HGFF: A Deep Reinforcement Learning Framework for Lifetime Maximization in Wireless Sensor Networks

Xiaoxu Han, Xin Mu, and Jinghui Zhong, *Senior Member, IEEE*

***Abstract*—Planning the movement of the sink to maximize the lifetime in wireless sensor networks (WSNs) is an essential problem. Many existing mobile sink techniques based on mathematical programming or heuristics have demonstrated the feasibility of the task. Nevertheless, the huge computational cost or the over-reliance on human knowledge can result in relatively low performance. To balance the need for high-quality solutions to minimize inference time, we propose a new framework to construct the movement path of the sink automatically. We cast the lifetime maximization problem as an optimization task within a heterogeneous graph and learn movement policy for the sink by combining graph neural network (GNN) with deep reinforcement learning. Our approach comprises three key modules: 1) a heterogeneous GNN to learn representations of sites and sensors by aggregating features of neighbor nodes and extracting hierarchical graph features; 2) a multihead attention mechanism that allows the sites to attend to information from sensor nodes, which highly improves the expressive capacity of the learning model; and 3) a greedy policy that learns to append the next best site in the solution incrementally. We design twelve types of static and dynamic maps to simulate different WSNs in the real world, and extensive experiments are conducted to evaluate and analyze our approach. The empirical results show that our approach consistently outperforms the existing methods on all types of maps. Notably, our approach significantly extends the simulation lifetime without sacrificing a large increase in inference time.**

***Impact Statement*—Mobile sink methods are effective in prolonging the lifetime of WSNs. They balance the nodes' energy dissipation throughout the network. Planning the movement path of the sink is a critical challenge, as the states of each node in the network change dynamically. Previous methods based on mathematics or heuristics failed to find good solutions in a short period of time, due to the need for high-computational cost or suboptimal human knowledge. Our proposed approach overcomes these limitations. First, features containing complex network topology and dynamic node information are extracted by the graph-based neural network. Second, the movement policy is learned automatically through reinforcement learning, without prior knowledge. Experiments were carried out under different network settings, and the results show that our method outperforms the previous methods significantly. The proposed method has made notable contributions to maximize the lifetime of WSNs, thereby advancing its practical applications.**

***Index Terms*—Attention mechanism, deep reinforcement learning (DRL), heterogeneous graph neural network (GNN), wireless sensor network (WSN).**

I. Introduction

OVER the past few decades, wireless sensor networks (WSNs) have been widely applied in the real world, such as traffic control [1], military target tracking [2], and biomedical health monitoring [3]. A typical WSN consists of a sink and numerous sensor nodes that are used to obtain information from the environment (e.g., temperature, pressure, and location). Normally, the data are collected by sensor nodes, transmitted to the sink through multihop communication, and eventually sent to the users from the sink. As the sensor nodes tend to be disposable and energy-constrained, the lifetime of WSN is limited and often defined as the duration time of the network until one or more sensor nodes run out of energy for the first time. Moreover, the inaccessibility of sensor deployment results in high-recharging costs. Therefore, how to maximize the lifetime becomes one of the most important issues in the WSN community.

Leveraging sink mobility has been shown to be an effective way to maximize the lifetime of WSNs [4], [5]. Specifically, a mobile sink can move to various locations within the sensing region while collecting data from sensors. In this way, the sensors located near the sink change over time, which brings more balanced energy dissipation throughout the network. Thus, the lifetime of WSN with a mobile sink is prolonged. Efficiently scheduling the movement of the mobile sink is vital for the network's lifetime. Yet, determining the optimal traversal path for the sink remains a formidable challenge, known to be an NP-Hard problem [6].

Received 25 October 2023; revised 18 May 2024; accepted 9 November 2024. Date of publication 13 November 2024; date of current version 31 March 2025. This work was supported in part by the Major Key Project of Peng Cheng Laboratory under Grant PCL2023A09; in part by the National Science Foundation of China under Grant 62476096, Grant 62106114, and Grant 62076098; and in part by the Guangdong Basic and Applied Basic Research Foundation under Grant 2023A1515012291. This article was recommended for publication by Associate Editor Kyriakos G. Vamvoudakis upon evaluation of the reviewers' comments. *(Corresponding authors: Xin Mu; Jinghui Zhong.)*

Xiaoxu Han is with the South China University of Technology, Guangzhou 510006, China, and also with Peng Cheng Laboratory, Shenzhen 518000, China (e-mail: ftxiaoxu.han@mail.scut.edu.cn).

Xin Mu is with Peng Cheng Laboratory, Shenzhen 518000, China (e-mail: mux@pcl.ac.cn).

Jinghui Zhong is with the South China University of Technology, Guangzhou 510006, China (e-mail: jinghuizhong@scut.edu.cn).

Digital Object Identifier 10.1109/TAI.2024.3497926

Various approaches to optimizing the sink's movement to prolong the WSN's lifetime have been proposed, including operational research-based (OR) methods [7], [8], [9] and heuristic methods [4], [10], [11], [12], [13], [14]. Formulating the problem into a mathematical programming model [e.g., the mixed integer linear programming model (MILP)], the OR methods [e.g., branch-and-bound (B&B)] are usually guaranteed to find the optimal solution. Nevertheless, the exponential complexity of OR methods hinders their application to large-scale real-world instances. Heuristic methods can get satisfactory results within bearable time. Whereas well-designed and task-specific prior knowledge from human experts and trial-and-error are required, the optimality still cannot be guaranteed.

An alternative research direction for guiding the sink's path to maximize WSN's lifetime is based on reinforcement learning (RL). RL methods have been shown to be able to automatically learn good heuristics within brief solving time, as demonstrated by its outstanding performance in tackling combinational optimization problems, such as traveling salesman problem (TSP) [15], vehicle routing problem (VRP) [16], and so on [17], [18], [19], [20]. Without prior knowledge and labeled data, RL can learn effective policy from scratch. Given that the movement of mobile sink can be modeled as a Markov decision process (MDP), several works for planning the traverse path of sink based on RL have been proposed [21], [22], [23], [24]. They leverage the tabular-based Q-learning method [25] to find the optimal path, where the Q value is updated using the Bellman equation during trial-and-error. While the existing RL-based works still face the following technical challenges.

1) *Curse of Dimensionality and Continuous Spaces*: Maintaining Q values in a table for every possible state-action pair can become infeasible for large or continuous state spaces. Thus, they cannot handle the high-dimensional space brought by large-scale WSNs, which may consist of hundreds or thousands of nodes. Besides, information loss is caused by the inability to handle continuous data input of WSNs.
2) *Unable to Handle Dynamic Scenario*: In dynamic WSNs, the location of sensors may change at any time. Yet, the existing methods cannot adapt well to changes without further exploration. In addition, they lack the ability to generalize and have difficulty in processing unseen dynamic scenarios.

To tackle the issues and limitations above, we propose a novel deep reinforcement learning (DRL) framework with heterogeneous graph-based feature fusion (HGFF[1]) to address the lifetime maximization problem in WSNs. We first abstract the problem as an optimization on the heterogeneous graph with two types of nodes, i.e., sites and sensors. Then, extracting features from the original WSNs graph input, the GNN captures the information of the node in relation to its surrounding graph nodes. Specifically, the node representations are learned using a combination of learnable type embedding and message-passing aggregation. The introduction of type embedding is to learn the heterogeneity of node features and better exploit graph structure based on the node type information. Meanwhile, an attention-based global feature fusion operation is proposed to enable the site nodes to attend over features of all the sensor nodes in the graph. We use double Q-learning [26] method to learn greedy policy for guiding the movement of the sink. Following the algorithm pattern first produced in [17], our framework constructs the route of mobile sink incrementally by greedily selecting the site node with the highest Q value, which is estimated based on the node embeddings. To demonstrate the superiority of the proposed method across different WSNs, we compare HGFF with different existing methods on numerous types of WSNs. The empirical results show that HGFF achieves significant improvement in prolonging the lifetime of WSNs.

Our main contributions to this research are as follows.

1) We present a new DRL framework HGFF for lifetime maximization in WSN. HGFF generates effective heuristics to guide the movement of the sink in an end-to-end manner by DRL, which requires neither huge computation consumption nor human knowledge.
2) We model a WSN as a heterogeneous graph and use graph-based feature fusion techniques to enhance the representations for nodes. To be specific, learnable type embedding is incorporated into the GNN to learn the heterogeneous information of different types of nodes. Moreover, different from the neighbor node aggregation in common graph attention networks, we adopt a global-based attention mechanism to further extract the relativity information. The improved node representation thus improves the quality of the solution.
3) We conduct extensive experiments to verify the effectiveness of our proposed approach. The empirical results have shown that HGFF consistently outperforms other methods across various types of maps, including challenging large-scale networks and dynamic WSNs.

The rest of the article is structured as follows. In Section II, we review different research using mathematical methods, heuristic methods, and RL-based methods in the problem of lifetime maximization WSN context. Then, in Section III, we define the mathematical model of the movement problem for the sink. In Section IV, we present the MDP formulation and introduce our framework in detail. Section V presents the experimental settings for evaluating our proposed method and the empirical results, which includes the setup of experiments, the results, and the analysis of our method with existing methods. Finally, we conclude the article and look forward to future work in Section VI.

II. Related Work

How to prolong the lifetime of network is a critical issue in WSNs due to the limited energy of sensors. Various techniques for extending the lifetime have been proposed, varying by their network lifetime definitions, objective functions, and

[1]Source codes are available in: https://github.com/xiaoxuh/HGFF_Maximize-the-lifetime-of-WSN-with-DRL.

considered topologies [27]. Cross-layer optimization is leveraged to allocate resources effectively, addressing various design constraints [28], [29], [30], [31], [32], [33]. Additionally, constructing dynamic routes for sensors is another essential way to extend network lifetime [34], [35], [36], [37], [38], [39]. Furthermore, via using multitier network architectures, researchers organize the network into clusters managed by high-energy cluster heads to facilitate data relay [40], [41], [42], [43], [44], [45].

A line of research concentrates on utilizing the mobility of sinks to prolong the lifetime of WSNs [5], [7], [8], [9], [10], [11], [12], [13], [14], [46], [47], [48]. It has been demonstrated that static sinks perform badly in terms of energy efficiency [49]. The main reason for this phenomenon is the "energy hole" problem [50], namely, the sensor nodes around the sink are prematurely depleted of energy due to the need to forward more data, which is likely to form a sensor network energy consumption bottleneck, resulting in a short lifetime. Therefore, the mobile sink has been adopted by various methods to improve the lifetime of WSNs. In this work, we focus on planning the route of the mobile sink to maximize the lifetime for energy-constrained WSNs.

