Learn to Charge Electric Vehicles in Large Scale.

Wenyong Zhang
Department of Computer Science
The Chinese University of Hong Kong
Shatin, Hong Kong
`1155114001@link.cuhk.edu.hk`

Liangliang Hao
Department of Mechanical and Automation
The Chinese University of Hong Kong
Shatin, Hong Kong
`llhao@link.cuhk.edu.hk`

Abstract

In this project we consider the problem of dynamic resource allocation problem faced by a charging station with stochastic Electric vehicle (EV) arrivals, random renewable generation and time-varying electricity price. Similar scheduling problems can be found in many diverse fields, including packet transmission in communication system, resource allocation in cloud computing and job scheduling in a service center. In these cases, processing costs are usually time-varying, jobs' arrivals are randomized, workloads and departures are also randomized.

In addition to the randomness, the schedule problems have some particularly disturbing features. Typically, each EV has a specific deadline, if the system fails to satisfy the total workloads of the job before its deadline expires, the system will be penalized. Also, for the EV charging problem, the station has extremely large state space and action space due to hundreds of chargers, and it has to meet energy demands of EVs by their departure time.

To tackle all issues mentioned above, we created an environment in OpenAI Gym to model the operation of the charging station. Then, based on the real-world data, we used some state-of-art deep-reinforcement learning algorithms, such as Proximal Policy Optimisation (PPO) algorithm and deep deterministic policy gradient (DDPG) method, to solve the scheduling problem. Compared with the offline optimality, numerical results shows that the deep reinforcement learning based online scheduling algorithms can achieve relatively good performance.

1 Introduction

As a promising alternative to fossil-fuel car, EV mobility is expanding its shares in the auto-market at a rapid pace. In 2018, the global electric car fleet exceeded 5.1 million, up 2 million from the previous year (3). With the increase in adoption of Electric Vehicles (EVs), proper utilisation of the charging station is an emerging challenge. Operating the station without optimisation leads to some serious problems, such as overstays of EVs, failure of fully charging EVs by their departure. As a result, a deregulated charging station suffer huge revenue loss.

The scheduling problem is indeed a sequential decision process, and can be solved by dynamic programming (DP). However, conventional dynamic programming (DP) based approaches require a model on the uncertainty, *e.g.*, randomness in the transition dynamics of EVs' states(arrive time, energy demands, leave time), renewable generation and electricity price, and any methods in the field of DP are usually impractical for online application. Also, the DP method suffers the curse of dimensionality.

In this paper, we formulate the scheduling problem as a Markov Decision Process (MDP) with unknown transition probability and propose a deep reinforcement learning framework to charge EVs online. In reinforcement earning (RL), the learning agent observe a state from the environment in each step, and then chooses one action according to its policy. After taking the action, the state

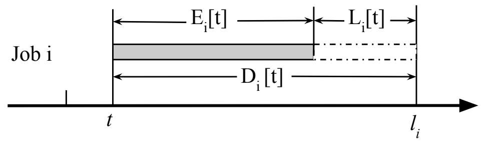

Figure 1: State of i^{th} EV at time t: E_t^i is the remaining job size, D_t^i is the remaining processing time before leave at time l_i.

transits to a new state and the agent receives a reward. The goal of the agent is to learn the best policy in order to maximize its accumulated reward.

1.1 Related Work

In recent years, several reinforcement learning based algorithms have been proposed for EV charging (8; 9; 1). In (8), authors consider the charging problem of single EV at home, whereas in this project we consider the jointed scheduling of multiple EVs in a public charging station with action space. In (1), a non-linear kernel averaging regression operator is applied to fit the action-value function. The drawback of this approach is that the determination of the kernel function and its parameters greatly affect its performance. In (9), Long Short-Term Memory (LSTM) is applied as one part of the representation network for feature extraction in order to capture time dependencies of time-series data, *e.g.*, solar generation, electricity price.

In reinforcement earning (RL), the learning agent observe a state from the environment in each step, and then chooses one action according to its policy. After taking the action, the state transits to a new state and the agent receives a reward. The goal of the agent is to learn the best policy in order to maximize its accumulated reward. For RL problem with continuous actions, the deep deterministic policy gradient (DDPG) (4) and Proximal Policy Optimisation (PPO) (6) are two state-of-art algorithms.

2 Assumptions and System Model Formulation

In Section 2.1, we describe basis problem settings and assumptions. In Section 2.2, we formulate the stochastic deadline scheduling as a Markov decision process.

2.1 Problem description and assumptions

This EV charging problem has the following settings and assumptions.

1. Time is discrete. The total scheduling horizon is $\mathcal{T} \doteq \{0, 1, \ldots, T-1\}$, indexed by t. All EVs leave at the beginning of terminal time T.

