Hindawi
Wireless Power Transfer
Volume 2022, Article ID 9921885, 10 pages
https://doi.org/10.1155/2022/9921885

Research Article

Optimal Path Planning for Wireless Power Transfer Robot Using Area Division Deep Reinforcement Learning

Yuan Xing,[1] **Riley Young**,[1] **Giaolong Nguyen**,[1] **Maxwell Lefebvre**,[1] **Tianchi Zhao**,[2] **Haowen Pan**,[3] **and Liang Dong**[4]

[1]*Department of Engineering and Technology, University of Wisconsin-Stout, Menomonie, WI 54751, USA*
[2]*Department of Electrical and Computer Engineering, University of Arizona, Tucson, AZ 85721, USA*
[3]*Changzhou Voyage Electronics Technology LLC, Changzhou, China*
[4]*Department of Electrical and Computer Engineering, Baylor University, Waco, TX 76706, USA*

Correspondence should be addressed to Yuan Xing; xingy@uwstout.edu

Received 26 October 2021; Accepted 31 January 2022; Published 4 March 2022

Academic Editor: Narushan Pillay

This paper aims to solve the optimization problems in far-field wireless power transfer systems using deep reinforcement learning techniques. The Radio-Frequency (RF) wireless transmitter is mounted on a mobile robot, which patrols near the harvested energy-enabled Internet of Things (IoT) devices. The wireless transmitter intends to continuously cruise on the designated path in order to fairly charge all the stationary IoT devices in the shortest time. The Deep Q-Network (DQN) algorithm is applied to determine the optimal path for the robot to cruise on. When the number of IoT devices increases, the traditional DQN cannot converge to a closed-loop path or achieve the maximum reward. In order to solve these problems, an area division Deep Q-Network (AD-DQN) is invented. The algorithm can intelligently divide the complete charging field into several areas. In each area, the DQN algorithm is utilized to calculate the optimal path. After that, the segmented paths are combined to create a closed-loop path for the robot to cruise on, which can enable the robot to continuously charge all the IoT devices in the shortest time. The numerical results prove the superiority of the AD-DQN in optimizing the proposed wireless power transfer system.

1. Introduction

The wireless power transfer technique is proved to be the most effective solution to the charging problem as the number of the IoT devices grows drastically, since it is impossible to replace the batteries of all IoT devices [1]. In recent years' Consumer Electronics Show (CES), a large number of wireless power transfer products have come into consumers' sights. There are two types of wireless power transmission products: near-field and far-field. In near-field wireless power transfer, the IoT devices, which are charged by resonant inductive coupling, have to be placed very close to the wireless transmitters (less than 5 cm) [2]. In far-field wireless power transfer, the IoT devices use the electromagnetic waves from transmitters as the power resource and the effective charging distance ranges from 50 centimeters to 1.5 meters [3–5]. Compared to the near-field transmitters, the far-field wireless power transmitters can charge the IoT devices (including the mobile IoT devices) that are deployed in a larger space.

However, the far-field wireless power transfer is still in its infancy for two reasons. First, the level of power supply is very low due to the long distance between the power transmitters and the energy harvesters. In [6], the authors mentioned that the existing far-field RF energy harvesting technologies can only achieve nanowatts-level power transfer, which is too tiny to power up the high-power-consuming electronic devices. In [3], the authors investigated the RF beamforming in radiative far field for wireless power transfer. The authors demonstrated that, with beamforming techniques, the level of the energy harvesting can be boosted. However, as the distance between the transceivers increases to 1.5 meters, the amount of the harvested energy is less than 5 milliwatts, which is still not

ideal to power up the high-energy-consuming devices. Second, most of the existing wireless charging systems can only effectively charge stationary energy harvesters. In [7], a set of wireless chargers (Powercast [8]) are deployed on the square area. The Powercast transmitters can adjust the transmission strategies to optimize energy harvested at the stationary energy harvesters. In [9], the Powercast wireless charger is mounted on the moving robot. Therefore, the charger is a mobile wireless charger, which can adjust the transmission patterns of the stationary sensors while moving. However, the number of the IoT devices to be charged is too small. In order to wirelessly charge multiple IoT devices, some researchers proposed using Unmanned Aerial Vehicle (UAV) to implement the wireless power transfer [10–13]. The UAV is designed to plan the optimal path to charge the designated IoT devices. However, it is very inefficient to use UAV to charge the IoT devices, since UAV has very high power consumption and very short operational time. Installing the wireless power emitter on the UAV will further shorten the operational time of UAV.

In order to enhance the level of the energy harvesting and efficiency in charging a large number of energy-hungry IoT devices, in this paper, we assembled the wireless power transfer robot and applied deep reinforcement learning algorithm to optimize its performance. In the system, the wireless transmitter aims to find the optimal path for the wireless power transfer robot. The robot cruises on the path, which can charge each IoT device in the shortest time. DQN has been widely used to play the complicated games which have a large number of system states even when the environment information is not entirely available [14]. Lately, a lot of researchers have started to implement DQN in solving the complicated wireless communication optimization problems because the systems are very complicated and environment information is time-varying and hard to capture [15–18]. In particular, the researchers applied deep reinforcement learning to plan the optimal path for autodrive robots [19–22] and the robots can quickly converge to the optimal path. Henceforth, we found that DQN is a perfect match to solve our proposed optimization problem. However, those papers either only proposed the theoretical model or could not implement wireless power transfer functions. To the best of our knowledge, we are the first ones to implement the automatic far-field wireless power transfer system in the test field and invent a DQN algorithm to solve it. In our system, the entire test field is evenly quantified into the square spaces. The time is slotted with the same interval. We consider the relative location of the robot in the test field as the system state, while we define the direction to move in the next time slot. At the beginning of each time slot, the wireless power transfer robot generates the system state and takes it as the input to DQN. The DQN can generate the Q values for each possible action and the one with the maximum Q value is picked to guide robot's move during the current time slot.

As the number of IoT devices increases and the testing field becomes more complicated, the traditional DQN cannot generate the close-loop path for the robot to cruise on, which does not satisfy the requirement of charging every regular time interval. In order to deal with this problem, area division deep reinforcement learning is proposed in this paper. At first, the algorithm divides the whole test field into several areas. In each area, DQN is utilized to calculate the optimal path. Next, the entire path is formulated with the paths of each separated area. In this way, a closed loop is guaranteed and the numerical results prove that the calculated path is also the optimal path.

2. System Model

The symbols used in this paper and the corresponding explanations are listed in Table 1.

