Ride-Hailing Service aware Electric Taxi Fleet Management using Reinforcement Learning

Paul Silva[†], YoungJoo Han[§], Young-Chon Kim*[‡], Dong-Ki Kang*[‡]
[†]*Dept. of Information Technology*, [‡]*Dept. of Computer Engineering*, *Jeonbuk National University*, Jeonju, Korea.
{paulsilvap, yckim, dongkikang}@jbnu.ac.kr
[§]*Car Cloud Development Group*, *Hyundai Motor Group*, Seoul, Korea.
yjhan@hyundai.com,

***Abstract*—Recently, the adoption of electric taxis (e-taxis) has become an essential option in many countries for the reduction of carbon emissions from large cities. However, it is generally not easy to design a sophisticated e-taxi management system due to the complex mixture of charging overheads, ride-hailing service quality for passengers, and uncertain traffic conditions. This paper proposes an *Intelligent E-taxi ride-Hailing Service* (I-EHS) controller that maximizes the satisfaction of served passengers while guaranteeing reliable charging for each e-taxi. Our controller integrates the reinforcement learning (RL) based e-taxi dispatcher and the heuristic-based e-taxi allocator, so as to derive the delicate e-taxi control with acceptable training overhead. Through the experiments based on the OpenAI-Gym framework, we show that our I-EHS controller efficiently finds the solution without prior knowledge of the traffic environment.**

***Index Terms*—Electric taxi, reinforcement learning, charging scheduling, hailing service.**

I. Introduction

Environmental concerns such as global warming and climate change caused by the emission of greenhouse gases has contributed to the adoption and development of clean energy sources and technologies. Owing to the financial support of governments that try to reduce the emission of CO_2 to the environment, many public transportation companies have deployed electric taxis (e-taxis), with the gradual replacement of gasoline-based vehicles [1]. However, despite the continuous development of battery technology and the expansion of charging infrastructures, the charging overhead is still a critical concern in the rapid adoption of e-taxis. For example, the most popular EV product, the Tesla Model3 with a battery capacity of 50kWh has a range (based on the Environmental Protection Agency, EPA) of 423km on a single charge [2]. This range is still shorter than that of conventional gasoline vehicles. Furthermore, the charging time of EVs is much longer than the refueling time of gasoline vehicles. Common EVs require more than 50 minutes to fully charge the battery even using the DC-based high-speed battery charger (50kW). If using the AC-based slow charger (5-7kW), EVs generally need 7-12 hours to fully charge the battery. This frequent and long charging slows down the adoption of EVs as e-taxis.

There are various studies for EV charging management. Li et al. [3] presented the charging scheduling for EVs based on the dynamic traffic flow. The authors designed the microscopic traffic flow model (MTFM) for spatiotemporal-based charging management. Lv et al. [4] formulated a combined charging-driving navigation (CCDN) model in order to find the optimal EVs driving on the electrified highway network (EHN). The authors presented the chronological searching approach to minimize both the traveling cost and the traveling time. Long et al. [5] proposed the real-time charging policy for plug-in EVs. They developed an ordinal optimization (OO) approach to enable scalable charging scheduling for multiple EVs. Especially, there have been a number of studies on e-taxis charging management. Yuan et al. [6] proposed the proactive partial charging (p^2Charging) in order to optimize the e-taxis charging schedule based on the dynamic passenger demand. The authors presented the mixed-integer programming (MIP) based formulation to match the passenger demand and the e-taxi supply. However, they assumed perfect knowledge of passenger behavior and e-taxi traveling patterns, which is impractical. Cilio et al. [7] presented an approach to allocate charging stations (CS) to urban areas for e-taxis in order to achieve rapid charging. They integrated the iterative clustering algorithm and the numerical optimization technique to maximize CS utilization. However, they did not consider the temporal uncertainty of e-taxis mobility and did not cover the dynamic charging scheduling problem. Ma et al. [8] investigated the characteristic of the charging problem involving CS and networked e-taxis. They designed the Stackelberg game-theoretic model to find the relationship between CS and e-taxis. However, they did not explicitly consider the service quality of served e-taxi passengers.

