A General Framework for Charger Scheduling Optimization Problems

A Preprint

Xual Li
Center for Advanced Computer Studies
School of Computing and Informatics
University of Louisiana at Lafayette
Lafayette, LA 70503
xuan.li1@louisiana.edu

Miao Jin
Center for Advanced Computer Studies
School of Computing and Informatics
University of Louisiana at Lafayette
Lafayette, LA 70503
miao.jin@louisiana.edu

January 15, 2020

Abstract

This paper presents a general framework to tackle a diverse range of NP-hard charger scheduling problems, optimizing the trajectory of mobile chargers to prolong the life of Wireless Rechargeable Sensor Network (WRSN), a system consisting of sensors with rechargeable batteries and mobile chargers. Existing solutions to charger scheduling problems require problem-specific design and a trade-off between the solution quality and computing time. Instead, we observe that instances of the same type of charger scheduling problem are solved repeatedly with similar combinatorial structure but different data. We consider searching an optimal charger scheduling as a trial and error process, and the objective function of a charging optimization problem as reward, a scalar feedback signal for each search. We propose a deep reinforcement learning-based charger scheduling optimization framework. The biggest advantage of the framework is that a diverse range of domain-specific charger scheduling strategy can be learned automatically from previous experiences. A framework also simplifies the complexity of algorithm design for individual charger scheduling optimization problem. We pick three representative charger scheduling optimization problems, design algorithms based on the proposed deep reinforcement learning framework, implement them, and compare them with existing ones. Extensive simulation results show that our algorithms based on the proposed framework outperform all existing ones.

Keywords Wireless rechargeable sensor networks · Mobile charger scheduling · Deep reinforcement learning

1 Introduction

Wireless Rechargeable Sensor Network (WRSN), consisting of a group of sensors with rechargeable batteries and one or multiple mobile chargers, has become a promising solution to the energy limitation problem in Wireless Sensor Networks (WSNs). However, inefficient path planning of chargers may result in not just the waste of energy but also the death of nodes and failure of network tasks, considering a mobile charger needs to travel close to a sensor node to charge. Therefore, charger scheduling optimization has become a popular research topic that focuses on optimizing the trajectory of a mobile charger to prolong the life of a WRSN system.

Charger scheduling optimization problems can be roughly classified into two groups. One provides a budget including the total charging time or energy spent on both charging and traveling. A charger seeks a path to maximize some objective function including the total energy charged on nodes or number of nodes charged within the budget Liang et al. [2017], Chen et al. [2016]. The other requires a number of sensor nodes to be charged. A charger seeks a path to minimize some objective function including the total energy spent on the road, or the travel distance, or the total charging time Shi et al. [2011], Xie et al. [2012], He et al. [2014], Lu et al. [2015], Li and Jin [2019].

Charger scheduling optimization problems are in general NP-hard. They are tackled by exact, approximation, and heuristic algorithms. Specifically, an exact algorithm with provable optimality guarantee uses enumeration strategy but fails when the size of dataset increases. An approximation algorithm with provable solution quality is in general desirable. However, the computational complexity of an approximation algorithm may also go sky-high when the size of dataset or requirement of the solution quality increase. A heuristic algorithm is another alternative, fast to compute although lack of optimality or quality guarantee. Overall, existing solutions require problem-specific design and a trade-off between the solution quality requirement and computing time.

It is obvious that existing algorithm design on charger scheduling optimization problems fails to exploit the common characteristic of these problems - instances of the same type of problem are solved repeatedly with similar combinatorial structure but different data Bello et al. [2016], Khalil et al. [2017]. Instead, we consider searching an optimal charger scheduling as a trial and error process, and the objective function of a charging optimization problem as reward, a scalar feedback signal for each search. Specifically, we model charger scheduling optimization problems with a weighted graph and consider the objective function as a cumulative reward of charging sensor nodes along a path in one cycle. We then build a deep reinforcement learning - a fundamental approach combined with deep neural network to optimize strategy to maximize the cumulative reward - based charger scheduling optimization framework. The biggest advantage of the framework is that a diverse range of domain-specific charger scheduling strategy can be learned automatically from previous experiences, i.e., different graphs with various sizes. A framework also simplifies the complexity of algorithm design for individual charger scheduling optimization problem.

We introduce the framework with four steps:

Graph construction: We use a weighted graph to model charger scheduling optimization problems.

Graph representation: We apply the structure2vec technique Dai et al. [2016], Khalil et al. [2017] to represent the weighted graph.

Deep reinforcement learning based framework: The framework includes all the key components of a typical reinforcement learning algorithm and functions specifically designed for charger scheduling optimization problems.

Deep-Q-Network algorithm: A Deep-Q-Network (DQN) is applied to learn the policy, i.e. an optimal charger scheduling strategy.

The rest of this paper is organized as follows: Section 2 gives a brief review of charger scheduling optimization problems and introduction of the basic concepts of deep reinforcement learning. Sections 3, 4, and 5 introduce three representative charger scheduling optimization problems, respectively, where the one in 5 has not been studied yet. Section 6 provides the detail of proposed deep reinforcement learning-based charger scheduling optimization framework and shows in steps the algorithms designed to tackle the three charging problems based on the proposed framework. Section 7 presents the simulation results and performance comparison of the algorithms designed based on the proposed framework and existing ones. Section 8 concludes the paper.

2 Related Works

2.1 Charger Scheduling Optimization

Charger scheduling optimization has been a popular research topic that focuses on optimizing the trajectory of a mobile charger to prolong the life of a WRSN system. Charger scheduling optimization problems can be roughly classified into two groups.

The first group is that a charger with given budget (e.g., the total charging time, or the total energy spent on charging and traveling) seeks a path to achieve a maximum objective function (e.g., the total energy charged to nodes, or the total number of nodes being charged) Liang et al. [2017], Chen et al. [2016]. In Liang et al. [2017], a charger with energy bound seeks a path to maximize the charging rewards. The authors consider two scenarios - sensors can be charged to full capacity at one time or a certain energy level by several times - and provide a 4-approximation algorithm. In Chen et al. [2016], a charger seeks a charging path to maximize the number of mobile sensor nodes charged within a given charging time. The authors discretize the moving trajectory of each sensor node by the time and provide a quasi-polynomial time approximation algorithm.

The second group is that a charger is required to charge a given set of sensor nodes and seeks a path to achieve a minimum objective function (e.g., the total energy spent on the road, the total travel distance, or the total charging time) Shi et al. [2011], Xie et al. [2012], He et al. [2014], Lu et al. [2015], Li and Jin [2019], Ma et al. [2018], Xu et al. [2020]. A solution may not exist for problems of this category. In Shi et al. [2011], the authors consider a scenario where a mobile charger periodically travels inside a network to charge each sensor node. To maximize the ratio of the charger's vacation time over the cycle time, the authors prove that its optimal traveling path in each renewable cycle is the shortest Hamiltonian cycle and propose a heuristic solution. In Xie et al. [2012], the authors apply multi-node wireless energy transfer technology and extend the study in Shi et al. [2011] to a scenario where multiple nodes can be charged at the same time. In He et al. [2014], the authors introduce a tree-based charging schedule to minimize the travel distance of a charger in robotic sensor networks. To guarantee the charging schedule depletion free for any robot, they provide theoretical guidance on the setting of remaining energy threshold at which robots request energy replenishment. In Lu et al. [2015], the authors consider minimizing both the travel distance of a charger and the charging delay of sensor nodes as a set of nested Traveling Salesman Problems. In Ma et al. [2018], a mobile charger charges multiple sensors simultaneously to minimize the travel distance. In Xu et al. [2020], the authors consider a multi-node charging setting where multiple neighboring sensors can be charged simultaneously by a single charger. They provide an approximation algorithm to minimize the longest delay of sensors waiting for charge.

We pick three representative problems that cover the two major groups and design issues in WRSN in Sections 3, 4, and 5, respectively. We use them as concrete examples to illustrate how our deep reinforcement learning-based framework can be applied and provide solutions with superior performance in the following sections.

2.2 Deep Reinforcement Learning

Given a goal-directed agent in an uncertain environment, the agent interacts with the environment through observations, actions, and feedback (rewards) on actions Sutton et al. [1998]. At each time step t, the agent observes current state s_t, and chooses action a. Then the state of the environment transits to next one s_{t+1} and the agent receives a reward r_t. $T(s_{t+1}|s_t, a)$ is a transition probability function indicating the probability that the environment will transfer to state s_{t+1} if the agent take action a at state s_t. The agent learns through previous experiences to select an action and make the expected cumulative discounted rewards $E[\sum_{t=0}^{\inf} \gamma r_t]$ in the future maximized, where γ is the discount rate between 0 and 1.