As shown in Figure 1, a mobile robot that carries two RF wireless power transmitters cruises on the calculated path to radiate the RF power to K nearby RF energy harvesters. Both the power transmitter and the RF power harvesters are equipped with one antenna. The power received at receiver k, $k \in \mathcal{K} = \{1, 2, \ldots, K\}$, is

$$p_k = \frac{\eta G_{\text{tx}} G_{\text{rx}} (\lambda/4\pi)^2}{l_p (L+\mu)^2} p_{\text{tx}}, \tag{1}$$

where p_{tx} is the transmit power; G_{tx} is the gain of the transmitter's antenna; G_{rx} is the gain of the receiver's antenna; L is the distance between the transmitter and harvester k; η is the rectifier efficiency; λ denotes the wavelength of the transmitted signal; l_p denotes the polarization loss; μ is the adjustable parameter due to Friis's free space equation. Since the effective charging area is critical in determining the level of energy harvesting and it is the parameter to be adjusted at the transmitter, equation (1) is reexpressed using the effective area:

$$p_k = \frac{\eta S_{\text{tx}} S_{\text{rx}} \cos\alpha}{l_p \lambda^2 (L+\mu)^2} p_{\text{tx}}, \tag{2}$$

where S_{tx} is the maximum effective transmit area; S_{rx} is the effective received area; α is the angle between the transmitter and the vertical reference line.

Since we consider the mobile energy harvesters in the system, the distance and effective charging area may vary over the time; we assume that the time is slotted and the position of any mobile device within one time slot is constant. In time slot n, the power harvested at receiver k can be denoted as

$$p_k(n) = \frac{\eta S_{\text{tx}} S_{\text{rx}} \cos\alpha(n)}{l_p \lambda^2 (L(n)+\mu)^2} p_{\text{tx}}. \tag{3}$$

For a mobile energy harvester, the power harvested in different time slots is determined by the angle between the transmitter and the vertical reference line $\alpha(n)$ together with the distance between the transmitter and the harvester $L(n)$ in the time slot.

In our model, the mobile transmitter is free to adjust the transmit angle $\alpha(n)$ and $L(n)$ as it can move around the IoT devices. We assume that the effective charging is counted only when $\alpha(n) = 0$ and $L(n) <= 45$ cm.

https://doi.org/10.1155/2022/9921885 Published online by Cambridge University Press

Table 1: Symbols and explanations.

Symbol	Explanation
K	The number of energy harvesters
η	Rectifier efficiency
G_{tx}	Gain of transmitter's antenna
G_{rx}	Gain of receiver's antenna
λ	Wavelength of transmitted signal
l_p	Polarization loss
μ	Friis's free space adjustable parameter
L	Distance between transmitter and harvester
p_{tx}	Transmit power
p_k	Received power
α	Angle between transmitter and the vertical reference line
S_{tx}	Maximum effective transmit area
S_{rx}	Effective received area
n	Time instant
$\mathbf{pos}(h, v)$	Position h and v units to left and upmost edges
$\mathbf{o}_k$	Position of kth energy harvester
$\mathbf{eff}_k$	Effective charging area for kth IoT devices
s	Present system state
s'	Next system state
$\mathbf{a}^n$	Action taken at n
T	Total time consumption
$p_{s,s'}(\mathbf{a})$	Transition probability from state s to state s' taking action $\mathbf{a}$
$w(s, \mathbf{a}, s')$	Reward function at state s taking action $\mathbf{a}$
$\mathbf{acc}_{\mathbf{o}_{k-1}}$	Indicator whether $k-1$ harvesters have been charged
ζ	Unit price for reward function
π	Optimal strategy
$Q(s, \mathbf{a})$	Cost function at state s taking action $\mathbf{a}$
γ	Reward decay
$\sigma(s', \mathbf{a})$	Learning rate for Q-learning
$\mathbf{p}_i$	Selected location for ith area
$\mathscr{W}_i$	ith area

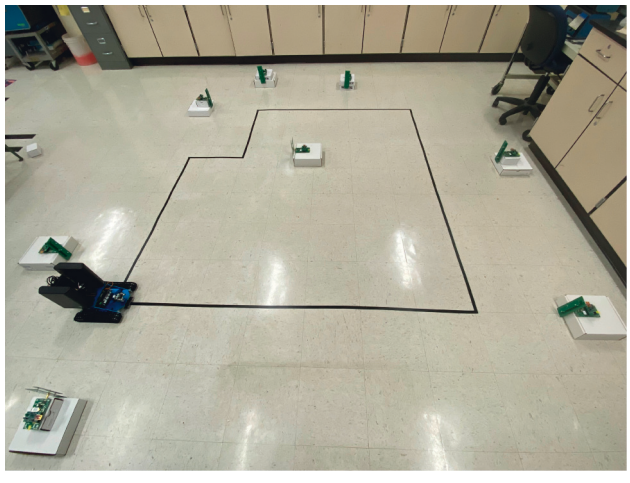

Figure 1: Mobile wireless power transmitter cruises on the calculated path to charge multiple harvested energy-enabled IoT devices.

3. Problem Formulation

In this paper, the optimization problem is formulated as a Markov Decision Process (MDP) and reinforcement learning (RL) algorithm is utilized to solve the problem. Furthermore, DQN algorithm is applied to address the large number of system states.

3.1. Problem Formulation. In order to model our optimization problem as an RL problem, we define the test field consisting of same area unit square, whose side length is 30 cm. $K = 8$ harvested energy-enabled IoT devices are deployed in the test field, whose orders are 0, 1, 2, 3, 4, 5, 6, and 7, respectively. The map is shown in Figure 2. The system state s^n at time slot n is defined as the position of a particular square where the robot is currently located at in the test field, which is specified as $s^n = \mathbf{pos}(h, v)$, where h is the distance between the present square and the leftmost edge, which is counted by the number of squares. v indicates the distance between the present square and the upmost edge, which is counted by the number of squares. For example, the No. 5 IoT device can be denoted as $\mathbf{o}_5 = \mathbf{pos}(2, 0)$. The shadow area adjacent to No. k IoT devices indicates the effective charging area for the respective IoT devices, which is denoted as $\mathbf{eff}_k$. For example, the boundary of effective charging areas for No. 6 IoT device is highlighted in red. We define the direction of movement in a particular time slot n as the actions $\mathbf{a}^n$. The set of possible actions $\mathscr{A}$ consists of 4 different $\mathscr{A} = \{\mathbf{U}, \mathbf{D}, \mathbf{L}, \mathbf{R}\}$, where $\mathbf{U}$ is moving upward one unit, $\mathbf{D}$ is moving downward one unit, $\mathbf{L}$ is moving left one unit, and $\mathbf{R}$ is moving right one unit.

Given the above, the mobile wireless charging problem can be formulated as minimizing the time duration T for the robot to complete running one loop at the same time the robot has to pass through one of the effective areas of each IoT device.

$$\mathscr{P}_1: \begin{aligned} &\underset{\{\mathbf{a}^n\}}{\text{minimize}} \quad T, \\ &\text{subject to} \quad s^0 = s^T, \\ &\qquad\qquad \exists s^n \in \mathbf{eff}_k, \forall k \in \mathscr{K}, n = 1, 2, \ldots, T. \end{aligned} \tag{4}$$

Time duration for the robot to complete running one loop is defined as T. The starting position is the same as the last position, since the robot cruises in a loop. In the loop, the robot has to pass through at least one of the effective charging areas of each IoT device.