The optimization problem for scheduling the sink movement to maximize lifetime can be modeled as different mathematical optimization models. Leveraging the MILP model, Basagni et al. [4] attempt to maximize the lifetime of WSNs with constraints including a movement restriction for the sink traveling between two different sites, and the limited energy cost of the mobile sink in scheduling. The method that sets an objective to maximize the accumulated sojourn time of the mobile sink in event-driven applications, a convex optimization model to determine the optimal trajectory of a mobile sink without relying on predefined structures is introduced in [9]. To make the optimization model closer to reality, more elements are considered in the mathematical problem. The storage capacity of sensors is considered in [7], for the case of delayed data delivery is permitted. The authors propose a more complicated model based on linear programming (LP), which decides the data delivery of sensors according to whether the sink is in the position most conducive to achieve the longest network lifetime. Behdani et al. [8] propose a column generation algorithm both to prohibit delay tolerance and allow delay tolerance WSNs. Although the mathematical optimization models are bound to be a high-quality solution, the consuming time and computing power costs are unbearable for large-scale problems.

Some works concentrate on designing heuristic methods manually to solve the problem. A greedy maximum residual energy (GMRE) heuristic rule is proposed in [4], which greedily selects the site with the largest residual energy of surrounding sensors to visit. To get a decent solution within tolerable consuming time, Liang et al. [10] present a three-stage heuristic method that iteratively does local improvements on a distance-constrained shortest path problem that approximates the original lifetime maximization problem. Metaheuristic methods are also utilized to solve the lifetime maximization problem, as they are capable of finding near-optimal solutions by exploring the space of possible solutions efficiently. Nonetheless, they can be very time-consuming when facing large-scale problems, owing to the repeated evaluations of the objective function. Some existing methods introduce ant colony optimization (ACO) algorithm [11], [12], [13] to improve the lifetime of WSNs. In recent years, hyperheuristic algorithms also have been leveraged to solve the combinational optimization problem in WSNs. A hyperheuristic framework is proposed in [14], which can automatically construct a solution to optimize the moving path of the sink to prolong the lifespan of WSNs based on several heuristic rules, while the above methods are highly dependent on the predefined heuristics. There is no guarantee that human knowledge is necessarily optimal.

Avoiding the design of complicated computing mechanisms and massive feature engineering, RL-based methods are applied in WSNs to increase the lifetime. Many works focus on improving energy efficiency and prolonging the lifetime of cluster-based WSNs, in which RL methods are mainly applied to select cluster heads and decide the visit of the sink to collect data. For the clustering or the election of cluster head, the tabular Q-learning method is leveraged in [21], [22], and [23] to learn the optimal cluster head to address energy challenges in WSNs. As for finding the shortest path to collect data, the tabular Q-learning method is also used in [24] to improve the movement route of the mobile sink reaching the cluster heads to collect data. However, the state space covered by Q-table is limited in the above works, which results in a lack of scalability and cannot be used in dynamic networks. In this article, we propose a DRL approach, which is able to handle more complex WSNs with vast state space and dynamic situations. Besides, in contrast to the above methods for cluster-based protocol WSNs, we focus on the problem of scheduling the traversal path of the sink to maximize the lifetime in an instance model of flat-based protocol WSNs, which is more suitable for the network with low-redundancy signals.

III. Problem Definition

This section describes the process of transmitting data from the sensor to the sink in WSNs, using the notations provided in Table I. We then present the mathematical formulation of the problem of maximizing the lifetime of WSNs.

A WSN consists of a sink α which can stop in a set of locations (i.e., sites) $L = \{l_1, \ldots, l_n\}$, and a large set of homogeneous sensor nodes $N = \{n_1, \ldots, n_m\}$ scattered in a geographic area following the sensor-updating function f. The distribution of sensors in $k+1$ round can be defined as $N^{k+1} = f(N^k)$. In particular, the distribution of sensors in the static sensor network does not change over time, i.e., $N^k = N^{k+1} = f(N^k)$. Both the sensor nodes and the sink have a maximum transmission range $d_{\max}$. Each sensor node n_i is assumed to generate information at a fixed rate of z_i during its life span. Hence, the data produced by n_i in the time interval Δt (the minimum time of each round) can be denoted as $z_i * \Delta t$. Then, the data will be forwarded to the sink node in one hop if the Euclidean distance $d_{i,\alpha}$ between the sensor node n_i with the sink α is smaller than $d_{\max}$, otherwise, through a multihop

TABLE I
PROBLEM NOTATION

Symbol	Definition
$L=\{l_1,\ldots,l_n\}$	The set of sites
$N=\{n_1,\ldots,n_m\}$	The set of sensor nodes
α	The sink
T	The lifetime of the WSN
z_i	The sensing data rate of n_i
$d_{\max}$	The maximum transmission range of sensor nodes
$d_{i,j}$	The Euclidean distance between node i and node j
t_j	The sojourn time of the sink stays at l_j
Γ_j^k	Whether the sink settles in site l_j in kth round
E_i	The initial energy of n_i
$U^k=\{u_1^k,\ldots,u_m^k\}$	The residual energy of sensors at kth round
u_i^k	The residual energy of sensor n_i in kth round
et_i^k	The consumption rate of sending one bit of data for n_i in kth round
er_i^k	The consumption rate of receiving one bit of data for n_i in kth round
f_i^k	The amount of data sent by n_i in kth round
g_i^k	The amount of data received by n_i in kth round
$loc(\alpha)^k$	The site where the sink stays at kth round

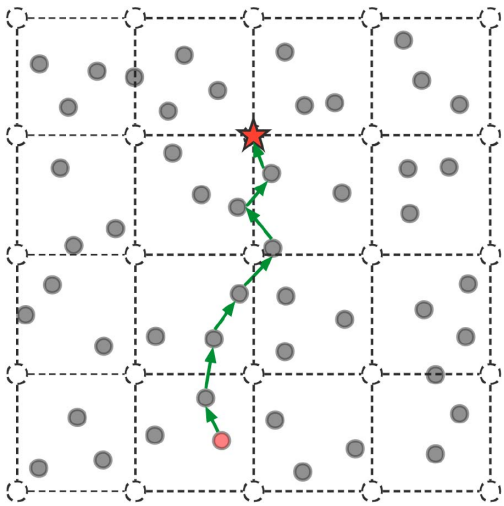

Fig. 1. Forward route of data transmission from the source sensor to the sink. The dashed circle denotes the sites, the red five-pointed star refers to the sink, and the solid circle represents the sensor nodes. The solid red circle refers to the source sensor, and the arrow represents the data flow of two nodes within the transmission range.

transmission tree if $d_{i,\alpha} > d_{\max}$. The data transmission from the sensor node to the sink in WSN is shown in Fig. 1.

At each round k, the sink reaches one of the possible sites in L or does not move (i.e., stays at the location in the previous round $k-1$). The traveling time $\dot{t}$ between two sites is ignored, as $\dot{t} \ll \Delta t$. When the sink reaches a new site or leaves the previous site, a data packet containing the current location of the sink will be sent to all sensors to make them aware of the new site of the sink. We denote the sojourning time of the sink at each location l_j as t_j. Therefore, the lifetime T of the WSN can be demonstrated as follows:

$$T = \sum_{j=1}^{n} t_j$$

where $t_j = \sum_k \Gamma_j^k$. Γ_j^k is a binary variable that denotes whether the sink settles in site l_j at the kth round: if the sink stays at site l_j, the Γ_j^k will be 1; otherwise, it is 0. For the case of WSN with a single sink, only one site places the sink at each round k: $\sum_{j=1}^{m} \Gamma_j^k = 1$.

In the beginning, each sensor node n_i has the initial energy E_i and the WSN dies when one of the sensors exhausts its energy. The data flow of a sensor node involves the sending of data generated by itself and the transmission of data coming from other nodes j ($d_{i,j} < d_{\max}$). The incoming data flow at n_i consists of data sent by other nodes, and the outgoing data flow includes its sending data and the data originating from other sensors.

Both processes of sending and receiving data of sensor node n_i require energy. The transmission power consumption is closely coupled with the route selection. The consumption rates of sending and receiving data of sensor n_i in kth round can be denoted as et_i^k and er_i^k. We consider the er_i^k, $\forall k$, as a constant, and that the consumption rate $et_{i,j}^k$ for data sending from sensor i to sensor (or the sink) j is proportional to the Euclidean distance $d_{i,j}$, as is shown in (1). The a and b are constant coefficients related to the transmission media properties

$$et_{i,j}^k = a \times d_{i,j}^2 + b. \tag{1}$$

The multihop transmission tree H is leveraged to decide the data transmission routes between sensors and the sink, which can be constructed by the flow augmentation algorithm (FA) [51]. H represents the shortest cost path routing from the origin node (i.e., sensor) to the destination node (i.e., sink) through computing link costs reflecting both the communication energy consumption rates and the residual energy of the two end nodes. That is to say, the FA algorithm takes the distribution of sensors, the residual energy of sensors, and the location of the sink as input, and outputs the amount of data sent f_i^k (or received g_i^k) by sensor n_i in the kth round, which can be formulated as

$$[f_i^k, g_i^k] = \text{FA}\ (N^k, loc(\alpha)^k, U^k). \tag{2}$$

Based on (1) and (2), the energy consumption rate per unit time of sensor n_i at kth round is $et_i^k \cdot f_i^k + er_i^k \cdot g_i^k$. Thus, the energy cost of each sensor n_i during its lifetime in the WSN can be defined as follows:

$$\sum_k (et_i^k \cdot f_i^k + er_i^k \cdot g_i^k).$$

Following the definition in [14], the problem of maximizing the lifetime of the WSN can be formulated as a MILP model

$$\max \quad T = \sum_{j=1}^{n} t_j \tag{3}$$

$$\text{s.t.} \quad t_j = \sum_k \Gamma_j^k, l_j \in L \tag{4}$$

$$\sum_{j=1}^{m} \Gamma_j^k = 1 \tag{5}$$

$$\sum_{k} (et_i^k \cdot f_i^k + er_i^k \cdot g_i^k) - E_i \leqslant 0, \forall n_i \in N \tag{6}$$

$$g_i^k + z_i * \Delta t = f_i^k, \forall n_i \in N. \tag{7}$$

Equation (6) is an energy constraint representing that the total energy cost of each node during its lifetime is less than the initial energy. Besides, the data flow conservation (7) ensures that the amount of incoming data with the data produced by the sensor n_i equals the amount of its outflow data.

