Wireless Power Transfer

ARTICLE

Open Access

https://doi.org/10.48130/wpt-0025-0004

*Wireless Power Transfer* **2025**, 12: e010

# Deployment optimization of laser chargers in self-organizing power transfer internet of things

Junzhe Hu[1,2], Chuanwen Luo[1,2*], Yi Hong[1,2], Guopeng Wang[3], Xin Fan[1,2] and Deying Li[4]

[1] *School of Information Science and Technology, Beijing Forestry University, Beijing 100083, China*
[2] *Engineering Research Center for Forestry-Oriented Intelligent Information Processing of National Forestry and Grassland Administration, Beijing 100083, China*
[3] *Engineering Research Center of Integration and Application of Digital Learning Technology, Ministry of Education, Beijing 100816, China*
[4] *School of Information, Renmin University of China, Beijing 100872, China*
* Corresponding author, E-mail: chuanwenluo@bjfu.edu.cn

## Abstract

Recently, the Internet of Things (IoT) has played an important role in many fields. Nevertheless, the fast and uneven energy consumption of IoT Devices (IoTDs) significantly limits the lifetime of IoT networks. One of the effective solutions is to deploy Laser Static Chargers (LSCs) to power IoTDs. However, deploying LSCs to cover all IoTDs will consume enormous costs. To prolong the lifetime of IoT and reduce the deployment costs of LSCs, in this paper, we first propose a novel IoT network named Self-organizing Power Transfer IoT with Laser Static Chargers (SPTIoT-LSC), where IoTDs are equipped with laser transmission and reception modules allowing energy transfer between IoTDs, and several LSCs are deployed into the network to charge IoTDs. Based on SPTIoT-LSC, we study the Minimizing Laser Chargers Coverage(MLCC) problem, which aims to minimize the number of LSCs deployed in SPTIoT-LSC while enabling all IoTDs to work continuously. Then we prove its NP-hardness. To solve the problem, we propose two sub-algorithms: the Layered Charging Scheduling Strategy (LCSS) algorithm and Deploy Chargers based on the Multi-agent deep deterministic policy gradient (DCM) algorithm to maximize the working time of IoTDs with given LSCs and corresponding positions and deploy given LSCs in SPTIoT-LSC, respectively. Based on the above sub-algorithms, we propose an approximation algorithm to solve the MLCC problem. Finally, extensive experiments are proposed to verify the efficiency of the proposed algorithm and the superiority of SPTIoT-LSC.

**Citation:** Hu J, Luo C, Hong Y, Wang G, Fan X, et al. 2025. Deployment optimization of laser chargers in self-organizing power transfer internet of things. *Wireless Power Transfer* 12: e010 https://doi.org/10.48130/wpt-0025-0004

## Introduction

Internet of Things (IoT) is the network of physical objects, encompassing sensors, vehicles, instruments, and so on. This interconnected web enables these devices to gather and exchange data[1]. Boasting attributes like low cost, convenient deployment, and self-organization of IoT Devices (IoTDs), IoT networks have been applied in various fields, including agricultural production, smart cities, and environmental monitoring[2–4], etc.

While IoTDs have clear advantages, their practical applications also face significant challenges. Specifically, IoTDs' limited battery capacity restricts their ability to operate sustainably. The rapid and uneven energy depletion of IoTDs significantly hinders the widespread practical utilization of IoT networks. To address this limitation, one of the effective solutions is to deploy Laser Static Chargers (LSCs) to power IoTDs by using laser charging, a kind of Wireless Power Transfer (WPT) technique[5–7]. However, deploying LSCs to cover all IoTDs to supply sufficient energy will consume enormous costs. Motivated by the above analysis, to prolong the lifetime of IoT while reducing the deployment costs of LSCs, we first propose a novel network framework called Self-organizing Power Transfer Internet of Things (SPTIoT), where each IoTD is equipped with a laser transmission module and a laser reception module, enabling laser energy transferred between IoTDs, as shown in Fig. 1. The SPTIoT network consists of several IoT devices with rechargeable batteries. All IoTDs form a self-organizing power transfer network by transmitting laser energy to each other in their neighbourhood. In the network, each IoTD can be used both as a receiver to receive energy from other IoTDs within its neighbourhood and as a charger to charge neighbour IoTDs. As a receiver, the laser reception module of the IoTD receives laser energy, converts the laser energy into electrical energy, and stores it in rechargeable batteries. As a charger, the laser transmission module of the IoTD converts its electrical energy into laser energy and transmits it to other IoTDs. Unlike general IoTs, the SPTIoT distinguishes itself by forming a self-organizing power transfer network, where IoTDs with lower energy can be charged by the IoTDs within their neighbourhood. This innovative architecture not only helps balance the energy distribution across IoTDs and extend the overall lifespan of the network but also significantly reduces the number of LSCs required, because covering a portion of IoTDs can meet the energy requirements of all IoTDs in SPTIoT.

Then, based on the SPTIoT, comprehensively considering the energy requirements of IoTDs and deployment costs of LSCs, we study the Minimizing Laser Chargers Coverage (MLCC) problem. The objective of the problem is to minimize the number of LSCs deployed in SPTIoT while ensuring that all IoTDs can work continuously.

The contributions of this paper can be summarized as follows:

(1) We propose a novel network architecture called the Self-organizing Power Transfer Internet of Things (SPTIoT), allowing energy transfer between IoTDs. We propose the concept of self-organizing wireless charging network for the first time;

(2) We first identify the SPTIoT with LSCs (SPTIoT-LSC) network model, where LSCs are deployed in the network to replenish energy for IoTDs. Based on SPTIoT-LSC, we study the Minimizing Laser Chargers Coverage (MLCC) problem, whose objective is to minimize the number of LSCs deployed in the network while enabling all IoTDs to work continuously. Then we prove that the MLCC problem is NP-hard;

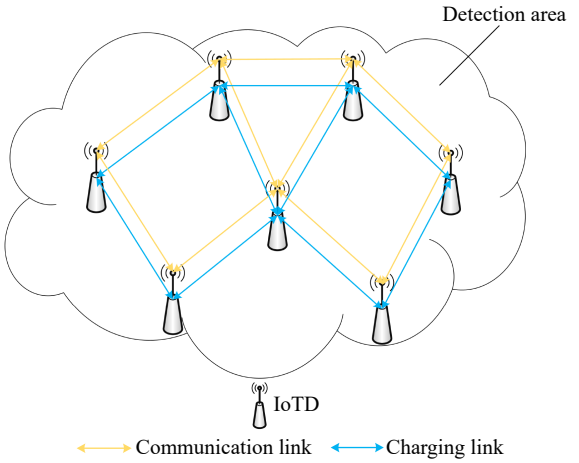


**Fig. 1** Architecture of the SPTIoT network.

(3) To solve the MLCC problem, we first propose a sub-problem, Charging Scheduling Problem (CSP), to maximize the working time of IoTDs with the given LSCs and their positions. Then the Layered Charging Scheduling Strategy (LCSS) algorithm is proposed to solve the CSP problem. After that, the Deploy Chargers based on Multi-agent deep deterministic policy gradient (DCM) algorithm is employed to deploy the given LSCs based on the CSP problem. Finally, we propose an approximation algorithm MLCC Algorithm (MLCCA) to solve the MLCC problem based on the above two algorithms;

(4) Extensive simulations are presented to demonstrate the efficiency of the proposed algorithm for the MLCC problem as well as the superiority of the SPTIoT compared with general IoT.

## Related works

This section discusses relevant research and puts forward the differences between previous literature and this paper. We classify the investigated problems into two types: wireless charging in IoT, and deployment optimization of static chargers.

### Wireless charging in IoT

Commonly used wireless charging methods include magnetic resonance coupling, solar charging, laser charging, etc. In the study by Xie et al.[8], they utilized a wireless charging vehicle with the magnetic resonant coupling charging method to provide periodic recharging for sensors in a wireless sensor network. However, magnetic resonant coupling technology suffers from limited charging distances, facing the problems of high deployment cost and low charging efficiency. In previous studies[9–11], solar charging was used to prolong the lifetime of IoTDs. However, the solar charging method is weather-dependent, and in bad weather or at night, IoTDs cannot be charged. Laser charging, in contrast, offers a reliable and stable energy supplyment, less susceptible to weather variations. The attenuation of laser beam transmission in air is very small[12], making it an effective long-distance charging method. In previous studies[13,14], laser charging technology was used to replenish energy for electric vehicles, overcoming the drawbacks including long charging times and short charging ranges. In a study by Fu et al.[7], an Unmanned Aerial Vehicle (UAV) was utilized to collect data from IoTDs and charge IoTDs using laser charging technology. By optimizing the UAV's trajectory, they maximized its residual energy while meeting all IoTDs' energy requirements. In the research of Zhang et al.[15], they used a laser-beamed WPT technology and presented a multitier tile grid-based spatial structure to charge Aerial User Equipment (AUE) in a high-altitude platform.

### Deployment optimization of static chargers

A large body of literature is dedicated to minimizing the deployment cost of static chargers in IoT networks while meeting the energy requirements of IoT devices. Liao & Jiang[16] studied the problem: minimizing the number of chargers deployed on grid points at a fixed height to make the wireless rechargeable sensor network sustainable. And then they presented two greedy algorithms to solve it. In the research of Chen & Jiang[17], a Particle Swarm Charger Deployment (PSCD) algorithm was proposed to deploy chargers by using the Particle Swarm Optimization (PSO) method. The results indicated its superiority compared with two greedy algorithms proposed by Liao & Jiang[16]. In the study by Chien et al.[18], the authors presented a layoff algorithm combining the Simulated Annealing-based (SA) method to optimize the chargers' positions in the WRSN. The proposed algorithm can reduce the number of chargers efficiently according to their simulation results. In the research of Li et al.[19], they studied how to efficiently deploy wireless static chargers in UAV networks. They presented a binary integer programming method to determine the minimum number of wireless static chargers as well as the location of each charger so that the energy requirements of UAVs are satisfied during flight. Liu et al.[20] investigated the problem of charging nodes with uncertain mobility by static chargers. They proposed a genetic-algorithm-based multi-objective optimization scheme to address the charger deployment problem. Lin et al.[21] proposed a hybrid search and removal strategy to discover the minimum number of chargers required to cover all sensor nodes. You et al.[22] studied a fundamental issue of wireless charger placement with obstacles and proposed a greedy algorithm for placing chargers. In summary, there are many articles studying the optimization of static charger deployment. However, the aforementioned studies neglected the energy transfer between devices, thereby necessitating the deployment of more chargers to ensure comprehensive coverage for IoTDs, which is more costly.