Adapting to the different positions, the agent chooses different action at each time slot. Henceforth, we can model our proposed system as a Markov chain. In the system, we use the current position to specify a particular state s. $\mathscr{S}$ denotes the system state set. The starting state s^0 and final state s^T are the same, since the robot needs to move and return to the starting point. The MDP process can be described as the agent chooses an action $\mathbf{a}$ from $\mathscr{A}$ at a specific system state s. After that, a new system state s' will be transit into. $p_{s,s'}(\mathbf{a})$, $s, s' \in \mathscr{S}$ and $\mathbf{a} \in \mathscr{A}$, denotes the probability that system state transits from s to s' with $\mathbf{a}$.

The reward of the MDP is denoted as $w(s, \mathbf{a}, s')$, which is defined for the condition that system state transits from s to state s'. The optimization problem is formulated as reaching s^T in the fewest transmission time slots; henceforth, the reward has to be defined to motivate the mobile robot that

https://doi.org/10.1155/2022/9921885 Published online by Cambridge University Press

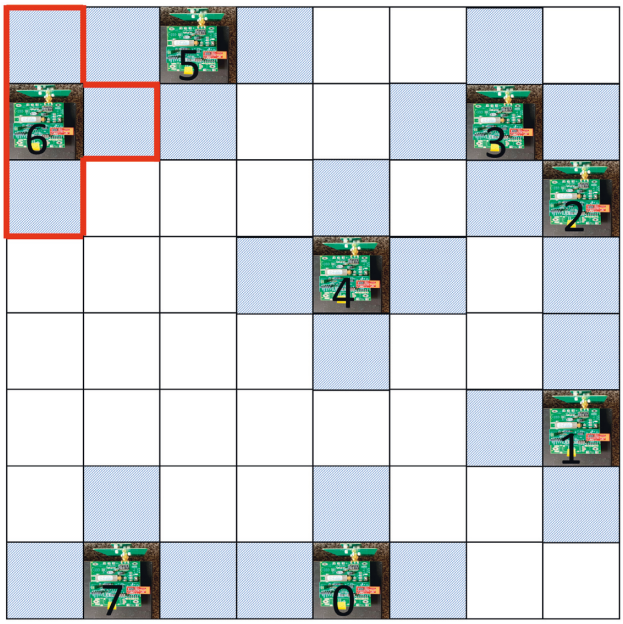

Figure 2: The entire test field consists of same space unit square. $K = 8$ harvested energy-enabled IoT devices are deployed in the test field. The shadow area adjacent to each IoT device indicates the effective charging area for the respective IoT devices. For example, the boundary of effective charging areas for No. 6 IoT device is highlighted in red.

does not repeatedly pass through any effective charging area of any IoT devices. Besides, the rewards at different positions are interconnected with each other, since the goal of the optimization is to pass through the effective charging areas of all the IoT devices. We assume that the optimal order to pass through all the IoT devices is defined as $\mathbf{o}_0, \mathbf{o}_1, \ldots, \mathbf{o}_7$. $\mathbf{o}_k = 0, 1, \ldots, 7$. Specifically, the reward function can be expressed as

$$w(s, \mathbf{a}, s') = \begin{cases} \mathbf{o}_k \zeta, & s' \in \mathbf{eff}_{\mathbf{o}_k}, \mathbf{acc}_{\mathbf{o}_{k-1}} = 1, \\ -1, & \text{otherwise.} \end{cases} \tag{5}$$

In the above equation, $\mathbf{acc}_{\mathbf{o}_{k-1}} = 1$ if the robot has already passed through as effective area of the $\mathbf{o}_{k-1}$th IoT device; and ζ denotes the unit price of the harvested energy.

As we have defined all the necessary elements for MDP, we can characterize the formulated problem as a stochastic shortest path search that starts at s^0 and ends at s^T. At each system state s, we derive the best action $\mathbf{a}^*(s)$ which can generate the maximum reward. The optimal policy sets are defined as $\pi = \{\mathbf{a}(s): s \in \mathcal{S}\}$.

3.2. Optimal Path Planning with Reinforcement Learning. If the systematic dynamics obey a specific transition probability, reinforcement learning will be the perfect match to solve the optimization problems. In this section, Q-learning [23] is first introduced to solve the proposed problem. After that, to address the large states and actions sets, the DQN algorithm [14] is utilized to determine the optimal action for each particular system state.

3.2.1. Q-Learning Method. The traditional Q-learning method is widely used to solve the dynamic optimization problem provided that the number of the system states is moderate. Corresponding to each particular system state, the best action can be determined to generate the highest reward.

$Q(s, \mathbf{a})$ denotes the cost function, which uses a numerical value to describe the cost of taking action $\mathbf{a}$ at state s. At the beginning of the algorithm, all the cost function is zero since no action has ever been taken to generate any consequence $Q(s, \mathbf{a}) = 0$. All the Q values are saved in the Q table. Only one cost function is updated in each time slot as the action is taken and the reward function is calculated. The cost function is updated as

$$\begin{aligned} Q(s, \mathbf{a}) = {} & (1 - \sigma(s, \mathbf{a}))Q(s, \mathbf{a}) \\ & + \sigma(s, \mathbf{a})[w(s, \mathbf{a}, s') + \gamma f(s', \mathbf{a})], \end{aligned} \tag{6}$$

where

$$f(s', \mathbf{a}) = \max_{\mathbf{a} \in \mathcal{A}} Q(s', \mathbf{a}). \tag{7}$$

The learning rate is defined as $\sigma(s', \mathbf{a})$.

When the algorithm initializes, the Q table is empty since no exploration has been made to obtain any useful cost function to fill the Q table. Since the agent has no experience about the environment, the random action selection is implemented at the beginning of the algorithm. A threshold $\epsilon_c \in [0.5, 1]$ is designed to start the exploration. In each time slot, a numerical value $p \in [0, 1]$ is generated and compared with the threshold. If $p \geq \epsilon_c$, action $\mathbf{a}$ is picked as

$$\mathbf{a} = \max_{\mathbf{a} \in \mathcal{A}} Q(s, \mathbf{a}). \tag{8}$$

However, provided that $p < \epsilon_c$, an action is randomly selected from the action set $\mathcal{A}$.

After iteratively updating the value in the Q table, the Q value converges. We can calculate the best action corresponding to each action and state by

$$\pi^*(s) = \arg\max_{\mathbf{a} \in \mathcal{A}} Q^*(s, \mathbf{a}), \tag{9}$$

which corresponds to finding the optimal moving direction for each system state explored during the charging process.

3.2.2. DQN Algorithm. The increase in the number of IoT devices has led to an increase in the number of system states. Suppose that Q-learning algorithm is used; a very large Q table has to be created and the convergence speed is too slow. DQN algorithm is more compatible since there is a deep neural network in the structure that can be well trained and take immediate action to determine the best action that is going to be taken.