Recently, machine learning-based methods for EV charging have been studied to solve the uncertainty of the environment. Wan et al. [9] proposed the reinforcement learning (RL)-based data-driven method for optimal single residential EV charging. They designed the Markov Decision Process (MDP) model that penalizes the economic cost of EV charging schedules. Qian et al. [10] presented the EV charging navigation control using RL. Their proposed method can derive the optimal EV navigation that minimizes the traveling time and the charging cost under uncertain traffic conditions and charging prices. Lin et al. [11] solved the EV routing problem with time windows (EVRPTW) by using deep reinforcement learning (DRL). They integrated the Attention model with a graph embedding

* Corresponding authors: yckim@jbnu.ac.kr, dongkikang@jbnu.ac.kr

component to improve the quality of generated solutions for EVRPTW. Unfortunately, all of them require a large dataset and long learning time, due to the complex structure of neural network (NN) models and the heavy reliance on RL.

In this paper, we propose the ride-hailing service aware e-taxi fleet management using a RL method. The goal of our work is to maximize the ride-hailing service quality for e-taxi passengers while guaranteeing the charging of e-taxis on time, under uncertain environments such as irregular traffic conditions. Compared to existing works, the contribution of our paper is as follows:

- We present an explicit formulation considering both charging demands and ride-hailing service quality simultaneously. Our work is able to find the proper trade-off between reliable charging and service satisfaction.
- We adopt a Deep-Q-Network (DQN), to solve the uncertainty of traffic conditions, CS occupancy, and passenger requests that are difficult to model deterministically.
- We design an hybrid approach using both RL and an heuristic approach, for e-taxi control. In our proposed controller, the e-taxi allocator uses heuristics to map CS/passengers to e-taxis meanwhile the e-taxi dispatcher uses RL for e-taxi deployment. This structure enables to decrease the RL training overhead.

In order to investigate the performance of our work, we implement an e-taxi management simulator based on the OpenAI-Gym framework [12]. Upon a grid-shaped road network, we show that our I-EHS controller is able to derive the acceptable service quality of randomly generated passengers while ensuring reliable charging for each e-taxi.

II. Architecture of Proposed System

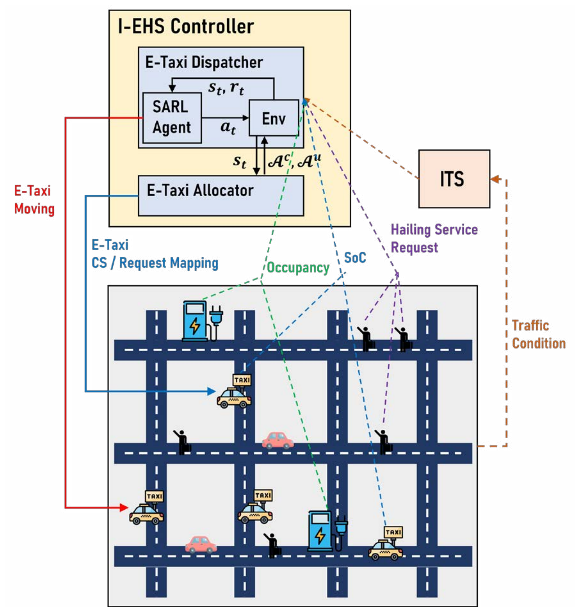


Fig. 1: Structure of Intelligent E-Taxi Ride-Hailing Service (I-EHS) Controller.

Fig. 1 shows the structure of our proposed Intelligent E-Taxi Ride-Hailing Service (I-EHS) controller interacting with the Intelligent Transportation System (ITS), for e-taxi driving and charging management. There are two modules in the I-EHS controller: the e-taxi allocator and the e-taxi dispatcher. The e-taxi allocator carries out two roles by checking the state-of-charge (SoC) of e-taxis and the ride-hailing service requests. First, if there is an e-taxi that needs to be charged, the e-taxi allocator finds the nearby CS, and lets the e-taxi move to that CS. Second, given the service requests, the e-taxi allocator assigns each passenger to each e-taxi if these have enough battery. The e-taxi dispatcher controls the moving direction of each e-taxi. The RL agent of the dispatcher finds optimal actions for e-taxi deployment, based on the current SoC of e-taxis, CS occupancy, traffic condition, and request arrival pattern. We present the role of each object as follows:

