, the state and action space and thus the size of the neural network would grow exponentially with the number of agents. Thus, we will look at modifications to the current algorithm and other algorithms that are better suited to cope with a large number of agents that need to cooperate with each other, such as the algorithms presented in [5], [8] and [4]. Last, we will introduce multiple objectives such as electricity cost and influence on the battery capacity of the cars. Such problem settings raise the need for multi-objective reinforcement learning algorithms in order to find the right balance between these objectives [9].

REFERENCES

[1] BJ Claessens, Stijn Vandael, Frederik Ruelens, Klaas De Craemer, and Bart Beusen. 2013. Peak shaving of a heterogeneous cluster of residential flexibility carriers using reinforcement learning. In *IEEE PES ISGT Europe 2013*. IEEE, 1–5.

[2] Kristien Clement-Nyns, Edwin Haesen, and Johan Driesen. 2008. The impact of uncontrolled and controlled charging of plug-in hybrid electric vehicles on the distribution grid. *Proceedings of EET-2008 3rd European Ele-Drive Transportation Conference*.

[3] Jing Dong, Changzheng Liu, and Zhenhong Lin. 2014. Charging infrastructure planning for promoting battery electric vehicles: An activity-based approach using multiday travel data. *Transportation Research Part C: Emerging Technologies* 38 (2014), 44 – 55. https://doi.org/10.1016/j.trc.2013.11.001

[4] Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2017. Counterfactual Multi-Agent Policy Gradients. *CoRR* abs/1705.08926 (2017). arXiv:1705.08926

[5] Jayesh K Gupta, Maxim Egorov, and Mykel Kochenderfer. 2017. Cooperative Multi-agent Control Using Deep Reinforcement Learning. In *Autonomous Agents and Multiagent Systems*, Gita Sukthankar and Juan A Rodriguez-Aguilar (Eds.). Springer International Publishing, Cham, 66–83.

[6] Diederick P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

[7] Kevin Mets, Tom Verschueren, Wouter Haerick, Chris Develder, and Filip De Turck. 2010. Optimizing smart energy control strategies for plug-in hybrid electric vehicle charging. In *2010 IEEE/IFIP Network Operations and Management Symposium Workshops*. IEEE, 293–299.

[8] Peng Peng, Quan Yuan, Ying Wen, Yaodong Yang, Zhenkun Tang, Haitao Long, and Jun Wang. 2017. Multiagent Bidirectionally-Coordinated Nets for Learning to Play StarCraft Combat Games. *CoRR* abs/1703.10069 (2017). arXiv:1703.10069

[9] Diederik M Roijers, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. 2013. A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research* 48 (2013), 67–113.

[10] Nasrin Sadeghianpourhamami, Johannes Deleu, and Chris Develder. 2018. Definition and evaluation of model-free coordination of electrical vehicle charging with reinforcement learning. (sep 2018). arXiv:1809.10679

[11] N. Sadeghianpourhamami, N. Refa, M. Strobbe, and C. Develder. 2018. Quantitive analysis of electric vehicle flexibility: A data-driven approach. *International Journal of Electrical Power and Energy Systems* 95 (2018), 451–462.

[12] N. Sadeghianpourhamami, N. Refa, M. Strobbe, and C. Develder. 2018. Quantitive analysis of electric vehicle flexibility: A data-driven approach. *International Journal of Electrical Power and Energy Systems* 95 (2018), 451–462.

[13] John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. 2015. High-Dimensional Continuous Control Using Generalized Advantage Estimation. *CoRR* abs/1506.02438 (2015). arXiv:1506.02438

[14] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. *CoRR* abs/1707.06347 (2017). arXiv:1707.06347

[15] Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement learning - an introduction*. MIT Press.

[16] Richard S. Sutton, David Mcallester, Satinder Singh, and Yishay Mansour. 1999. Policy Gradient Methods for Reinforcement Learning with Function Approximation. *In Advances in Neural Information Processing Systems 12* (1999), 1057–1063.