The deep neural network in the structure has the system state as the input and the Q value for each action is defined as the output. Henceforth, the function of the neural network is to generate the cost function for particular state and action. We can describe the cost function as $Q(s, \mathbf{a}, \theta)$, where θ is the weight on the neuron nodes in the structure. As we collect

https://doi.org/10.1155/2022/9921885 Published online by Cambridge University Press

the data when different actions are taken in different time slot, the neural network is trained to update the weight of the neural network, which can output a more precise Q value:

$$Q(s, \mathbf{a}, \theta) \approx Q^*(s, \mathbf{a}). \tag{10}$$

There are two identical neural networks existing in the structure of DQN [24]: one is called the evaluation network eval_net, and the other is called the target network target_net. Since these two deep neural networks have the same structure, multiple hidden layers are defined for each network. We use the current system state s and the next system state s' as the input to eval_net and target_net, respectively. We use $Q_e(s, \mathbf{a}, \theta)$ and $Q_t(s, \mathbf{a}, \theta')$ to define the output of two deep neural networks eval_net and target_net. In the structure, in order to update the value of the weight of neuron nodes, we only continuously train the evaluation network eval_net. The target network is not trained. It periodically duplicates the weights of the neurons from the evaluation network (i.e., $\theta' = \theta$). The loss function is described as follows, which is used to train eval_net:

$$\text{Loss}(\theta) = E\left[\left(y - Q_e(s, \mathbf{a}, \theta)\right)^2\right]. \tag{11}$$

We use y to represent the real Q value, which can be expressed as

$$y = w(s, \mathbf{a}, s') + \epsilon \max_{\mathbf{a}' \in \mathscr{A}} Q_t(s', a', \theta'). \tag{12}$$

We denote the learning rate as ϵ. The idea of backpropagation is utilized to update the weight of eval_net; as a result, the neural network is trained.

The experience reply method is utilized to improve the training effect, since it can effectively eliminate the correlation among the training data. Each single experience includes the system state s, the action $\mathbf{a}$, and the next system state s', together with the reward function $w(s, \mathbf{a}, s')$. We define the experience set as $\text{ep} = \{s, \mathbf{a}, w(s, \mathbf{a}, s'), s'\}$. In the algorithm, D individual experiences are saved and, in each training epoch, only D_s (with $D_s < D$) experiences are selected from D. As the training process is completed, target_net copies the weight of the neurons from the evaluation network (i.e., $\theta' = \theta$). D different experiences are generated from ep, while only D_s are picked to train the evaluation network eval_net. The total number of training iterations can be denoted as U. Both evaluation network and target network share the same structure, in which the deep neural networks have N_l hidden layers.

3.2.3. Dueling Double DQN. In order to leverage the performance of DQN, which can effectively select the optimal action to charge multiple harvesters in a time-varying channel conditions, we redesign the structure of the deep neural network by using Dueling Double DQN. Doubling DQN is an advanced version of DQN which can prevent the overestimating problem appearing throughout the training [24]. Dueling Double DQN can efficiently solve the overestimating problem throughout the training process. In the same training epochs, Dueling Double DQN is proved to outperform the original DQN in learning efficiency.

In traditional DQN, as shown in equation (12), the target network target_net is designed to derive the cost function for a particular system state. Nevertheless, because we do not update the weight of the target network target_net in each training epoch, the training error will increase while training, hence prolonging the training procedure. In Doubling DQN, both the target network target_net and the evaluation network eval_net are used to calculate the cost functions. We use evaluation network eval_net to calculate the best action for system state s'.

$$y = w(s, \mathbf{a}, s') + \epsilon \max_{\mathbf{a}' \in \mathscr{A}} Q_e\left(s', \arg\max_{\mathbf{a} \in \mathscr{A}} Q(s', \mathbf{a}, \theta), \theta'\right). \tag{13}$$

The latest research proves that the training error can be dramatically reduced using the Doubling DQN structure [24].

In traditional DQN, we only define the cost function Q value as the output of the deep neural network. The Dueling DQN is invented to speed up the convergence of the deep neural network by designing 2 individual streams of the output for the deep neural network. We use the output value $V(\mathbf{s}, \theta, \beta)$ to represent the first stream of the neural network. It denotes the cost function for a specific system state. We name the second stream of the output as advantage output $A(s', \mathbf{a}, \theta, \alpha)$, which is utilized to illustrate the advantage of using a specific action to a system state [25]. We define α and β as the parameters to correlate the output of two streams and the neural network. The cost function can be denoted as

$$Q(s, \mathbf{a}, \theta, \alpha, \beta) = V(s, \theta, \beta) + \left(A(s', \mathbf{a}, \theta, \alpha) - \frac{1}{|A|}\sum_{a'} A(s', \mathbf{a}, \theta, \alpha)\right). \tag{14}$$

The latest research proves that Dueling DQN can speeds up the training procedure by efficiently annihilating the additional freedom while training the deep neural network [25].

3.3. Area Division Deep Reinforcement Learning. In this paper, the optimization problem can be seen as calculating the optimal close-loop path which generates the maximum accumulated reward. However, the traditional DQN shows the difficulty converging to the optimal path because of the complicated experimental field. In order to leverage the performance of traditional DQN, we invent an AD-DQN in this paper. At first, the experimental field is divided into multiple separate parts. DQN is run on each part individually to obtain the optimal path for the robot, respectively. Finally, the entire close-loop path is formulated using the path on each part. In area division, the whole area is defined as $\mathscr{W}$. The whole area is divided at multiple specific locations. $\mathbf{p}_i \in \mathscr{P}$.

https://doi.org/10.1155/2022/9921885 Published online by Cambridge University Press

The criterion to pick $\mathbf{p}_i$ is finding the squares, which exist in more than one effective charging area of the IoT devices.

$$\begin{aligned} &\forall \mathbf{p}_i \in \mathscr{P}, \\ &\mathbf{p}_i \in \mathbf{eff}_m, \\ &\mathbf{p}_i \in \mathbf{eff}_n, \\ &m, n = 0, 1, \ldots, K-1, m \neq n. \end{aligned} \tag{15}$$

For each $\mathbf{p}_i$, we define $\mathscr{N}_i = \{\mathbf{p}_i\}$. We define set $\mathscr{K}_e = \left\{\mathbf{o}_{\arg \mathbf{p}_i \in \mathbf{eff}_j,\, j=0,1,\ldots,K-1}\right\}$. In the clockwise direction, we find that the IoT device $\mathbf{o}_i$ has the shortest distance to $\mathbf{p}_i$, and then add both $\mathbf{o}_i$ and the effective charging area of $\mathbf{o}_i$ to $\mathscr{N}_i$. The new area can be expressed as

$$\mathscr{N}_i = \mathscr{N}_i \cup \{\mathbf{o}_i\} \cup \{\mathbf{eff}_i\}. \tag{16}$$

Next, we find the IoT device having the shortest distance to the IoT device $\mathbf{o}_i$ that is just added to set $\mathscr{N}_i$, and then add both the new IoT device and the effective charging area of it to $\mathscr{N}_i$. Iteratively, we find that all the IoT devices besides the ones in $\mathscr{K}_e$ are included in one $\mathscr{N}_i$. Finally, classify all the rest squares to the nearest $\mathscr{N}_i$. $\{\mathscr{N}_i\} = \mathscr{W}$.

In each area, the DQN is run to determine the optimal path for the robot. In each area, the starting point is the same as the position of $\mathbf{p}_i$; the end point is one of the effective charging squares of the furthest IoT device from the starting point in the same area. After the optimal path is calculated for each individual area, the close-loop optimal path for the entire area can be synthesized. The algorithm is shown in Algorithm 1.