- **E-Taxi:** At the initial time step, each e-taxi departs from a certain starting point among the roads (e.g., location of e-taxi company). All the e-taxis traverse the roads to serve passengers. The e-taxi allocator in the I-EHS controller receives ride-hailing service requests from passengers and then maps the requests to the e-taxis that have enough SoC level, by using the pre-defined heuristics. The e-taxis that accept the requests, start to drive to the assigned passengers and take them to their destination. After then, the e-taxis start to traverse and wait for requests again. Whenever the e-taxi allocator finds a certain e-taxi that is running out of electricity, it sends the e-taxi to the CS. If the SoC level is too low, the e-taxi may fail to reach the CS and be stranded on the road with a dead battery.
- **Passenger:** Passengers on roads send the ride-hailing service requests to the I-EHS controller. The request contains the current location of the passenger and the destination. Note that the passengers have their distinct *patience level* which is the indicator of how long they can wait for the e-taxi to arrive. If the expected arrival time is too long, the passenger may cancel his/her request which will degrade the e-taxi company's service quality.
- **Charging Station (CS):** There is a fixed number of CS in the whole city. Each CS has a limited number of charging slots. Note that not only our controlled e-taxi tries to charge at CS, but also other EVs use the charging service of CS. The e-taxi has to wait a while if all the charging slots of the CS are fully occupied. Obviously, the distribution of slot occupancy ratio of CS has spatiotemporal characteristics [1]. An elaborate policy for CS selection is essential to prevent the e-taxi from often waiting at a busy CS.
- **Traffic Condition:** We define the traffic condition as the required driving time to pass through the road. The traffic condition of all the roads dynamically changes over time. We also assume that the I-EHS controller is able to collect the traffic condition status from the ITS, in real-time.

With respect to the continuously retrieved information (i.e., traffic condition, ride-hailing service request, SoC level, and

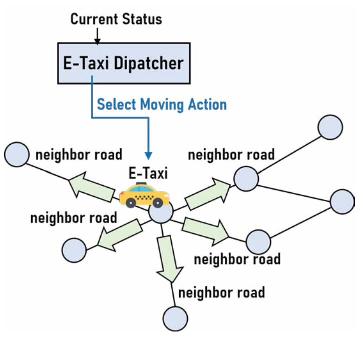


Fig. 2: Procedures of E-Taxi Dispatcher

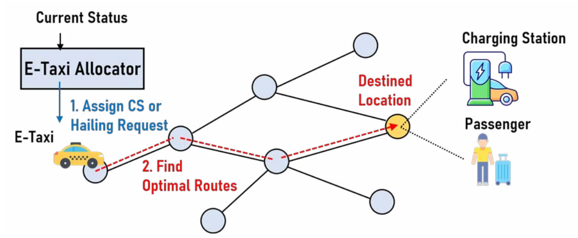


Fig. 3: Procedures of E-Taxi Allocator

CS occupancy ratio), the e-taxi dispatcher of the I-EHS controller finds the optimal e-taxi deployment. To do this, the e-taxi dispatcher adopts the method of single-agent reinforcement learning (SARL). The environment repeatedly collects the transportation status in the city and provides it as the state variable to the RL agent. In the training phase, the RL agent updates the weight parameters of the approximated Q-function and improves the policy. In the inference phase, the RL agent derives the Q-function values for action selection corresponding to the state variable. The training phase and the inference phase can be carried out alternatively, in an online manner. We unfold the procedures of the I-EHS controller in detail. Step 1) the e-taxis traverse the roads based on the policy of the RL agent. Step 2) the arbitrary passengers randomly appear on certain locations of roads, and send the ride-hailing service requests to the I-EHS controller. Step 3) the e-taxi allocator in the I-EHS controller seeks the available e-taxis to assign the requests and maps the idle e-taxis to the waiting passengers. Step 4) the RL agent in the e-taxi dispatcher selects the action based on the state variable. Step 5) the RL agent integrates the new sample data into the replay memory for continuous online learning. Fig. 2 and 3 show the e-taxi dispatcher and e-taxi allocator, respectively. The detailed routing path can be determined by using conventional routing algorithms. To do this, we exploit the *Dijkstra* algorithm [13].