In this article, we focus on the maximization of network lifetime as the performance indicator, which is critical for resource-constrained WSNs. Extending the network lifetime necessitates efficient energy management, which involves minimizing the energy consumed by each node during its operation. Furthermore, the overall network lifetime sets the minimum threshold for the lifespan of individual sensors within the network. Therefore, maximizing the network lifetime significantly contributes to the sustainability and overall effectiveness of the entire network.

IV. Proposed Method

In this section, we illustrate our framework formally, including the design of MDP, the structure of the neural network for evaluating the agent's state-action value, and the training method. First, the overall framework of our method is introduced. Second, we formulate the movement process of the sink to maximize WSN's lifetime as an MDP. Then, we represent the WSN as a weighted undirected heterogeneous graph. In addition, the way to learn the representation of nodes in WSN based on the GNN and multihead attention is given. Finally, we introduce the training algorithm in detail.

A. Overview

The lifetime maximization with mobile sink problem of different WSNs has the same structure but differ in the geographic area, distribution, and sensor-updating function. Due to the requirement for high-quality human knowledge, it is challenging to design optimal heuristic rules to solve the problem. Different from previous work mainly using heuristics, a DRL-based framework HGFF is proposed to learn the optimal movement strategy for the sink in this article.

First, we formalize the WSN as a weighted undirected heterogeneous graph. Due to the different roles and attribute information, the sensors and sites are naturally seen as two types of nodes in the heterogeneous graph. Then, the learnable node type embedding is incorporated with the GNN to learn the representations of sensors and possible sites the sink visits with the relationship between them. To further enhance representations, we introduce the multihead attention to "fuse" features, which means learning and leveraging the compatibility between embeddings of the two kinds of nodes. Different from the attention in common graph attention networks, we focus on all the sensor nodes in the WSN graph instead of only the neighbor nodes. Such a method is more suitable for dynamic WSNs, in which the surrounding sensors are changing at each time step.

Based on the double deep Q-learning (DQN) algorithm [52], the agent learns the policy of guiding movement of the sink to maximize the network's lifetime from thousands of training experiences. The overall framework of the agent's Q-network is shown in Fig. 2. The Q-network takes the state of the virtual WSN environment, which is represented as graph data, and outputs the evaluation of the value for agent staying at each site: $Q(s, a)$. To learn the heterogeneity of nodes, the representation of nodes will be enhanced by combining learnable type embeddings explicitly in aggregating information from neighbors and combining the information to update representations in the next iteration. Building on the updated embeddings in graph learning, we aim to learn the compatibilities of sites with sensor nodes to improve the representations, which are the bases for estimating the Q-value. The estimated Q-value is updated during trial-and-error and will be greedily utilized to obtain the optimal moving strategy in the inference of the model.

B. MDP Formulation

We formalize the moving process of the sink in a WSN as an MDP. The purpose of such modeling is to address the problem: determine the route for the mobile sink, together with the sojourn times at which the sink stays on each site, so as to maximize the lifetime of the WSN. The MDP can be defined as a five-tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma \rangle$. S represents the environmental information and the possible configurations of the WSN environment. A denotes the available action set of the agent and T defines the probability distribution over possible next states. $\gamma \in [0, 1]$ is the discounted factor. It is worth mentioning that the Markov property is satisfied in the virtual WSN environment. Given the present state of the WSN environment, the future decision of the sink (i.e., select a site to move) is independent of the past. In other words, the state of the WSN environment includes all the relevant information about its history: $\mathbb{P}[S_{t+1}|S_t] = \mathbb{P}[S_{t+1}|S_1, \ldots, S_t]$.

Based on the MDP, we simulate the decision process of the site where the sink stays at each time step (i.e., Γ_j^k) until the overall WSN dies. The main parts of MDP are defined as follows.

1) States: The state for the agent includes all the attribute information of the sensor nodes and sites, which is composed of the following global information.

a) Property of node: sensor, site, and if it belongs to the set of sites, whether there is a sink on this site.
b) Current coordinate position (including x-axis and y-axis) of the sink, sensors, and sites.
c) Residual energy of the sensor nodes.
d) Energy consumption per unit time of sensor nodes.
e) Euclidean distance $d_{i,j}$ between every two nodes if $d_{i,j} < d_{\max}, i, j \in L \cup N$.

2) Actions: Since the time for the sink to move between the site nodes is ignored, we assume that any one of the sites is available to the sink at each time step. Thus, the action for the

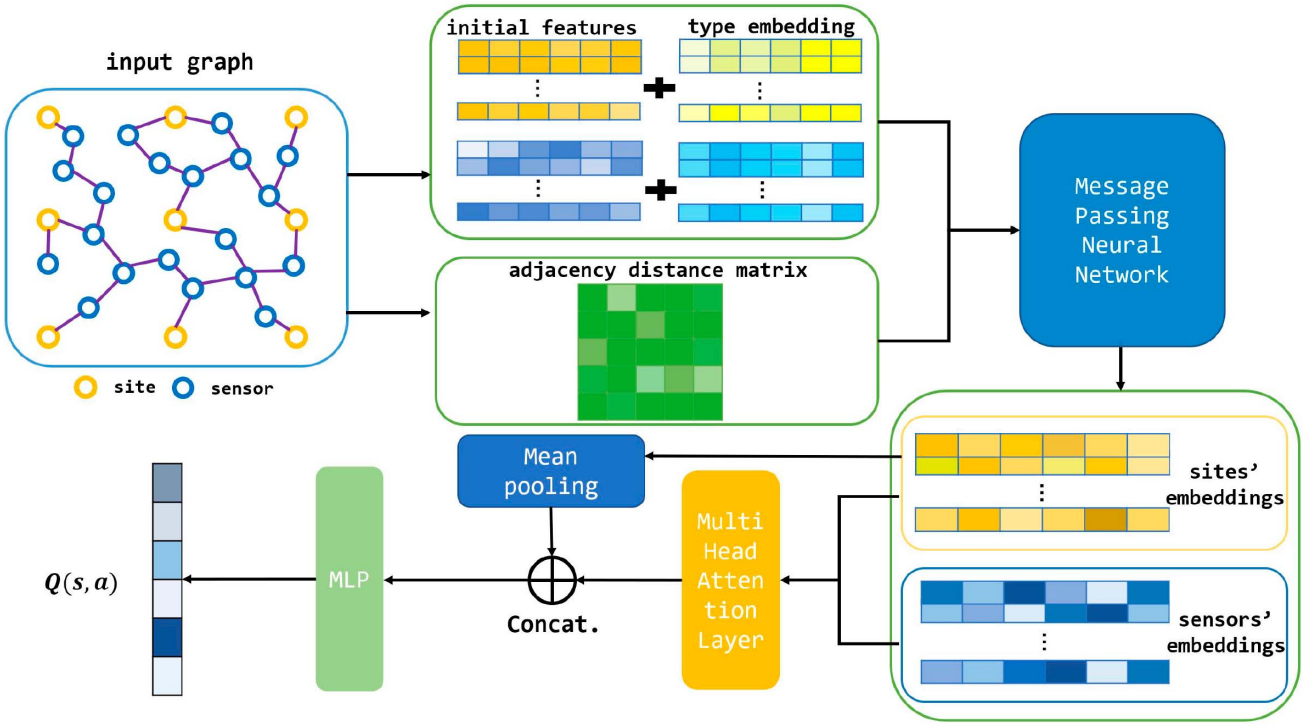

Fig. 2. Overall framework of HGFF.

agent is to select a site from the set of sites, and the length of action space is equivalent to the range of site set L: $|A| = |L|$.

3) Rewards: In fact, maximizing the WSN's lifetime T is equivalent to maximizing the accumulated sojourn times of the sink at the sites, and also equivalent to maximizing the expected episode's length of the agent. Intuitively, we define the reward as a constant: $r_t = 1$, for each time step t. The goal of the agent is to learn the optimal movement policy π for the sink to maximize the expected return with a discount factor γ: $E_\pi\{\sum_{t=0}^{\infty}\gamma^t r_t\}$. With $r_t = 1$, the goal of the agent can also be written as $\max\ E_\pi\{\sum_{t=0}^{T}\gamma^t\}$.

4) Transitions: At each time step, after the agent makes the decision to choose one site to visit, the environment will process the decision to step into the next state definitely: $\mathcal{T}: \mathcal{S} \times \mathcal{A} \to \mathcal{S}$. The energy consumption of each sensor node due to sending and receiving data will be calculated to obtain the new state.

5) Terminal Criterion: When the residual energy $u_i \leqslant 0, \forall i \in N$ happens, the episode ends.

C. Learning Over Graphs

1) Graph Representation Formulation: We define the WSN as a weighted heterogeneous graph: $G = (V, E, w, \phi, \psi)$, where the set of nodes V is constituted of two kinds of nodes: sensors in set N and sites in set L. The set of edges E represents the connection between two nodes that are able to communicate with each other (i.e., they are within the transmission range $d_{\max}$). w refers to a function mapping undirected edges to their weight values: $E \to W$, where $W = \{W_e : \forall e \in E\}$ means the set of weights. ϕ is the node type mapping function: $V \to T_v$, where $T_v = \{\phi(v) : \forall v \in V\}$ denotes the set of possible node types. ψ is the edge type mapping function: $E \to T_e$, where $T_e = \{\psi(e) : \forall e \in E\}$ denotes the set of possible edge types. In the graph G, there are two types of nodes and one type of edges: $|T_v| = 2$, $|T_e| = 1$. It is worth noting that G must be a fully connected graph, otherwise, there will be isolated nodes that cannot send data to the outside.

We design five features for sensor node n_i at each time step k: the mark for the type of node, the x-coordinate of the node, the y-coordinate of the node, the ratio of the residual energy to the initial energy in kth time step: u_i^k/E_i, and the energy consumption per unit time in kth time step: $et_i^k \cdot p_i^k + er_i^k \cdot q_i^k$. For the features of site l_j, they are the mark for the type of node, the x-coordinate of the node, and the y-coordinate of the node. Note that the elements for both features are normalized, except for the identification mark for different nodes.