(i) Define $\mathscr{E} = \{\mathbf{eff}_k, k = 0, 1, \ldots, K-1\}$. Among $\mathscr{E}$, find all the area division points $\mathbf{p}_i$ by $\{\mathbf{p}_i\} = \{\mathbf{pos}(h, v) | \mathbf{pos}(h, v) \in \mathbf{eff}_m, \mathbf{pos}(h, v) \in \mathbf{eff}_n, m, n \in \mathscr{K}\}$

(ii) The number of area divison points is defined as $|\mathscr{P}|$.

(iii) $i = 1, \ldots, |\mathscr{P}|$. The number of the area to be divided is $|\mathscr{P}| + 1$.

(iv) $\mathscr{K}_e = \left\{\mathbf{o}_{\arg \mathbf{p}_i \in \mathbf{eff}_j,\, j=0,1,\ldots,K-1}\right\}$.

(v) **for** $i = 1, \ldots, |\mathscr{P}|$:

(vi) $\mathbf{r}_1 = \mathbf{p}_i$. $\mathbf{r}_2 = \mathbf{p}_i$.

(vii) **while** $\nexists \mathbf{o}_g \in \mathscr{N}_i, \mathbf{o}_g \in \mathscr{N}_{\mathscr{T}\setminus\{i\}}$

(viii) **if** $i <= |\mathscr{P}|$

(ix) In the clockwise direction, find the the IoT devices, that has the shortest distance to $\mathbf{r}_1$. The order of the IoT device is: $g = \text{argmin}_{\mathbf{o}_i \notin \mathscr{K}_e} |\mathbf{o}_i - \mathbf{r}_1|$. $\mathscr{N}_i$ is updated as: $\mathscr{N}_i = \mathscr{N}_i \cup \{\mathbf{o}_g\} \cup \{\mathbf{eff}_g\}$. $\mathbf{r}_1 = \mathbf{o}_g$.

(x) **else**

(xi) In the counterclockwise direction, find the the IoT devices, that has the shortest distance to $\mathbf{r}_2$. The order of the IoT device is: $g = \text{argmin}_{\mathbf{o}_i \notin \mathscr{K}_e} |\mathbf{o}_i - \mathbf{r}_2|$. $\mathscr{N}_i$ is updated as: $\mathscr{N}_i = \mathscr{N}_i \cup \{\mathbf{o}_g\} \cup \{\mathbf{eff}_g\}$. $\mathbf{r}_2 = \mathbf{o}_g$.

(xii) **en d**

(xiii) **en dw hile**

(xiv) **en d**

(xv) **for** $i = 1, \ldots, |\mathscr{P}|$:

(xvi) Define set $\mathscr{W}_i$

(xvii) $\mathscr{W}_i = \mathscr{N}_i \cup \{\mathbf{pos}(h, v) | \text{arg}_{h,v}\text{min}_{\mathbf{pos}(h,v) \in \mathscr{W}} |\mathbf{pos}(h, v) - \mathbf{o}_k|,\ k \notin \mathscr{K}_e\}$

(xviii) **en d**

(xix) **for** $i = 1, 2, \ldots, |\mathscr{P}| + 1$:

(xx) **for** $j = 1, 2, \ldots, |\mathscr{J}|$:

(xxi) The starting point is defined as $\mathbf{p}_i$. The end point is defined as $e_j \in \mathbf{eff}_c$, $j \in \mathscr{J}$.

(xxii) The weight of the neuron nodes θ are randomly generated for the eval_net and the weights are copied by target_net$\theta' = \theta$. $u = 1$. $D = d = 1$.

(xxiii) **while** $u < U$ **s** $= s^0$. $t = 1$.

(xxiv) A probability is generated as a numerical parameter $p \in [0, 1]$.

(xxv) **if** $D > 200$ **an d** $p \geq \epsilon_{ch}$

(xxvi) **a** $= \max_{\mathbf{a} \in \mathscr{A}} Q(s, \mathbf{a})$

(xxvii) **else**

(xxviii) Randomly choose the action from action set $\mathscr{A}$.

(xxix) **en d**

(xxx) **while** $s' \neq s^T$

(xxxi) The state transit into s' after taking the action. $d = d + 1$. $\text{ep}(d) = \{s, \mathbf{a}, w(s, \mathbf{a}, s'), s'\}$. Suppose D keeps unchanged if it goes over the experience pool's limitation, $d = 1$; otherwise, \$ $D = d$\$. $t = t + 1$. $s = s'$. After enough data has been collected in experience pool, eval_net is trained using D of D_s experiences. Minimize the loss function Loss(θ) using Back-propagation. target_net copies the weight from eval_net periodically.

(xxxii) **en dw hile**

(xxxiii) **en dw hile**

(xxxiv) The optimal path of the entire test field is synthesized with the optimal path in each $\mathscr{W}_i$.

4. Experimental Results

The implementation of the proposed wireless power transfer system is shown in Figure 3.

In the test field, 8 harvested energy-enabled IoT devices are placed as Figure 2 indicates. The top view of the test field can be seen as a 2D map. Henceforth, the map is modeled and inputted into the computer. Then the AD-DQN algorithm is implemented in computer using Python and the optimal charging path can be derived. At the same time, a wireless power transfer robot is assembled. Two Powercast RF power transmitters TX91501 [8] are mounted on two sides of the Raspberry Pi [26] enabled intelligent driving robot. Each transmitter is powered by 5 V power bank and continuously emits 3 Watts RF power. The infrared patrol

https://doi.org/10.1155/2022/9921885 Published online by Cambridge University Press

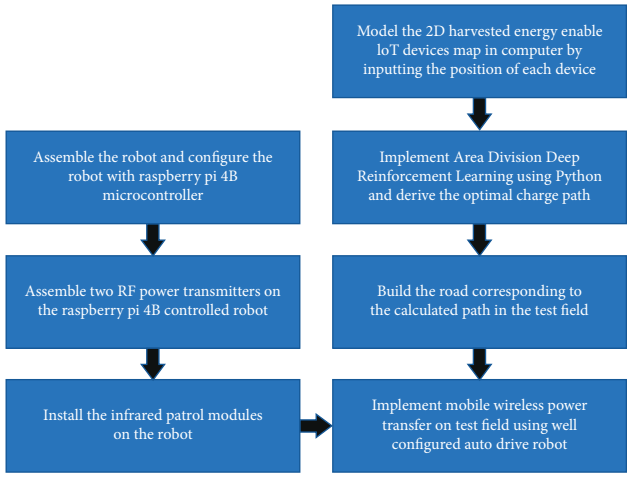

Figure 3: Flowchart of wireless power transfer implementation.

module is installed on the robot to implement the autodrive on the test field; henceforth, the robot can automatically cruise on along the path and continuously charge the multiple IoT devices, as shown in Figure 1. To the best of our knowledge, we are the first ones to implement the automatic wireless power transfer system in the test field and invent AD-DQN algorithm to design the optimal path for the wireless power transfer robot. Since we are the first ones to design and implement the mobile far-field wireless power transfer system, there is no hardware reference design we can refer to and use for validation. So the validation of our work is done in the software aspect. But referring to the flowchart, our mobile wireless power transfer system can be replicated.