III. Markov Decision Process (MDP) Model and Problem Formulation

Our main goal is to find the optimal policy of the RL agent of the I-EHS controller, so as to maximize the ride-hailing service quality for passengers. In order to design the RL agent policy improvement, we present the detailed Markov Decision Process (MDP) model as follows:

- **State:** The state vector at time step t is defined as $s_t = (e_t^1, \cdots, e_t^{N^e}, u_t^1, \cdots, u_t^{N^u}, l_t^1, \cdots, l_t^{N^l}, c_t^1, \cdots, c_t^{N^c})$ contains the state of each e-taxi e, the state of each passenger u, the state of each road l, and the state of each CS c, where $e_t^i = (p_t^i, b_t^i, w_t^i)$, $u_t^j = (\text{src}_t^j, \text{dst}_t^j, o_t^j)$. For the i-th e-taxi at time step t, p_t^i is the road position, b_t^i is the SoC level, and w_t^i is the working status: *traversing*, *serving*, and *charging*. For the j-th passenger, src_t^j is the current road position, dst_t^j is the destination's road position, and o_t^j is the passenger's status: *standing*, *served*, and *no exist*. l_t^k is the required driving time to pass the k-th road and c_t^q is the waiting time for charging at the q-th CS.
- **Action:** The action vector at time step t is defined as $a_t = (a_t^1, \cdots, a_t^{N^e})$ where $a_t^i = (\text{nl}_t^{i,1}, \cdots, \text{nl}_t^{i,\text{MAX}})$. $\text{nl}_t^{i,\text{idx}}$ represents the idx-th neighbor road of the current position p_t^i. MAX is the possible biggest number of neighbor roads for all the road positions.
- **Reward:** The RL agent gets the reward r_t corresponding to the pair of $(s_t = s, a_t = a)$. The reward r_t is represented as follows:

$$r_t = \sum_i^{N^e} v_t^i + \sum_j^{N^u} g_t^j, \tag{1}$$

$$v_t^i = \begin{cases} \alpha \cdot (L(p_t^i, \text{src}_t^j))^{-1}, & \text{if } \mathcal{A}^u(j; s_t) = i, \\ 0, & \text{if } \mathcal{A}^c(i; s_t) > 0, \end{cases} \tag{2}$$

$$g_t^j = \begin{cases} -\delta, & \text{if } \mathcal{A}^u(j; s_t) = -1 \ \forall j, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

  Here, $L(A, B)$ is the sum of required driving time to cover the roads from position A to B which is determined by the optimal routing algorithm as represented in Figure 3. $\mathcal{A}^u$ and $\mathcal{A}^c$ are output functions of the e-taxi allocator. α and δ are predefined positive coefficients.
- **State transition probability:** In this paper, we assume a stochastic environment where the next state vector s_{t+1} is influenced by not only the pair of (s_t, a_t) but also the randomness of the traffic condition and the waiting time at the CS. Obviously, the SoC level b and road position p of the e-taxi are iteratively updated every time step.

In Algorithm 1, the e-taxi allocator maps a certain CS to the e-taxi that needs to be charged. In line 03, $\underline{b}_t^i$ and η are predefined minimum SoC level and the threshold value for charging, respectively. In line 04, the e-taxi allocator finds the best CS that minimizes the sum of driving time and waiting time for charging, corresponding to the state of the associated e-taxi. ϵ, loc^q and $D(p_t^i, \text{loc}^q)$ are unit energy usage, location of the q-th CS and the physical distance from p_t^i to loc^q, respectively. In Algorithm 2, the e-taxi allocator maps a certain passenger to the traversing e-taxi. In line 04, for the j-th

Algorithm 1 CS Allocation

```
INPUT: $s_t$
OUTPUT: $\mathcal{A}^c(i;s_t),\ \forall i$
01: for $i = 1,\cdots,N^e$ do
02:     $\mathcal{A}^c(i;s_t) = -1$
03:     if $b_t^i < \underline{b}_t^i + \eta$ & $w_t^i ==$ traversing then
04:         $q^*$ = $\text{argmin}_q(L(p_t^i,\text{loc}^q) + c_t^q) : \epsilon \cdot D(p_t^i,\text{loc}^q) \le b_t^i$
05 :        $\mathcal{A}^c(i;s_t) = q^*$
06 :        $w_t^i \leftarrow$ charging
07 :    end if
08 : end for
```