It is clear from previous discussions that there are many studies on wireless charging in IoT and the deployment optimization of static chargers. However, they have not studied the energy transfer between IoTDs. Inspired by the above research, this paper, investigates the Minimize Laser Chargers Coverage problem in SPTIoT networks by considering laser charging technology, deployment optimization of chargers, and energy transfer between IoTDs, to minimize the number of LSCs deployed in SPTIoT and satisfy the energy requirements of all IoTDs.

## Models and definition

In this section, we first introduce the network model. Then, we give the laser charging model for charging IoTDs. Finally, the formal definition of our problem is presented.

### Network model

In this paper, we consider the network architecture of SPTIoT with Laser Static Chargers (SPTIoT-LSC), where a set of $n$ IoTDs $S = \{s_1, s_2, \cdots, s_n\}$ is randomly located at a two-dimensional square detection area $A \subseteq \Re^2$ for a monitoring mission and several LSCs are deployed into the area to replenish energy for IoTDs. In the network, each IoTD $s_i \in S$ has a two-dimensional coordinate $(x_i, y_i)$. All IoTDs have the uniform battery capacity $E_M$ and energy threshold $E_T$. The IoTD stops working when its remaining energy is less than $E_T$. Since IoTDs are in different positions and may take different tasks in the mission, each IoTD $s_i \in S$ has its unique energy consumption rate $\delta_i$.

To satisfy the charging requirements of the network, multiple LSCs are deployed in the detection area to replenish energy for IoTDs, which can be collected in the set $C = \{c_1, c_2, \cdots, c_m\}$. For simplic-

ity, we assume that IoTDs and LSCs have the same laser transmission distance $R_c$. For any IoTD $s_i \in S$, it can be charged by $c_j \in C$ if and only if $d^c_{i,j} \le R_c$, where $d^c_{i,j}$ denotes the Euclidean distance between $s_i$ and $c_j$. Similarly, any IoTD $s_i \in S$ can be charged by $s_k \in S$ if and only if $d^s_{i,k} \le R_c$, where $d^s_{i,k}$ denotes the Euclidean distance between $s_i$ and $s_k$. Let $\mathcal{N}^s_i = \{s_k | k \neq i, d^s_{i,k} \le R_c\}$ represent the set of neighbor IoTDs of $s_i \in S$.

Furthermore, let $T_w$ represent the working period during which all IoTDs must remain continuously operational in SPTIoT-LSC. We discretize $T_w$ evenly into $\Gamma$ time slots, with each slot having a duration of $l = \frac{T_w}{\Gamma}$. The set of time slots is denoted as $\mathcal{T} = \{1, 2, \cdots, \Gamma\}$. For any IoTD $s_i \in S$, we utilize $E^r_i(\tau)$ to signify its remaining energy at time slot $\tau$, with the initial energy set as $E^r_i(0) = E_M$.

## Laser charging model

We consider the one-on-one laser charging model. We use a linear laser energy harvesting model[23,24] with efficiency $\omega$ to derive the energy transmission link from the $c_j$ to $s_i$. The power of $s_i$ received from $c_j$ can be expressed as:

$$P^{j,i}_c = \begin{cases} \frac{(1-\delta_s)\eta_{el}\omega A_c \chi e^{-\alpha d^c_{i,j}}}{(D + d^c_{i,j}\Delta_\theta)^2} P_c, & \mathrm{d^c_{i,j}} \le \mathrm{R_c} \\ 0, & \mathrm{d^c_{i,j}} > \mathrm{R_c} \end{cases} \quad (1)$$

where, $P_c$ is the source power of LSC, $\delta_s$ represents the the power splitting factor to separate the energy transfer link and communication link, $\eta_{el}$ is the electricity-to-laser conversion efficiency, $A_c$ is the area of the charging panel of each IoTD, $\chi$ represents the optical efficiency of the combined transmission-receiver, $\alpha$ is the laser attenuation coefficient, $D$ denotes the size of the initial laser beam, $\Delta_\theta$ represents the angular expansion of the laser beam.

Similar to the laser charging model from LSC to IoTD, for any pair of $s_i \in S$ and $s_k \in S$, we can obtain the power of $s_i$ received from $s_k$ as:

$$P^{k,i}_s = \begin{cases} \frac{(1-\delta_s)\eta_{el}\omega A_c \chi e^{-\alpha d^s_{i,k}}}{(D + d^s_{i,k}\Delta_\theta)^2} P_s, & \mathrm{d^s_{i,k}} \le \mathrm{R_c} \\ 0, & \mathrm{d^s_{i,k}} > \mathrm{R_c} \end{cases} \quad (2)$$

where, $P_s$ is the source power of IoTD.

## Problem definition

In this subsection, we investigate the Minimizing Laser Chargers Coverage (MLCC) problem, whose goal is to minimize the number of LSCs deployed into the detection area while ensuring the energy of each IoTD in $S$ is greater than or equal to $E_T$ during the working period $T_w$.

We first define the binary variables $a^k_i(\tau)$ and $b^j_i(\tau)$ as follows.

$$a^k_i(\tau) = \begin{cases} 1, & s_i \text{ is charged by } s_k \text{ at time slot } \tau \\ 0, & otherwise \end{cases} \quad (3)$$

$$b^j_i(\tau) = \begin{cases} 1, & s_i \text{ is charged by } c_j \text{ at time slot } \tau \\ 0, & otherwise \end{cases} \quad (4)$$

Then, we give the expression for the remaining energy of $s_i \in S$ at any time slot $\tau > 0$:

$$E^r_i(\tau) = E^r_i(\tau - 1) + \sum_{k=1}^{n} a^k_i(\tau) P^{k,i}_s l + \sum_{j=1}^{m} b^j_i(\tau) P^{j,i}_c l - \delta_i l - \sum_{v=1}^{n} a^i_v(\tau) P_s l \quad (5)$$

Finally, the MLCC problem can be mathematically formulated as:

$$\mathbb{P}: \min \ m \quad (6)$$

s.t.

$$\begin{aligned} &\mathrm{C1}: \ E^r_i(\tau) \ge E_T, \forall s_i \in S, \forall \tau \in T \\ &\mathrm{C2}: \ \Sigma^n_{k=1} a^k_i(\tau) + \Sigma^m_{j=1} b^j_i(\tau) \le 1t, \forall s_i \in S, \ \forall \tau \in T \\ &\mathrm{C3}: \ \Sigma^n_{i=1} b^j_i(\tau) \le 1, \forall c_j \in C, \ \forall \tau \in T \\ &\mathrm{C4}: \ \Sigma^n_{i=1} a^k_i(\tau) \le 1, \forall s_k \in S, \forall \tau \in T \end{aligned} \quad (7)$$

where, the constraint C1 ensures the remaining energy of each IoTD is upper than or equal to $E_T$ at any time slot $\tau \in \mathcal{T}$, the constraint C2 limits that one IoTD can only be charged by one IoTD or one LSC at each time slot $\tau \in \mathcal{T}$, the constraint C3 states one LSC can only charge one IoTD at every time slot $\tau \in \mathcal{T}$ and the constraint C4 guarantees one IoTD can only charge another one IoTD at each time slot $\tau \in \mathcal{T}$.

**Theorem 1.** The MLCC problem is NP-hard.

*Proof.* We consider a special case of the MLCC (scMLCC) problem with four conditions: (1) $P_s = 0$; (2) $P_c = +\infty$; (3) for each $s_i \in S$, $E_M - \delta_i T_w < E_T$; (4) LSCs can only be deployed at the positions that coincide with IoTDs. Based on the above four conditions, the objective of the scMLCC problem is to minimize the number of LSCs, which can only be deployed at the locations coinciding with IoTDs, for coveraging all IoTDs to satisfy their charging requirements.

The decision version of the scMLCC problem, called *K*-scMLCC, is that given a set of IoTDs *S* and a positive integer *K*, does there exists a positive integer *m* (*m* ≤ *K*) such that *m* LSCs deployed at the locations coinciding with *m* IoTDs can cover all IoTDs?

To prove the scMLCC problem is NP-hard, we use the Minimum Dominating Set (MDS) problem for reduction, which has proven NP-hard[25]. The decision version of MDS, called *J*-MDS, is defined as: given an undirected graph *G*(*V*, *E*) and a positive integer *J*, does there exists a subset $V' \subseteq V$ and $|V'| \le J$ satisfying that for each node $u \in V \setminus V'$, there must be at least one node $v \in V'$ such that the edge $(u,v) \in E$?

For given an instance of *J*-MDS, with *G*(*V*,*E*), $V = \{v_1, \cdots, v_n\}$ and a positive integer *J*, we construct the instance of *K*-scMLCC problem as follows:

- $K = J$;
- $|S| = |V|$;
- each $s_i \in S$ corresponds to node $v_i \in V$ on graph *G*;
- any pair of $s_i$, $s_j \in S$ are neighbors iff edge $(v_i, v_j) \in E$.

In the following, we will prove that *J*-MDS problem has a YES answer if *K*-scMLCC has a YES answer.

'Sufficiency'. Suppose *m*(*m* ≤ *K*) LSCs, whose locations coincide with*m* IoTDs, can coverage all IoTDs. Let $S' \in S$ be the set of these *m* IoTDs and $V' \subseteq V$ be the set of nodes on graph *G* corresponding to IoTDs in *S*'. Apparently, the subset $V' \subseteq V$ is a dominating set in *G* and $|V'| = |S'| = m \le K = J$.

'Necessity'. Suppose $V' \subseteq V$ is the dominating set in *G* and $|V'| \le J$. Let $S' \subseteq S$ be the set of IoTDs corresponding to nodes in $V'$ on graph $G$. Apparently, any $s_i \in S \setminus S'$ has at least one neighbor in *S*'. *m*(*m* = |*S*'|) LSCs deployed at the locations coinciding with the IoTDs in *S*' can coverage all IoTDs and $m = |S'| = |V'| \le J = K$.

