Charger Scheduling Optimization Framework

Xuan Li
Center for Advanced Computer Studies
University of Louisiana at Lafayette
Lafayette, LA 70504

Miao Jin
Center for Advanced Computer Studies
University of Louisiana at Lafayette
Lafayette, LA 70504
miao.jin@louisiana.edu

***Abstract*—Wireless Rechargeable Sensor Network (WRSN), consisting of sensors with rechargeable batteries and mobile chargers, has become a promising solution to the energy limitation problem in Wireless Sensor Networks (WSNs). Charger scheduling optimization focuses on optimizing the trajectory of mobile chargers to prolong the life of a WRSN system. Charger scheduling optimization problems are in general NP-hard. Previous solutions using traditional algorithms often require problem-specific design and a trade-off between the performance and computing time. An insight into these optimization problems is that a domain-specific charger scheduling strategy could be learned automatically when the objective function of an optimization problem is considered as a reward. We model charger scheduling optimization problems using a weighted graph and consider the objective function as a cumulative reward of charging sensors along a charging path in one cycle. We then build a deep reinforcement learning based framework to solve a diverse range of charger scheduling optimization problems. The biggest advantage of the framework is that an optimal charger scheduling strategy can be learned 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 based on the proposed framework with traditional ones on a set of selected charger scheduling optimization problems. They outperform all existing algorithms.**

***Index Terms*—Wireless rechargeable sensor networks, Mobile charger scheduling, Deep reinforcement learning**

I. 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 node and failure of network task, considering a mobile charger needs to travel close to a sensor node to charge. Therefore, charger scheduling optimization has become a popular research area 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 types. One provides a budget (e.g., the total charging time, the total energy spent on charging and traveling). A charger seeks a path to maximize an objective function (e.g., the total energy charged to nodes, the total number of nodes charged) [1], [2]. The other gives a number of sensor nodes to be charged. A charger seeks a path to minimize an objective function (e.g., the total energy spent on the road, the total travel distance, the total charging time) [3]–[7].

Charger scheduling optimization problems are in general NP-hard. They are solved by exact, approximation, or heuristics algorithms. Exact algorithms use enumeration strategy but fail under the large size of a dataset. Approximation algorithms are desirable solutions with provable guarantees on the distance of the returned solution to the optimal one. However, approximation algorithms do not always exist for charger scheduling optimization problems. The computational complexity of approximation algorithms may also go sky-high with the large size of a dataset or high requirement of optimality. Heuristic algorithms are alternative solutions, fast to compute but lack of optimality guarantee. Overall, previous solutions with traditional algorithms often require problem-specific design and a trade-off between the performance and computing time.

It is obvious that traditional algorithm design fails to exploit the common characteristic of these charger scheduling optimization problems. An insight into these problems is that a domain-specific charger scheduling strategy could be learned automatically when the objective function of an optimization problem is considered as a reward. We model charger scheduling optimization problems using 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 based framework to tackle a diverse range of charger scheduling optimization problems. The biggest advantage of the framework is that an optimal charger scheduling strategy can be learned 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.

Specifically, 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 [8], [9] to compute a p-dimensional feature vector and node embedding to represent the weighted graph and a single vertex.

Deep reinforcement learning based framework: The framework includes the key components of a typical reinforcement learning algorithm and functions designed for charger

Xuan Li and Miao Jin are partially supported by NSF CNS 1320931.

978-1-7281-2522-0/19/$31.00

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: Sec. II gives a brief review of charger scheduling optimization problems and the introduction of the basic concept of deep reinforcement learning. Sec. III provides in detail the proposed framework to tackle charger scheduling optimization problems. Sec. IV-B presents the simulation and comparison results. Sec. V concludes the paper.

II. Related Works

A. Charger Scheduling Optimization

Charger scheduling optimization has been a popular research area 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 types.