Algorithm 2 Passenger Allocation

```
INPUT: $s_t$
OUTPUT: $\mathcal{A}^u(j;s_t),\ \forall j$
01: for $j = 1,\cdots,N^u$ do
02:     $\mathcal{A}^u(j;s_t) = -1$
03:     if $o_t^j == standing$ then
04 :        $i^*$ = $\text{argmin}_i L(p_t^i,\text{src}_t^j)$ :
            $\epsilon \cdot (D(p_t^i,\text{src}_t^j) + D(\text{src}_t^j,\text{dst}_t^j)) < \underline{b}_t^i + \eta,$
            $A^c(i;s_t) < 0$ & $w_t^i ==$ traversing.
05:         $A^u(j;s_t) = i^*$
06 :        $w_t^{i*} \leftarrow serving$
07 :        $o_t^j \leftarrow served$
08 :    end if
09 : end for
```

passenger, the e-taxi allocator searches the best e-taxi that achieves the minimum waiting time of the passenger. The constraint terms in line 04 prevent an e-taxi with an insufficient SoC level to accept the passenger request.

IV. Reinforcement Learning with Deep Q-Network

In order to evaluate the policy, we define the Q-function value, $Q_\pi(s,a)$ given the state s and the action a as follows:

$$Q_\pi(s,a) = \mathbb{E}_\pi[r_t + \gamma Q_\pi(s_{t+1},a_{t+1})|s_t = s, a_t = a], \quad (4)$$

where $\mathbb{E}$ is the expectation over the state-action trajectories, and $Q^* = \max_\pi(s,a)$ is the optimal Q-function value for the pair of (s,a). π is the policy for e-taxis control, that is the mapping given state s to the certain action a. γ is the discount factor used to penalize the delayed rewards. We use the deep Q-learning [14] where exploits the deep neural network (DNN) as the function approximator. Given the parameter θ, we can estimate the true Q-function value as $Q(s,a;\theta) \approx Q^*(s,a)$. We present the loss function $L(\theta)$ as follows:

$$L(\theta) = \mathbb{E}_{s,a\sim\rho}[(y - Q(s,a;\theta))^2]. \quad (5)$$

The target value is $y = \mathbb{E}_{s'}[r+\gamma\cdot\text{argmax}_{a'}Q(s',a';\theta)|(s,a)]$ where s', a', and ρ are next state, next action, and the probability distribution over pairs of (s,a), respectively. Every time step, the RL agent collects the sample transition (s,a,r,s') and inserts it to the experience replay $\mathcal{D}$. The detailed procedures for training are shown in Algorithm 3. After the e-taxi dispatcher completes the training, the I-EHS controller would be able to conduct the inference for e-taxi control. Given the state vector s_t and the parameter θ, the I-EHS controller derives the optimal driving action, CS allocation, and passenger allocation for each e-taxi in the city. The control steps are presented in Algorithm 4.

Algorithm 3 RL Agent Training

```
01: Initialize $\mathcal{D}$ and $\theta$
02 : for episode = $1,\cdots$ do
03 :    for $t = 1,\cdots T$ do
04 :        select $a_t$ by DQN using $\varepsilon$-greedy
05 :        conduct $a_t$ and get $s_{t+1}$
06 :        get $r_t$ by $\mathcal{A}^c(i;s_t), \mathcal{A}^u(j;s_t)\ \forall i,j$
07 :        $s_{t+1} \leftarrow s_t$
08 :        insert transition $(s_t,a_t,r_t,s_{t+1})$ to $\mathcal{D}$
09 :        sample minibatch of $H$ transitions from $\mathcal{D}$
10 :        get $y_h = r_h + \gamma\text{max}_a Q(s_{h+1},a;\theta)\ \forall h = 1,\cdots,H$
11 :       update $\theta$ by minimizing $\sum_{\forall h}\mathbb{E}[(y_h - Q(s_h,a_h;\theta))^2]$
12 :    end for
13 : end for
```

Algorithm 4 E-Taxi Control

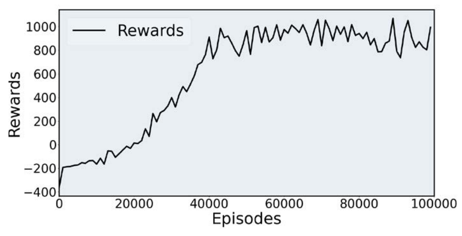
```
INPUT: $\text{s}_t, \theta$
OUTPUT : $a_t, \mathcal{A}^c(i;s_t), \mathcal{A}^u(i;s_t)\ \forall i,j$
01 : $a_t \longleftarrow \text{argmax}_a Q(s_t,a;\theta),$ by the e-taxi dispatcher
02 : $\mathcal{A}^c(i;s_t)\ \forall i \longleftarrow$ Algorithm 1, by the e-taxi allocator
03 : $\mathcal{A}^u(j;s_t)\ \forall j \longleftarrow$ Algorithm 2, by the e-taxi allocator
```