Therefore, the scMLCC problem is NP-hard. Since the scMLCC problem is a special case of the MLCC problem, the MLCC problem is also NP-hard.

## Methodology

In this section, we propose an approximation algorithm called MLCC Algorithm (MLCCA) to solve the MLCC problem.

Before introducing the MLCCA algorithm, we first study a subproblem, the Charging Scheduling Problem (CSP), as shown in **Definition 1**, whose goal aims to provide a charging scheduling strategy of LSCs and IoTDs to maximize the working time of all IoTDs based on the given number and positions of LSCs. Then, we design

the Layered Charging Scheduling Strategy (LCSS) algorithm to solve the sub-problem, which is a subroutine of the MLCCA.

The MLCCA algorithm consists of two steps. Firstly, we divide the monitoring area $A$ evenly into $L \times L$ grids. Secondly, we propose the Deploy Chargers based on Multi-agent deep deterministic policy gradient (DCM) algorithm to deploy the LSCs into the monitoring area $A$ to charge the IoTDs by using the Multi-Agent Deep Reinforcement Learning (MADRL) method, where the output of the LCSS algorithm is utilized as one of the metrics of the reward function in the DCM algorithm.

In the following, we first give a detailed description of the CSP problem.

**Definition 1 (CSP):** Given a set $S = \{s_1, s_2, \cdots, s_n\}$ of $n$ IoTDs, a set $C = \{c_1, c_2, \cdots, c_m\}$ of $m$ LSCs, a set $\mathcal{T} = \{1, 2, \cdots, \Gamma\}$ of time slots, working period $T_w$, the objective of Charging Scheduling Problem (CSP) is to maximize the working time $T_c$ of IoTDs such that the remaining energy of each IoTD is greater than or equal to $E_T$ at every time slot $\tau \in [1, \frac{T_c}{l}]$.

The mathematical formulation of the CSP problem can be expressed as:

$$\mathbb{P}: \max\ T_c \tag{8}$$

s.t.

$$\begin{aligned}
&\mathrm{C1}:\ T_c \le T_w \\
&\mathrm{C2}:\ E_i^r(\tau) \ge E_T,\ \forall s_i \in S,\ \forall \tau \in [1, \frac{T_c}{l}] \\
&\mathrm{C3}:\ \Sigma_{k=1}^{n} a_i^k(\tau) + \Sigma_{j=1}^{m} b_i^j(\tau) \le 1,\ \forall s_i \in S,\ \forall \tau \in [1, \frac{T_c}{l}] \\
&\mathrm{C4}:\ \Sigma_{i=1}^{n} b_i^j(\tau) \le 1t,\ \forall c_j \in C,\ \forall \tau \in [1t, \frac{T_c}{l}] \\
&\mathrm{C5}:\ \Sigma_{i=1}^{n} a_i^k(\tau) \le 1t,\ \forall s_k \in St,\ \forall \tau \in [1, \frac{T_c}{l}]
\end{aligned} \tag{9}$$

where, the contsraint C1 states that $T_c$ is less than or equal to the working period$T_w$, the constraint C2 ensures the remaining energy of each IoTD is greater than or equal to $E_T$ at any time slot $\tau \in [1, \frac{T_c}{l}]$, the constraint C3 limits that any one IoTD can only be charged by one IoTD or one LSC at each time slot $\tau \in [1, \frac{T_c}{l}]$, the constraint C4 states one LSC can only charge one IoTD at every time slot $\tau \in [1, \frac{T_c}{l}]$, and the constraint C5 guarantees one IoTD can only charge another one IoTD at each time slot $\tau \in [1, \frac{T_c}{l}]$.

## Algorithm for the CSP problem

In this subsection, we give a detailed description of the Layered Charging Scheduling Strategy (LCSS) algorithm, as shown in Algorithm 1.

Before describing the algorithm, we first introduce some notations. Let $ly_i$ denote the minimum number of hops from the nearest LSC to $s_i$, which is named the layer number of $s_i$. Let $S_o = \{s_{\rho_1}, s_{\rho_2}, \cdots, s_{\rho_n}\}$ be an ordered set of IoTDs arranged in ascending order according to their remaining energy. If more than one IoTD has the same remaining energy, then sort them in descending order by their layer numbers. If more than one IoTD has the same layer number, then sort them in ascending order by their subscript numbers.

The LCSS algorithm consists of four steps.

In the first step, we initialize $T_c = 0$.

In the second step, for each $s_i \in S$, we compute the set $\mathcal{N}_i^c = \{c_j | d_{i,j}^c \le R_c\}$ of its neighbor LSCs.

In the third step, we compute $ly_i$ for each $s_i \in S$, where the computation process is that for each $c_j \in C$, we use the Dijkstra algorithm to compute the hops from $c_j$ to each IoTD, and then for each $s_i \in S$, we select the smallest one as $ly_i$ from all LSCs to $s_i$.

**Algorithm 1.** Layered charging scheduling strategy.

**Input:** The number of LSCs $m$, the set $C$ of $m$ LSCs, the set $S$ of $n$ IoTDs, time slot length $l$. The set $\mathcal{T}$ of $\Gamma$ time slots;
**Output:** $T_c$;

Initialize $T_c = 0$;
For each $s_i \in S$, compute $\mathcal{N}_i^c = \{c_j | d_{i,j}^c \le R_c\}$;
For each $s_i \in S$, compute $ly_i$ by using Dijkstra algorithm;
**for** $\tau$ from 1 to $\Gamma$ **do**
  For each $s_i \in S$, let $E_i^r(\tau) = E_i^r(\tau - 1) - \delta_i l$.
  Sort all IoTDs and obtain $S_o = \{s_{\rho_1}, \cdots, s_{\rho_n}\}$;
  **for** $i$ from 1 to $n$ **do**
    Let $C_{\rho_i} = \{c_j | c_j \in \mathcal{N}_{\rho_i}^c \wedge \Sigma_{q=1}^{n} b_q^j(\tau) = 0\}$;
    Let $S_{\rho_i}^u = \{s_k | s_k \in \mathcal{N}_{\rho_i}^s \wedge ly_k = ly_{\rho_i} - 1 \wedge \Sigma_{q=1}^{n} a_q^k(\tau) = 0 \wedge E_k^r(\tau) - P_s l > E_{\rho_i}^r(\tau)\}$;
    Let $S_{\rho_i}^e = \{s_k | s_k \in \mathcal{N}_{\rho_i}^s \wedge ly_k = ly_{\rho_i} \wedge \Sigma_{q=1}^{n} a_q^k(\tau) = 0 \wedge E_k^r(\tau) - P_s l > E_{\rho_i}^r(\tau)\}$;
    **if** $C_{\rho_i} \neq \varnothing$ **then**
      $c_v = \arg min\{d_{\rho_i,v}^c | c_v \in C_{\rho_i}\}$;
      $b_{\rho_i}^v(\tau) = 1$, $E_{\rho_i}^r(\tau) = min\{E_M, E_{\rho_i}^r(\tau) + P_c^{v,\rho_i} l\}$;
    **else if** $S_{\rho_i}^u \neq \varnothing$ **then**
      $s_v = \arg max\{E_v^r(\tau) | s_v \in S_{\rho_i}^u\}$;
      $E_{\rho_i}^r(\tau) = E_{\rho_i}^r(\tau) + P_s^{v,\rho_i} l$;
      $a_{\rho_i}^v(\tau) = 1, E_v^r(\tau) = E_v^r(\tau) - P_s l$;
    **else if** $S_{\rho_i}^e \neq \varnothing$ **then**
      $s_v = \arg max\{E_v^r(\tau) | s_v \in S_{\rho_i}^e\}$;
      $E_{\rho_i}^r(\tau) = E_{\rho_i}^r(\tau) + P_s^{v,\rho_i} l$;
      $a_{\rho_i}^v(\tau) = 1, E_v^r(\tau) = E_v^r(\tau) - P_s l$;
    **end if**
    Let $S_o = S_o \backslash \{s_{\rho_i}\}$ and re-sort $S_o = \{s_{\rho_{i+1}}, s_{\rho_{i+2}}, \cdots, s_{\rho_n}\}$;
  **end for**
  **if** $min\{E_i^r(\tau) | 1 \le i \le n\} < E_T$ **then**
    **break;**
  **end if**
  $T_c = l\tau$;
**end for**
**return** $T_c$;

In the fourth step, we schedule the charging process in the network, i.e., for any time slot $\tau$, we update $E_i^r(\tau)$ for each $s_i \in S$ according to equation (5). At each time slot $1 \le \tau \le \Gamma$, the following substeps are executed:

- For each $s_i \in S$, update the remaining energy $E_i^r(\tau) = E_i^r(\tau - 1) - \delta_i l$.
- We obtain $S_o = \{s_{\rho_1}, \cdots, s_{\rho_n}\}$ by sorting all IoTDs.
- For each $1 \le i \le n$, we execute the following steps to find the LSC or IoTD that can supply energy for $s_{\rho_i}$ to charge $s_{\rho_i}$. Firstly, we compute the set $C_{\rho_i} = \{c_j | c_j \in \mathcal{N}_{\rho_i}^c \wedge \Sigma_{q=1}^{n} b_q^j(\tau) = 0\}$ of the LSCs that can charge $s_{\rho_i}$ and the set $S_{\rho_i}^u = \{s_k | s_k \in \mathcal{N}_{\rho_i}^s \wedge ly_k = ly_{\rho_i} - 1 \wedge \Sigma_{q=1}^{n} a_q^k(\tau) = 0 \wedge E_k^r(\tau) - P_s l > E_{\rho_i}^r(\tau)\}$ of the IoTDs with layer number equal to $ly_{\rho_i} - 1$ that can charge $s_{\rho_i}$ and the set $S_{\rho_i}^e = \{s_k | s_k \in \mathcal{N}_{\rho_i}^s \wedge ly_k = ly_{\rho_i} \wedge \Sigma_{q=1}^{n} a_q^k(\tau) = 0 \wedge E_k^r(\tau) - P_s l > E_{\rho_i}^r(\tau)\}$ of the IoTDs with layer number equal to $ly_{\rho_i}$ that can charge $s_{\rho_i}$. Secondly, we find the LSC or IoTD to charge $s_{\rho_i}$ in the light of three cases: (1) $C_{\rho_i} \neq \varnothing$, (2) $S_{\rho_i}^u \neq \varnothing \wedge C_{\rho_i} == \varnothing$, and (3) $S_{\rho_i}^e \neq \varnothing \wedge C_{\rho_i} == \varnothing \wedge S_{\rho_i}^u == \varnothing$.