2. The station has I charging positions, label $i \in \{1, \ldots, I\}$. The continuous charging rate a_t^i of charger i is bound in the closed interval $[0, C^+]$, where C^+ is the maximum charging rate. Each charger cannot process two or more jobs simultaneously, and each EV can only be processed by one charger at any time t. But any charger can switch from processing one EV to another at different time without incurring any switching cost.

3. The arrivals of new EVs are randomized and are not known to the station. For any newly arrived EV, it reports its state $x_t^i \doteq [D_t^i, E_t^i]$ to the station, whereD_t^i is the deadline(departure time) and E_t^i is the remaining workload at period t. Without loss of generality, we use the state of the occupied EV to represent the state of position i, and $x_t^i \doteq t, 0$ denotes that the positions is empty.

   An EV leaves the station at the beginning of its deadline D. The station will be penalized $\$f(x)$ if an EV is not completed by its deadline, with x denoting the unfinished workloads. Figure 1 shows the state of an EV.

2.2 Markov decision process

We are now ready to formulate the problem as a Markov Decision Process (MDP) by introducing states, actions, transition dynamics, reward process and the objective. Our problem can be formulated as Markov Decision Process (MDP) defined by a 5 tuple $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$. $\mathcal{S}$ and $\mathcal{A}$ are the state space and action space, respectively. $P(s_{t+1} \mid s_t, a_t)$ is the transition probability from state s_t to s_{t+1} under action a_t.

1. **System state**: The state $s_t \doteq (\{x_t^i\}_{i=1}^I, g_t, c_t, \mathcal{U}_t)$, which contains states of waiting EVs $\{x_{i,t}\}_{i=1}^N$, renewable generation $g(t)$ and electricity price c_t, and EVs arrival set $\mathcal{U}_t$.
2. **Action**: The action $a_t \doteq \{a_t^i\}_{i=1}^I$ is charging rate of the station. It is feasible to charge an EV, *i.e.*, $0 \le E_t^i - a_t^i \le B$, where B is the maximum battery capacity.
3. **State transition**: At period t, state evolution of i^{th} position depends on the system state and action. If the EV occupied at the i^{th} position has not reached its deadline, *i.e.* $D_t^i > t+1$, then
$$x_{t+1}^i = [D_t^i, (E_t^i - a_t^i)^+], \ \forall\, i, t,$$
where $(x)^+ = max(0, x)$. Otherwise, the occupied EV reaches its deadline and will leave the station, *i.e.* $D_t^i \le t+1$, and new job (D', E') in the set $\mathcal{U}_t$ may enter this position:
$$x_{t+1}^i = [D', E'], \ \forall\, i, t.$$
The transition of the renewable generation g_t, electricity price c_t and the EV arrivals set $\mathcal{U}_t$ at period t are exogenous Markov processes and are independent the operations of the station.
4. **Cost**: At period t, the cost r_t consists the charging cost and the non-completion penalties, *i.e.*,
$$r_t(s_t, a_t) = (\sum_{i=1}^N (a_t^i) - g_t)^+ c_t + \sum_{i=1}^N g[(E_t^i - a_t^i)^+]\mathbb{1}[D_t^i = t+1],$$
in which $\mathbb{1}(\cdot)$ is the indicator function. The return R_t is the total discounted cost from time-step t onwards, *i.e.*, $R_t = \sum_{k=t}^T \gamma^{k-t} r(s_k, a_k)$, where $0 < \gamma < 1$. Value functions are defined to be the expected discounted reward, $V_t^\pi(s) = \mathbb{E}[R_t \mid S_t = s; \pi]$, and $Q^\pi(s,a) = \mathbb{E}[R_t \mid S_t = s, A_t = a; \pi]$.
5. **Objective**: The goal is to find an optimal policy that minimize the cumulative cost for any start state, *i.e.*, $\min_\pi \mathbb{E}[R_1 \mid S_1 = s; \pi]$.