[17] Konstantina Valogianni, Wolfgang Ketter, and John Collins. 2015. A multiagent approach to variable-rate electric vehicle charging coordination. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 1131–1139.

[18] Konstantina Valogianni, Wolfgang Ketter, John Collins, and Dmitry Zhdanov. 2014. Effective management of electric vehicle storage using smart charging. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*.

[19] Stijn Vandael, Bert Claessens, Damien Ernst, Tom Holvoet, and Geert Deconinck. 2015. Reinforcement learning of heuristic EV fleet charging in a day-ahead

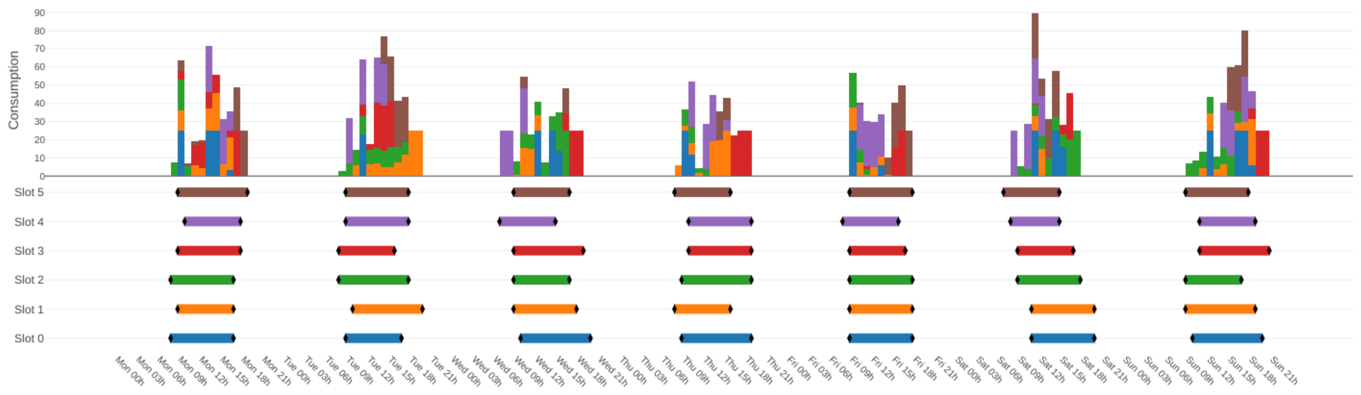

Figure 3: Electricity consumption per charging station. In the upper graph, a stacked bar chart is shown with the car consumption per timestep. In the lower graph, each horizontal bar represents the time span in which the car was connected to the charging station.

electricity market. *IEEE Transactions on Smart Grid* 6, 4 (2015), 1795–1805.

[20] Peter Weldon, Patrick Morrissey, and Margaret O'Mahony. 2018. Long-term cost of ownership comparative analysis between electric vehicles and internal combustion engine vehicles. *Sustainable Cities and Society* 39 (2018), 578–591.

A ENVIRONMENT PARAMETERS

- Number of charging stations: 10
- Gaussian distribution for arrival time:
  - Mean: 8.0
  - Standard deviation: 1.0
- Gaussian distribution for time until leaving:
  - Mean: 9.0
  - Standard deviation: 1.0
- Range of required charge percentage at arrival: $[20, 100]$
- Range of action per car: $[0, 25]$

B PPO HYPERPARAMETERS

- Number of local steps: 2048
- Batch size: 32
- Clipped Surrogate Objective epsilon ϵ: 0.2
- Number of epochs: 10
- γ: 0.99
- Generalized advantage estimation λ: 0.95
- Neural network of actor and critic:
  - Number of hidden layers: 2
  - Non-linearity: tanh
  - Number of units per layer: 256
  - learning rate α: 0.001
  - Optimizer: Adam [6]
  - Gradient norm clipping threshold: 50.0