The first type is that a charger with a given budget (e.g., the total charging time, 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, the total number of nodes charged) [1], [2]. In [1], a charger with energy bound seeks a path to maximize the charging rewards. The authors consider two scenarios: the sensors can be charged to full capacity one time or a certain energy level several times. A 4-approximation algorithm is given in this paper. In [2], a charger seeks a path that maximizes the number of sensor nodes charged within a given charging time. The authors consider mobile sensor nodes, so they discretize the moving trajectory of each sensor node according to its moving time. A quasi-polynomial time approximation algorithm is provided in the paper.

The second type is that a charger needs to charge a number of given 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, the total charging time) [3]–[7]. A solution may not exist for problems of this category. In [3], 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. They propose a heuristic solution. In [4], the authors consider the multi-node wireless energy transfer technology in WRSNs and extend the study in [3] to a scenario that multiple nodes can be charged at the same time. In [5], the authors provide a tree-based charging schedule for robotic sensor networks to minimize the travel distance of the charger. To guarantee the charging schedule depletion free for any robot, the authors provide theoretical guidance on the setting of the remaining energy threshold at which the robots request energy replenishment. In [6], the authors consider minimizing both the travel distance of the charger and the charging delay of sensor nodes by a set of nested Traveling Salesman Problem (TSP). In [7], a mobile charger seeks a path to charge sensor nodes before their charging deadlines under the constraint of maintaining the k-coverage ability of a sensor network on the monitored area, at the same time, minimizing the total travel distance. The authors introduce a dynamic programming based solution that suffers high computational complexity when the size of WRSNs increases.

B. 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 [10]. In each time step t, the agent observes current state s_t, and chooses action a. Then the state of the environment transits to state 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 s_{t+1} if the agent take action a at state s_t. The agent learns to select an action and try to 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 at state S 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 reinformed 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 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 for 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. The function approximation technique [11] is commonly used. In 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) [12] applies deep neural networks as function approximators, combined with different techniques including the experience replay method [13]. Considering the exponentially increased size of station-action pairs in charger scheduling optimization problems, we choose DQN to build the framework to solve a diverse range of Charger Scheduling Optimization problems in WRSNs.

III. Charger Scheduling Optimization Framework

We show the framework by four steps and use three representative charger scheduling optimization problems as concrete examples to illustrate the algorithm design based on the framework.

A. Selected Charger Scheduling Optimization Problems

We illustrate our framework using three representative charger scheduling optimization problems.

- **Mobile Network Charging Path Optimization Problem [2]:** Given a network composed of a set of mobile nodes, the mobile network charging path optimization problem is to find a closed tour for a mobile charger to maximize the number of charged nodes under the constraints of a required charging level at each mobile node and a maximum timespan of the charger. Note that the mobility pattern of the mobile nodes (i.e., their moving trajectories) are known to the charger.
- **Fully Charging Reward Maximization Problem [1]:** Given a static sensor network with a subset of sensor nodes sending charging requests, the fully charging reward maximization problem is to find a closed tour for a mobile charger such that the sum of prizes collected from all charged sensors is maximized, subject to that the total amount of energy consumed on sensor charging and the traveling of the mobile charger is no greater than its energy capacity. Assuming that each sensor sending request will be charged to its full energy capacity, where the prize assigned to a sensor is proportional to the amount of energy it will be charged.
- **Optimal k-coverage Charging Problem [7]:** An area is called k-covered if any point of the region is within the sensing range of at least $k > 1$ sensor nodes. Given a set of static sensor nodes randomly deployed over a planar region such that every point of the monitored region has been at least k-covered initially, the optimal k-coverage charging problem is to schedule a charging path of a mobile charger that selects and charges a number of sensor nodes before their deadlines to guarantee k-coverage of the region. At the same time, the charger seeks a path with a minimum traveling distance.

B. 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 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 the 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:** The graph is a directed one. There exists an edge $\overrightarrow{v_i v_j}$ in graph G if and only if the inequality below holds

$$\frac{B - B_i(t_0)}{r_c} + \frac{d_{ij}}{s} \leq D_j \tag{1}$$

where $B_i(t_0)$ denotes the residual energy of sensor node v_i at charger departure time t_0, r_c the energy transfer rate of charger, d_{ij} the Euclidean distance between v_i and v_j, s the average speed of a charger, and D_j the charging deadline of v_j calculated by the residual energy and energy consumption rate at v_j. Note that we use the charger departure time t_0 to construct the graph, but later t_0 will be updated to the exact charging time at v_i.