  (1) $C_{\rho_i} \neq \varnothing$.

  Let $c_v = \arg min\{d_{\rho_i,v}^c | c_v \in C_{\rho_i}\}$. Update $b_{\rho_i}^v(\tau) = 1$, $E_{\rho_i}^r(\tau) = min\{E_M, E_{\rho_i}^r(\tau) + P_c^{v,\rho_i} l\}$.

  (2) $S_{\rho_i}^u \neq \varnothing \wedge C_{\rho_i} == \varnothing$.

Let $s_v = \arg max\{E_v^r(\tau)|\, s_v \in S_{\rho_i}^u\}$. Update $a_{\rho_i}^v(\tau) = 1$, $E_{\rho_i}^r(\tau) = E_{\rho_i}^r(\tau) + P_s^{v,\rho_i} l$ and $E_v^r(\tau) = E_v^r(\tau) - P_s l$.

(3) $S_{\rho_i}^e \neq \varnothing \wedge C_{\rho_i} == \varnothing \wedge S_{\rho_i}^u == \varnothing$.

Let $s_v = \arg max\{E_v^r(\tau)|\, s_v \in S_{\rho_i}^e\}$. Update $a_{\rho_i}^v(\tau) = 1$, $E_{\rho_i}^r(\tau) = E_{\rho_i}^r(\tau) + P_s^{v,\rho_i} l$ and $E_v^r(\tau) = E_v^r(\tau) - P_s l$.

Thirdly, update $S_o = S_o \setminus \{s_{\rho_i}\}$ and re-sort $S_o = \{s_{\rho_{i+1}}, s_{\rho_{i+2}}, \cdots, s_{\rho_n}\}$.

• If $min\{E_i^r(\tau)|\, 1 \le i \le n\} < E_T$, then jump out of the loop, otherwise, let $T_c = l\tau$.

Finally, we can obtain the value of $T_c$.

**Theorem 2.** Time complexity of the LCSS algorithm is $O(\Gamma n^2 + mn^2)$.

*Proof.* The primary time complexity of the LCSS algorithm involves the following calculations:

Firstly, the computation of $\mathcal{N}_i^c$ for each $s_i \in S$ necessitates a running time of $O(m)$. Consequently, the overall complexity of computing $\mathcal{N}_i^c$ across all $s_i \in S$ amounts to $O(mn)$.

Secondly, we repeatedly apply the Dijkstra algorithm $m$ times to compute the minimum hops from each $c_j \in C$ to each IoTD $s_i \in S$, which contributes a complexity of $O(mn^2)$.

Thirdly, we need $\Gamma$ iterations to update $E_i^r(\tau)$ for each $s_i \in S$ at any time slot $1 \le \tau \le \Gamma$. In the interior of the loop, first, we need $O(nlogn)$ to obtain the ordered set $S_o$. Second, for any 1 ≤ i ≤ n, the computation of $C_{\rho_i}$, $S_{\rho_i}^u$ and $S_{\rho_i}^e$ incurs a complexity of $O(n(g_c + 2g_s))$, where $g_c = max\{|\mathcal{N}_{\rho_i}^c|\, 1 \le i \le n\}$ and $g_s = max\{|\mathcal{N}_{\rho_i}^s|\, 1 \le i \le n\}$. Third, for any 1 ≤ i ≤ n, $s_{\rho_i}$ is removed from $S_o$ and $S_o$ is re-sorted. Since only when $s_{\rho_i}$ is charged by another IoTD $s_v$, we need to re-sort $S_o$ by moving $s_v$ to the correct position in $S_o$ according to its remaining energy, the re-sorting process requires at most $n$ steps and the time complexity is $O(n^2)$.

Taken together, we can conclude that the time complexity of the LCSS algorithm is $O(mn + mn^2 + \Gamma n(g_c + 2g_s) + \Gamma nlogn + \Gamma n^2) = O(mn^2 + \Gamma n^2)$, since $g_c \le n$ and $g_s \le n$.

## Markov game formulation for deploying LSCs

Given $m$ LSCs and $L \times L$ grids, we hope to deploy these LSCs into optimal grids of the detection area such that LSCs can serve IoTDs more effectively and IoTDs can work for a longer period. To this end, we formulate the deployment process of LSCs as a discrete-time Markov Game[26], employing the MADRL method to find a solution.

We utilize the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma)$ to represent the Markov Game for LSC deployment, where each LSC functions as an agent. $\mathcal{S}$ is the global state space of the environment, containing the locations of all IoTDs and LSCs. In the initial state, $m$ LSCs are placed in the center of the intermediate grid, which is the grid in row $\lceil \frac{L}{2} \rceil$ and column $\lceil \frac{L}{2} \rceil$.The set of joint actions $\mathcal{A}$ comprises the potential actions of all agents. The reward functions of agents are captured in $\mathcal{R}$. $\mathcal{P}$ is the state transition probability from the current state to the next state, and $\gamma \in [0,1]$ is the discount factor. Additionally, due to the partial observability of the Markov Game[27], agents can only observe local information from the environment. We use observation space $O$ to represent the set of local information observed by agents.

(1) Time Steps

We set $t = 1 \cdots T$ as the time steps of each episode in MADRL. Each episode has $T$ time steps, and at every time step, each agent takes action to change its position by transferring to one of the adjacent grids. After $T$ steps, the final deployment of LSCs is obtained. We set $T = L$ to ensure that each agent can reach any grid in the area within $T$ time steps.

(2) Observation Space $O$

We set $\mathbf{o^t} = \{o_1^t, \cdots, o_m^t\}$ to denote the joint observations of all agents at time step $t$. Since the IoTDs are placed beforehand and their positions no longer change, each agent obtains their location information in advance. Hence, the observation space $o_j^t$ of j-th agent at step $t$ is denoted as $o_j^t = \{(x_j^t, y_j^t), \{(x_i, y_i)|s_i \in S\}\}$, where $(x_j^t, y_j^t)$ denotes the current coordinate of j-th agent at time step $t$.

(3) Action Space $\mathcal{A}$

We set $\mathbf{a^t} = \{a_1^t, \cdots, a_m^t\}$ to denote the joint actions of all agents at time step $t$. Each agent can move to the center of one adjacent grid or keep still in any time step. Concretely, at any time step $t$, the action $a_j^t$ of j-th agent can be expressed as a five-dimensional vector from the set {(0,0,0,0,1), (0,0,0,1,0), (0,0,1,0,0,), (0,1,0,0,0), (1,0,0,0,0)}. Each component of this vector corresponds to a specific movement direction:

- $a_j^t = (0,0,0,0,1)$ indicates that j-th agent transfers to adjacent grid on the left.
- $a_j^t = (0,0,0,1,0)$ represents that j-th agent moves to adjacent grid on the right.
- $a_j^t = (0,0,1,0,0)$ signifies a transfer to the adjacent grid in front.
- $a_j^t = (0,1,0,0,0)$ denotes that j-th agent moves to adjacent grid behind it.
- $a_j^t = (1,0,0,0,0)$ means that j-th agent remains in its current grid.

(4) Reward Function $\mathcal{R}$

At any time step t > 0, each agent takes action $a_j^t$, moving to a new position. We obtain $tf^t$, which is equal to the output of Algorithm 1 when these agents are located in new positions after taking joint actions $\mathbf{a^t}$. At time step $t$ = 0, $tf^0$ is determined by executing Algorithm 1 when agents are in the initial positions.

Then, the set of reward functions $\mathbf{r^t} = \{r_1^t, \cdots, r_m^t\}$ can be computed based on the value of $tf^t$. The reward function $r_j^t$ of j-th agent at time step $t$ is formulated as:

$$r_j^t = \frac{tf^t - tf^0 - \xi_j^t P_b}{T_w} \tag{10}$$

where, $P_b$ is the punishment if the j-th agent moves out of the boundary of the area or collides with other agents or IoTDs, $\xi_j^t \in \{0,1\}$ with $\xi_j^t = 1$ indicating that j-th agent accepts the punishment at time step $t$ and $\xi_j^t = 0$ otherwise.

## Algorithm for deploying LSCs

In this subsection, we propose Deploy Chargers based on Multi-agent deep deterministic policy gradient (DCM) algorithm to deploy $m$ LSCs in SPTIoT-LSC network. The Multi-Agent Deep Deterministic Policy Gradient (MADDPG)[28] algorithm is one of the MADRL methods with the paradigm of centralized training and decentralized execution. In the centralized training phase, agents use joint observations and actions of all agents to train the neural networks. In the decentralized execution phase, each agent only uses local observation to obtain its action.

We first introduce the fundamental components and corresponding notations of MADDPG, encompassing the actor networks, critic networks, target networks, and the replay buffer. The framework of MADDPG is shown in Fig. 2. Each agent learns a $Q$ value function serving as a critic and a policy function acting as an actor. The actor of j-th agent is built via a neural network (denoted as its parameters $\theta_j^\mu$). The actor network $\theta_j^\mu$ of $j$-th agent generates the action $a_j^t = \mu_j(o_j^t; \theta_j^\mu)$ based on the input observation $o_j^t$ at each time step $t$, where $\mu_j(o_j^t; \theta_j^\mu)$ is the output of the actor network. Similarly, the critic of j-th agent is also built via a neural network (denoted as its parameters $\theta_j^Q$). The critic network $\theta_j^Q$ of j-th agent gives the estimated $Q_j(\mathbf{o^t}, \mathbf{a^t}; \theta_j^Q)$ value based on the input joint observations $\mathbf{o^t}$ and joint actions $\mathbf{a^t}$ at any time step $t$. Any j-th agent possesses a target actor network (represented as parameters $\theta_j^{\mu'}$) and a target critic network (represented as parameters $\theta_j^{Q'}$) to make the training process more stable. Let $\mathcal{D}$ represent the replay buffer to store the

transition $(\mathbf{o^t},\mathbf{a^t},\mathbf{r^t},\mathbf{o^{t+1}})$. We can randomly select a mini-batch $\mathcal{K}$ of transitions from $\mathcal{D}$ for training the networks, effectively breaking the correlation among sequential samples and enhancing stability of the training process.