V. Evaluation

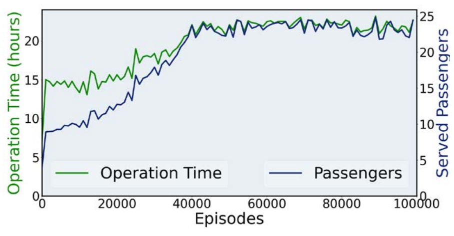
To evaluate our work, we implemented the e-taxi management simulator following design patters from OpenAI-Gym. Both the simulator and the I-EHS controller were implemented in Python. All experiments are performed on an Intel Core i7-4790 CPU @ 3.60GHZ with 16 GB RAM and Nvidia GTX 1080Ti GPU.

A. Environment Settings

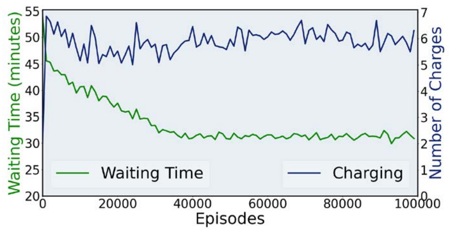
We considered a $10 * 10$ grid scenario that contains 100 intersections and 180 roads with four CS in the following grid locations [4,4], [0,9], [9,0] and [9,9]. Five passenger requests are randomly generated across the grid with different positions with the maximum patience level per passenger being of one hour. So, if no passenger request is selected during this period of time, a new set of 5 passenger requests is again randomly generated. The battery capacity for our e-taxi was set to b = 24kWh, similar to a Nissan Leaf, while the battery consumption was set as ϵ = 0.4kW. The charging rate is 20kW and the e-taxi will perform a full charge every time it visits a CS. In the case of the traffic condition, we assigned driving times dynamically bounded between 1 and 6 minutes, which updates every 10 steps. In a similar way, the waiting time at a CS follows a uniform distribution bounded between 0 and 72 minutes. Each episode in our simulation starts with the I-EHS controller having access to the state vector and concludes in the following two cases: the e-taxi fully depletes its battery



(a)



(b)



(c)

Fig. 4: Behaviour of I-EHS Controller during Training

because it failed to reach a CS on time or the maximum number of time steps is reached. Each time step is equal to 6 minutes and the maximum number of steps per episode is 240, which represents a total of 24 hours. We considered an episode to last this long, given that is common for taxis to work a double shift [15]. The E-Taxi dispatcher is trained for 100,000 episodes to learn the optimal e-taxi deployment management strategy which takes approximately 6 hours to complete. We set the learning rate at 0.001, γ at 0.9 and implemented a decaying epsilon strategy that starts at 1.0 and decreases with time all the way down to 0.05 where it remains constant until the training is completed. This favor exploration at the beginning and exploitation at the last stages. The DNN inside the DQN contains 2 hidden layers and each layer has 64 units. The parameters of the network are initialized randomly and are optimized by using RMSprop during training. The experience replay D stores up to 100000 sample transitions and the DQN samples randomly batches of size 32.

B. Experimental Results Analysis

We defined the following 4 evaluation metrics:

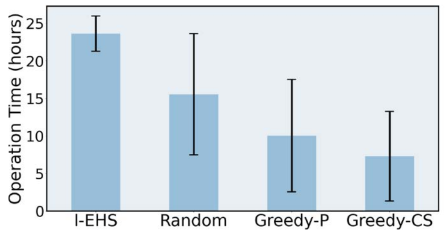
- The **e-taxi operation time** measures total time *serving* a passenger + total time *charging* at CS + total time *traversing* the roads.
- The **passenger waiting time** measures the average time that a passenger has a *standing* status during an episode.
- The number of **served passengers** measures the total number of passengers that reached *served* status during an episode.
- The number of **charges** measures the total number of times an e-taxi reached *charging* status during an episode.