C. Graph Representation

As we have mentioned in Sec. II-B, deep Q-learning applies parametrized function $Q(S, A; \Theta)$ to approximate the state-action value function $Q(S, A)$ where $Q(S, A; \Theta)$ is parameterized by Θ with size much smaller than $Q(S, A)$, i.e., the combinations of all pairs of state and action. It is expected that the evaluation function Q has well summarized the state S of the current problem solving, i.e., incorporating the current partial solution including the selected vertices and their order into the graph model constructed in Sec. III-B. Q should also have an estimation of the reward value of a new node if it is to be added as an action A in current state S. It is challenging to accurately describe both S and A on a graph. They may depend on global and local statistics of the current graph. There exist different graph representation techniques. We apply the structure2vec technique [8], [9], a deep learning architecture over graphs to compute a p-dimensional node embedding of each vertex and a p-dimensional feature vector of the graph. The state-action value function Q 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.

D. 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 charged sensor nodes before the constraints of a charger or requirements of a network have been satisfied. A step, denoted by t, refers to a single sensor node charging 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 [8], [9] 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 [8], [9] 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}:

$$r(s_t, v) = f(s_{t+1}) - f(s_t). \tag{2}$$

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 the agent. Specifically, at time step t, an action v_t is selected by

$$v_t = \begin{cases} \arg\max_{v \in \bar{s}_t} Q(s_t, v; \Theta) & \text{with probability } 1-\varepsilon \\ \text{select a random vertex } v_t \in \bar{s}_t & \text{otherwise} \end{cases} \tag{3}$$

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

- **Mobile Network Charging Path Optimization Problem:** The objective function f counts the number of selected vertices, i.e., charged mobile nodes. The Insertion function g simply inserts the selected vertex v at time step t to the end of state s_t. 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 at time step t include all non-selected vertices in current state s_t. The reward function $r(s_t, v)$ is defined as

$$r(s_t, v) = 1 \tag{4}$$

  because inserting one vertex to a partial solution contributes the objective function f by one count.
- **Fully Charging Reward Maximization Problem:** The objective function f counts the total prizes collected from selected vertices, i.e., charged mobile nodes. The prize at each vertex is proportional to the energy charged to its full capacity. 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:

$$g(s_t, v) = \arg\min_i \{E_{0i} + E_{it}\} \tag{5}$$

  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. 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 at time step t include all non-selected vertices in current state s_t. The reward function $r(s_t, v)$ is defined as

$$r(s_t, v) = E_v \tag{6}$$

  where E_v is the energy charged to get the full capacity of v.
- **Optimal k-coverage Charging Problem:** The objective function f counts the total traveling distance of a charger. To maximize f, we turn f to negative. The stop function checks whether the current charging path has guaranteed the k-coverage requirement of the area. If it is the case, the current episode/charging cycle can terminate. Actions at time step t contain vertices who are not in s_t and has at least one edge from vertices in s_t. 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. Assuming g inserts v as the i-th node of the partial solution at s_t, the reward function $r(s_t, v)$ is defined as

$$r(s_t, v) = d_{i-1,i+1} - (d_{i-1,i} + d_{i,i+1}), \tag{7}$$

  where d_{ij} is the euclidean distance between nodes v_i and v_j.

E. Deep-Q-Network (DQN) Algorithm

We adopt the deep-Q-network algorithm introduced in [9] to learn the parameters Θ of the state-action value function $Q(S,v;\Theta)$. The adopted algorithm updates the parameters Θ after n steps, instead of each step as a standard deep Q-learning one, to have a more accurate estimation of the objective function. Experience replay method in [13] is also applied to update Θ. The agent's experience in each step is stored in a dataset. When we update Θ, the sample batch is randomly selected from the dataset. This method can break the data correlation and avoid oscillations with the parameters. As a result, it could increase data efficiency. The batch size is 32.