The agent takes action $A = \{a_1, ..., a_n\}$ at state $S = \{s_1, ..., s_m\}$ based on a policy denoted by $\pi(S, A)$. A policy can be either deterministic or stochastic. If the action space is discrete and the policy is deterministic, we choose value-based reinforcement learning, e.g. Q-learning. If the action space is continuous and the policy is stochastic, we then choose policy-based reinforcement learning, e.g. policy-gradient. Considering the action space of charger scheduling optimization problems is discrete, i.e., a charger decides which exact node to charge next, we choose the value-based reinforcement learning technique, Q-learning, to build the framework.

Q-learning is a model-free reinforcement learning that does not require an agent with full knowledge of the whole environment. The agent simply maintains a Q-table, which stores Q-value where Q stands for quality/reward of each state-action pair. The agent will select the action that maximizes Q in the current state. However, it is infeasible to learn all the state-action pairs for most practical problems. Therefore, function approximation technique Hornik [1991] is commonly used. For Q-learning, a function approximator $Q(S, A; \Theta)$ is parameterized by Θ with size much smaller than the combination of all possible state-action pairs. Deep Q-Network (DQN) Riedmiller [2005] applies deep neural networks as function approximators, combined with different techniques including the experience replay method Mnih et al. [2013]. Considering the exponentially increased size of station-action pairs in charger scheduling optimization problems, we choose DQN to build the framework for charger scheduling optimization problems in WRSNs.

3 Mobile Network Charging Path Optimization Problem

A battery-powered mobile sensor usually takes a pre-designed mobility pattern to do the jobs such as scientific exploration. The mobile network charging path optimization problem studied in Chen et al. [2016] seeks an optimal path for a mobile charger to charge mobile sensor nodes as many as possible within a fixed time horizon.

3.1 Network Model

The network model in Chen et al. [2016] assumes a set of battery-powered mobile sensor nodes that need to be recharged periodically. Denote $V = \{v_i | 1 \leq i \leq n\}$ as the set of nodes deployed over a planar Field of Interest (FoI) and $p_i(t)(1 \leq i \leq n)$ as the position of v_i at time t. A sensor node v_i is equipped with a rechargeable battery with capacity B. The trajectory of a mobile sensor node can be a curve of any form without imposing any constraint on its mobility pattern, but its trajectory (i.e., the moving position $p_i(t)$ for node v_i at any t) is known to the charger. A mobile sensor node remains stationary when being charged.

3.2 Problem Formulation

Assume the moving speed of a mobile charger is faster than the upper-bound of the moving speed of a mobile sensor. A mobile charger travels from a starting point, charges mobile sensor nodes one by one to a required battery level denoted as α before returning to the end point. The mobile charger needs to maximize the charged sensor nodes within a maximum charging timespan denoted as C. The total charging time along the charging path P is denoted as $\Gamma(P)$, including the traveling time of the charger, the charging time of the sensor nodes, and the total sojourn time when the charger stays at a position without charging any mobile sensor. The total charging time $\Gamma(P)$ needs to be less than the maximum charging timespan C.

The mobile network charging path optimization problem is defined as the following.

Definition 1. *Given a required battery level α and a maximum charging timespan C, the mobile network charging path optimization problem is to schedule an optimal path:*

$$P^* = \arg \max_{\Gamma(P) \leq C} |\Lambda(P)| \tag{1}$$

where $\Lambda(P)$ is the set of nodes charged on the path P.

3.3 Problem Hardness

The mobile network charging path optimization problem is APX-hard.

Theorem 1. *The mobile network charging path optimization problem is APX-hard, or NP-hard.*

Proof. The proof can be found in the reference Chen et al. [2016]. □

3.4 Algorithm

A quasi-polynomial time algorithm that achieves a poly-logarithmic approximation is introduced in Chen et al. [2016] . The trajectory of each mobile sensor node is discretized by a time step Δt to construct a directed acyclic graph. Vertices of the graph represent the discretized points along the trajectories of mobile sensor nodes, the starting and ending positions of the charger. The approximation algorithm recursively decomposes the problem of searching an optimal charging path into sub-problems of searching sub-paths.

The computational complexity of the algorithm is $O(n_d \min(n_d, C) \log C)^L$, where n_d is the size of nodes after discretization, C is the maximum charging timespan, and L is the recursion level of the algorithm. Let n represent the original size of mobile sensor nodes, then $n_d = n\frac{C}{\Delta t}$. For a network with large size n and a long charging timespan C, it is obvious that the approximation algorithm has to sacrifice the solution quality by either increasing the time step size Δt or decreasing the recursion level L.

4 Fully Charging Reward Maximization Problem

Given a set of energy-critical sensors and a mobile charger with a fixed energy capacity, the fully charging reward maximization problem studied in Liang et al. [2017] seeks an optimal tour for the mobile charger to maximize the sum of charging energy rewards.

4.1 Network Model

The network model in Liang et al. [2017] assumes a set of stationary sensor nodes, $V = \{v_i | 1 \leq i \leq n\}$, deployed over a planar FoI with locations, $P = \{p_i | 1 \leq i \leq n\}$. Each sensor node v_i is equipped with a rechargeable battery with a capacity of B.

4.2 Problem Formulation

A mobile charger with an energy capacity IE sends its departure time denoted as t_0 from the service station to each sensor node. When receiving the message, node v_i estimates its residual energy at t_0 denoted as $B_i(t_0)$. If $B_i(t_0)$ is less than an energy threshold α, i.e., $B_i(t_0)/B \leq \alpha$, v_i calculates a positive integer number $\pi_i (1 \leq \pi_i \leq n^2)$ that models the gain/reward if being charged. The reward is proportional to the amount of energy required to charge v_i to its full capacity. Node v_i then sends a charging request $(id, p_i, \pi_i, B_i(t_0))$, including node v_i ID, position p_i, charging reward π_i, and residual energy $B_i(t_0)$ at t_0, to the charger.

The mobile charger seeks a charging path P that maximizes the total reward of nodes charged. Note that the total amount of energy consumed in charging sensor nodes and traveling along the P should be no larger than the energy capacity IE of the mobile charger.

The fully charging reward maximization problem is defined as the following.

Definition 2 (Fully charging reward maximization problem)**.** *The fully charging reward maximization problem is to schedule a charging path P*

$$\max \sum_{v_i \in P} \pi_i \tag{2a}$$

$$\text{s.t.} \sum_{v_i \in P} (B - B_i(t_0)) + \Omega(P) * \xi \leq IE \tag{2b}$$

where $\Omega(P)$ is the length of path P, and ξ is the amount of energy consumption per traveling unit of the charger.

4.3 Problem Hardness

The fully charging reward maximization problem is NP-hard.

Theorem 2. *The fully charging reward maximization problem is NP-hard.*

Proof. The proof is contained in the reference Liang et al. [2017]. □

4.4 Algorithm

A 4-approximation algorithm is introduced in Liang et al. [2017]. The basic idea of the algorithm is to reduce the fully charging reward maximization problem to the orienteering problem with an approximation algorithm introduced in Bansal et al. [2004]. The orienteering problem is to find a path P in a graph G from a starting vertex to an ending one such that the total reward collected from vertices along P, $\sum_{v_i \in P} \pi_i$ is maximized, and the length of P is no longer than the length of a maximum path $\Omega(P_{\max})$.

The time complexity of the 4-approximation algorithm is $T_{\text{ori}}(3|V(G)|, 2(|V(G)| + |E(G)|))$, where $T_{\text{ori}}(|V(G)|, |E(G)|)$ is the time complexity of the approximation algorithm in Bansal et al. [2004].

5 Optimal k-coverage Charging Problem

One basic requirement of WSN deployment is full area coverage of a field of interest (FoI). Multiple coverage, where each point of the FoI is covered by at least k different sensors with $k > 1$ (k-coverage), is often applied to increase the sensing accuracy of data fusion and enhance the fault tolerance in case of node failures Yang et al. [2006], Simon et al. [2007], Liggins et al. [2008], Z. Zhou and Gupta [2009], Bai et al. [2011], Li et al. [2015]. A common practice to achieve k-coverage is a high density of sensor nodes randomly distributed over the monitored FoI Kumar et al. [2004], Hefeeda and Bagheri [2007].

Optimal k-coverage charging problem studies the mobile charger scheduling scenario in which the k-coverage ability of a network system needs to be maintained. A node sends a charging request with its position information and a charging deadline estimated based on its current residual energy and battery consumption rate. A mobile charger seeks a path to charge sensor nodes before their charging deadlines under the constraint of maintaining the k-coverage ability of the monitored are. At the same time, the charger tries to maximize the energy usage efficiency, i.e., minimizing the energy consumption on traveling per tour.

The Optimal k-coverage charging problem has not been studied yet. We describe the network model, formulate the problem, analyze its hardness, and propose a dynamic programming-based algorithm in detail in the following sections.