For the software, we use TensorFlow 0.13.1 together with Python 3.8 in Jupyter Notebook 5.6.0 as the software simulation environment to train the AD-DQN. The number of hidden layers is 4 and each hidden layer owns 100 nodes. The learning rate is less than 0.1. The mini-batch size is 10. The learning frequency is 5. The training starting step is 200. The experience pool is greater than 20000. The exploration interval is 0.001. The target network replacement interval is greater than 100. Reward decay is 0.99.

First, different reward functions are tested for the optimal one. Reward one $\mathbf{reward}_1$ is defined using equation (5). The unit price is defined as $\zeta = 4$. Reward two $\mathbf{reward}_2$ is defined as

$$w_2(s, \mathbf{a}, s') = \begin{cases} \zeta, & s' \in \text{eff}_{\mathbf{o}_k}, \mathbf{acc}_{\mathbf{o}_{k-1}} = 1, \\ -1, & \text{otherwise}, \end{cases} \tag{17}$$

where $\zeta = 4$. Reward three $\mathbf{reward}_3$ is defined with equation (5); however, $\zeta = 2$. Two factors are observed for the performance of different rewards, which are average reward during the training and average time consumption during the training.

Based on the procedures of AD-DQN in Algorithm 1, the experimental field is divided into two areas along the only shared effective charging area for both device 2 and device 3. In area I, IoT devices 2, 3, 4, 5, and 6 are included, while in area II, IoT devices 0, 1, 2, 6, and 7 are included.

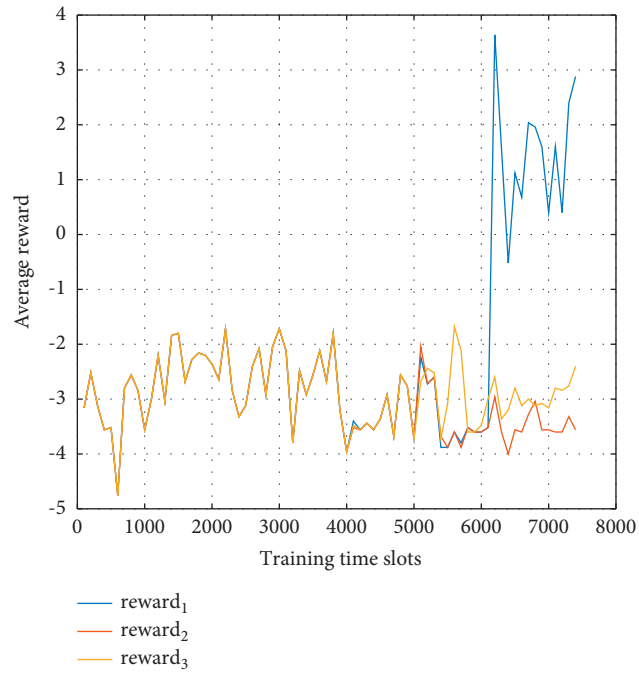

Figure 4: The average rewards of $\mathbf{reward}_1$, $\mathbf{reward}_2$, and $\mathbf{reward}_3$ versus the training episodes in area I of the experimental field.

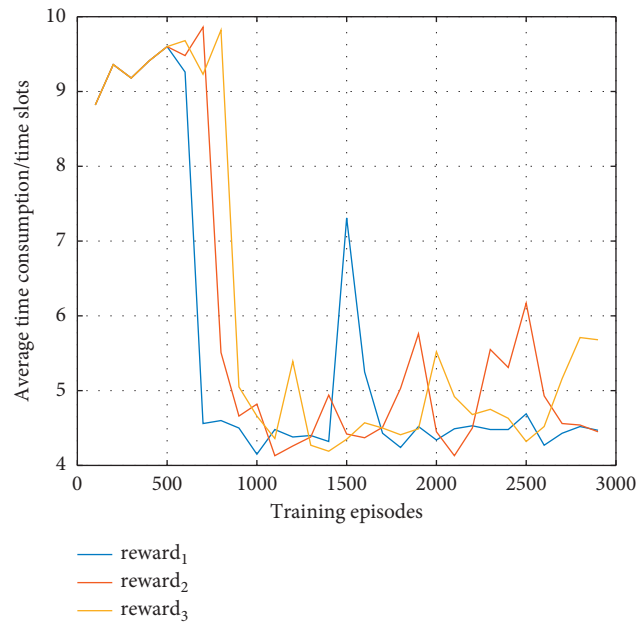

Figure 5: The average time consumption achieved by $\mathbf{reward}_1$, $\mathbf{reward}_2$, and $\mathbf{reward}_3$ versus the training episodes in area I of the experimental field.

In area I, the performances of three different rewards are compared in Figures 4 and 5.

In area II, the performances of three different rewards are compared in Figures 6 and 7.

From Figures 4 and 5, we can observe that $\mathbf{reward}_1$ is optimal. Since all three rewards perform similarly on the time consumption, $\mathbf{reward}_1$ is the highest reward among all,

https://doi.org/10.1155/2022/9921885 Published online by Cambridge University Press

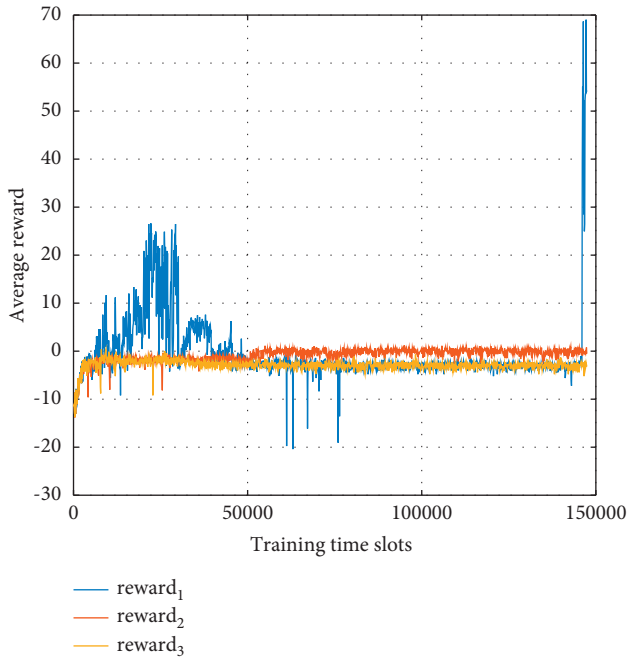

Figure 6: The average rewards of **reward**$_1$, **reward**$_2$, and **reward**$_3$ versus the training episodes in area II of the experimental field.