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

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

$$y_l = \begin{cases} r(s_l, v) + \gamma \max_{v'} Q(s_{l+1}, v'; \Theta) \\ \qquad \text{if } s_{l+1} \text{ non-terminal and } v' \in \bar{s_l} \\ r(s_l, v) \\ \qquad \text{otherwise} \end{cases} \tag{8}$$

    if s_{t+1} satisfy the stop function **then**
      Break
    end if
  end for
end for

IV. Simulation

We compare the performance of the DQN algorithm based on the proposed framework with traditional ones on the selected charger scheduling optimization problems.

A. Common Simulation Setting

The simulation area is a Euclidean square with size varying from $[100,100]m^2$ to $[1000,1000]m^2$. We deploy sensors under uniform random distribution with the size, denoted by n, varying from 10 to 200. The battery capacity of each sensor, denoted by B, is 10.8KJ with an initial battery level randomly chosen from $[0,B]$ [2]. A mobile charger starts from the center of the monitored area with an average speed of 5m/s and then comes back to the starting point after a cycle of charging. The average energy transfer rate is 40 W. Note that we run multiple independent simulations for a set of given parameters and then choose the average value.

B. Mobile Network Charging Path Optimization Problem

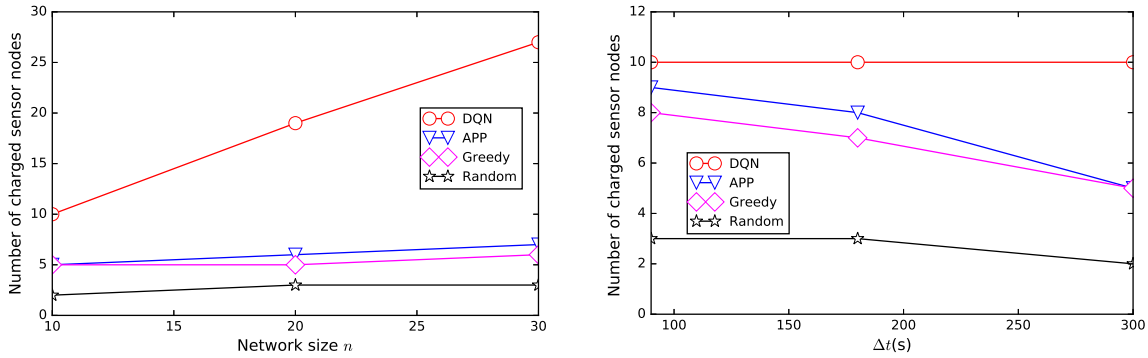
A quasi-polynomial time algorithm that achieves a poly-logarithmic approximation is introduced in [2] to solve the optimum charging path problem. The algorithm discretizes the trajectory of each mobile node by a time step Δt and then constructs a directed acyclic graph with vertices representing the discretized points on the trajectories of all nodes and the starting and ending positions of the charger.

The computational complexity of the approximation 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. Given 20 mobile nodes to charge with the maximum charging timespan $C = 6$ hours, if the time step size is set to $\Delta t = 0.1$ second, n_d will be $n\lfloor C/0.1 \rfloor = 4.32E+6$ and the time complexity will be $O(4.32E+6 * 2.16E+5 * \log 2.16E+5)^L = 5.0E+12$. It is obvious that the approximation algorithm has to sacrifice the performance by increasing the time step size Δt and decreasing the recursion level L when the network size is large.

We compare the DQN algorithm based on the proposed framework with the greedy algorithm, the random algorithm, and the approximation algorithm (APP) in [2]. Specifically, we set the simulation area $[100,100]m^2$ and the maximum timespan of charger $C = 30$ minutes. we choose the random waypoint model to simulate the trace of a mobile sensor with its average speed randomly chosen from $[0,2]$ m/s. For the approximation algorithm in [2], we set the recursion level $L = 3$.