5.1 Network Model

We consider a set of stationary sensor nodes, $V = \{v_i | 1 \leq i \leq n\}$, deployed over a planar FoI denote as A with locations, $P = \{p_i | 1 \leq i \leq n\}$. For each sensor node v_i, we assume a disk sensing model with sensing range r. If the Euclidean distance between a point $q \in A$ and node position p_i is within distance r, i.e., $||p_i - q||_{L_2} \leq r$, then the point q is covered by sensor node v_i, and we use $v_i(p) = 1$ to represent it, as shown in equation (3):

$$v_i(q) = \begin{cases} 1, & ||p_i - q||_{L_2} \leq r \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Definition 3 (Full Coverage)**.** *If for any point $q \in A$, there exists at least one sensor node covering it, i.e., $\sum_{i=1}^{n} v_i(q) \geq 1$, then area A is full covered.*

Definition 4 (K-Coverage)**.** *If for any point $q \in A$, there exist at least $k \geq 1$ sensor nodes covering it, i.e., $\sum_{i=1}^{n} v_i(q) \geq k$, then area A is k-covered.*

It is obvious that full coverage is a special case of k-coverage when $k = 1$.

5.2 Problem Formulation

A sensor node v_i is equipped with a rechargeable battery with capacity B. $B_i(t)$ denotes the residual energy of sensor node v_i at time t. A mobile charger sends its departure time denoted as t_0 from service station to each sensor node. When receiving the message, node v_i estimates its residual energy $B_i(t_0)$ at t_0. If it is less than an energy threshold α, i.e., $B_i(t_0)/B \leq \alpha$, sensor node v_i sends a charging request (id, p_i, D_i) including its ID, position p_i, and charging deadline D_i to the charger. A charging deadline, i.e., an energy exhausted time, is estimated based on the residual energy $B_i(t_0)$ at t_0 and an average battery consumption rate denoted as β_i. Specifically, $D_i = B_i(t_0)/\beta_i$. Note that nodes may have different energy consumption rates.

A mobile charger with an average moving speed s is responsible for selecting and charging a number of sensor nodes before their charging deadlines to maintain k-coverage of area A, and it also seeks a path with a minimum energy consumption on traveling. We assume that the time spent on charging path is less then the operation time of sensors, so a sensor node only needs to be charged once in each tour. Unless under an extremely dense sensor deployment, we consider that a mobile charger charges sensor nodes one by one because the energy efficiency reduces dramatically with distance; the energy efficiency drops to 45 percent when the charging distance is 2m (6.56 ft) Kurs [2007]. We denote r_c the energy transfer rate of a mobile charger.

The charging time is defined as the following:

Definition 5 (Charging Time)**.** *Denote P a charging path and $t_P(v_i)$ the charging time along P at node v_i. If P goes from nodes v_i to v_j, the charging time begins at node v_j is*

$$t_P(v_j) = \begin{cases} t_P(v_i) + \frac{B - B_i(t_P(v_i))}{r_c} + \frac{d_{ij}}{s}, \\ \qquad \textit{if}\; t_P(v_i) + \frac{B - B_i(t_P(v_i))}{r_c} + \frac{d_{ij}}{s} \leq D_j \\ \textit{inf}, \\ \qquad \textit{otherwise}, \end{cases} \tag{4}$$

where d_{ij} is the Euclidean distance between nodes v_i and v_j and s is the average moving speed of a mobile charger. The residual energy $B_i(t)$ is estimated as $B_i(t) = B_i(t_0) - \beta_i * (t - t_0)$.

The optimal k-coverage charging problem is then be formulated as the following.

Definition 6 (Optimal k-coverage charging problem)**.** *Given a set of sensor nodes $V = \{v_i | 1 \leq i \leq n\}$, randomly deployed over a planar region A with locations $P = \{p_i | 1 \leq i \leq n\}$ such that every point of A has been at least k covered initially, the optimal k-coverage charging problem is to schedule a charging path P*

$$\min \; |P| \tag{5a}$$

$$\text{s.t.} \; \sum_{i=1}^{n} v_i(q) \geq k, \forall q \in A. \tag{5b}$$

$$t_P(v_i) \leq D_i, \forall v_i \in P. \tag{5c}$$

Note that a charger does not need to respond all the nodes sending requests.

5.3 Problem Hardness

The optimal k-coverage charging problem is NP-hard.

Theorem 3. *The optimal k-coverage charging problem is NP-hard.*

Proof. To prove the NP-hardness of optimal k-coverage charging problem, we prove that an NP-hard one, traveling salesman problem with deadline that seeks a minimum traveling cost to visit all the nodes before their deadlines, can be reduced to the trivial case of the optimal k-coverage charging problem in polynomial time.

We consider a trivial case of the optimal k-coverage charging problem: we require $k = 1$ and assume that the initial deployment of sensor nodes has no coverage redundancy. A mobile charger needs to charge all the sensor nodes sending requests before their deadlines to maintain a full coverage. It is straightforward to see that a solution of traveling salesman problem with deadline is also a solution of the trivial case of the optimal k-coverage charging problem and vice versa. Since even finding a feasible path for traveling salesman problem with deadline is NP-complete Savelsbergh [1985], the optimal k-coverage charging problem is then NP-hard. □

5.4 Dynamic Programming Based Algorithm

5.4.1 Area Segmentation and Time Discretization

Considering that the sensing region of a sensor node v_i is centered at p_i with radius r_i, disk-shape sensing regions of a network divide a planar FoI A into a set of subregions, marked as $A = \{a_i | 1 \leq i \leq m\}$. Then $\sum_{i=1}^{n} v_i(a_i)$ is the number of sensors with a_i within their sensing ranges. In the definition of optimal k-coverage charging problem, we assume $\sum_{i=1}^{n} v_i(a_i) \geq k$ with the initial deployment of a network. Denote $r(a_i)$ the number of sensor nodes sending charging requests with sensing regions including subregion a_i. Three cases exist for subregion a_i:

Case I: $\sum_{i=1}^{n} v_i(a_i) - r(a_i) \geq k$: all the requests are not essential.

Case II: $\sum_{i=1}^{n} v_i(a_i) = k$ and $\sum_{i=1}^{n} v_i(a_i) - r(a_i) < k$: a mobile charger needs to charge all the sensor nodes sending requests with their sensing regions containing subregion a_i before their charging deadlines.

Case III: $\sum_{i=1}^{n} v_i(a_i) > k$ and $\sum_{i=1}^{n} v_i(a_i) - r(a_i) < k$: a mobile charger needs to charge at least $k - \sum_{i=1}^{n} v_i(a_i) + r(a_i)$ sensor nodes sending requests with their sensing regions containing subregion a_i before their charging deadlines.

A table denoted as T with size m, is constructed to store the minimum number of sensors to charge for each a_i. Specifically, $T[i] = k - \sum_{i=1}^{n} v_i(a_i) + r(a_i)$. If the value is negative, we simply set $T[i]$ to zero.

For a sensor node v_i sending request, we divide its time window $[t_0, t_0 + D_i]$ into a set of time units $\{t_i^k | 0 \leq k \leq D_i\}$, where $t_i^0 = t_0$ and $t_i^{D_i} = D_i$. We represent node v_i with a set of discretized nodes $\{v_i(t_i^k) | 0 \leq k \leq D_i\}$, where $v_i(t_i^k)$ represents node v_i at time t_i^k.

5.4.2 Graph Construction

We construct a directed graph denoted as G with vertices and edges defined as follows.

Vertices. The vertex set $V(G)$ includes the discretized sensor nodes sending charging requests, i.e., $\{v_i(t_i^k) | 0 \leq k \leq D_i\}$.

Edges. There exists a directed edge $\overrightarrow{v_i(t_i^k)v_j(t_j^{k'})}$ from $v_i(t_i^k)$ to $v_j(t_j^{k'})$ in the edge set $E(G)$ if and only if

$$t_i^k + \frac{B - B_i}{r_c} + \frac{d_{ij}}{s} > t_j^{k'-1},$$

$$t_i^k + \frac{B - B_i}{r_c} + \frac{d_{ij}}{s} \leq t_j^{k'},$$

where $k' > 0$.

A directed edge ensures that a mobile charger arrives at sensor node v_j before its charging deadline with the charging time being the arrival time of the charger.