3 Methods and Algorithms

The first algorithm we can choose is the deep deterministic policy gradient (DDPG) method (4). So why we don't want to use Deep Q-Learning? In traditional RL algorithms such as Q-learning or TD-learning we would estimate the value or utility based on different actions. Suppose we have policy π. The reward we are going to collect at the next state s' is $r(s, \pi(s), s')$. The expected reward from a stochastic policy is $\hat{r}(s, \pi(s), s') = \sum_a p_\pi(s, a, s') r(s, a, s')$. However, in our problem, the action is in a continuous space, the curse of dimensionality dominates, that is, the total number of actions increases exponentially as the democratization precision increases. But in DQN we get our action by calculating: $a_k = argmax_a Q(s_k, a)$. The number of a is too large and the optimization of a will be a great challenge. Therefore, we want to use policy gradient method which is used to deal with continuous action space. In traditional policy gradient method, the policy is a conditional distribution on the state, that is: $a_k \sim \pi_\theta(a|s_k)$. For we can better decide the charging rate, we can throw away the randomness and produce an deterministic action for continuous space action and then evaluate and update parameters accordingly, *i.e.*, $a_k = \mu(s_k|\theta^\mu)$. Hence at each time step we can just produce one action based on policy $\mu(s_k)$ and make further updates based on it. In our case, we use neural networks to produce the action of the next time step based on current state. The gradients will be calculated by first backpropagating from loss function we will define to the neural network(which is automatically taken care of by deep learning libraries Pytorch). This method is from the paper Continuous Control with Deep Reinforcement Learning, which is the published paper at ICLR 2016.

The other method we can use is Proximal Policy Optimization (PPO) algorithm (6), which is a continuous control method with important sampling and has method to choose the step size that can make the algorithm have better convergence. We will conduct some experiments using these two state-of-art algorithms and try to make some improvement based on their performance.

3.1 DDPG

For DDPG, we need to first emphasize our model structure. Since the action given the state is deterministic, the policy is a neural network(Actor) and in our problem, we formulate it as a two-layer MLP, with each layer containing 128 neurons. The output of the actor network plays the role on the input of the critic network. The training process on critics is similar to deep Q learning, but in this circumstance the action is the continuous value given by the actor network. The process on actor is as same as policy gradient method. Another issue we want to mention is that $\tau << 1$ which is the weight given to the target network. This means that the target values are constrained to change slowly, greatly improving the stability of learning. In our project, the observation $o_t \doteq (\{x_{i,t}\}_{i=1}^N, g_t, c_t, I_t, t) \in \mathbb{R}^{2N+4}$, $a_t \doteq \{a_{i,t}\}_{i=1}^N \in \mathbb{R}^N$. In our MDP setting, some of the states can not be observed, so we represent the state by o_t rather than s_t. where N is the number of chargers in the station.

3.2 PPO

For PPO method, we want to emphasize our stochastic policy which is the diagnal Gaussian policy. A multivariate Gaussian distribution (or multivariate normal distribution, if you prefer) is described by a mean vector, μ, and a covariance matrix, Σ. A diagonal Gaussian distribution is a special case where the covariance matrix only has entries on the diagonal. As a result, we can represent it by a vector. A diagonal Gaussian policy always has a neural network that maps from observations to mean actions, $\mu_\theta(s)$. There are two different ways that the covariance matrix is typically represented. The first way: There is a single vector of log standard deviations, $\log \sigma$, which is not a function of state: the $\log \sigma$ are standalone parameters. (Our implementations of A2C and PPO do it this way.) The second way: There is a neural network that maps from states to log standard deviations, $\log \sigma_\theta(s)$. It may optionally share some layers with the mean network. Then we talk about sampling:Given the mean action $\mu_\theta(s)$ and standard deviation $\sigma_\theta(s)$, and a vector z of noise from a spherical Gaussian $(z \sim \mathcal{N}(0, I))$, an action sample can be computed with

$$a = \mu_\theta(s) + \sigma_\theta(s) \odot z$$

where $\odot$ denotes the elementwise product of two vectors.

To evaluate the likelihood of each state-action pair, we can calculate the log-likelihood which is of a k -dimensional action a, for a diagonal Gaussian with mean $\mu = \mu_\theta(s)$ and standard deviation $\sigma = \sigma_\theta(s)$, is given by

$$\log \pi_\theta(a|s) = -\frac{1}{2}\left(\sum_{i=1}^{k}\left(\frac{(a_i - \mu_i)^2}{\sigma_i^2} + 2\log \sigma_i\right) + k \log 2\pi\right).$$

4 Experiments

In Section 4.1, we introduce our customised environment and the data source. Then, in Section 4.2, we use three different state-of-art algorithms to learn to charge EVs.

4.1 Environment and Data Source

Following the MDP, we create a customised environment in OpenAI Gym. The observation $o_t \doteq (\{x_t^i\}_{i=1}^I, g_t, c_t, I_t, t) \in \mathbb{R}^{2N+4}$, where x_t^i is the state of position i, g_t is the renewable generation, c_t is the electricity price, I_t the number of newly arrived EVs, and t is the time index. Note that the observation o_t is slightly different with the system state s_t, because some states are partly observable to the station.