To include the distance information between different nodes in the representation, we explicitly define the weight $W_{e_{ij}}$ of connection between node i and node j be the normalized Euclidean distance of two nodes

$$w(e_{ij}) = \text{normlize}(d_{ij}).$$

The information on the above nodes and edges is used as the initial feature to train the GNN in the next section.

2) Message Passing Neural Networks (MPNNs) With Learnable Node-Type Embedding: To leverage type information to improve training, we incorporate node type embedding into the original MPNN [53], which is a general learning framework used widely in many famous graph networks with specific implementations. Based on the input node feature $\mathbf{x}_v \in \mathbb{R}_0^d$

(d = 5) and the structure of graph G constructed in Section IV-C1, we aim to learn an d_h (we use $d_h = 64$) dimensional representation vector $\mathbf{h}_v^l$ for each vertex $v \in V$, where l denotes the current iteration or network layer. First, the input feature $\mathbf{x}_v$ is input to a linear projection with parameters W^x and B^x: $\boldsymbol{\mu}_v^0 = W^x \boldsymbol{x}_v + B^x$. Then, $\mathcal{L}$ rounds of message passing will be leveraged to update the embeddings, according to (8) and (9).

Although the original MPNN can be used to model homogeneous graphs, the omission of node or edge types could make it suboptimal for heterogeneous graphs. To emphasize the heterogeneity between two kinds of nodes, we allocate an embedding $\boldsymbol{t}_{\phi(v)}^l$ for each type of vertex $\phi(v), \forall v \in V$, where the contents of $\boldsymbol{t}_{\phi(v)}^l$ are adjusted during the training process. Therefore, the node embedding $\boldsymbol{h}_v^l$ consists of two parts: node embedding $\boldsymbol{\mu}_v^l$ in original MPNN and learnable node type embedding $\boldsymbol{t}_{\phi(v)}^l$, where $\phi(v)$ is the type of node v. The embedding $\boldsymbol{h}_v^l$ of node v in iteration l is defined as $\boldsymbol{h}_v^l = \boldsymbol{\mu}_v^l || \boldsymbol{t}_{\phi(v)}^l$, where $||$ denotes the concatenation operation.

Based on the aggregation from the neighbor nodes, the GNN updates node representation iteratively following:

$$\boldsymbol{m}_v^{l+1} = M_l\left(\boldsymbol{h}_v^l, \left\{\boldsymbol{h}_u^l\right\}_{u\in N(v)}, \left\{\omega_{uv}\right\}_{u\in N(v)}\right) \tag{8}$$

$$\boldsymbol{h}_v^{l+1} = U_l(\boldsymbol{h}_v^l, \boldsymbol{m}_v^{l+1}) \tag{9}$$

where M_l denotes the message function and U_l represents the update function. The message function aggregates information from neighborhood nodes $\left\{\boldsymbol{h}_u^l\right\}_{u\in N(v)}$ with adjacent edges $\left\{\omega_{uv}\right\}_{u\in N(v)}$, and pass on the information to the next layers. The node representation is updated iteratively through the nonlinear transformation in the update function.

D. Attention-Based Feature Fusion

Aggregating the embeddings from neighbors, the GNN in Section IV-C2 learns the representations of sites and sensor nodes: $\{\boldsymbol{h}_v^{\mathcal{L}}\} = \{\boldsymbol{h}_l^{\mathcal{L}}\} \cup \{\boldsymbol{h}_n^{\mathcal{L}}\}$, where $v \in V, l \in L, n \in N$. To further learn the association between nodes and enhance the model's expressive capacity, we adopt the attention mechanism [54], which requires a query and a collection of key-value pairs to map the output. Formally, we define the queries consist of site nodes' embeddings $\{\boldsymbol{h}_l^{\mathcal{L}}\}$ and keys and values $\{\boldsymbol{h}_n^{\mathcal{L}}\}$ come from the sensor nodes' embeddings. The queries, keys, and values are calculated by the linear transformations of the sites' embeddings and sensors' embeddings, respectively,

$$\boldsymbol{q_i} = W_q * \boldsymbol{h}_l^{\mathcal{L}}, \boldsymbol{k_j} = W_k * \boldsymbol{h}_n^{\mathcal{L}}, \boldsymbol{v_j} = W_v * \boldsymbol{h}_n^{\mathcal{L}}$$

where $\forall i \in L, \forall j \in N$, parameter W_q is learnable matrice $(d_q * d_h)$, and W_k, W_v have the same dimension $d_k * d_h$. The compatibility of the query $\boldsymbol{q}_i$ of site node i with the key $\boldsymbol{k}_j$ of sensor node$j$: $u_{ij} \in \mathbb{R}$ is computed following the dot-product in

$$u_{ij} = \frac{\boldsymbol{q}_i^T \boldsymbol{k}_j}{\sqrt{d_k}}. \tag{10}$$

Based on the compatibilities u_{ij}, the attention weights $a_{ij} \in [0, 1]$ can be computed by the softmax function

$$a_{ij} = \frac{e^{u_{ij}}}{\sum_j e^{u_{ij}}}.$$

Then, a convex combination of $\boldsymbol{v}_j$ is used to produce vector $\boldsymbol{h}_i'$:

$$\boldsymbol{h}_i' = \sum a_{ij} \boldsymbol{v}_j.$$

Due to the stability that comes from learning different types of messages from neighbors, multihead attention with $M = 8$ heads is leveraged to compute the value in Section IV-D repeatedly with M different learnable matrices. After calculating various attention maps, we combine all the learned representations. Specifically, the dimensionality of query or key is d_h/M. The resulting vectors are denoted as $\boldsymbol{h}_{im}'$ for $m \in \{1, \ldots, M\}$ and learnable matrices W_m^O are used to project back the vectors to d_h dimension vector. Finally, the multihead attention value $\boldsymbol{h}_i^*$ for site node i can be defined as follows:

$$\boldsymbol{h}_i^* = \sum_{m=1}^{M} W_m^O \boldsymbol{h}_{im}'. \tag{11}$$

Selectively focusing on the most important parts of the sensors' embeddings following the above way, the site node embeddings $\{\boldsymbol{h}_i^*\}_{i\in L}$ integrated the global sensor information are obtained. Further, the mean pooling of sites' embeddings $\sum_{i=1}^{L} \boldsymbol{h}_i^*/L$ is concatenated with $\{\boldsymbol{h}_i^*\}_{i\in L}$, and then a multilayer perceptron (MLP) function is leveraged to output the final evaluation of the value of site nodes that the agent may visit at time step t: $Q(s_t, a_t)$.

E. Training Method

The double DQN algorithm [52] is employed to learn the optimal policy for planning the traversal path of the sink. Untangling the selection and evaluation in the update of the Q value, the double DQN method reduces the overestimation error by learning two value functions, one of which is used to decide the greedy policy and the other to decide the value. The learning target Y_t of double DQN algorithm at t time step is shown in

$$Y_t = R_{t+1} + \gamma Q(s_{t+1}, \text{argmax}_a Q(s_{t+1}, a; \theta_t), \theta_t^-) \tag{12}$$

where θ_t refers to the parameters of the Q-network, θ_t^- are the parameters of the target network, and γ refers to the discount factor. The parameters of Q-network θ is updated as follows:

$$\theta_{t+1} \leftarrow \theta_t + \alpha(Y_t - Q(s_t, a_t; \theta_t))\nabla_{\theta_t} Q(s_t, a_t; \theta_t) \tag{13}$$

where α is a scalar step size. The pseudocode of the training method is described in Algorithm 1.

F. Complexity Analysis

1) Time Complexity: The time complexity of HGFF is primarily derived from the MPNN and the multihead attention layer. For MPNN, the complexity mainly depends on the number of edges E in the graph. The time complexity of the multihead attention layer is driven by the dot product operations between the query and the key matrices, which scale with the product of the number of sensors (n) and the number of sites (m). Consequently, the overall time complexity of our method can be written as $O(E + n \times m)$.

Algorithm 1: The Double DQN Algorithm

Initialize the replay buffer $\mathcal{D}$ and network Q_θ with random weights θ and target network Q_{θ^-} with $\theta^- = \theta$
for episode 1 to $\mathcal{M}$ **do**
  while not done **do**
    Get current state s_t
    Compute Q values according to network model Q_θ based on s_t and select an action a following *epsilon greedy*:

$$a = \begin{cases} argmax_a Q_\theta(s_{t+1}, a) & \text{with prob } 1-\epsilon \\ \text{sample from action space} & \text{with prob } \epsilon \end{cases}$$

    Execute a in the environment
    Get reward r_t and next state s_{t+1}
    Store transition (s_t, a, r_t, s_{t+1}) in $\mathcal{D}$
    Sample mini-batch experience from $\mathcal{D}$
    For the sampled experience, compute the update target in (12) and update the parameters θ according to (13)
  end while
  Every C steps perform $\theta^- = \theta$
end for

2) Space Complexity: The space complexity of HGFF primarily depends on the number and size of parameters in the neural network and the representation of the graph in memory. The total number of parameters in our model is $0.07M$. The "M" stands for million. The compactness of our model facilitates ease of deployment and operation in diverse network environments. Besides, the implementation of our approach on a centralized system or the mobile sink, both of which possess substantially more computing resources than individual sensor nodes.

V. Experiments

In this section, we introduce the experimental setup and results of our proposed method. In the experimental setup, the designed WSN environments for evaluating different approaches are introduced. Then, we present the indicators to probe the performance of different algorithms. Moreover, the baseline methods and their hyperparameters are illustrated in detail. Then, the hyperparameters during the training of our method and the baseline DRL-based method are given. In the experimental results, we show the results of the conducted extensive experiments on both static and dynamic maps, as well as some analysis. In addition, we further evaluate the effectiveness of the component of our method via ablation studies. What is more, the visualization of the sink mobile path is given, which directly reflects the effectiveness of our method.

A. Experimental Setup

1) Design of Virtual Environments: Two kinds of virtual environments are designed to simulate WSNs: stationary and dynamic environments. In stationary environments, the deployment of sensor nodes does not change during the lifetime of WSNs, and the energy load balancing of the network only depends on the movement of the mobile sink. Further, to cater to most situations in the real-world, dynamic WSN environments are also designed to test our proposed method, that is, during the operation of the network, the deployment of sensor nodes will change over time. The dynamic environments pose a great challenge to the response time of the algorithm. Besides, the initial location distribution of sensors is randomly generated.