Theorem 4. *G is a directed acyclic graph (DAG).*

Proof. Suppose there exists a cycle in G. Assume vertices $v_i(t_i^k)$ and $v_j(t_j^{k'})$ are on the cycle. Along the directed path from $v_i(t_i^k)$ to $v_j(t_j^{k'})$, it is obvious that $t_i^k < t_j^{k'}$. However, along the directed path from $v_j(t_j^{k'})$ to $v_i(t_i^k)$, we have $t_j^{k'} < t_i^k$. Contradiction, so G is a directed acyclic graph. □

Definition 7 (Clique)**.** *A set of nodes $\{v_i(t_i^k)|0 \leq k \leq D_i\}$ in G is defined as a clique if they correspond to the same node v_i at different time units.*

Definition 8 (Feasible Path)**.** *A path P in G is a feasible one if it passes no more than one vertex of a clique. At the same time, charging along P satisfies the k-coverage requirement of the given network.*

5.4.3 Dynamic Programming Algorithm

We design a dynamic programming-based algorithm to find an optimal charging path for the k-coverage charging problem.

To make sure that the computed charging path passes no more than one discretized vertex of a sensor node, we apply the color coding technique introduced in Alon et al. [1995] to assign each vertex a color. Specifically, we generate a coloring function $c_v : V \rightarrow \{1, ..., n\}$ that assigns each sensor node a unique node color. Each sensor node then passes the node color to its discretized ones. A path in G is said to be colorful if each vertex on it is colored by a distinct node color. It is obvious that a colorful path in G passes no more than one discretized vertex of a sensor node.

To take into the consideration of traveling distance from service station to individual sensor node, we add an extra vertex denoted as v_0 and connect it with directed edges to vertices in G, i.e., $\{v_i(t_i^k)|k = 0\}$. The length of edge $\overrightarrow{v_0 v_i(t_i^0)}$ is the Euclidean distance between the service station and sensor node v_i. The table T constructed in Sec. 5.4 is stored at v_0.

We first topologically sort the new graph, i.e., $G + v_0$. Then we start from v_0 to find colorful paths by traversing from left to right in linearized order. Specifically, v_0 checks neighbors connected with outgoing edges and sends table T to those contributing to the decrease of at least one table entry. Once a vertex $v_i(t_i^0)$ receives T, $v_i(t_i^0)$ checks the subregions within its sensing range and updates the corresponding entries of T. $v_i(t_i^0)$ also generates a color set $C = \{c(v_i(t_i^0))\}$ and stores with T, which indicates a colorful path of length $|C|$.

Similarly, suppose the algorithm has traversed to vertex $v_i(t_i^k)$, we check each color set C stored at $v_i(t_i^k)$ and its outgoing edge $v_j(t_j^{k'})$. If $c(v_j(t_j^{k'})) \notin C$ and charging $v_j(t_j^{k'})$ helps decrease at least one entry of T associated with C, we add the color set $C = \{C + c(v_j(t_j^{k'}))\}$ along with the updated T to the collection of $v_j(t_j^{k'})$.

After the update of the last vertex in linearized order, we check the stored Ts in each node and identify those with all zero entries. A color set associated with a T with all zero entries represents a colorful path that is a feasible solution of the k-coverage problem.

A path can be easily recovered from a color set. The basic idea is to start from vertex $v_i(t_i^k)$ with a color set C. We check the stored color sets of vertices connected to $v_i(t_i^k)$ with incoming edges. Assume we identify a neighbor node $v_j(t_j^{k'})$ storing a color set $C - c(v_i(t_i^k))$, then we continue to trace back the path from $v_j(t_j^{k'})$ with a color set $C - c(v_i(t_i^k))$. When we trace back to v_0, we have recovered the whole charging path. Among all feasible charging paths, the one with a minimal traveling distance is the optimal one.

Lemma 1. *The algorithm returns an optimal solution of the k-coverage charging problem, i.e., a feasible path maximizing the energy usage efficiency, if it exists.*

Proof. We first show that the algorithm returns a feasible path. A path returned by the algorithm is a colorful one that guarantees the path passes a sensor node no more than once. In the meantime, charging time at each sensor node along the path is before its deadline, otherwise a directed edge along the path won't exist. Array T with all zero entries makes sure that the k-coverage is maintained.

When the algorithm has traversed to the i^{th} node in linearized order, each colorful path passing through the node has been stored in the node. □

Note that the computational complexity of the dynamic programming algorithm can increase exponentially in the worst case because the stored color sets at a vertex can increase exponentially to the size of sensor nodes n.

6 Charger Scheduling Optimization Framework

We show the framework by four steps and use the three representative charger scheduling optimization problems discussed in Secs. 3, 4, and 5 as concrete examples to illustrate the algorithm design based on the framework.

6.1 Graph Construction

Given a set of sensor nodes represented by $V = \{v_i | 1 \leq i \leq n\}$ deployed over a planar region with the initial positions denoted by $P = \{p_i | 1 \leq i \leq n\}$, the sensing range of a sensor node v_i is r and the sensing model is a disk. Each node v_i is equipped with a rechargeable battery of capacity B. When the residual energy of v_i represented as $B_i(t)$ at time t is below a specific threshold, the sensor node v_i will send a charging request to a mobile charger. The charger collects all charging requests before leaving from base station denoted by v_0 at t_0. The average speed of the charger is s.

We use weighted graph $G(V, E, \omega)$ to model the charger scheduling optimization problems. Specifically,

Vertices. A vertex $v_i \in V(G)$ represents a sensor node sending charging request.

Edges. An edge $e(v_i, v_j) \in E(G)$ indicates a possible charging path of a charger from sensor node v_i to v_j without the violation of any constraints.

Edge Weight. The weight $\omega(v_i, v_j)$ assigned to edge $e(v_i, v_j)$ represents the Euclidean distance of nodes v_i and v_j.

More constraints are added to the graph model for each charger scheduling optimization problem. Specifically,

- **Mobile Network Charging Path Optimization Problem:** The graph is a fully connected undirected one with an edge connecting every pair of sensor nodes. The edge weight changes dynamically because the sensor nodes are mobile ones. With the assumption that the exact location of each mobile node is known to a charger at any time t, we discretize the time and update the graph in each time step.
- **Fully Charging Reward Maximization Problem:** The graph is a fully connected undirected one with an edge connecting every pair of sensor nodes. Each vertex v_i is additionally associated with a positive integer with range $[1, n^2]$ to model the gain of charging v_i by a mobile charger. A sensor with less residual energy is assigned a larger prize as it needs to be charged more urgently.
- **Optimal k-coverage Charging Problem:**
  Vertices. The vertex set $V(G)$ includes all the sensor nodes, i.e., $\{v_i | 1 \leq i \leq n\}$ and a start vertex denoted as v_0. Each vertex v_i has a deadline D_i. The location of v_0 is the service station and the deadline of v_0 is inf. Note that the deadline of sensor nodes without sending any request is set as 0.
  Edges. For any sensor nodes v_i and v_j in $V(G)$, d_{ij} is the euclidean distance between v_i and v_j. There exists an edge $\overrightarrow{v_i v_j}$ in $V(G)$ if and only if the inequality below holds

$$\frac{B - B_i(t_0)}{r_c} + \frac{d_{ij}}{s} \leq D_j \tag{6}$$

  where $B_i(t_0)$ denotes as the residual energy of sensor node i at charger departure time t_0, r_c is the energy transfer rate of the mobile charger and s is its average moving speed. If a charging path P goes from node v_i to node v_j, the charging time beginning at node v_j is given by Def. 5.

  Definition 9 (Feasible Path). *A path P in G is a feasible one if it starts from and ends at v_0, does not traverse repeated vertex, and charges vertices before their charging deadlines. At the same time, charging along P satisfies the k-coverage requirement of the given network.*

6.2 Graph Representation

As we have mentioned in Sec. 2.2, deep Q-learning applies parametrized function $Q(S, A; \Theta)$ to approximate a large-sized state-action value function $Q(S, A)$ that includes the combination of all pairs of states and actions, where $Q(S, A; \Theta)$ is parameterized by Θ with size much smaller than $Q(S, A)$. It is expected that the state-action value function $Q(S, A; \Theta)$ has well summarized the state S of the current problem solving, i.e., incorporating current partial solution including the selected vertices and their order into the graph model constructed in Sec. 6.1. The state-action value function $Q(S, A; \Theta)$ should also have an estimation of the reward value when a new vertex is picked considered as taking an action A in current state S.

However, it is challenging to accurately describe both S and A on a graph. They may depend on the global and local statistics of the current graph. There exist different graph representation techniques. We apply structure2vec Dai et al.

[2016], Khalil et al. [2017], a deep learning-based graph embedding technique, to compute a p-dimensional node embedding for each vertex and a feature vector with the same dimension for the graph. The state-action value function $Q(S, A; \Theta)$ can then be defined by the computed feature vector and node embedding parameterized by Θ. Parameters Θ will be learned later using deep reinforcement learning algorithm.