1) Impact of Network Size: Figure 1(a) and Table I 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 charges a significantly higher number of mobile sensors than any other algorithm. At the same time, the computing time of DQN remains stable with an increase of n. By contrast, the computing time of the approximation algorithm in [2] increases dramatically. Overall, the DQN algorithm outperforms all other algorithms.

2) Impact of Time Step Size: The time step size Δt has a big impact on the performance of the approximation algorithm (APP) in [2] as shown in Figure 1(b) and Table I. The number of charged mobile sensors of the approximation algorithm in [2] increases with a decreased time step size Δt and comes close to the DQN algorithm with the cost of sky-high computing time. On the contrary, Δt has no effect on



(a) Impact of network size

(b) Impact of time step size

Fig. 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.

TABLE I
COMPUTING TIME UNDER DIFFERENT NETWORK SIZE n AND TIME STEPSIZE Δt

Alg.	Computing time in Fig. 1(a)		Computing time in Fig. 1(b)	
	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

the DQN algorithm because of the way to construct the graph as introduced in Sec.III-B. The DQN algorithm again outperforms all other algorithms.

C. Fully Charging Reward Maximization Problem

Paper [1] provides a 4-approximation algorithm to solve the fully charging reward maximization problem. The authors reduce the original problem to the classical orienteering one solved by the algorithm in [14].

We evaluate the DQN algorithm based on the proposed framework, the approximation algorithm (APP) in [1], the minimum spanning tree (MST), the Capacitated Minimum Spanning Tree (CMST), and the greedy algorithm. We compare the computing time and the total energy spent on sensor charging of these algorithms. We also study the impact of varying network size and charger capacity on the performances.

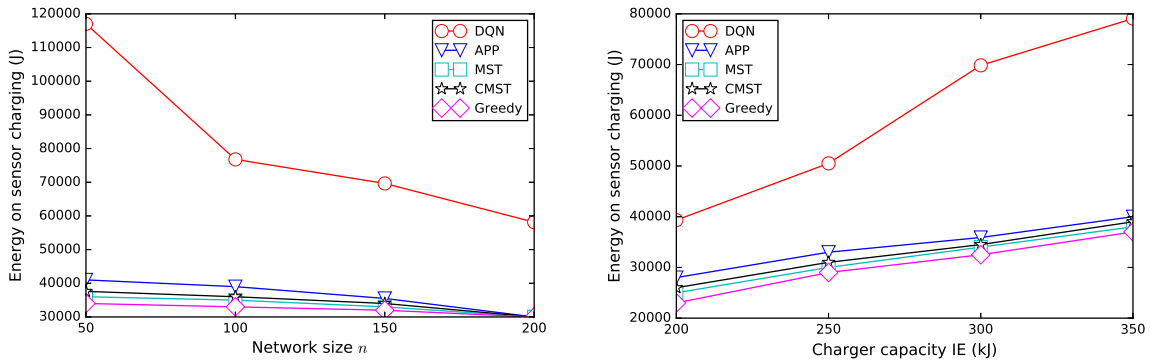
1) Impact of Network Size: We assume a mobile charger with an energy capacity $IE = 300$KJ and 600 J/m on average spent on traveling [1]. The remaining energy threshold is 20% when a sensor sends a charging request. Figure 2(a) and Table II compare the energy spent on sensor charging and the corresponding 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. With the increased network size, the computing time of the approximation algorithm (APP) in [1] increases dramatically while the DQN algorithm remains stable. Overall, the DQN algorithm outperforms all other algorithms.

2) Impact of Charger Capacity: Similarly, we assume a mobile charger with 600 J/m on average spent on traveling [1] and a sensor sends a charging request with the remaining energy less than 20%. Figure 2(b) and Table II 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 sensors increases for all algorithms. 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.

D. Optimal k-coverage Charging Problem

A dynamic programming based algorithm is introduced in [7] that provides an exact solution to the optimal k-coverage charging problem. The algorithm works on a directed acyclic graph constructed by discretizing the time with the computational complexity exponential to the size of a network.