Then, we give the methods to update the actor networks, critic networks and target networks.

The actor network of *j*-th agent is updated by using sampled policy gradient for the policy function $J_j$:

$$\nabla_{\theta_j^\mu} J_j = \frac{1}{|\mathcal{K}|} \sum_{k' \in \mathcal{K}} \nabla_{\theta_j^\mu} \mu_j\left(o_j^k;\theta_j^\mu\right) \nabla_{a_j^k} Q_j\left(\mathbf{o^k},\mathbf{a^k};\theta_j^Q\right)|_{a_j^k=\mu_j\left(o_j^k;\theta_j^\mu\right)} \tag{11}$$

where $k^{'} = (\mathbf{o^k},\mathbf{a^k},\mathbf{r^k},\mathbf{o^{k+1}})$ is a transition in $\mathcal{K}$.

The critic network of j-th agent is updated by minimizing the Mean Square Error (MSE) loss:

$$L\left(\theta_j^Q\right) = \frac{1}{|\mathcal{K}|} \sum_{k' \in \mathcal{K}} \left[Q_j\left(\mathbf{o^k},\mathbf{a^k};\theta_j^Q\right) - y_j^k\right]^2 \tag{12}$$

where, $y_j^k = r_j^k + \gamma Q_{j'}(\mathbf{o^{k+1}},\mathbf{a^{k+1}};\theta_j^{Q'})|_{a_j^{k+1}=\mu'(o_j^{k+1};\theta_j^{\mu'})}$.

The parameters of the target networks of j-th agent are updated through a soft update, that is:

$$\theta_j^{\mu'} \leftarrow \epsilon\theta_j^\mu + (1-\epsilon)\theta_j^{\mu'} \tag{13}$$

$$\theta_j^{Q'} \leftarrow \epsilon\theta_j^Q + (1-\epsilon)\theta_j^{Q'} \tag{14}$$

where, $\epsilon$ is a small constant.

The details of the DCM algorithm are summarized in Algorithm 2, which consists of two phases: training phase (**Lines** 1–26) and executing phase (**Lines** 27–35).

In the training phase, our target is to train the actor and critic networks of agents. The training phase contains $N_e$ episodes and each episode has $T$ time steps. First, we initialize the parameters of neural networks $\theta_j^\mu$, $\theta_j^{\mu'}$, $\theta_j^Q$, and $\theta_j^{Q'}$ of each $c_j \in C$ and execute Algorithm 1. to acquire $tf^0$ and the replay buffer $\mathcal{D}$ is cleared out. Then, at each episode, we initialize positions of $m$ LSCs and an exploration noise $\mathcal{N}$. After that, for each time step $t$ at any one episode, any j-th agent derives the action $a_j^t = \mu_j(o_j^t;\theta_j^\mu) + \mathcal{N}$ by inputing the local observation $o_j^t$ in the actor network, and then executes its action. Subsequently, agents get their rewards $\mathbf{r^t}$ and next observations $\mathbf{o^{t+1}}$ and store the transition $(\mathbf{o^t},\mathbf{a^t},\mathbf{r^t},\mathbf{o^{t+1}})$ in the replay buffer $\mathcal{D}$. Then, the set $\mathcal{K}$ of random transitions are selected from $\mathcal{D}$ to train the actor, critic, and target networks. The critic network of j-th agent is updated by minimizing the MSE loss function $L\left(\theta_j^Q\right)$ in Eqn (12) and the actor network of j-th agent is updated by using the policy gradient for policy function $J_j$ in Eqn (11). The updates of the target actor network and target critic network of j-th agent are shown in Eqns (13) and (14), respectively.

In the executing phase, we leverage the well-trained actor networks to determine the actions of agents. Any j-th agent inputs its local observation $o_j^t$ in its actor network and obtains an action $a_j^t = \mu_j(o_j^t;\theta_j^\mu)$ for execution. After $T$ steps, we obtain the final positions of LSCs and execute Algorithm 1 to obtain $T_c$.

## Algorithm for MLCC problem

In this subsection, we propose an approximation algorithm to solve the MLCC problem, which is called MLCC Algorithm (MLCCA). The steps of the MLCCA algorithm are as follows (Algorithm 3).

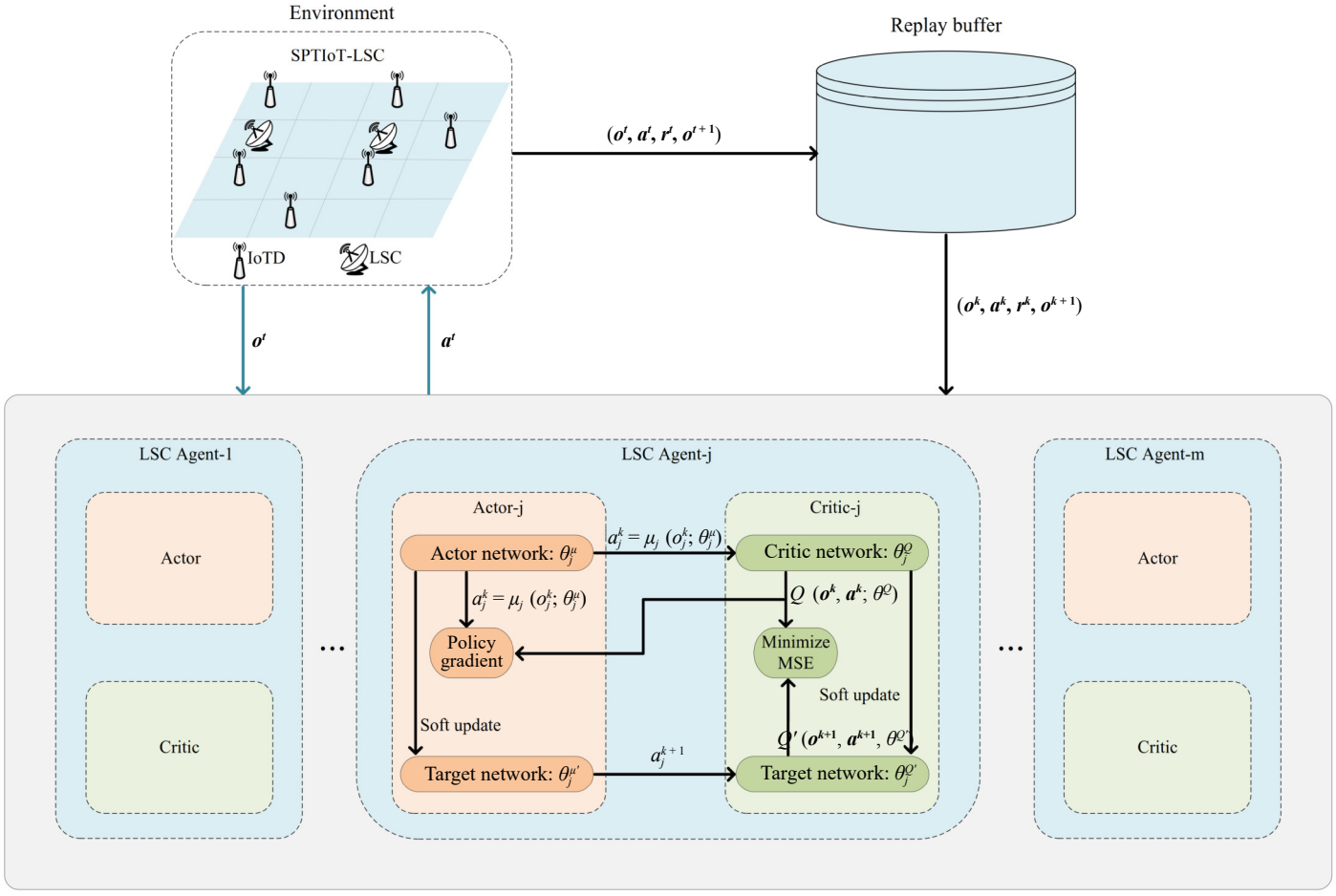


**Fig. 2** The framework of the MADDPG algorithm. The blue arrows represent the interaction among the agents and the environment. The black arrows are used to update the parameters of the networks.

First, we divide the monitoring area $A$ evenly into $L \times L$ grids.

Second, for $m$ from 1 to $n$, we execute Algorithm 2 to obtain the deployment result of $m$ LSCs and $T_c$. If $T_c == T_w$ then we obtain the value $m$, otherwise, let $m = m + 1$.

## Simulation results

In this section, we demonstrate and evaluate the performance of the proposed MLCCA algorithm and the ascendency of SPTIoT through a series of meticulously designed experiments. We first evaluate the convergence of the MADDPG approach under different learning rates and compare the performance of MADDPG with Deep Deterministic Policy Gradient (DDPG). Then, we compare the MLCCA algorithm with other baseline algorithms to demonstrate its effectiveness. Finally, we compare the performance of the SPTIoT network with the general IoT network to highlight its advantages.