6.3 Deep Reinforcement Learning Framework

States, actions, transition, rewards, and policy are key components of any typical reinforcement learning algorithm. We add objective function, insertion function, and stop function into the deep reinforcement learning based charging framework. Before we discuss these key components of the framework, we will briefly explain some terms borrowed from reinforcement learning. An episode in the framework refers to a complete sequence of sensor nodes charged under constraints or requirements. A step, denoted by t, refers to charging a single sensor node in an episode.

Objective function: An objective function, denoted by f, reflects the quality of a partial solution to a charging problem. The definition of f varies from one charging problem to another, e.t., from maximizing the number of charging nodes to minimizing the traveling distance. To keep consistent, we turn f to negative when the problem requires minimizing some value.

Insertion function: An insertion function, denoted by g, is designed to insert vertex v to the best position in a partial solution s_{t+1}, e.g., $s_{t+1} := g(s_t, v)$.

Stop function: A stop function checks the problem constraints and determines when to stop the current episode.

States: A state s_t is an ordered list of visited vertices at time step t, represented as a p-dimensional vector using the structure2vec Dai et al. [2016], Khalil et al. [2017] graph representation technique. It is a partial solution $s_t \subseteq V(G)$. The initial state is denoted by $s_0 = (v_0)$. A stop function will determine the termination state s_{end}.

Actions: Actions include all the candidate vertices at state s_t. Such candidacy depends on the definition of the individual charging problem. We will discuss it later. Similar to a state, an action is represented as a p-dimension node embedding using the structure2vec Dai et al. [2016], Khalil et al. [2017] graph embedding technique.

Transition: Suppose v is a candidate vertex. After taking the action v at time step t, state s_t is transitioning to $s_{t+1} := g(s_t, v)$ where v is inserted to the best position by an insertion function g.

Rewards: A reward function, denoted by $r(s_t, v)$, reflects the change of the objective function f after taking action v at state s_t and transitioning to state s_{t+1}:

$$r(s_t, v) = f(s_{t+1}) - f(s_t). \tag{7}$$

It is obvious that the cumulative reward of an episode equals the objective function of the termination state, i.e. $\sum_{i=1}^{n} r(s_i, v_i) = f(s_{\text{end}})$ assuming the episode takes n steps.

Policy: We choose an epsilon greedy policy. The policy will choose an action achieving the current highest reward. However, with a small probability, it will instead randomly select the next action to prevent the stuck of a local optimal solution. Specifically, at time step t, an action v_t is selected by

$$v_t = \begin{cases} \arg\max_{v \in \bar{s}_t} Q(s_t, v; \Theta), & \text{with probability } 1 - \epsilon \\ \text{select a random vertex } v_t \in \bar{s}_t, & \text{otherwise} \end{cases} \tag{8}$$

We will illustrate how to tailor the model for each of the selected charger scheduling optimization problems.

6.3.1 Mobile Network Charging Path Optimization Problem:

Objective function: The objective function f counts the number of selected vertices, i.e., charged mobile nodes.

Insertion function: The Insertion function g simply inserts the selected vertex v at time step t to the end of state s_t.

Stop function: The stop function checks s_{t+1} to make sure that the total charging time is no larger than a given maximum timespan of the charger. Otherwise, the selected vertex v will be removed.

Actions: Actions at time step t include all non-selected vertices in current state s_t.

Rewards: The reward function $r(s_t, v)$ is defined as

$$r(s_t, v) = 1 \tag{9}$$

because inserting one vertex to a partial solution contributes the objective function f by one count.

6.3.2 Fully Charging Reward Maximization Problem:

Objective function: The objective function f counts the total prizes collected from selected vertices, i.e., mobile nodes charged. The prize at each vertex is proportional to the energy charged to its full capacity.

Insertion function: The Insertion function g finds a position to insert v at s_t that minimizes the energy of charger spent on road, which is defined as:

$$g(s_t, v) = \arg\min_{i} \{E_{0i} + E_{it}\} \tag{10}$$

where $E_{0i} + E_{it}$ is the energy of charger spent on path when charging vertex v as the i-th one among the selected vertices.

Stop function: The stop function checks s_{t+1} to make sure that the total amount of energy consumed on sensor charging and the traveling is no greater than the energy capacity of a mobile charger . Otherwise, the selected vertex v will be removed.

Actions: Actions at time step t include all non-selected vertices in current state s_t.

Rewards: The reward function $r(s_t, v)$ is defined as

$$r(s_t, v) = E_v \tag{11}$$

where E_v is the energy charged to get the full capacity of v.

6.3.3 Optimal k-coverage Charging Problem:

Objective function: The objective function f counts the total traveling distance of a charger. To maximize f, we turn f to negative.

Insertion function: The Insertion function g finds a position to insert v at s_t that maximizes the reward and ensures each node charged before its deadline.

Stop function: : The stop function checks whether the current charging path guarantees the k-coverage requirement of the area. If the requirement is satisfied or R(S,v) is -inf, the algorithm terminates.

States: A state S is a partial solution $S \subseteq V(G)$, an ordered list of visited vertices. The first vertex in S is v_0.

Actions: Let $\bar{S}$ contain vertices not in S and has at least one edge from vertices in S. An action is a vertex v from $\bar{S}$ returning the maximum reward. After taking the action v, the partial solution S is updated as

$$S' := (S, v), \text{where } v = \arg\max_{v \in \bar{S}} Q(S, v) \tag{12}$$

(S, v) denotes appending v to the best position after v_0 in S that introduces the least traveling distance and maintains all vertices in the new list a valid charging time.

Rewards: The reward function $R(S, v)$ is defined as the change of the traveling distance when taking the action v and transitioning from the state S to a new one S'. Assume v_i, v_j are two adjacent vertex in S, v_0 is the first vertex in the S, and v_t is the last vertex in the S.

The reward function $R(S, v_k)$ is defined as follows:

$$R(S, v_k) = \begin{cases} -\min(d_{ik} + d_{kj} - d_{ij}, d_{tk} + d_{k0} - d_{t0}), & t_{S'}(v) \neq \text{inf} \\ -\text{inf}, & \text{otherwise} \end{cases} \tag{13}$$

where d_{ij} is the euclidean distance between nodes v_i and v_j, and $t_{S'}(v)$ is the updated charging time of node in path formed by S' after inserting the v_k.

6.4 Deep-Q-Network (DQN) Algorithm

We adopt the deep-Q-network algorithm introduced in Khalil et al. [2017] to learn the parameters Θ of the state-action value function $Q(S, v; \Theta)$. The adopted algorithm updates the parameters Θ every n steps instead of every single step, to have a more accurate estimation of the objective function. Particularly, we apply experience replay method in Mnih et al. [2013] to update Θ. Experience learned in each step is stored in a dataset. When we update Θ, a sample batch with size 32 is randomly selected from the dataset. Experience replay method helps break data correlation and avoid oscillations with the parameters.

The input of the network is the p-dimension vector featured by graph embedding network and output is the optimal solution for the current state. The advantage of DQN is that it can handle delayed rewards, which represent the way to optimize the objective function. In each step of the algorithm, the graph embedding method will be used to update the current partial solution and the new p-dimension vector which contains the newest information will be used for the next step.

Algorithm 1 summaries the major steps of the algorithm.

Algorithm 1 DQN Algorithm

```
1: Initialize replay memory $\mathcal{H}$ to capacity $C$
2: **for** each episode **do**
3:   Initialize state $s_1 = (v_0)$
4:   **for** step $t = 1$ **to** $n$ **do**
5:     Select $v_t$ by (8)
6:     Add $v_t$ to partial solution by insertion function: $s_{t+1} := g(s_t, v)$
7:     Calculate reward $r(s_t, v)$
8:     Store tuple $(s_t, v, r(s_t, v), s_{t+1})$ to $\mathcal{H}$
9:     Sample random batch $(s_l, v_l, r(s_l, v_l), s_{l+1})$ from $\mathcal{H}$
10:    Update the network parameter $\Theta$ by squared loss function $(y_l - Q(s_l, v_l; \Theta))^2$, where
```

$$y_l = \begin{cases} r(s_l, v) + \gamma \max_{v'} Q(s_{l+1}, v'; \Theta) \\ \qquad \text{if } s_{l+1} \text{ non-terminal and } v' \in \bar{s}_l \\ r(s_l, v) \\ \qquad \text{otherwise} \end{cases} \tag{14}$$

```
11:    **if** $s_{t+1}$ satisfy the stop function **then**
12:      Break
13:    **end if**
14:  **end for**
15: **end for**
```

7 Performance Comparison

We implement and compare the performance of DQN algorithms designed based on the proposed deep reinforcement learning-based charger scheduling optimization framework with existing ones on the selected charger scheduling optimization problems.