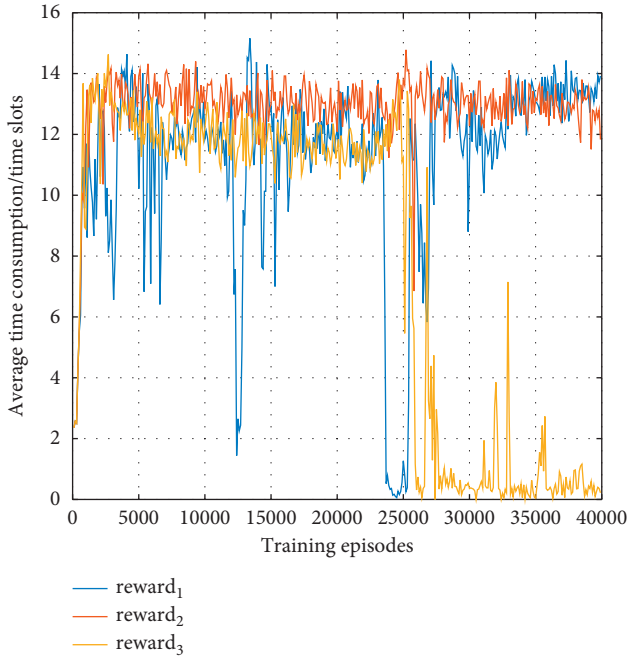

Figure 7: The average time consumption achieved by **reward**$_1$, **reward**$_2$, and **reward**$_3$ versus the training episodes in area II of the experimental field.

which means that **reward**$_1$ can effectively charge most of the IoT devices compared with the other two rewards.

From Figures 6 and 7, we can observe that **reward**$_3$ performs best on the time consumption to complete one episode; however, **reward**$_1$ is much more average reward

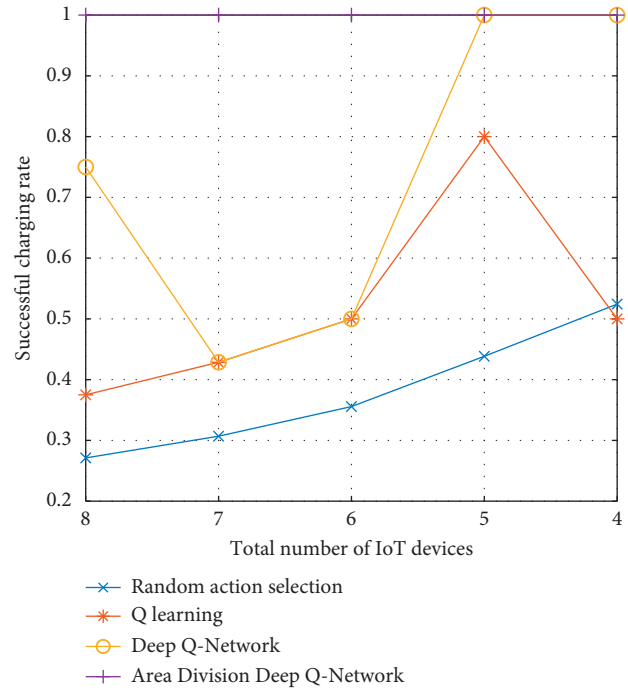

Figure 8: The effective charging rate of random action selection, Q-learning, DQN, and AD-DQN versus the total number of IoT devices.

than **reward**$_3$. That can be explained as follows: compared with **reward**$_1$, **reward**$_3$ can only effectively charge fewer number of the IoT devices.

Overall, **reward**$_1$ has optimal performance in both areas I and II; henceforth, **reward**$_1$ is used to define the reward for AD-DQN.

In Figures 8 and 9, the performances of four different algorithms are compared. The random action selection randomly selects the action in the experimental test field. Same as AD-DQN, **reward**$_1$ is used as the reward of Q-learning and DQN.

We define the successful charging rate as the number of IoT devices that can be successfully charged in one complete charging episode over the total number of the IoT devices. From Figure 8, we can observe that random action selection has the worst successful charging rate. That can be explained as follows: random action selection never converges to either suboptimal or optimal path. Q-learning has a better performance than random action selection; however, it is outperformed by the other two algorithms, since Q-learning can only deal with the simple reinforcement learning model. DQN performs better than Q-learning and random action selection; however, it is outperformed by the AD-DQN, since the rewards for different states are defined as interconnected; even the reward decay is 0.99; DQN still cannot learn the optimal solution. When the total number of the IoT devices decreases, both DQN and AD-DQN perform the same since the decrease of the number of the IoT devices degrades the interconnections between different system states. From Figure 9, we can observe that, compared with the other algorithms, AD-DQN is not the one consuming

https://doi.org/10.1155/2022/9921885 Published online by Cambridge University Press

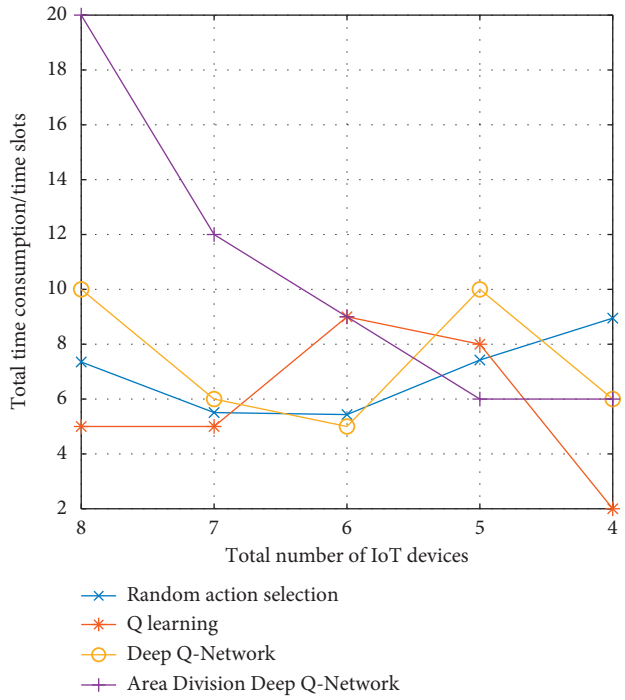

Figure 9: The average time consumption of random action selection, Q-learning, DQN, and AD-DQN versus the total number of IoT devices.

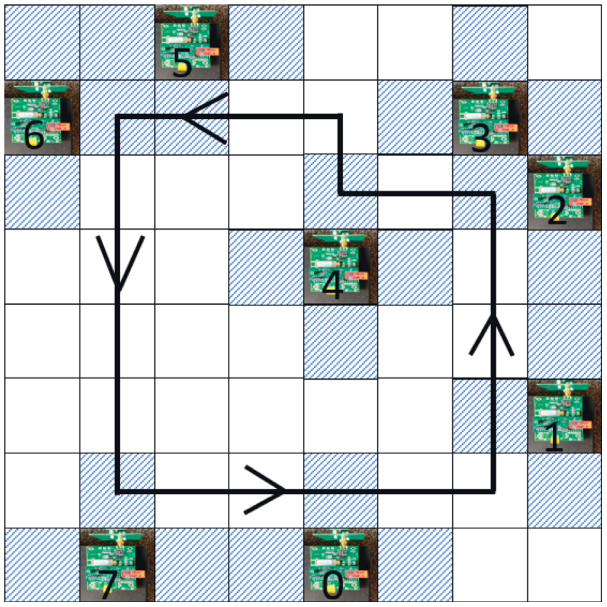

Figure 10: The optimal path determined by AD-DQN. Bold black line indicates the path for the wireless power transfer robot.

the least time slots to complete one charging episode; however, AD-DQN is still the optimal algorithm, since all the other algorithms cannot achieve 100% effective charging rate; hence they consume fewer time slots to complete one charging episode.