TABLE II
Twelve Types of Maps, Which Differ in the Geographic Area, the Number of Sensors and Sites

Index	Sensors	Sites	Map Size
1	30	5×5	100×100
2	50	5×5	100×100
3	100	5×5	100×100
4	100	10×10	150×150
5	200	5×5	100×100
6	200	10×10	150×150
7	100	5×15	50×150
8	100	10×10	100×100
9	300	10×10	150×150
10	500	20×20	150×150
11	800	20×20	500×500
12	1000	25×25	1000×1000

Specifically, twelve types of WSN environments are designed for both static and dynamic cases, respectively. The initial location distribution of sensors in each type of dynamic map is consistent with that in the corresponding static map. All the scenarios are listed in Table II, which differ in map size and the number of sensors and sites. In particular, map 7 refers to the sensors and sites distributed in a rectangle sensing region (50×150). Besides, we set half of the sites randomly to be inaccessible in map 8 to simulate accidental situations in the real world. For other maps, the complexity is growing with the number of sensors and sites gradually increasing and distributed on a larger map. To test the scalability of our algorithm, we designed maps 11 and 12, which feature more sensors in large areas. The more complex maps bring greater challenges to the performance of the algorithm and computing resources.

To validate the performance of methods, we generate twelve maps for each type of WSN environment, respectively. As described in Section III, the deployment of sensors is randomly generated and the locations of sites are arranged in squares. In each map, the initial 2-D coordinate of every sensor is sampled uniformly at random in the square of size $[0, \text{mapsize}] \times [0, \text{mapsize}]$. In dynamic environments, the coordinates of sensors are changed at each time step. To simulate such a dynamic of sensors, the coordinate of the sensor i at time step t is determined by sampling a normal distribution $\mathcal{Z}$, in which the mean is the initial location of sensor i and the variance is set to 3. For the routing of data from sensors to the sink, the FA algorithm [51] is leveraged to construct the multihop flow routing tree, and the parameters are as follows. The constant coefficients a, b for computing the energy consumption of sending data in (1) are set to 50 nJ/bit and 100 $pJ/bit/m^2$, respectively. The maximum transmission range $d_{\max}$ in maps 1–10 is set to be

30 m and this value is 50 and 100, respectively, for maps 11 and 12, considering the increase in network coverage area on the two maps. The sensing rate z_i is 1 bit/s and the least sojourn time Δt is 3600 s.

2) Evaluation Metrics: The *average lifetime* and *computation time* are leveraged as indicators to evaluate the quality of methods. The *average lifetime* intuitively reflects the optimality of the solution, which can be denoted as $\overline{T}$. The *computation time* C_t refers to the time elapsed for the sink to make the decision on the site to stay next time step, in seconds, which reflects the efficiency of methods to obtain solutions. The calculation of $\overline{T}$ and C_t is defined as follows:

$$\overline{T} = \frac{\sum_{i=0}^{n} T_i}{n}$$
$$C_t = \frac{\text{time}_{\text{tot}}}{T}$$

where T_i represents the lifetime of the WSN in i episode, time_{tot} means the running time of an episode from resetting the environment until the WSN dies, and the T defined in Section III means the cumulative decision count of sink in an episode. Note that the corresponding actual lifetime of the WSN in i episode is $T_i \cdot \Delta t$ in seconds. For the proposed method and RL-based baseline methods, the two test indicators are measured by averaging the results of 20 test episodes after training, in which the agent performs action selection greedily. The inference time of the proposed method and all the compared methods are evaluated on a workstation with Intel(R) Core(TM) i9-10900K CPU @ 3.70 GHz.

3) Methods for Comparison: In our experiments, several existing algorithms are selected for comparison, which can represent three categories of methods: heuristic methods, hyperheuristic methods, and DRL methods.

a) Heuristic methods: For the heuristic methods, a rule-based method: GMRE method [4] is reproduced in our experiments, which leads the sink to move to the site node with the most residual energy greedily. In addition, the ACO method [55] is also implemented in the experiments. With relatively strong global search ability, the ACO algorithm has been shown able to get near-optimal solutions for the lifetime maximization problem in WSNs [12], [13]. The details of the two methods are as follows.

1) *GMRE*: Intuitively, GMRE proposed a heuristic ζ that the sink moves to the site with the largest residual energy of the sensors within a certain distance $\mathcal{L}$

$$\zeta = \max\ u_i^t, \forall l_j \in L, n_i \in N, d(l_j, n_i) < \mathcal{L}.$$

Following the rule ζ, the site which covers sensors with maximal residual energy is selected.

2) *ACO*: In nature, real ants can find the shortest path from a food source to a nest by leveraging pheromones without vision. Inspired by real ants, the ACO algorithm [55] is proposed to solve combinational optimization problems. In the ACO algorithm, there is a group of artificial ants to search for the optimal solutions according to the heuristic information and global pheromones. The heuristic information is predefined manually before the search, and the pheromones are updated based on the results of the historical exploration of ants. In this article, the heuristic functions in [13] are used to compute the initial pheromone and update the selection probability of each ant, including *average communication hop*, *average residual energy,* and *maximum step distance*. In this article, the experimental settings of the ACO algorithm [13] are set to their recommended optimal values, as detailed in Table III.

TABLE III
HYPERPARAMETERS FOR THE ACO ALGORITHM

Parameter	Value
Number of artificial ants	10
Exploitation rate to construct solution	0.95
Pheromone reinforcement rate	0.5
Weight of heuristic information	0.1
Weight of the average communication hops	10

TABLE IV
HYPERPARAMETERS FOR THE SL-GEP ALGORITHM

Parameter	Value
Evolving generations	500
Population size	10
Length of the head in main program	10
Length of head in ADFs	3
Number of ADFs in each chromosome	2

b) Hyperheuristic methods: Searching for the optimal solution among different heuristics automatically, the hyperheuristic methods are proposed to improve performance and flexibility. In this article, a self-learning gene expression programming (SL-GEP) algorithm [14] is selected as one of the baselines to compare with our proposed method. In SL-GEP, the chromosome representation of an individual consists of subfunctions, and the high-level heuristic is obtained by the combination of these functions. Besides, the subfunctions are made up of linking functions and the automatically defined functions (ADFs), which are a combination of handcraft low-level heuristic functions. The population is evolved by the use of evolutionary operations, such as mutation, crossover, and selection based on a modified differential evolution method to obtain individuals with higher fitness values. In our experiments, the parameter settings for the SL-GEP method [14] are configured to their recommended optimal values, as listed in Table IV.

c) DRL methods: To verify the effectiveness of our proposed framework, the double DQN method [52] (the detail of the method is in Section IV-E) is implemented as one of the baseline methods in the experiments, where the agent's Q-network is implemented by the neural network with fully connected layers.

4) Training Settings: The proposed method is trained in an end-to-end way with the embedding dimension $d_h = 64$, and the embeddings are learned with three massaging and updating

TABLE V
HYPERPARAMETERS FOR THE DRL-BASED ALGORITHMS

Parameter Name	Symbol	Value
Batch size	$\mathcal{N}$	64
Size of replay buffer	$\|D\|$	50 k
Training episodes	$\mathcal{M}$	100 k
Learning optimizer	-	Adam
Learning rate	α	1e-4
Maximal exploration probability	ϵ_{init}	0.99
Minimal exploration probability	ϵ_{fin}	0.01
Exploration decaying rate per episode	ϵ_{decay}	5e-5
Discount factor	γ	0.98

iterations. After the feature fusion in multihead attention training, the embeddings for site nodes are concatenated with 64-dimensional mean pooling of sensor nodes' embeddings. Then, the vectors are inputted to a one-layer MLP with hidden dimension 128 and ReLU activation function. For the baseline double DQN method, the Q-network of the agent is implemented by three fully connected layers with 64 hidden units and ReLU activation functions. Except for the above Q-network settings, the proposed method and RL baseline method share the training settings listed in Table V. For the hardware, the training of the neural network is conducted on one NVIDIA GeForce RTX 3080.

B. Experimental Results

1) Experiments on Stationary WSNs: According to the setup of experiments in Section V, we conduct a performance comparison between the proposed method and the baseline methods. In our experiments, the mathematical programming-based method is also used to find a solution with high quality. The mathematical programming-based method is implemented by the GNU Linear Programming Kit package (GLPK-4.65).[2]

The empirical results on twelve types of maps with static sensors are shown in Table VI, with the peak in bold. The value of $\overline{T}$ on each type of map is calculated by the mean of finally achieved lifetimes, which comes from twelve randomly generated scenarios of each type of map. For HGFF and baseline double DQN method, we run 20 test episodes and obtain the mean lifetime in each scenario. For the hyperheuristic method SL-GEP, the lifetime of each scenario is the mean of the lifetimes obtained by the final trained heuristic rule in 20 test episodes. For the random search method ACO, the results are obtained by the mean of 30 searches. The value of C_t is collected from the inference time of the final trained model. In DRL-based methods and hyperheuristic methods, C_t is collected by the mean of test episodes. Since all the calculations are online in the mathematical method and heuristic methods, C_t records the time from the beginning of the algorithms to the completion of finding its solution.

As shown in the tables, HGFF can obtain the maximal lifetime on all types of maps among the heuristics, hyperheuristic methods, and DRL-based algorithms. The mathematical method GLPK is usually guaranteed to find the optimal solution or proven bounds on the optimality gap. Therefore, it is an unexpected result that HGFF outperforms the GLPK method on map 1, which indicates that our method might potentially yield better solutions than those derived from traditional theoretical optimizations. The increased complexity of map 2, featuring a higher number of sensors, poses greater computational challenges. In this scenario, the precise calculations of GLPK proved more effective than our DRL-based approach. It is worth noting that our method is able to get a high-quality route for the mobile sink with less time. The computation time of HGFF is dozens of times less than that of GLPK and ACO, which shows that HGFF is more applicable and scalable. The stochastic search method ACO performs best among the heuristic methods and hyperheuristic methods. However, it still cannot obtain the high-quality solutions that HGFF gets, especially since its reasoning takes a long time. In particular, HGFF shows great adaptability through the good results achieved on map 7, which finally achieves twice the network lifetime ratio than the heuristic-based method. Note that the sensors on map 7 are deployed in a rectangle sensing area. The performance comparison on maps 7 and 8 means that HGFF is more suitable in the real world, where there are sensing areas of different shapes and numerous occurrences of emergency. Due to the extremely large time consumption and memory usage, the LP method implemented by GLPK is only carried on maps 1 and 2, which requires the least computational overhead of all maps.