7.1 Mobile Network Charging Path Optimization Problem

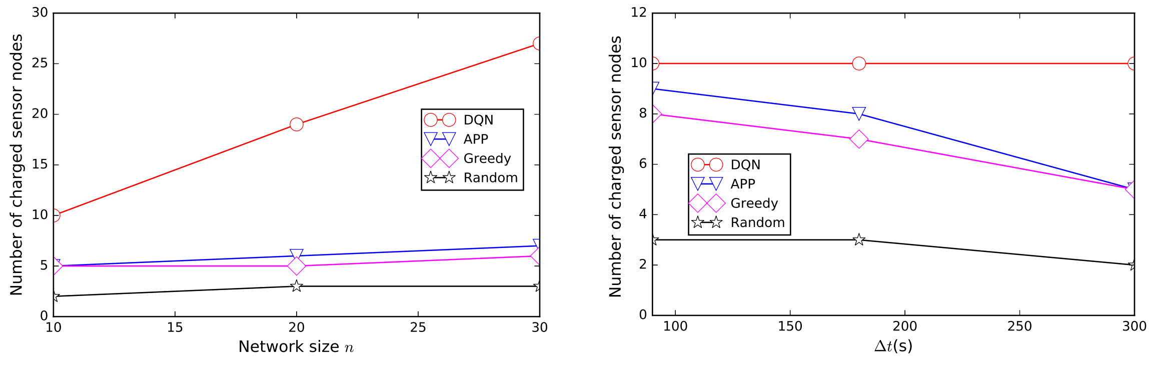
We compare the DQN algorithm designed based on the proposed framework with the quasi-polynomial time approximation algorithm (APP) introduced in Chen et al. [2016] and two other heuristic algorithms, including a greedy one and a random one.

However, the computing time of the APP algorithm in Chen et al. [2016] is very sensitive to the network size n, time step size Δt, and charging timespan C and can increase dramatically to days with large n, small Δt, or long C.

To control the running time, we choose networks with moderate sizes and a small timespan of charging $C = 30$ minutes. We compare the computing time and the number of sensors charged of these algorithms and study the impact of varying network size and charger capacity on the performances in Secs. 7.1.2 and 7.1.3, respectively. Note that we choose a low recursion level $L = 3$ for the APP algorithm in Chen et al. [2016] to control the running time.

7.1.1 Simulation Setting

We deploy sensors by uniform random distribution with the size n varying from 10 to 30 in an Euclidean square with size $[100, 100]m^2$. A charger starts from the center of monitored area and ends at the upper right corner. The speed of charger is 5m/s. We use the random way point model to simulate the trace of a mobile sensor node with its average speed randomly chosen from $[0, 2]$ m/s.



(a) Impact of network size (b) Impact of time step size

Figure 1: (a) The number of mobile sensors charged with the maximum timespan of charger $C = 30$ minutes and $\Delta t = 300$s under a varying network size of n. (b) The number of mobile sensors charged with the maximum timespan of charger $C = 30$ minutes and $n = 10$ under a varying time step size.

The battery capacity B of each sensor is 10.8KJ Chen et al. [2016]. The initial battery level of node i is randomly chosen from $[0, B]$. $g_i[(1-\epsilon)\alpha]$ is the time to charge node i to $(1-\epsilon)\alpha$. The required charging level $\alpha = 90\%$, $\epsilon = 0.1$. The average energy transfer rate is 40 W.

7.1.2 Impact of Network Size

Figure 1(a) and Table 1 compare the number of mobile sensors charged within the maximum timespan and the corresponding computing time of each algorithm, respectively, with $\Delta t = 300$s and a varying network size of n. With an increased network size, more mobile sensors can be charged, but the DQN algorithm consistently charges a significantly higher number of mobile sensors than any other algorithm. At the same time, the computing time of the DQN algorithm remains stable with an increase of n. By contrast, the computing time of the APP algorithm in Chen et al. [2016] increases dramatically. Overall, the DQN algorithm outperforms all others.

7.1.3 Impact of Time Step Size

The time step size Δt has a big impact on the performance of the APP algorithm in Chen et al. [2016] as shown in Figure 1(b) and Table 1. The number of charged mobile sensors of the APP algorithm in Chen et al. [2016] increases with a decreased time step size Δt but at the cost of sky-high computing time. On the contrary, Δt has no effect on the DQN algorithm because of the way to construct a graph as introduced in Sec.6.1. The DQN algorithm again outperforms all other algorithms.

7.2 Fully Charging Reward Maximization Problem

Paper Liang et al. [2017] provides a 4-approximation algorithm to solve the fully charging reward maximization problem by reducing the original problem to a classical orienteering one Bansal et al. [2004].

We compare the DQN algorithm designed based on the proposed framework with the approximation algorithm (APP) in Liang et al. [2017], the minimum spanning tree (MST), the Capacitated Minimum Spanning Tree (CMST), and the greedy one. We compare the computing time and the total energy spent on charging sensors and study the impact of varying network size and charger capacity on the performances of these algorithms in Secs. 7.2.2 and 7.2.3, respectively.

7.2.1 Simulation Setting

The simulation area is an Euclidean square $[1000, 1000]m^2$. We randomly deploy sensors with size n ranging from 50 to 200. A mobile charger with energy capacity $IE = 300$KJ, travels with an average speed 5m/s from the center of monitored area and back to it with an average 600 J/m spent on traveling Liang et al. [2017]. The battery capacity B of each sensor is 10.8KJ Liang et al. [2017]. The initial battery level of node v_i is randomly chosen from $(0, B]$. A sensor node sends a charging request when its energy is below 20% of its capacity B.

Table 1: Computing time under different network size n and time stepsize Δt

	Computing time in Fig. 1(a)		Computing time in Fig. 1(b)	
Alg.	n	Comp. Time (s)	Δt (s)	Comp. Time (s)
DQN	10	22	90	22
APP		580		417329
Greedy		0.01		0.033
Random		0.0006		0.0007
DQN	20	53	180	22
APP		28011		7507
Greedy		0.044		0.02
Random		0.0024		0.0006
DQN	30	70	300	22
APP		232080		580
Greedy		0.11		0.01
Random		0.005		0.0006

7.2.2 Impact of Network Size

Figure 2(a) and Table 2 compare the energy spent on sensor charging and the corresponding computing time of each algorithm, respectively, with the network size n increased from 50 to 200. It is clear that the DQN algorithm spends much more energy in charging sensors than any other algorithm. Considering a charger with a fixed energy capacity and a group of sensors scattered in a field, the energy spent on charging sensors decreases with an increased network size for all algorithms. However, with the increased network size, the computing time of the APP algorithm in Liang et al. [2017] increases dramatically while the DQN algorithm remains stable. Overall, the DQN algorithm outperforms all other algorithms.

7.2.3 Impact of Charger Capacity

Figure 2(b) and Table 2 compare the energy spent on sensor charging and the corresponding computing time of each algorithm, respectively, with a varying charger capacity. With the charger capacity IE increased from 200KJ to 350KJ, the energy spent on charging sensor nodes increases for all algorithms too. The DQN algorithm consistently charges much more energy on sensors than any other algorithms. At the same time, the computing time of the DQN algorithm keeps stable while the APP algorithm increases dramatically. Overall, the DQN algorithm still outperforms all other algorithms.

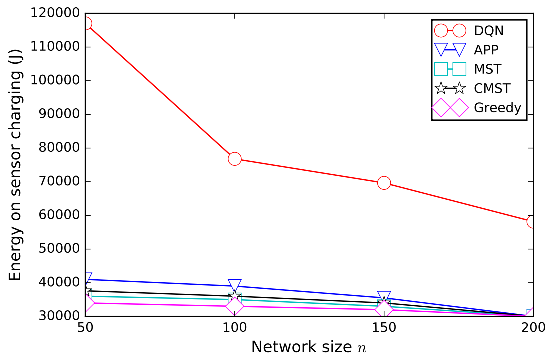
7.3 Optimal k-coverage Charging Problem

We evaluate the DQN algorithm designed based on the proposed framework and compare with the dynamic programming based algorithm and three other heuristic algorithms including Ant Colony System (ACS) based algorithm, Random algorithm, and Greedy algorithm, as explained briefly in Sec. 7.3.2.

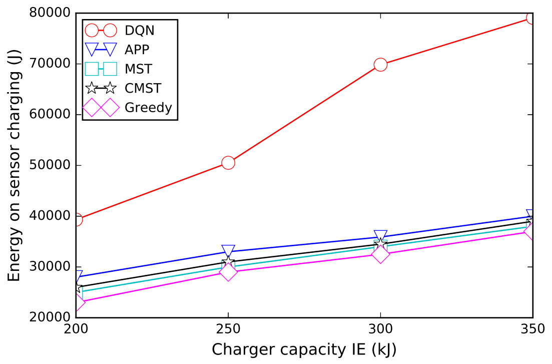
We compare the computing time to find a feasible charging path and the energy of a charger spent on traveling of these algorithms. Considering network settings may affect the performance, we study the impact of varying network size n, coverage requirement k, and remaining energy threshold α in Secs. 7.3.3, 7.3.4, and 7.3.5, respectively. Specifically, we assume a monitored area is at least k-coverage initially. Sensor node v_i estimates the charging deadline D_i based on its residual energy $B_i(t_0)$ and the experimental energy consumption rate in Zhu et al. [2009].