Fig. 4 shows the results of the training part. We are using an exponential moving average (EMA) with an $\alpha = 0.95$ for better visualization. As seen in fig. 4a, during the first 40,000 episodes we can observe the effect of the decaying epsilon strategy on the training, then the model converges around episode 60,000. The e-taxi operation time and the number of served passengers have a strong correlation with the way the model converges, as seen in fig. 4b. At the beginning of training, the agent failed to complete a full working day given that at that point it has no idea of which was the optimal policy. Not completing the working day also meant that the number of served passengers was small. As the training continues, the agent becomes better at understanding the rules of the environment given that a better policy is being learnt. This translates into the agent being able to complete a full working day which in turns means more time for the e-taxi to serve a passenger. And, as it turns out, the number of served passengers did increase, as expected. On the other hand, in Fig. 4c we can observe the passenger waiting time and the e-taxi number of charges. At the beginning, the waiting time of the passenger was high because the agent did not understand how the traffic conditions affected its environment. As the agent becomes better and better at understanding the traveling time between the e-taxi position and the passenger position, the passenger waiting time starts to gradually decrease.

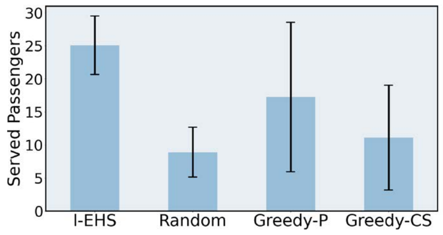
To verify the efficiency of our approach at inference time, we compared it with the following baselines for 1000 episodes:

- **Random:** the controller randomly selects the serving, charging, or moving actions for the e-taxi.
- **Greedy-CS:** the controller assigns the e-taxi to serve the closest passenger request to the center and sends it to charge to the closest CS.
- **Greedy-P:** the controller assigns the e-taxi to serve the passenger request with the minimum travel time and sends it to charge to closest CS.

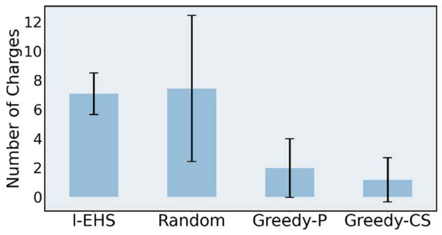
Fig. 5 shows the overall performance of the I-EHS controller against the baselines with respect to the operation time, served passengers and number of charges, which are better understood from the e-taxi perspective. All graphs show as well the standard deviation of the samples collected during the duration of the test. As seen in fig. 5a, the I-EHS controller made sure that the e-taxi completed a full working day which is not the case for the rest of the baselines. Greedy-P and Greedy-CS were not even able to complete one work shift, most of the time. Completing a double work shift allowed the e-taxi to serve a higher number of passengers during the episode but it also means that the e-taxi needed to charge more times, otherwise it would have run out of battery, as seen in the results in fig. 5b and 5c. The opposite is not necessarily true, charging more times do allow the e-taxi to be available to serve more passengers, but without a strategy, as confirmed by the random baseline, the e-taxi won't fulfill that purpose. It would move around aimlessly depleting the battery. On the other hand, for the Greedy-P and Greedy-CS baselines the number of served passengers is high compared to the Operation Time.



(a)



(b)



(c)

Fig. 5: Performance Comparison - E-Taxi Perspective

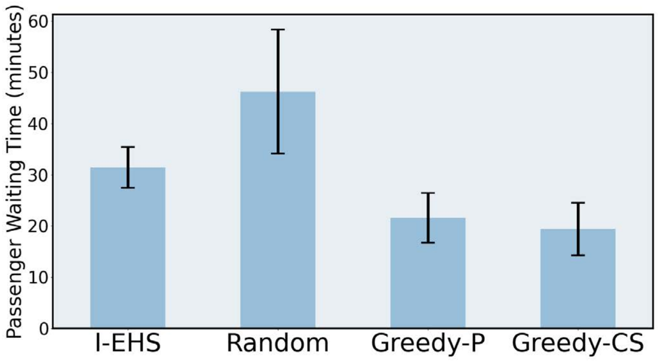


Fig. 6: Performance Comparison - Passenger Perspective

This is due to the fact that these strategies prioritize serving passengers over reliable charging which, in the view of an e-taxi operator, might not be beneficial as they end up with several units without enough battery to serve future passengers.