Overall, our algorithm can consistently outperform all the compared methods on the static WSNs scenarios across all 12 types of maps. However, the MILP methods implemented by GLPK cost superabundant computational resources and time, which makes it hard to operate in WSNs with more sensors.

2) Experiments on Dynamic WSNs: The empirical results on twelve types of maps with dynamic sensors are shown in Table VII, with the peak in bold. In dynamic maps, the initial distribution of sensors on each type of map is the same as on the stationary map, and the random perturbation method of each sensor follows Section V-A1. There is an interesting phenomenon that the lifetime obtained by HGFF on most dynamic maps is longer than on corresponding static maps. This means that the location dynamics of sensor nodes have a positive effect on balancing the energy consumption of WSNs. However, heuristic methods perform worse on some dynamic maps than static ones, which shows that the dynamic map to some extent increases the difficulty of algorithmic decision-making. Accordingly, the inference time of all methods is increased due to the extra computation caused by a more complex simulation of dynamic WSNs. Similar to the performance on stationary maps, HGFF also performs very well on dynamic map 7. Besides, with the rising complexity of maps, the reasoning time of most methods gets longer on both static and dynamic maps. In summary, HGFF significantly outperforms the baseline methods on both stationary and dynamic maps.

3) Ablation Study: To probe the efficiency of HGFF, we conduct ablation experiments to investigate the influence of introducing type embeddings into initial features and the necessity

[2]https://www.gnu.org/software/glpk/

TABLE VI
COMPARISON ON STATIONARY WSNS RANGE FROM MAPS 1 TO 12

	MILP Method		Heuristic				Hyperheuristic		DRL Methods			
	GLPK		ACO		GMRE		SL-GEP		DDQN		HGFF	
Map	$\overline{T}$	$C_t(s)$	$\overline{T}$	$C_t(s)$	$\overline{T}$	$C_t(s)$	$\overline{T}$	$C_t(s)$	$\overline{T}$	$C_t(s)$	$\overline{T}$	$C_t(s)$
1	30.78(≈)	3.42	28.73(≈)	1.058	17.57(−)	0.002	27.71(−)	0.032	28.95(≈)	0.055	**32.98**	0.065
2	**56.11**(+)	18.59	43.42(−)	4.447	37.56(−)	0.002	42.1 (−)	0.011	43.54(−)	0.156	47.27	0.225
3	N/A	N/A	50.39(≈)	18.37	45.22(−)	0.007	50.1 (≈)	0.047	49.41(−)	0.753	**53.34**	0.844
4	N/A	N/A	22.64(≈)	6.63	17.2 (−)	0.006	20.7 (−)	0.025	21.21(−)	0.305	**25.15**	0.418
5	N/A	N/A	79.62(−)	101.7	58.4 (−)	0.035	73.6 (−)	0.284	72.96(−)	2.94	**83.79**	3.842
6	N/A	N/A	30.22(≈)	25.71	19.75(−)	0.025	24.25(−)	0.09	28.05(−)	1.3	**32.2**	1.64
7	N/A	N/A	28.67(−)	7.669	18.8 (−)	0.04	27 (−)	0.022	50.95(≈)	0.733	**54.48**	0.786
8	N/A	N/A	47.32(≈)	20.15	45.13(−)	0.07	46.9 (≈)	0.047	46.9 (≈)	0.72	**50.98**	0.915
9	N/A	N/A	35.13(≈)	53.293	17.6 (−)	0.029	32 (−)	0.23	31.54(−)	3.52	**36.74**	4.48
10	N/A	N/A	35.37(−)	85.36	22.1 (−)	0.082	34.2 (−)	0.525	34.58(−)	9.61	**38.63**	10.5
11	N/A	N/A	43.26(≈)	485.37	10.4 (−)	0.248	38.6 (−)	2.358	44.65(≈)	18.96	**45.49**	19.37
12	N/A	N/A	23.29(−)	373.51	8.25(−)	0.234	19.5 (−)	1.91	26.43(−)	20.11	**29.17**	22.34

Note: Symbols ≈, +, and − represent the corresponding method is similar to, significantly better, and worse than the HGFF method according to the Wilcoxon rank-sum test at $\alpha = 5\%$. The same symbols are also used in Table VII.

TABLE VII
COMPARISON ON DYNAMIC WSNS RANGE FROM MAPS 1 TO 12

	Heuristic				Hyperheuristic		DRL Methods			
	ACO		GMRE		SL-GEP		DDQN		HGFF	
Map	$\overline{T}$	$C_t(s)$	$\overline{T}$	$C_t(s)$	$\overline{T}$	$C_t(s)$	$\overline{T}$	$C_t(s)$	$\overline{T}$	$C_t(s)$
1	31.89(−)	1.03	11.33(−)	0.003	32.83(−)	0.003	32.64(≈)	0.102	**35.43**	0.104
2	46.97(−)	4.43	10.89(−)	0.003	52.66(≈)	0.012	50.47(≈)	0.383	**54.08**	0.313
3	55.62(−)	18.24	17.88(−)	0.005	65.12(≈)	0.037	56.28(−)	1.31	**65.78**	1.33
4	22.58(≈)	6.66	8.92(−)	0.028	25.25(≈)	0.024	21.05(−)	0.38	**25.49**	0.384
5	79.73(−)	103.73	19.95(−)	0.014	58.75(−)	0.2	83.75(−)	5.62	**93.47**	5.875
6	28.48(−)	27.09	11.7 (−)	0.016	20 (−)	0.051	31.47(−)	1.93	**34.86**	2.54
7	28.79(−)	7.69	19.4 (−)	0.037	32.7 (−)	0.021	57.83(−)	1.24	**61.14**	1.224
8	58.98(−)	20.28	45.8 (−)	0.028	66.7 (≈)	0.045	64.55(−)	1.21	**68.16**	1.35
9	29.72(−)	47.38	17.1 (−)	0.053	20.67(−)	0.085	35.22(−)	4.84	**42.94**	6.56
10	42.5 (−)	84.72	20.2 (−)	0.089	38.5 (−)	0.12	42.74(−)	12.27	**46.82**	13.07
11	34.2 (−)	352.79	10.65(−)	0.204	38.7 (−)	2.265	47.19(−)	19.13	**52.85**	20.09
12	26.37(−)	436.87	7.75(−)	0.192	19.11(−)	1.68	29.98(≈)	21.43	**33.49**	23.74

of attention-based feature fusion. To explore the performance improvements brought by the two components, three methods are tested on several types of static and dynamic maps: HGFF without feature fusion, HGFF without both type embedding and feature fusion, and HGFF. The experimental results of the three methods on randomly sampled maps are summarized in Table VIII.

As we can see, the performance gap between the HGFF without the feature fusion method and the HGFF without the two components method is shown in the third row and the second row, which reflects the improvement brought by type embedding. What is more, the comparison between HGFF without the feature fusion method and HGFF shows the effectiveness of the feature fusion mechanism. The results show that both two

TABLE VIII
ABLATION STUDIES FOR HGFF

Map	Map 1	Map 4	Map 7	Dyn Map 1	Dyn Map 4	Dyn Map 7
HGFF	32.98	25.15	54.48	35.43	25.49	61.14
W.o. feature fusion	31.83	23.55	52.94	34.52	24.1	60.16
W.o. node type embedding + w.o. feature fusion	31.71	23.43	52.6	34.29	23.82	60.16

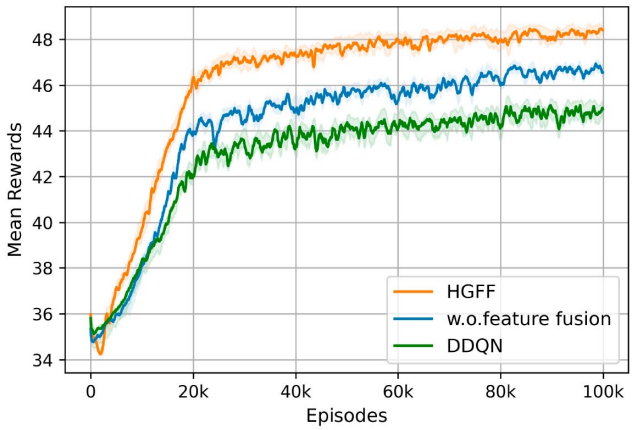

Fig. 3. Mean rewards of HGFF, HGFF w.o. feature fusion, and baseline DDQN method during training on map 1, [25%,75%] percentile interval is shaded.

components consistently boost performance. The enhancement resulting from feature fusion is evidently larger. Whereas, the type embedding only brings slight improvements. The reason for this phenomenon could be that the initial node features already encompass the heterogeneity information.

Take one scenario of map 1 as an example, we visualize the training process of HGFF, HGFF without feature fusion method, and baseline DDQN method. The mean rewards curve of the methods in training is shown in Fig. 3, and all the experimental results are performed at least five runs with different seeds. From the comparison of the baseline DDQN implemented by fully connected networks and HGFF without the feature fusion method, we find that heterogeneous graph learning plays an important role in improving performance. Besides, feature fusion is also very useful in enhancing the quality of node representations, which leads to further improvement.