7.3.1 Simulation Setting

We set up an Euclidean square $[500, 500]m^2$ as a simulation area and randomly deploy sensor nodes with size ranging from 32 to 80 in the square such that the area is at least k-coverage initially where k varies from 2 to 4. The sensing range r is 135m. The base and service station of charger are co-located in the center of the square. A charger with a



(a) Impact of network size



(b) Impact of charger capacity

Figure 2: (a) Energy spent on sensor charging for a mobile charger with a total energy capacity $IE = 300$KJ under a varying network size of n. (b) Energy spent on sensor charging for a mobile charger with a varying energy capacity IE and a fixed network size $n = 120$.

Table 2: Computing time under different network size n and charger capacity IE

Alg.	Computing time in Fig. 2(a)		Computing time in Fig. 2(b)	
	n	Comp. Time (s)	IE (kJ)	Comp. Time (s)
DQN	50	31	200	67
APP		35		90
MST		0.2		0.39
CMST		1.2		2.35
Greedy		0.0005		0.0004
DQN	100	57	250	67
APP		506		102
MST		0.35		0.4
CMST		2.45		2.37
Greedy		0.0007		0.0004
DQN	150	87	300	68
APP		1005		605
MST		0.46		0.42
CMST		3.56		2.95
Greedy		0.0009		0.0004
DQN	200	115	350	70
APP		3500		807
MST		0.58		0.49
CMST		4.28		2.97
Greedy		0.001		0.0004

starting point from the service station has an average traveling speed 5m/s and consumes energy $600J/m$ Liang et al. [2017]. The battery capacity B of each sensor is 10.8KJ Liang et al. [2017]. The remaining energy threshold α vary from 0.2 to 0.8. A sensor sends a charging request before the leaving of the charger from the service station. The sensor will include in the request the estimated energy exhausted time based on its current residual energy and energy consumption rate. To simulate such request, we consider the residual battery of a sensor is a uniform random variable B_i between $(0.54, 10.8]$KJ Shi et al. [2011]. The energy exhausted time $D_i = B_i/\beta_i$. We choose the energy consumption rate β_i from the historical record of real sensors in Zhu et al. [2009] where the rate is varying according to the remaining energy and arrival charging time to the sensor. The energy transfer rate r_c is 20W Chen et al. [2016]. The discredited time step-size is 1s.

7.3.2 Comparison Algorithms

We implement three heuristic algorithms for comparison: Ant Colony System (ACS) based algorithm, Random algorithm, and Greedy algorithm.

ACS algorithm solves the traveling salesmen problem with an approach similar to the foraging behavior of real ants Colorni et al. [1991], Dorigo and Gambardella [1997], Gutjahr [2000]. Ants seek path from their nest to food sources and leave a chemical substance called pheromone along the paths they traverse. Later ants sense the pheromone left by earlier ones and tend to follow a trail with a stronger pheromone. Over a period of time, the shorter paths between the nest and food sources are likely to be traveled more often than the longer ones. Therefore, shorter paths accumulate more pheromone, reinforcing these paths.

Similarly, ACS algorithm places agents at some vertices of a graph. Each agent performs a series of random moves from current vertex to a neighboring one based on the transition probability of the connecting edge. After an agent has finished its tour, the length of tour is calculated and the local pheromone amounts of edges along the tour are updated based on the quality of the tour. After all agents have finished their tours, the shortest one is chosen and the global pheromone amounts of edges along the tour are updated. The procedure continues until certain criteria are satisfied. When applying ACS algorithm to solve the the traveling salesmen problem with deadline, two local heuristic functions are introduced in Cheng and Mao [2007] to exclude paths that violate the deadline constraints.

We modify ACS algorithm introduced in Cheng and Mao [2007] for the optimal k-coverage charging problem. Agents start from and end at v_0. Denote $\tau_{ij}(t)$ the amount of global pheromone deposited on edge $\overrightarrow{v_i v_j}$ and $\Delta\tau_{ij}(t)$ the increased amount at the t^{th} iteration. $\Delta\tau_{ij}(t)$ is defined as

$$\Delta\tau_{ij}(t) = \begin{cases} \frac{1}{L^*} & \text{if } \overrightarrow{v_i v_j} \in P^* \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

where L^* is the traveling distance of the shortest feasible tour P^* at the t^{th} iteration. Global pheromone $\tau_{ij}(t)$ is updated according to the following equation:

$$\tau_{ij}(t) = (1-\theta)\tau_{ij}(t-1) + \theta\Delta\tau_{ij}(t), \tag{16}$$

where θ is the global pheromone decay parameter. The local pheromone is updated in a similar way, where θ is replaced by a local pheromone decay parameter and $\Delta\tau_{ij}(t)$ is set as the initial pheromone value.

We also modify the stop criteria of one agent such that the traveling path satisfies the requirement of k-coverage, or the traveling time of current path is inf, or the agent is stuck at a vertex based on the transition rule.

Random and Greedy algorithms work much more straightforward. The Random algorithm randomly chooses a next node not in the same clique of the existing path and with an outgoing edge from current one to charge. The Greedy algorithm always chooses the nearest node not in the same clique of the existing path with an outgoing edge from current one. Random and Greedy algorithms terminate when they either find a feasible path or are locally stuck. Note that for ACS and Random algorithms, we always run multiple times and choose the best solution.

7.3.3 Impact of Network Size

We set the coverage requirement $k = 3$ and the remaining energy threshold $\alpha = 0.45$ for a sensor node to send a charging request. Table 3 compares the performances including the computation time to find a feasible charging path and the energy spent on traveling when the network size n varies from 48 to 80. The energy spent on traveling decreases with an increased n since there are more redundant sensors to maintain the k-coverage requirement. Again, the DQN algorithm outperforms all other competing algorithms.

Table 3: Performance comparison under different sizes of sensor network n

Algorithm	n	$k = 3$, $\alpha = 0.45$		
		Computation Time (s)	Feasible Path Found	Traveling Energy (kJ)
Dynamic	48	156700	Yes	771
DQN		83	Yes	771
ACS		92	Yes	951
Random		0.0004	Yes	2085
Greedy		0.0004	Yes	888
Dynamic	64	455	Yes	702
DQN		20	Yes	702
ACS		19	Yes	702
Random		0.0003	No	–
Greedy		0.0003	Yes	846
Dynamic	72	362	Yes	567
DQN		32	Yes	567
ACS		57	Yes	567
Random		0.0003	Yes	1941
Greedy		0.0003	Yes	810
Dynamic	80	268	Yes	345
DQN		20	Yes	345
ACS		18	Yes	345
Random		0.0003	No	–
Greedy		0.0003	Yes	375

7.3.4 Impact of Coverage Requirement

We set the number of sensor nodes $n = 64$ and the remaining energy threshold $\alpha = 0.45$ for a sensor node to send a charging request. Table 4 compares the performances including the computation time to find a feasible charging path and the energy spent on traveling when the coverage requirement k varies from 2 to 4. The traveling energy increases with the increased k because more sensor nodes need to be charged to satisfy the coverage requirement. The dynamic programming algorithm with an exponentially increased computing time can only find a feasible charging path when k is small. By contrast, the computing time of the DQN algorithm including its training time grows slowly with the increase of k. The ACS algorithm performs better than the random and greedy algorithms, but overall, the performance of the DQN algorithm significantly outperforms all other competing algorithms.

7.3.5 Impact of Remaining Energy Threshold

Tables 5 and 6 give the performances of different algorithms with the remaining energy threshold α varying from 0.2 to 0.8 under two network settings: $n = 32$ and $k = 2$, and $n = 48$ and $k = 3$, respectively. With the increased remaining energy threshold, the traveling energy increases in both network settings. The Random and Greedy algorithms fail to detect a feasible path in many cases. The dynamic programming algorithm runs out of the memory when α is large. Both the ACS and DQN algorithms find feasible paths for all cases. However, the performance of the ACS algorithm decreases with the increase of α. The DQN algorithm consistently and significantly outperforms all other comparison algorithms including botht the traveling energy and computing time.