In Figure 10, the optimal path determined by AD-DQN is shown as the bold black line. The arrows on the path show the direction of the robot to move as we assume that the robot is regulated to cruise on the path in the counterclockwise direction. In this way, the robot can continuously charge all the IoT devices. The experimental demonstration is shown in Figure 1.

5. Conclusions

In this paper, we invent a novel deep reinforcement learning algorithm AD-DQN to determine the optimal path for the mobile wireless power transfer robot to dynamically charge the harvesting energy-enabled IoT devices. The invented algorithm can intelligently divide a large area into multiple subareas and implement the individual DQN in each area, finally synthesizing the entire path for the robot. Compared with the state of the art, the proposed algorithm can effectively charge all the IoT devices on the experimental field. The whole system can be used in a lot of application scenarios, like charging IoT devices in the dangerous area and charging medical devices.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Authors' Contributions

Yuan Xing designed the proposed wireless power transfer system, formulated the optimization problem, and proposed the innovative reinforcement learning algorithm. Riley Young, Giaolong Nguyen, and Maxwell Lefebvre designed, built, and tested the wireless power transfer robot on the wireless power transfer test field. Tianchi Zhao optimized the performance of the proposed deep reinforcement learning algorithm. Haowen Pan implemented the comparison on the system performance between the proposed algorithm and the state of the art. Liang Dong provided the theoretical support for far-field RF power transfer technique.

Acknowledgments

This work was supported by WiSys Technology Foundation under Spark Grant.

References

[1] F. Giuppi, K. Niotaki, A. Collado, and A. Georgiadis, "Challenges in energy harvesting techniques for autonomous self-powered wireless sensors," in *Proceedings of the 2013 European Microwave Conference*, pp. 854–857, Nuremberg, Germany, October 2013.

[2] A. M. Jawad, R. Nordin, S. K. Gharghan, H. M. Jawad, and M. Ismail, "Opportunities and challenges for near-field wireless power transfer: a review," *Energies*, vol. 10, no. 7, p. 1022, 2017.

[3] P. S. Yedavalli, T. Riihonen, X. Wang, and J. M. Rabaey, "Far-field rf wireless power transfer with blind adaptive

https://doi.org/10.1155/2022/9921885 Published online by Cambridge University Press

beamforming for internet of things devices," *IEEE Access*, vol. 5, pp. 1743–1752, 2017.

[4] Y. Xing and L. Dong, "Passive radio-frequency energy harvesting through wireless information transmission," in *Proceedings of the IEEE DCOSS*, pp. 73–80, Ottawa, ON, Canada, June 2017.

[5] Y. Xing, Y. Qian, and L. Dong, "A multi-armed bandit approach to wireless information and power transfer," *IEEE Communications Letters*, vol. 24, no. 4, pp. 886–889, 2020.

[6] Y. L. Lee, D. Qin, L.-C. Wang, and G. H. Sim, "6g massive radio access networks: key applications, requirements and challenges," *IEEE Open Journal of Vehicular Technology*, vol. 2, 2020.

[7] S. Nikoletseas, T. Raptis, A. Souroulagkas, and D. Tsolovos, "Wireless power transfer protocols in sensor networks: experiments and simulations," *Journal of Sensor and Actuator Networks*, vol. 6, no. 2, p. 4, 2017.

[8] Powercast, https://www.powercastco.com/documentation/, 2021.

[9] C. Lin, Y. Zhou, F. Ma, J. Deng, L. Wang, and G. Wu, "Minimizing charging delay for directional charging in wireless rechargeable sensor networks," in *Proceedings of the IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pp. 1819–1827, IEEE, Paris, France, April-May 2019.

[10] H. Yan, Y. Chen, and S.-H. Yang, "Uav-enabled wireless power transfer with base station charging and uav power consumption," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 11, pp. 12883–12896, 2020.

[11] Y. Liu, K. Xiong, Y. Lu, Q. Ni, P. Fan, and K. B. Letaief, "Uav-aided wireless power transfer and data collection in rician fading," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 10, pp. 3097–3113, 2021.

[12] W. Feng, N. Zhao, S. Ao et al., "Joint 3d trajectory design and time allocation for uav-enabled wireless power transfer networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 9, pp. 9265–9278, 2020.

[13] X. Yuan, T. Yang, Y. Hu, J. Xu, and A. Schmeink, "Trajectory design for uav-enabled multiuser wireless power transfer with nonlinear energy harvesting," *IEEE Transactions on Wireless Communications*, vol. 20, no. 2, pp. 1105–1121, 2020.

[14] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Playing atari with deep reinforcement learning," arXiv:1312.5602, 2013.

[15] Y. He, Z. Zhang, F. R. Yu et al., "Deep reinforcement learning-based optimization for cache-enabled opportunistic interference alignment wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 11, pp. 10433–10445, 2017.

[16] J. Foerster, I. A. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *Proceedings of the 29th Conference on Advances in Neural Information Processing Systems*, pp. 2137–2145, Barcelona, Spain, May 2016.

[17] Z. Xu, Y. Wang, J. Tang, J. Wang, and M. C. Gursoy, "A deep reinforcement learning based framework for power-efficient resource allocation in cloud rans," in *Proceedings of the 2017 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, Paris, France, May 2017.

[18] Y. Xing, H. Pan, B. Xu et al., "Optimal wireless information and power transfer using deep q-network," *Wireless Power Transfer*, vol. 2021, Article ID 5513509, 12 pages, 2021.

[19] C. Chen, J. Jiang, N. Lv, and S. Li, "An intelligent path planning scheme of autonomous vehicles platoon using deep reinforcement learning on network edge," *IEEE Access*, vol. 8, pp. 99059–99069, 2020.

[20] S. Wen, Y. Zhao, X. Yuan, Z. Wang, D. Zhang, and L. Manfredi, "Path planning for active slam based on deep reinforcement learning under unknown environments," *Intelligent Service Robotics*, vol. 13, pp. 1–10, 2020.

[21] S. Koh, B. Zhou, H. Fang et al., "Real-time deep reinforcement learning based vehicle navigation," *Applied Soft Computing*, vol. 96, Article ID 106694, 2020.

[22] R. Ding, F. Gao, and X. S. Shen, "3d uav trajectory design and frequency band allocation for energy-efficient and fair communication: a deep reinforcement learning approach," *IEEE Transactions on Wireless Communications*, vol. 19, no. 12, pp. 7796–7809, 2020.

[23] A. G. Barto, S. J. Bradtke, and S. P. Singh, "Learning to act using real-time dynamic programming," *Artificial Intelligence*, vol. 72, no. 1-2, pp. 81–138, 1995.

[24] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, vol. 2, p. 5p. 5, Phoenix, AZ, USA, February 2016.

[25] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, and N. De Freitas, "Dueling network architectures for deep reinforcement learning," arXiv:1511.06581, 2015.

[26] RaspberryPi, "Tx91501b-915mhz powercaster transmitter," 2021, https://www.raspberrypi.org/documentation/computers/raspberry-pi.html.

https://doi.org/10.1155/2022/9921885 Published online by Cambridge University Press