**Algorithm 2.** Deploy chargers based on MADDPG.

**Input:** The number of episodes $N_e$, time steps $T$, a small constant $\in$, the number of LSCs $m$, the set $S$ of IoTDs, the number of grids $L \times L$;
**Output:** Positions of $m$ LSCs, $Tc$;

**for** each $c_j \in C$ **do**
  Initialize parameters of actor and target network $\theta_j^{\mu}$ and $\theta_j^{\mu'}$;
  Initialize parameters of actor and target network $\theta_j^{Q}$ and $\theta_j^{Q'}$;
**end for**
Initially place $m$ agents in the center of the intermediate grid;
Obtain $tf^0$ by calling Algorithm 1;
Clear out the replay buffer $D$;
**for** episode from 1 to $N_e$ **do**
  Reset the positions of agents;
  Initialize the exploration noise $\mathcal{N}$;
  **for** time step $t$ from 1 to $T$ **do**
    **for** agent $j$ from 1 to $m$ **do**
      Get the observation $o_j^t$;
      Choose action $a_j^t = \mu_j(o_j^t;\theta_j^{\mu}) + \mathcal{N}$;
    **end for**
    Execute joint actions $a^t = \{a_1^t, ..., a_m^t\}$ of all agents;
    Execute Algorithm I to obtain $tf^t$;
    Obtain reward $r^t$ and next observations $o^{t+1}$;
    Store $(\mathbf{o^t}, \mathbf{a^t}, \mathbf{r^t}, \mathbf{o^{t+1}})$ in replay buffer $D$;
    Select random transitions $\mathcal{K}$ from $D$;
    **for** agent $j$ from 1 to $m$ **do**
      Update $\theta_j^{\mu}$ and $\theta_j^{Q}$ by equation (11) and (12);
      Update $\theta_j^{\mu'}$ and $\theta_j^{Q'}$ by equation (13) and (14);
    **end for**
  **end for**
**end for**
Reset the positions of agents;
**for** time step $t$ from 1 to $T$ **do**
  **for** agent $j$ from 1 to $m$ **do**
    Get the observation $o_j^t$;
    Obtain the action $a_j^t = \mu_j(o_j^t;\theta_j^{\mu})$;
    Execute the action $a_j^t$;
  **end for**
**end for**
Execute Algorithm I to obtain $T_c$;
**return** positions of $m$ agents and $T_c$;

The code is implemented by using MATLAB 2019b and Python 3.6. Table 1 lists some constant parameters utilized in our experiments.

### An instance of the MLCCA algorithm

As the example shown in Fig. 3, we set the configurations as $n$ = 45, $R_c$ = 50 m, $E_M$ = 10,000 J, $T_w$ = 3,600 s, $P_c$ = 3,000 W, $P_s$ = 200 W. We execute the MLCCA algorithm for the instance. Figure 3a–d are the deployment results of LSCs when $m$ = 1, 2, 3, 4, respectively. In these subfigures, the solid lines represent the movement paths of these LSCs during deployment and the dotted circles indicate the coverage areas of these LSCs. Only when $m$ = 4, the equality $T_c$ = 3,600 s = $T_w$ holds. Hence, we conclude that $m$ = 4 is the solution of MLCC problem in this instance.

### Convergence of the MADDPG approach

In this subsection, we first evaluate the convergence performance of the MADDPG algorithm under different learning rates. Then, we compare the performance of MADDPG with DDPG.

We set $n$ = 40, $m$ =3, $E_M$ = 10,000 J, $T_w$ = 3,600 s, $R_c$ = 50 m, $P_c$ = 3,000 W, $P_s$ = 200 W.

In Fig. 4a, we illustrate the convergence performance of the MADDPG algorithm under different learning rates $lr$ = 0.1, $lr$ = 0.01, and $lr$ = 0.001. As seen in the figure, a too high learning rate is unstable and has obvious fluctuation while a too low learning rate causes convergence to local optimal values. Notably, the best convergence effect is achieved with $lr$ = 0.01 compared to rates of 0.1 and 0.001. Consequently, the learning rate is set as 0.01 in subsequent experiments.

In Fig. 4b, we compare the convergence performance of MADDPG with DDPG, where the DDPG algorithm has the same htype parameters, observation space, action space, and reward functions. However, the difference is that the actor and critic networks of any LSC are trained only relying on its own observation and action and are not affected by the observations and actions of other LSCs. The

**Algorithm 3.** MLCCA.

**Input:** The set of IoTDs $S$;
**Output:** $m$;
Divide $A$ evenly into $L \times L$ grids;
**for** $m$ from 1 to $n$ **do**
  Execute Algorithm 2 to deploy $m$ LSCs and obtain $T_c$;
  **if** $T_c == T_w$ **then**
    **break**;
  **else**
    Let $m = m + 1$;
  **end if**
**end for**
**return** $m$;

**Table 1.** Experimental parameters.

| Parameters | Value | Parameters | Value |
|---|---|---|---|
| $A$ | [200 m, 200 m]$^2$ | $N_e$ | 6,000 |
| $L$ | 20 | $T$ | 20 |
| $E_T$ | 200 J | $\|\mathcal{D}\|$ | 25,600 |
| $l$ | 1 s | $\|\mathcal{K}\|$ | 256 |
| $\delta_s$ | $10^{-5}$ | $\epsilon$ | 0.005 |
| $\eta_{el}$ | 0.3 | $\gamma$ | 0.95 |
| $\omega A_c \chi$ | 0.004 m$^2$ | $\mathcal{N}$ | 0.1 |
| $\alpha$ | $10^{-6}$ m | $P_b$ | 50 |
| $D$ | 0.1 m | Layer type | Fully connected |
| $\Delta_\theta$ | $3.4 \times 10^{-5}$ | Optimizer | Adam |
| $\delta_i$ | [5 W, 20 W] | | |

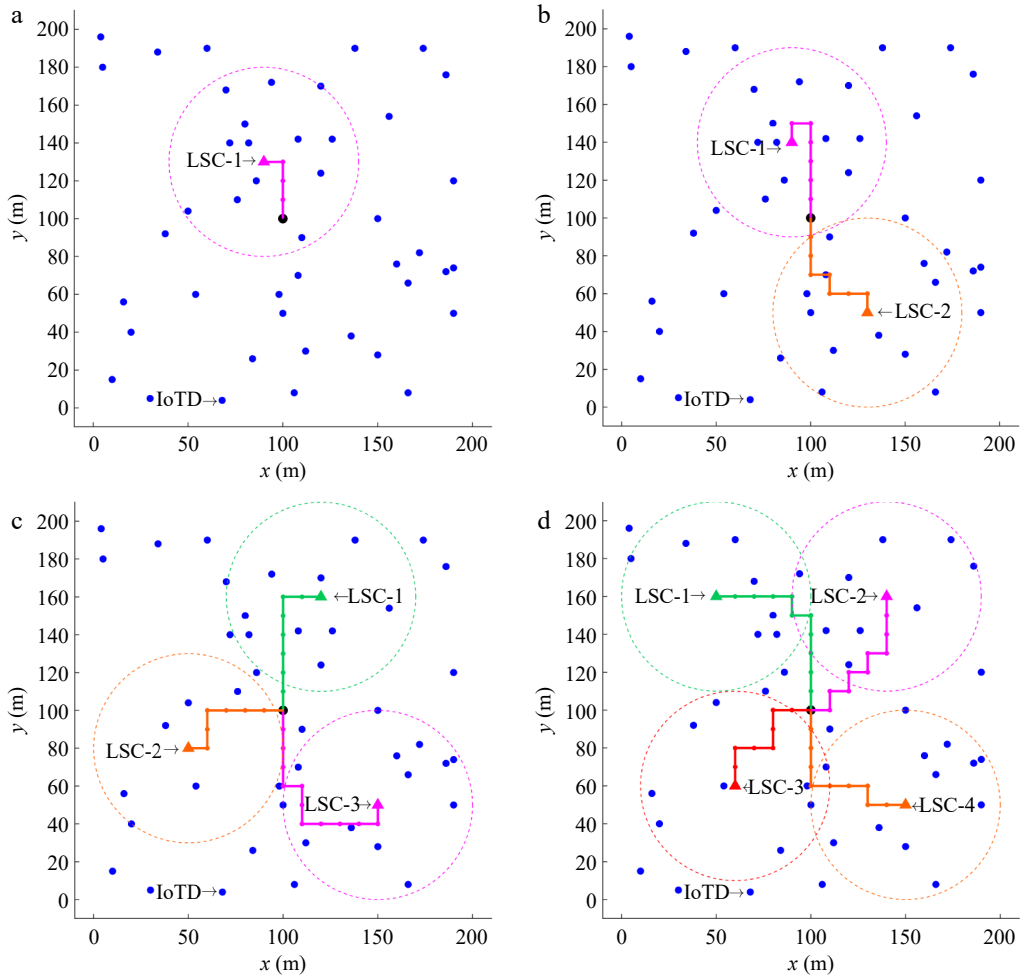


**Fig. 3** The deployment result of LSCs for a given instance obtained by the MLCCA algorithm. The deployment result (a) when m = 1, (b) when m = 2, (c) when m = 3, (d) when m = 4.

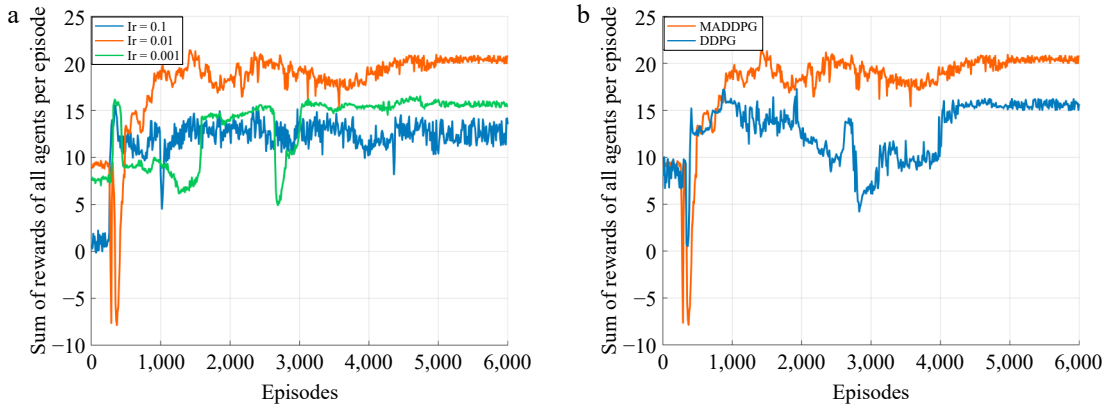


**Fig. 4** The convergence performance of the MADDPG approach. (a) The convergence evaluation under different learning rates. (b) Comparison between MADDPG and DDPG.

results show that the MADDPG algorithm outperforms the DDPG algorithm, reflecting the superiority of multi-agent collaborative working.