We evaluate the DQN algorithm based on the proposed framework, the dynamic algorithm in [7], the ant colony



(a) Impact of network size

(b) Impact of charger capacity

Fig. 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 II
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

system (ACS) based algorithm, the random algorithm, and the greedy algorithm. We compare the computing time to find a feasible charging path and the energy of a charger spent on traveling of these algorithms. We also study the impact of varying network coverage requirement and size on the performances. 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 [15]. We set the time step size to 1s when implementing the dynamic algorithm in [7].

TABLE III
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

1) Impact of Coverage Requirement: We set the number of sensor nodes $n = 64$ and the remaining energy threshold $\alpha = 0.45$ when a sensor sends a charging request. Table III 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

TABLE IV
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

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.

2) Impact of Network Size: We set the coverage requirement $k = 3$ and the remaining energy threshold $\alpha = 0.45$ when a sensor sends a charging request. Table IV 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.

V. CONCLUSIONS

We introduce a deep reinforcement learning based framework to solve a diverse range of charger scheduling optimization problems. The biggest advantage of the framework is that an optimal charger scheduling strategy can be learned 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 based on the proposed framework with traditional ones on a set of selected charger scheduling optimization problems. Extensive simulation results show the effectiveness of the framework in solving NP-hard charger scheduling optimization problems in WRSN.

REFERENCES

[1] W. Liang, Z. Xu, W. Xu, J. Shi, G. Mao, and S. K. Das, "Approximation algorithms for charging reward maximization in rechargeable sensor networks via a mobile charger," *IEEE/ACM Transactions on Networking*, vol. 25, no. 5, pp. 3161–3174, 2017.

[2] L. Chen, S. Lin, and H. 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*, pp. 101–110, ACM, 2016.

[3] Y. Shi, L. Xie, Y. T. Hou, and H. D. Sherali, "On renewable sensor networks with wireless energy transfer," in *INFOCOM, 2011 Proceedings IEEE*, pp. 1350–1358, IEEE, 2011.

[4] L. Xie, Y. Shi, Y. T. Hou, W. Lou, H. D. Sherali, and S. 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*, pp. 10–18, IEEE, 2012.

[5] L. He, P. Cheng, Y. Gu, J. Pan, T. Zhu, and C. Liu, "Mobile-to-mobile energy replenishment in mission-critical robotic sensor networks," in *INFOCOM, 2014 Proceedings IEEE*, pp. 1195–1203, IEEE, 2014.

[6] X. Lu, P. Wang, D. Niyato, D. I. Kim, and Z. Han, "Wireless charging technologies: Fundamentals, standards, and network applications," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1413–1452, 2015.

[7] X. Li and M. Jin, "Optimal k-coverage charging problem," *arXiv preprint,arXiv:1901.09129*, 2019.

[8] H. Dai, B. Dai, and L. Song, "Discriminative embeddings of latent variable models for structured data," in *International conference on machine learning*, pp. 2702–2711, 20S16.

[9] E. Khalil, H. Dai, Y. Zhang, B. Dilkina, and L. Song, "Learning combinatorial optimization algorithms over graphs," in *Advances in Neural Information Processing Systems*, pp. 6351–6361, 2017.

[10] R. S. Sutton, A. G. Barto, *et al.*, *Reinforcement learning: An introduction*. MIT press, 1998.

[11] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural networks*, vol. 4, no. 2, pp. 251–257, 1991.

[12] M. Riedmiller, "Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method," in *European Conference on Machine Learning*, pp. 317–328, Springer, 2005.

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

[14] N. Bansal, A. Blum, S. Chawla, and A. 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*, pp. 166–174, ACM, 2004.

[15] T. Zhu, Z. Zhong, Y. Gu, T. He, and Z.-L. Zhang, "Leakage-aware energy synchronization for wireless sensor networks," in *Proceedings of the 7th International Conference on Mobile Systems, Applications, and Services*, pp. 319–332, 2009.