Fig. 6 shows the performance with respect to the passenger waiting time. With the I-EHS controller strategy, the passenger waiting time was around 30 minutes and did not perform as well as the Greedy-P and Greedy-CS baselines. These results may imply that our strategy is not optimal, but that is not the case. Our aim was always to find the proper trade-off between service satisfaction and reliable charging. By using our strategy, the I-EHS controller learnt an optimal policy that confirms that the e-taxi has enough battery to continue serving passengers. This does not happen with the Greedy-P and Greedy-CS baselines. They might be faster to take a passenger request but the overall service satisfaction is affected by the fact that later in the day, there will not be enough e-taxis to fulfill passenger requests, which is not optimal.

VI. Conclusion

In this paper, we have proposed a I-EHS controller to optimize both the quality of ride-hailing service and reliable charging in an e-taxi fleet management. We developed a control approach that integrates a heuristic-based e-taxi allocation and a DRL-based e-taxi dispatching. Simulations show the effectiveness of our proposed controller. We aimed for simplicity and hope to add more decision variables in the future to increase the realism of the environment such as historical traffic conditions, real-life road network topology and electricity price. In the same way, the implementation of a multi-agent environment is left for a future update.

Acknowledgment

This work was supported by the National Research Foundation of Korea (NSF) grant funded by the Korean Government (2021R1I1A305872911)

References

[1] Y. Yuan, D. Zhang, F. Miao, J. Chen, T. He, and S. Lin, “p^2charging: Proactive partial charging for electric taxi systems,” in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pp. 688–699, 2019.

[2] “Tesla.” https://www.tesla.com/. Online.

[3] Y. Li, X. Liu, F. Wen, X. Zhang, L. Wang, and Y. Xue, “Dynamic charging scheduling for electric vehicles considering real-time traffic flow,” in *2018 IEEE Power & Energy Society General Meeting (PESGM)*, pp. 1–5, IEEE, 2018.

[4] S. Lv, Z. Wei, G. Sun, S. Chen, and H. Zang, “Ev charging-driving navigation in electrified highway network,” in *2020 IEEE Power & Energy Society General Meeting (PESGM)*, pp. 1–5, IEEE, 2020.

[5] T. Long, Q.-S. Jia, G. Wang, and Y. Yang, “Efficient real-time ev charging scheduling via ordinal optimization,” *IEEE Transactions on Smart Grid*, vol. 12, no. 5, pp. 4029–4038, 2021.

[6] Y. Yuan, D. Zhang, F. Miao, J. Chen, T. He, and S. Lin, “p^ 2charging: proactive partial charging for electric taxi systems,” in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pp. 688–699, IEEE, 2019.

[7] L. Cilio and O. Babacan, “Allocation optimisation of rapid charging stations in large urban areas to support fully electric taxi fleets,” *Applied Energy*, vol. 295, p. 117072, 2021.

[8] K. Ma, X. Hu, J. Yang, Z. Yue, B. Yang, Z. Liu, and X. Guan, “Electric taxi charging strategy based on stackelberg game considering hotspot information,” *IEEE Transactions on Vehicular Technology*, 2022.

[9] Z. Wan, H. Li, H. He, and D. Prokhorov, “A data-driven approach for real-time residential ev charging management,” in *2018 IEEE Power & Energy Society General Meeting (PESGM)*, pp. 1–5, IEEE, 2018.

[10] T. Qian, C. Shao, X. Wang, and M. Shahidehpour, “Deep reinforcement learning for ev charging navigation by coordinating smart grid and intelligent transportation system,” *IEEE transactions on smart grid*, vol. 11, no. 2, pp. 1714–1723, 2019.

[11] B. Lin, B. Ghaddar, and J. Nathwani, “Deep reinforcement learning for the electric vehicle routing problem with time windows,” *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[12] “Openai-gym, taxi-v3.” https://gym.openai.com/envs/Taxi-v3/. Online.

[13] E. W. Dijkstra *et al.*, “A note on two problems in connexion with graphs,” *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[14] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, “Playing atari with deep reinforcement learning,” *arXiv preprint arXiv:1312.5602*, 2013.

[15] “Uitp.” https://www.uitp.org/publications/global-taxi-benchmarking-study-2019/. Online.