## Performance comparison of different algorithms

In this subsection, we compare the performance of the MLCCA algorithm with four other algorithms under different settings and verify the superiority of our proposed algorithm. The baseline algorithms for comparison are listed as follows:

(1) **K-means**. The K-means algorithm consists of the following steps. In the first step, initialize $m = 1$. In the second step, we use the K-means[29] algorithm to obtain $m$ clustering centers by Euclidean distance clustering, and deploy $m$ LSCs to the $m$ clustering centers. In the third step, execute the LCSS algorithm to obtain $T_c$. Finally, if $T_c == T_w$ then we obtain the value $m$, otherwise, let $m = m + 1$ and return to the second step.

(2) **Greedy**. The Greedy algorithm is summarized in the following steps. In the first step, we divide the monitoring area $A$ evenly into

$L \times L$ grids. In the second step, for $j$ from 1 to $n$, we deploy the $j$-th LSC in the optimal grid and call LCSS algorithm to obtain $T_c$, where the optimal grid is defined as: j-th LSC deployed in this grid can obtain larger $T_c$ than deployed in any other grid. If $T_c == T_w$ then we obtain the value $m = j$, otherwise, let $j = j + 1$.

(3) **Simulated Annealing (SA)**[18]. We employ the Simulated Annealing[18] method to deploy LSCs, where the center of each grid is a candidate charger deployment location.

(4) **Wireless chArger placement wIth obsTacles (WAIT)**[22]. We employ the WAIT[22] method (ignoring obstacles) to deploy LSCs, and the goal is to minimize the number of deployed LSCs and meet the energy demands of all IoTDs.

In Fig. 5, we evaluate the performance of three algorithms by setting $E_M$ = 10,000 J, $T_w$ = 3,600 s, $R_c$ = 60 m, $P_c$ = 3,000 W, $P_s$ = 200 W, and changing $n$ from 10 to 40. It can be observed that the MLCCA algorithm outperforms the other algorithms and with the increasing of $n$, the number of LSCs increases. The SA and WAIT algorithms perform significantly worse than the other three algorithms. This is primarily because previous studies did not consider the energy transfer between IoTDs, needing to deploy more LSCs to cover all IoTDs.

In Fig. 6, we evaluate the performance of three algorithms by setting $n$ = 30, $E_M$ = =10,000 J, $T_w$ = 3,600 s, $P_c$ = 3,000 W, $P_s$ = 200 W, and changing $R_c$ from 40 to 100 m. The figure illustrates that the MLCCA algorithm outperforms the other algorithms and with the increasing of $R_c$, the performance difference between the five algorithms is getting smaller. The reason is that as the $R_c$ increases, LSCs can cover and provide energy to more IoTDs, thus narrowing the performance difference between the three algorithms. Additionally, the SA and WAIT algorithms notably lag behind the other three algorithms.

In Fig. 7, we compare the performance of three algorithms by setting $n$ = 30, $T_w$ = 3,600 s, $R_c$ = 60 m, $P_c$ = 3,000 W, $P_s$ = 200 W, and changing $E_M$ from 5,000 to 20,000 J. The figure shows that the performance of MLCCA algorithm is superior compared with the other algorithms. With the increasing of $E_M$, the curves corresponding to MLCCA, Greedy, and K-means algorithms gradually decrease, while the curves associated with SA and WAIT algorithms remain almost unchanged. This is because $n$ and $R_c$ remain unchanged, requiring the same number of LSCs to cover all IoTDs.

## Comparison between SPTIoT and the general IoT network

In this subsection, we undertake a comparative analysis of the SPTIoT and the general IoT network in terms of the number of LSCs required under various parameter settings. Notably, in the general IoT network, IoTDs are incapable of charging other IoTDs. The charging scheduling mechanism in the general IoT operates as follows: during each time slot $\tau$, each $c_j \in C$ charges the IoTD $s_i = \arg min\{E_i^r(\tau) | d_{i,j}^c \le R_c \wedge \sum_{c_v \in C \setminus \{c_j\}} a_i^v(\tau) = 0\}$. All results are illustrated in Fig. 8.

In Fig. 8a, we show the performance comparison of SPTIoT and general IoT network as we set $E_M$ = 10,000 J, $T_w$ = 3,600 s, $R_c$ = 60 m, $P_c$ = 3,000 W, $P_s$ = 200 W, and change $n$ from 10 to 40. The figure shows that the SPTIoT outperforms the general IoT network, and as $n$ increases, the number of LSCs in SPTIoT increases less than that in the general IoT network.

In Fig. 8b, we set $n$ = 30, $E_M$ = 10,000 J, $T_w$ = 3,600 s, $P_c$ = 3,000 W, $P_s$ = 200 W, and change $R_c$ from 40 to 100 m. The figure shows that as $R_c$ increases, the SPTIoT has better performance, and the difference between the number of LSCs in the two networks reduces. The reason is that with the increase of $R_c$, LSCs can cover and provide energy to more IoTDs, thus narrowing the performance difference between the two networks.

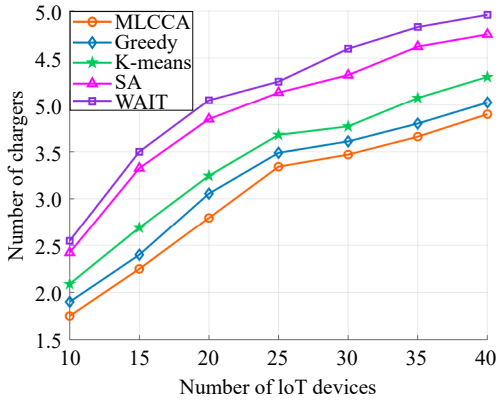


**Fig. 5** Performance comparison by changing the number of IoTDs $n$.

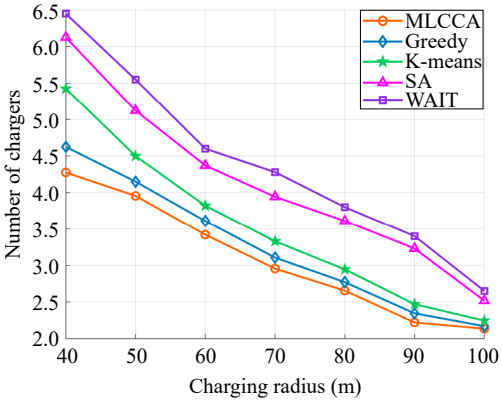


**Fig. 6** Performance comparison by changing the charging radius $R_c$.

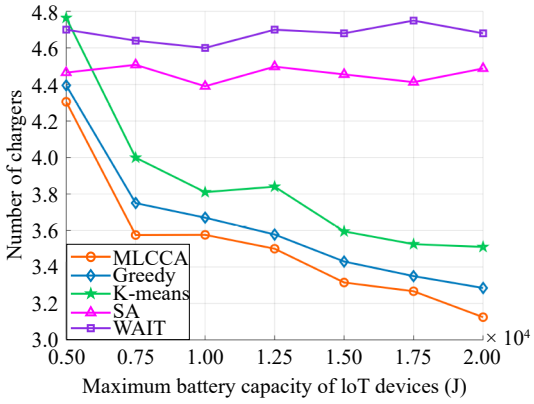


**Fig. 7** Performance comparison by varying the maximum battery capacity of IoTDs $E_M$.

In Fig. 8c, we set $n$ = 30, $E_M$ = 10,000 J, $T_w$ = 3,600 s, $R_c$ = 60 m, $P_s$ = 200 W, and change from $P_c$ from 2,000 to 8,000 W. The figure expresses that the SPTIoT maintains superior performance, and as $P_c$ increases, the number of LSCs in SPTIoT has significantly decreased, while it remains almost unchanged in the general IoT network. This is because in the general IoT, each IoTD needs to be covered by LSCs, and increasing $P_c$ will not change the coverage situation.

In Fig. 8d, we set $n$ = 30, $E_M$ = 10,000 J, $R_c$ = 60 m, $P_c$ = 3,000 W, $P_s$ = 200 W, and change from $T_w$ from 1,200 to 8,400 s. The figure illustrates that the SPTIoT has better performance. This superiority is attributed to the efficient charging scheduling mechanisms employed by SPTIoT, which enable it to effectively meet the energy demands of IoTDs over extended periods.

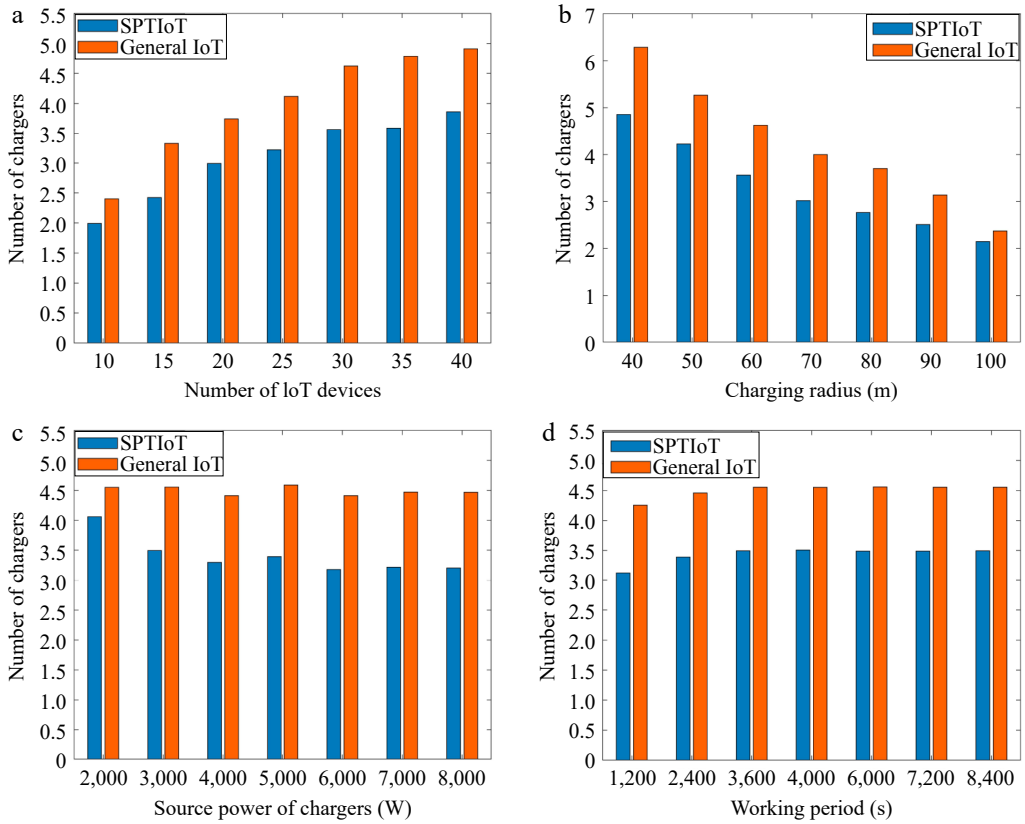


**Fig. 8** Performance comparison between SPTIoT and the general IoT network by varying $n$, $R_c$, $P_c$, $T_w$. Performance comparison by varying (a) the number of IoTDs $n$, (b) the charging radius $R_c$, (c) the source power of LSCs $P_c$, and (d) the working period $T_w$.