8 Conclusions

We introduce a deep reinforcement learning-based charger scheduling optimization framework. The biggest advantage of the framework is that a diverse range of domain-specific charger scheduling strategy can be learned automatically from previous experiences, i.e., different graphs with various sizes. A framework also simplifies the complexity of algorithm design for individual charger scheduling optimization problem. We compare the performance of algorithms designed based on the proposed framework with existing ones on a set of representative charger scheduling optimiza-

Table 4: Performance comparison under different coverage requirement k

Algorithm	k	$n = 64$, $\alpha = 0.45$ Computation Time (s)	Feasible Path Found	Traveling Energy (kJ)
Dynamic	2	0.102	Yes	249
DQN		16	Yes	249
ACS		5	Yes	249
Random		0.0006	Yes	249
Greedy		0.0007	Yes	288
Dynamic	3	455	Yes	702
DQN		20	Yes	702
ACS		19	Yes	702
Random		0.0003	No	–
Greedy		0.0003	Yes	846
Dynamic	4	–	–	–
DQN		71	Yes	1089
ACS		73	Yes	1188
Random		0.0006	No	–
Greedy		0.0005	Yes	1254

Table 5: Performance comparison under different remaining energy threshold α when $k = 2$, $n = 32$

Algorithm	α	$k = 2$, $n = 32$ Computation Time (s)	Feasible Path Found	Traveling Energy (kJ)
Dynamic	0.2	0.0012	Yes	405
DQN		13	Yes	405
ACS		3	Yes	405
Random		0.0002	No	–
Greedy		0.0003	Yes	420
Dynamic	0.4	9760	Yes	696
DQN		26	Yes	696
ACS		38	Yes	855
Random		0.0003	No	–
Greedy		0.0003	No	891
Dynamic	0.6	135742	Yes	1071
DQN		116	Yes	1071
ACS		232	Yes	2289
Random		0.0006	No	–
Greedy		0.0004	No	–
Dynamic	0.8	–	–	–
DQN		136	Yes	1080
ACS		400	Yes	2544
Random		0.002	No	–
Greedy		0.001	No	–

Table 6: Performance comparison under different remaining energy threshold α when $k = 3$, $n = 48$

Algorithm	α	$k = 3$, $n = 48$		
		Computation Time (s)	Feasible Path Found	Traveling Energy (kJ)
Dynamic	0.2	0.02	Yes	348
DQN		10	Yes	348
ACS		1	Yes	348
Random		0.0003	No	–
Greedy		0.0002	Yes	348
Dynamic	0.4	145602	Yes	750
DQN		81	Yes	750
ACS		90	Yes	930
Random		0.0004	Yes	2070
Greedy		0.0004	Yes	870
Dynamic	0.6	–	–	–
DQN		138	Yes	1020
ACS		466	Yes	1521
Random		0.001	No	–
Greedy		0.001	No	–
Dynamic	0.8	–	–	–
DQN		481	Yes	1272
ACS		4200	Yes	4788
Random		0.01	No	–
Greedy		0.02	No	–

tion problems. Extensive simulation and comparison results show that algorithms based on the proposed framework outperform all existing ones and achieve close to optimal results.

References

Weifa Liang, Zichuan Xu, Wenzheng Xu, Jiugen Shi, Guoqiang Mao, and Sajal K Das. Approximation algorithms for charging reward maximization in rechargeable sensor networks via a mobile charger. *IEEE/ACM Transactions on Networking*, 25(5):3161–3174, 2017.

Lin Chen, Shan Lin, and Hua Huang. Charge me if you can: Charging path optimization and scheduling in mobile networks. In *Proceedings of the 17th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 101–110. ACM, 2016.

Yi Shi, Liguang Xie, Y Thomas Hou, and Hanif D Sherali. On renewable sensor networks with wireless energy transfer. In *INFOCOM, 2011 Proceedings IEEE*, pages 1350–1358. IEEE, 2011.

Liguang Xie, Yi Shi, Y Thomas Hou, Wenjing Lou, Hanif D Sherali, and Scott F Midkiff. On renewable sensor networks with wireless energy transfer: The multi-node case. In *Sensor, mesh and ad hoc communications and networks (SECON), 2012 9th annual IEEE communications society conference on*, pages 10–18. IEEE, 2012.

Liang He, Peng Cheng, Yu Gu, Jianping Pan, Ting Zhu, and Cong Liu. Mobile-to-mobile energy replenishment in mission-critical robotic sensor networks. In *INFOCOM, 2014 Proceedings IEEE*, pages 1195–1203. IEEE, 2014.

Xiao Lu, Ping Wang, Dusit Niyato, Dong In Kim, and Zhu Han. Wireless charging technologies: Fundamentals, standards, and network applications. *IEEE Communications Surveys & Tutorials*, 18(2):1413–1452, 2015.

Xuan Li and Miao Jin. Optimal k-coverage charging problem. *arXiv preprint,arXiv:1901.09129*, 2019.

Irwan Bello, Hieu Pham, Quoc V Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2016.

Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. In *Advances in Neural Information Processing Systems*, pages 6351–6361, 2017.

Hanjun Dai, Bo Dai, and Le Song. Discriminative embeddings of latent variable models for structured data. In *International conference on machine learning*, pages 2702–2711, 2016.

Yu Ma, Weifa Liang, and Wenzheng Xu. Charging utility maximization in wireless rechargeable sensor networks by charging multiple sensors simultaneously. *IEEE/ACM Transactions on Networking*, 26(4):1591–1604, 2018.

Wenzheng Xu, Weifa Liang, Xiaohua Jia, Haibin Kan, Yinlong Xu, and Xinming Zhang. Minimizing the maximum charging delay of multiple mobile chargers under the multi-node energy charging scheme. *IEEE Transactions on Mobile Computing*, 2020.

Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*. MIT press, 1998.

Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

Martin Riedmiller. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *European Conference on Machine Learning*, pages 317–328. Springer, 2005.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Nikhil Bansal, Avrim Blum, Shuchi Chawla, and Adam Meyerson. Approximation algorithms for deadline-tsp and vehicle routing with time-windows. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 166–174. ACM, 2004.

Shuhui Yang, Fei Dai, Mihaela Cardei, Jie Wu, and Floyd Patterson. On connected multiple point coverage in wireless sensor networks. *Journal of Wireless Information Networks*, 2006, 2006.

Gyula Simon, Miklós Molnár, László Gönczy, and Bernard Cousin. Dependable k-coverage algorithms for sensor networks. In *Proc. of IMTC*, 2007.

Martin Liggins, David Hall, and James Llinas. *Handbook of Multisensor Data Fusion: Theory and Practice, Second Edition*. CRC Press, 2008.

S.R. Das Z. Zhou and H. Gupta. Variable radii connected sensor cover in sensor networks. *ACM Trans. Senor Networks*, 5(1):8:1–8:36, 2009.

X. Bai, Z. Yun, D. Xuan, B. Chen, and W. Zhao. Optimal multiple-coverage of sensor networks. In *INFOCOM, 2011 Proceedings IEEE*, page 2498–2506. IEEE, 2011.

Feng Li, Jun Luo, Wenping Wang, and Ying He. Autonomous deployment for load balancing k-surface coverage in sensor networks. *IEEE Transactions on Wireless Communications*, 14(1):279–293, 2015.

Santosh Kumar, Ten H. Lai, and József Balogh. On k-coverage in a mostly sleeping sensor network. In *Proceedings of the 10th Annual International Conference on Mobile Computing and Networking*, MobiCom '04, pages 144–158, 2004.

M. Hefeeda and M. Bagheri. Randomized k-coverage algorithms for dense sensor networks. In *INFOCOM, 2007 Proceedings IEEE*, pages 2376–2380, 2007.

Andre Kurs. *Power transfer through strongly coupled resonances*. PhD thesis, Massachusetts Institute of Technology, 2007.

Martin WP Savelsbergh. Local search in routing problems with time windows. *Annals of Operations research*, 4(1): 285–305, 1985.

Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *Journal of the ACM (JACM)*, 42(4):844–856, 1995.

Ting Zhu, Ziguo Zhong, Yu Gu, Tian He, and Zhi-Li Zhang. Leakage-aware energy synchronization for wireless sensor networks. In *Proceedings of the 7th International Conference on Mobile Systems, Applications, and Services*, pages 319–332, 2009.

A. Colorni, M. Dorigo, and V. Maniezzo. Distributed optimization by ant colonies. In *Proceedings of ECAL91 - European Conference on Artificial Life*, page 134–142, 1991.

Marco Dorigo and Luca Maria Gambardella. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on evolutionary computation*, 1(1):53–66, 1997.

Walter J. Gutjahr. A graph-based ant system and its convergence. *Future Gener. Comput. Syst.*, 16(9):873–888, 2000.

Chi-Bin Cheng and Chun-Pin Mao. A modified ant colony system for solving the travelling salesman problem with time windows. *Mathematical and Computer Modelling*, 46(9-10):1225–1235, 2007.