4) Parameter Study: To investigate the influence of learning rates and replay buffer sizes on the final lifetime of the WSN, we conduct multiple training runs with varied random seeds to assess performance variability. The results are illustrated in Fig. 4. The study reveals that moderate learning rates (1×10^{-4} and 1×10^{-3}) tend to yield better performance, characterized by higher final lifetimes and low variance, which indicates stable and effective convergence. Conversely, excessively high-learning rates led to instability and poor performance, while very low-learning rates resulted in slower convergence and suboptimal solutions. Regarding buffer size, larger buffer sizes (50k, 75k, and 90k) are found to produce better results. This suggests that a larger buffer helps the model

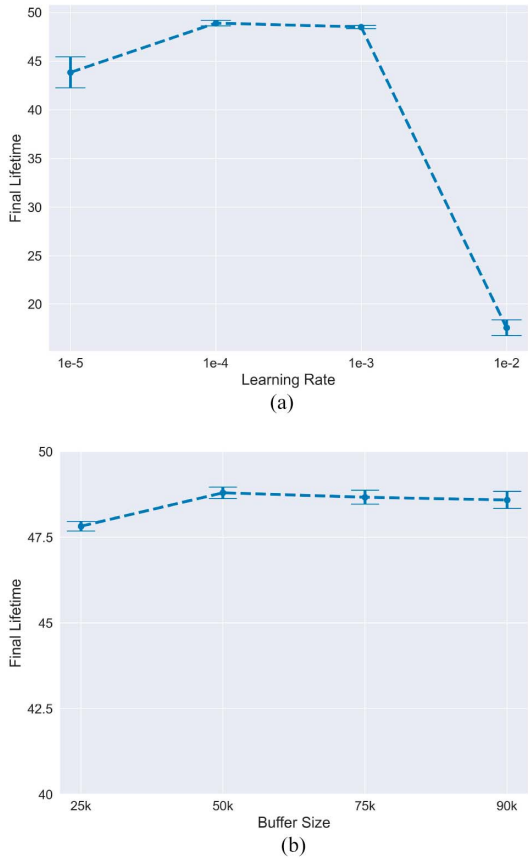

Fig. 4. Median lifetime of WSNs in final 10k train steps for the HGFF method, as a function of learning rate. The circle markers for each learning rate denote the mean lifetime, and the error bars show [25%,75%] percentile interval for the mean lifetime. (a) Impact of learning rates on final lifetime of WSN. b) Impact of buffer sizes on final lifetime of WSN.

maintain a diverse set of experiences, leading to more robust learning and better generalization. However, excessively large buffers might introduce inefficiencies and outdated experiences, while too small buffers could lead to overfitting and instability.

5) Visualization of the Sink Movement Path: According to the final trained model, we also visualize the movement path of the sink, which contains the selected site in sequential and the sojourn time on the site. We sampled several scenarios from map 1 which is deployed with the minimum number of sensors. To intuitively reflect the final performance of HGFF on maximizing lifespan, the route of sink constructed by baseline algorithms is also plotted for comparison. The movement paths of the sink constructed by HGFF and baseline methods on different scenarios are shown in Fig. 5.

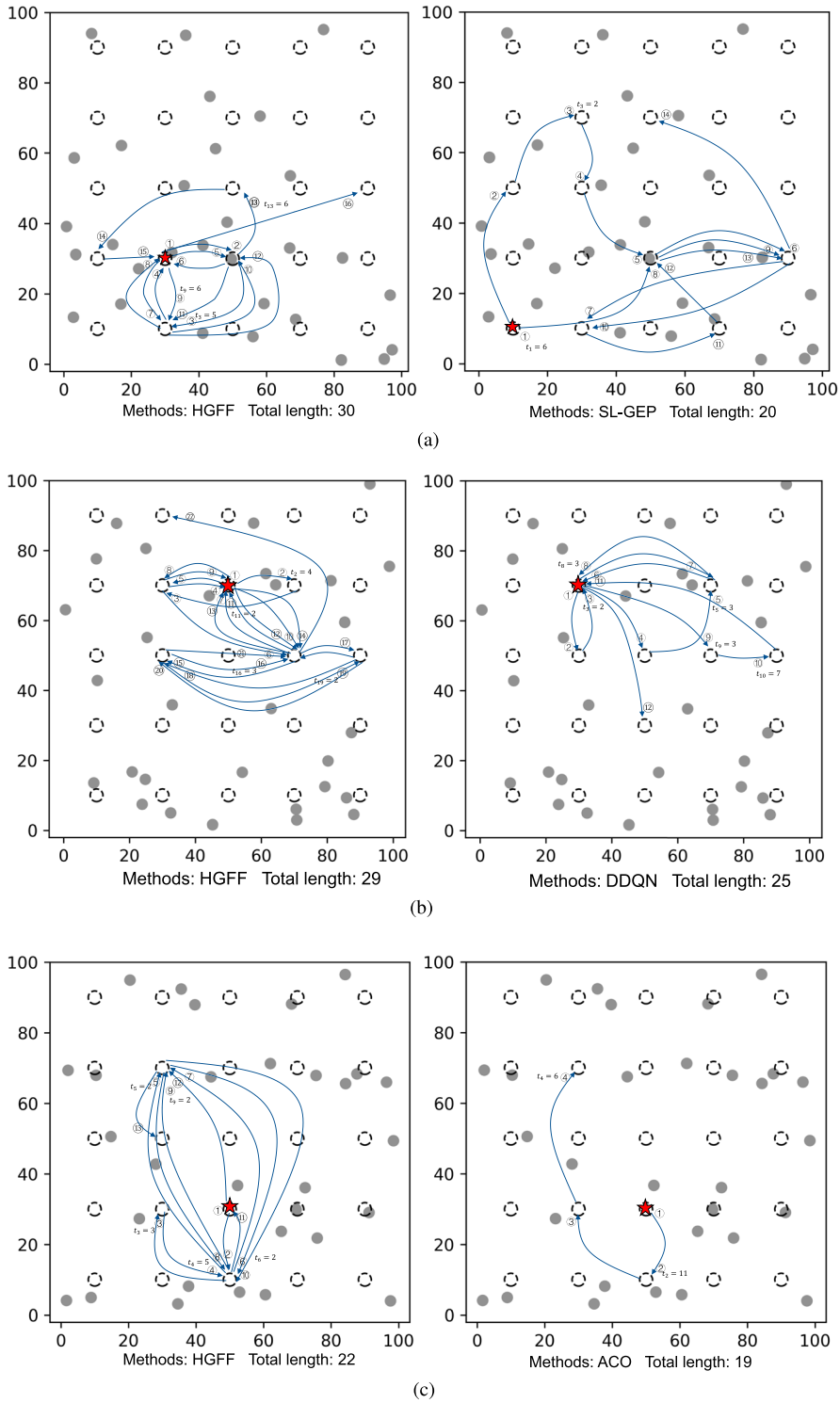

Fig. 5. Visualization of sink movement path obtained by HGFF and baseline methods. (a) Sink movement path obtained by HGFF and SL-GEP. (b) Sink movement path obtained by HGFF and DDQN. (c) Sink movement path obtained by HGFF and ACO.

The red five-pointed star denotes the sink, the dashed circle represents sites, and other gray solid points refer to sensors. The arrow reflects the sink's direction of movement. The number at the end of the arrow represents the movement order of the sink, and t_i indicates the sojourn time to stay at the position according to the movement order i. If there is no indication next to the arrow, it means $t_i = 1$. It is worth mentioning that before the running of the simulation of WSNs, the sink stays at the site that is closest to the center of the map. As we can see, HGFF can get the maximal route length among all the existing algorithms.

VI. Conclusion

In this article, a novel DRL-based framework called HGFF is proposed to address the NP-hard challenge of constructing the optimal route of the mobile sink to maximize the lifetime of WSN. Modeling the problem as an optimization on the heterogeneous graph, we introduce the learnable type embedding in graph learning to emphasize the heterogeneity of nodes. Moreover, the multihead attention technique is leveraged further to learn the relativity information between the heterogeneous nodes. The extensive experimental results show that our method is capable of learning solutions of good quality in a brief time without human knowledge. Besides, the proposed method also has referential significance in deciding the movement of the sink to different cluster heads in hierarchical routing protocol-based WSNs.

In the future, we would like to further study the problem of maximizing the lifetime of WSN where more than one sink exists. In addition, constructing the optimal data transmission route from sensors to the sink automatically is also an interesting future research direction.

References

[1] K. Nellore and G. P. Hancke, "A survey on urban traffic management system using wireless sensor networks," *Sensors*, vol. 16, no. 2, p. 157, 2016.

[2] C. V. Mahamuni and Z. M. Jalauddin, "Intrusion monitoring in military surveillance applications using wireless sensor networks (WSNs) with deep learning for multiple object detection and tracking," in *Proc. Int. Conf. Control, Automat., Power Signal Process. (CAPS)*, Piscataway, NJ, USA: IEEE, 2021, pp. 1–6.

[3] S. Li, B. Zhang, P. Fei, P. M. Shakeel, and R. D. J. Samuel, "Computational efficient wearable sensor network health monitoring system for sports athletics using IoT," *Aggression Violent Behav.*, 2020, Art. no. 101541.

[4] S. Basagni, A. Carosi, E. Melachrinoudis, C. Petrioli, and Z. M. Wang, "A new MILP formulation and distributed protocols for wireless sensor networks lifetime maximization," in *Proc. IEEE Int. Conf. Commun.*, vol. 8, Piscataway, NJ, USA: IEEE, 2006, pp. 3517–3524.

[5] V. Agarwal, S. Tapaswi, and P. Chanak, "A survey on path planning techniques for mobile sink in IoT-enabled wireless sensor networks," *Wireless Pers. Commun.*, vol. 119, pp. 211–238, 2021.

[6] J. Luo, "Mobility in wireless networks: Friend or foe: Network design and control in the age of mobile computing," Ph.D. dissertation, Lausanne, Switzerland: Verlag nicht ermittelbar, 2006.

[7] Y. Yun and Y. Xia, "Maximizing the lifetime of wireless sensor networks with mobile sink in delay-tolerant applications," *IEEE Trans. Mobile Comput.*, vol. 9, no. 9, pp. 1308–1318, Sep. 2010.

[8] B. Behdani, Y. S. Yun, J. C. Smith, and Y. Xia, "Decomposition algorithms for maximizing the lifetime of wireless sensor networks with mobile sinks," *Comput. Oper. Res.*, vol. 39, no. 5, pp. 1054–1061, 2012.

[9] F. Tashtarian, M. H. Y. Moghaddam, K. Sohraby, and S. Effati, "On maximizing the lifetime of wireless sensor networks in event-driven applications with mobile sinks," *IEEE Trans. Veh. Technol.*, vol. 64, no. 7, pp. 3177–3189, Jul. 2015.

[10] W. Liang, J. Luo, and X. Xu, "Prolonging network lifetime via a controlled mobile sink in wireless sensor networks," in *Proc. IEEE Global Telecommun. Conf. GLOBECOM*, Piscataway, NJ, USA: IEEE, 2010, pp. 1–6.

[11] J. Wang, J. Cao, R. S. Sherratt, and J. H. Park, "An improved ant colony optimization-based approach with mobile sink for wireless sensor networks," *J. Supercomput.*, vol. 74, no. 12, pp. 6633–6645, 2018.