## Conclusions

In this paper, we consider the Self-organizing Power Transfer Internet of Things (SPTIoT) network framework and investigate the Minimizing Laser Chargers Coverage (MLCC) problem, which aims at minimizing the number of LSCs deployed in SPTIoT and ensuring all IoTDs in SPTIoT work continuously. Then, we prove the MLCC problem is NP-hard. To solve the problem, we first propose a sub-problem, Charging Scheduling Problem (CSP), and corresponding Layered Charging Scheduling Strategy (LCSS) algorithm to solve it. Then we designed the Deploy Chargers based on the MADDPG (DCM) algorithm to deploy the given LSCs in the network. After that, we propose an approximation algorithm called the MLCC Algorithm (MLCCA) to solve the MLCC problem based on the LCSS algorithm and the DCM algorithm. Finally, we verify the effectiveness of the proposed algorithm as well as the superiority of SPTIoT compared to the general IoT with a large number of simulations.

## Author contributions

The authors confirm contribution to the paper as follows: study conception and design: Hu J, Luo C; data collection: Hu J; analysis and interpretation of results: Hu J, Luo C, Hong Y, Li D, Fan X, Wang G; draft manuscript preparation: Hu J, Luo C. All authors reviewed the results and approved the final version of the manuscript.

## Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (Grant Nos 62202054, 62002022) and the Young Elite Scientists Sponsorship Program by CAST (2023QNRC001).

## Data availability

The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

## Conflict of interest

The authors declare that they have no conflict of interest.

## Dates

Received 28 April 2024; Revised 1 November 2024; Accepted 18 December 2024; Published online 28 April 2025

## References

1. Gokhale P, Bhat O, Bhat S. 2018. Introduction to IOT. *International Advanced Research Journal in Science, Engineering and Technology* 5(1):41–44
2. Ma W, Yang X, Tian Z. 2024. Agriculture neutralization: perspective from intelligent agricultural machinery. *Circular Agricultural Systems* 4:e002
3. Ullah Z, Rehman AU, Wang S, Hasanien HM, Luo P, et al. 2023. IoT-based monitoring and control of substations and smart grids with renewables and electric vehicles integration. *Energy* 282:128924
4. Palazzari V, Mezzanotte P, Alimenti F, Fratini F, Orecchini G, et al. 2017. Leaf compatible 'eco-friendly' temperature sensor clip for high density monitoring wireless networks. *Wireless Power Transfer* 4(1):55–60
5. Yang P, Abusafia A, Lakhdari A, Bouguettaya A. 2023. Monitoring efficiency of iot wireless charging. *2023 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops), 13–17 March 2023, Atlanta, GA, USA*. USA: IEEE. pp. 306–8. doi: 10.1109/PerComWorkshops56833.2023.10150276

6. Xing Y, Pan H, Xu B, Tapparello C, Shi W, et al. 2021. Optimal Wireless Information and Power Transfer Using Deep Q-Network. *Wireless Power Transfer* 8:5513509
7. Fu Y, Mei H, Wang K, Yang K. 2021. Joint optimization of 3D trajectory and scheduling for solar-powered UAV systems. *IEEE Transactions on Vehicular Technology* 70(4):3972–77
8. Xie L, Shi Y, Hou YT, Lou W, Sherali HD, et al. 2014. Rechargeable sensor networks with magnetic resonant coupling. In *Rechargeable Sensor Networks: Technology, Theory, and Application*, eds. Chen J, He S, Sun Y. World Scientific. pp. 31–68. doi: 10.1142/9789814525466_0002
9. Sharma H, Haque A, Jaffery ZA. 2018. Solar energy harvesting wireless sensor network nodes: A survey. *Journal of Renewable and Sustainable Energy* 10(2):023704
10. Sharma H, Haque A, Jaffery ZA. 2018. An efficient solar energy harvesting system for wireless sensor nodes. *2018 2nd IEEE International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES), 22–24 October 2018, Delhi, India*. USA: IEEE. pp. 461–64. doi: 10.1109/ICPEICES.2018.8897434
11. Ram SK, Chourasia S, Das BB, Swain AK, Mahapatra K, et al. 2020. A solar based power module for battery-less IoT sensors towards sustainable smart cities. *2020 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), 6–8 July 2020, Limassol, Cyprus*. USA: IEEE. pp. 458–63. doi: 10.1109/ISVLSI49217.2020.00-14
12. Yang J, Zhu K, Zhu X, Wang J. 2021. Learning-based aerial charging scheduling for UAV-based data collection. *Wireless Algorithms, Systems, and Applications: 16th International Conference, WASA 2021, Nanjing, China, June 25–27, 2021, Proceedings, Part II 16*. Cham: Springer International Publishing. pp. 600–11. doi: 10.1007/978-3-030-86130-8_47
13. Rathod Y, Hughes L. 2019. Simulating the charging of electric vehicles by laser. *Procedia Computer Science* 155:527–34
14. Luo C, Liu N, Hou Y, Hong Y, Chen Z, et al. 2023. Trajectory optimization of laser-charged UAV to minimize the average age of information for wireless rechargeable sensor network. *Theoretical Computer Science 2023* 945:113680
15. Zhang L, Wang Y, Min M, Guo C, Sharma V, et al. 2023. Privacy-aware laser wireless power transfer for aerial multi-access edge computing: a colonel blotto game approach. *IEEE Internet of Things Journal* 10(7):5923–39
16. Liao JH, Jiang JR. 2014. Wireless charger deployment optimization for wireless rechargeable sensor networks. *2014 7th International Conference on Ubi-Media Computing and Workshops, 12–14 July 2014, Ulaanbaatar, Mongolia*. USA: IEEE. pp. 160–64. doi: 10.1109/U-MEDIA.2014.72
17. Chen YC, Jiang JR. 2016. Particle swarm optimization for charger deployment in wireless rechargeable sensor networks. *2016 26th International Telecommunication Networks and Applications Conference (ITNAC), 7–9 December 2016, Dunedin, New Zealand*. USA: IEEE. pp. 231–36. doi: 10.1109/ATNAC.2016.7878814
18. Chien WC, Cho HH, Chao HC, Shih TK. 2016. Enhanced SA-based charging algorithm for WRSN. *2016 International Wireless Communications and Mobile Computing Conference (IWCMC), 5–9 September 2016, Paphos, Cyprus*. USA: IEEE. pp. 1012–17. doi: 10.1109/IWCMC.2016.7577197
19. Li M, Liu L, Wang Y, Peng J, Xi J, et al. 2021. Efficient wireless static chargers deployment for UAV networks. *2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom), 30 September 2021 – 3 October 2021, New York City, NY, USA*. USA: IEEE. pp. 1483–90. doi: 10.1109/ISPA-BDCloud-SocialCom-SustainCom52081.2021.00200
20. Liu H, Zhong L, Liu Z, Lin F. 2022. A multi-objective genetic optimization algorithm for charger selection in static charger deployment scheme for WRSN. *2022 IEEE 14th International Conference on Advanced Infocomm Technology (ICAIT), 8-11 July 2022, Chongqing, China*. USA: IEEE. pp. 230–35. doi: 10.1109/ICAIT56197.2022.9862698
21. Lin T L, Chang H Y, Wang Y H. 2020. A novel hybrid search and remove strategy for power balance wireless charger deployment in wireless rechargeable sensor networks. *Energies* 13(10):2661
22. You W, Ren M, Ma Y, Wu D, Yang J, et al. 2023. Practical charger placement scheme for wireless rechargeable sensor networks with obstacles. *ACM Transactions on Sensor Networks* 20(1):1–23
23. Zhang Q, Fang W, Liu Q, Wu J, Xia P, et al. 2018. Distributed laser charging: A wireless power transfer approach. *IEEE Internet of Things Journal* 5(5):3853–64
24. Lahmeri MA, Kishk MA, Alouini MS. 2020. Stochastic geometry-based analysis of airborne base stations with laser-powered UAVs. *IEEE Communications Letters* 24(1):173–77
25. Hartmanis J. 1982. Computers and intractability: a guide to the theory of NP-completeness (Michael R. Garey and David S. Johnson). *SIAM Review* 24(1):90–91
26. Littman ML. 1994. Markov games as a framework for multi-agent reinforcement learning. In *Machine Learning Proceedings 1994*, eds. Cohen WW, Hirsh H. USA: Morgan Kaufmann. pp. 157–63. doi: 10.1016/B978-1-55860-335-6.50027-1
27. Zhang K, Liu Y, Liu J, Liu M, Başar T. 2020. Distributed learning of average belief over networks using sequential observations. *Automatica* 115:108857
28. Lowe R, Wu YI, Tamar A, Harb J, Abbeel P, et al. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in Neural Information Processing Systems 30 (NIPS 2017), 4-9 Dec 2017, Long Beach, USA*. *https://proceedings.neurips.cc/paper_files/paper/2017*
29. van de Velden M, D'Enza AI, Markos A. 2019. Distance-based clustering of mixed data. *Wiley Interdisciplinary Reviews: Computational Statistics* 11(3):e1456