We make use of a dataset consisting of EVs' travel timetables and charging patterns from January 1st to December 31st in 2017, recorded at public EV charging stations in UK (2). The solar generation

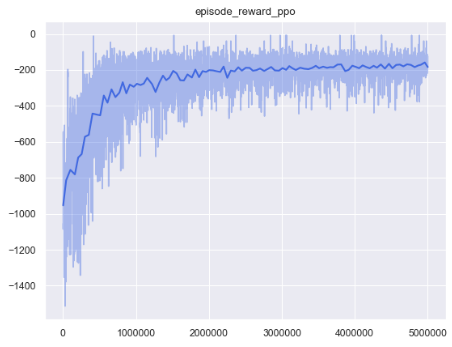

Figure 2: Learning curve of PPO.

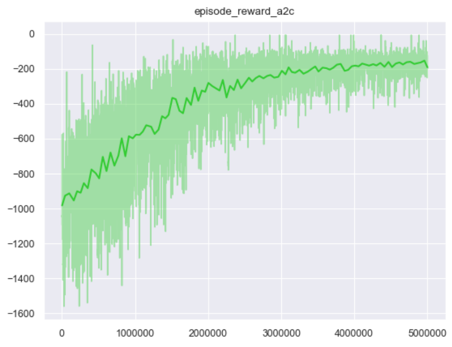

Figure 3: Learning curve of A2C.

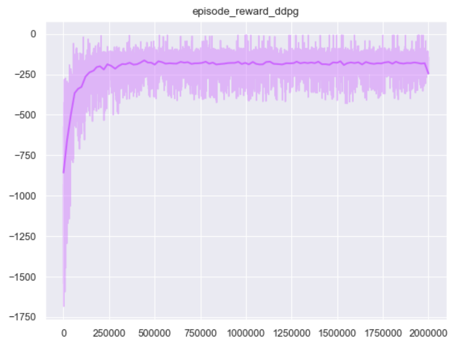

Figure 4: Learning curve of DDPG.

data is from Sheffield Solar (7) and the electricity price data is from Nord Pool (5). We consider an operation horizon with 24 hours and set T = 48, which means that 1 time slot equals $0.5h$. In total, the station has 100 chargers. We adopt the Nissan leaf 2018 model with a maximum $40kWh$ battery and $10kW/h$ charger. Any records in EV dataset with energy demand higher than $40kWh$, charging rate higher than $10kW/h$, parking time less than $0.5h$ or higher than $24h$ are deleted, resulting in $81,576$ valid records.

4.2 Results

We have run three methods, A2C, PPO, DDPG, for a station with 100 chargers, and plotted the smoothed episode rewards with respect to iteration steps in Figure 2-4. For A2C and PPO, we done training on 28 parallel environments for 5 million steps. We run DDPG on a single environment for 2 million steps. We can find that DDPG converge faster than another two methods (50k steps VS 4m steps), that's because DDPG has better sample efficiency.

We also evaluated time for learning of each method. We test 100 episodes for each method and draw the summary in Table 4.2, we can see DDPG has slightly better performance. In Table 4.2, we compute the upper bound for each episode by solving the offline optimisation problem using the CVXPY, **which assumed that all future randomness are known to the station**. Note that that any online scheduling algorithm should have no assumption like this.

5 Conclusion

In this project, we created an environment in OpenAI Gym to charge electric vehicle . Based on the real-world data, we applied some state-of-art deep-reinforcement learning algorithms, such as PPO algorithm and DDPG method, to solve the scheduling problem faced in the charging station.

Table 1: testing

	mean reward	episodes
methods		
DDPG	**-16.8**	100
A2C	-17.3	100
PPO	-17.8	100
upper bound	-2.9	–

Compared with the offline optimality, numerical results shows that the deep reinforcement learning based algorithms can achieve relatively good performance.

References

[1] A. Chiş, J. Lundén, and V. Koivunen. Reinforcement learning-based plug-in electric vehicle charging with forecasted price. *IEEE Transactions on Vehicular Technology*, 66(5):3674–3684, May 2017.

[2] DoT. Electric chargepoint analysis 2017: Public sector fasts. `https://www.gov.uk/government/statistics/electric-chargepoint-analysis-2017-public-sector-fasts`, 2017. Accessed: 2020-02-01.

[3] I.E.A. Global ev outlook 2018. Technical report, International Energy Agency, Paris, France, 2018.

[4] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[5] NP. Nord pool historical market data. `https://www.nordpoolgroup.com/historical-market-data/`, 2017. Accessed: 2020-02-01.

[6] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[7] S.U. Sheffield solar pv live. `https://www.solar.sheffield.ac.uk/pvlive/`, 2017. Accessed: 2020-02-01.

[8] K. Valogianni, W. Ketter, J. Collins, and D. Zhdanov. Effective management of electric vehicle storage using smart charging. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[9] 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*, 2018.