[12] P. Maheshwari, A. K. Sharma, and K. Verma, "Energy efficient cluster based routing protocol for WSN using butterfly optimization algorithm and ant colony optimization," *Ad Hoc Netw.*, vol. 110, 2021, Art. no. 102317.

[13] J.-h. Zhong and J. Zhang, "Ant colony optimization algorithm for lifetime maximization in wireless sensor network with mobile sink," in *Proc. 14th Annu. Conf. Genetic Evol. Comput.*, 2012, pp. 1199–1204.

[14] J. Zhong, Z. Huang, L. Feng, W. Du, and Y. Li, "A hyper-heuristic framework for lifetime maximization in wireless sensor networks with a mobile sink," *IEEE/CAA J. Autom. Sin.*, vol. 7, no. 1, pp. 223–236, Jan. 2020.

[15] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," in *Proc. 5th Int. Conf. Learn. Represent. (ICLR), Workshop Track Proc.*, 2017.

[16] W. Kool, H. van Hoof, and M. Welling, "Attention, learn to solve routing problems!" in *Proc. 7th Int. Conf. Learn. Represent. (ICLR)*, 2019.

[17] E. Khalil, H. Dai, Y. Zhang, B. Dilkina, and L. Song, "Learning combinatorial optimization algorithms over graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.

[18] T. Barrett, W. Clements, J. Foerster, and A. Lvovsky, "Exploratory combinatorial optimization with reinforcement learning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, 2020, pp. 3243–3250.

[19] Y. Bengio, A. Lodi, and A. Prouvost, "Machine learning for combinatorial optimization: A methodological tour d'horizon," *Eur. J. Oper. Res.*, vol. 290, no. 2, pp. 405–421, 2021.

[20] G. Wu, M. Fan, J. Shi, and Y. Feng, "Reinforcement learning based truck-and-drone coordinated delivery," *IEEE Trans. Artif. Intell.*, vol. 4, no. 4, pp. 754–763, Aug. 2023.

[21] A. Forster and A. L. Murphy, "CLIQUE: Role-free clustering with q-learning for wireless sensor networks," in *Proc. 29th IEEE Int. Conf. Distrib. Comput. Syst.*, Piscataway, NJ, USA: IEEE, 2009, pp. 441–449.

[22] I. Mustapha, B. M. Ali, A. Sali, M. F. A. Rasid, and H. Mohamad, "An energy efficient reinforcement learning based cooperative channel sensing for cognitive radio sensor networks," *Pervasive Mobile Comput.*, vol. 35, pp. 165–184, 2017.

[23] S. Soni and M. Shrivastava, "Novel learning algorithms for efficient mobile sink data collection using reinforcement learning in wireless sensor network," *Wireless Commun. Mobile Comput.*, vol. 2018, pp. 1–13, 2018.

[24] M. Krishnan and Y. Lim, "Reinforcement learning-based dynamic routing using mobile sink for data collection in WSNs and IoT applications," *J. Netw. Comput. Appl.*, vol. 194, 2021, Art. no. 103223.

[25] C. J. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, pp. 279–292, 1992.

[26] H. Hasselt, "Double q-learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 23, 2010.

[27] H. Yetgin, K. T. K. Cheung, M. El-Hajjar, and L. H. Hanzo, "A survey of network lifetime maximization techniques in wireless sensor networks," *IEEE Commun. Surveys Tut.*, vol. 19, no. 2, pp. 828–854, Second quart. 2017.

[28] W. Xu, Q. Shi, X. Wei, Z. Ma, X. Zhu, and Y. Wang, "Distributed optimal rate–reliability–lifetime tradeoff in time-varying wireless sensor networks," *IEEE Trans. Wireless Commun.*, vol. 13, no. 9, pp. 4836–4847, Sep. 2014.

[29] S. Ehsan, B. Hamdaoui, and M. Guizani, "Radio and medium access contention aware routing for lifetime maximization in multichannel sensor networks," *IEEE Trans. Wireless Commun.*, vol. 11, no. 9, pp. 3058–3067, Sep. 2012.

[30] J.-H. Jeon, H.-J. Byun, and J.-T. Lim, "Joint contention and sleep control for lifetime maximization in wireless sensor networks," *IEEE Commun. Lett.*, vol. 17, no. 2, pp. 269–272, Feb. 2013.

[31] S. M. Chowdhury and A. Hossain, "Different energy saving schemes in wireless sensor networks: A survey," *Wireless Pers. Commun.*, vol. 114, no. 3, pp. 2043–2062, 2020.

[32] S. R. Lahane and K. N. Jariwala, "Secured cross-layer cross-domain routing in dense wireless sensor network: A new hybrid based clustering approach," *Int. J. Intell. Syst.*, vol. 36, no. 8, pp. 3789–3812, 2021.

[33] X. Xue, R. Shanmugam, S. Palanisamy, O. I. Khalaf, D. Selvaraj, and G. M. Abdulsahib, "A hybrid cross layer with Harris-Hawk-optimization-based efficient routing for wireless sensor networks," *Symmetry*, vol. 15, no. 2, p. 438, 2023.

[34] C. G. Cassandras, T. Wang, and S. Pourazarm, "Optimal routing and energy allocation for lifetime maximization of wireless sensor networks with nonideal batteries," *IEEE Trans. Control Netw. Syst.*, vol. 1, no. 1, pp. 86–98, Mar. 2014.

[35] D. Ye and M. Zhang, "A self-adaptive sleep/wake-up scheduling approach for wireless sensor networks," *IEEE Trans. Cybern.*, vol. 48, no. 3, pp. 979–992, Mar. 2018.

[36] V. Anand and S. Pandey, "New approach of GA–PSO-based clustering and routing in wireless sensor networks," *Int. J. Commun. Syst.*, vol. 33, no. 16, 2020, Art. no. e4571.
[37] Y. Xu, W. Jiao, and M. Tian, "An energy-efficient routing protocol for 3d wireless sensor networks," *IEEE Sensors J.*, vol. 21, no. 17, pp. 19550–19559, 2021.
[38] Z. Ding, L. Shen, H. Chen, F. Yan, and N. Ansari, "Energy-efficient relay-selection-based dynamic routing algorithm for IoT-oriented software-defined WSNs," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 9050–9065, Sep. 2020.
[39] I. S. Alshawi, Z. A. Abbood, and A. A. Alhijaj, "Extending lifetime of heterogeneous wireless sensor networks using spider monkey optimization routing protocol," *TELKOMNIKA (Telecommun. Comput. Electron. Control)*, vol. 20, no. 1, pp. 212–220, 2022.
[40] O. Younis, M. Krunz, and S. Ramasubramanian, "Node clustering in wireless sensor networks: Recent developments and deployment challenges," *IEEE Netw.*, vol. 20, no. 3, pp. 20–25, May/Jun. 2006.
[41] P. Rawat and S. Chauhan, "Particle swarm optimization-based energy efficient clustering protocol in wireless sensor network," *Neural Comput. Appl.*, vol. 33, pp. 14147–14165, 2021.
[42] S. A. Sert, "A novel hybrid grey wolf optimization methodology for resource constrained networks," in *Proc. 30th Signal Process. Commun. Appl. Conf. (SIU)*, Piscataway, NJ, USA: IEEE, 2022, pp. 1–4.
[43] D. L. Reddy, C. Puttamadappa, and H. Suresh, "Merged glowworm swarm with ant colony optimization for energy efficient clustering and routing in wireless sensor network," *Pervasive Mobile Comput.*, vol. 71, 2021, Art. no. 101338.
[44] R. Ramya and T. Brindha, "A comprehensive review on optimal cluster head selection in WSN-IoT," in *Proc. Adv. Eng. Softw.*, vol. 171, 2022, Art. no. 103170.
[45] N. Kumar, P. Rani, V. Kumar, P. K. Verma, and D. Koundal, "TEEECH: Three-tier extended energy efficient clustering hierarchy protocol for heterogeneous wireless sensor network," *Expert Syst. Appl.*, vol. 216, 2023, Art. no. 119448.
[46] M. Gatzianas and L. Georgiadis, "A distributed algorithm for maximum lifetime routing in sensor networks with mobile sink," *IEEE Trans. Wireless Commun.*, vol. 7, no. 3, pp. 984–994, Mar. 2008.
[47] Y. Yang, M. I. Fonoage, and M. Cardei, "Improving network lifetime with mobile wireless sensor networks," *Comput. Commun.*, vol. 33, no. 4, pp. 409–419, 2010.
[48] C. Tunca, S. Isik, M. Y. Donmez, and C. Ersoy, "Ring routing: An energy-efficient routing protocol for wireless sensor networks with a mobile sink," *IEEE Trans. Mobile Comput.*, vol. 14, no. 9, pp. 1947–1960, Sep. 2015.
[49] M. I. Khan, W. N. Gansterer, and G. Haring, "Static vs. mobile sink: The influence of basic parameters on energy efficiency in wireless sensor networks," *Comput. Commun.*, vol. 36, no. 9, pp. 965–978, 2013.
[50] J. Ren, Y. Zhang, K. Zhang, A. Liu, J. Chen, and X. S. Shen, "Lifetime and energy hole evolution analysis in data-gathering wireless sensor networks," *IEEE Trans. Ind. Inform.*, vol. 12, no. 2, pp. 788–800, Apr. 2016.
[51] J.-H. Chang and L. Tassiulas, "Maximum lifetime routing in wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol. 12, no. 4, pp. 609–619, Aug. 2004.
[52] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 30, 2016.
[53] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2017, pp. 1263–1272.
[54] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.
[55] M. Dorigo and K. Socha, "An introduction to ant colony optimization," in *Handbook Approximation Algorithms Metaheuristics*, 2nd Ed. Chapman and Hall/CRC, 2018, pp. 395–408.

Xiaoxu Han received the B.S. degree in information management and information system from the School of Information Management, Shanxi University of Finance and Economics, Taiyuan, China, in 2018, and the M.S. degree in software engineering from the School of International Education, Tianjin University, Tianjin, China, in 2021. She is currently working toward the Ph.D. degree in electronic information with the School of Future Technology, South China University of Technology, Guangzhou, China.

Her research interests include reinforcement learning, evolutionary computation, and combinatorial optimization.

Xin Mu received the Ph.D. degree in computer science and technology from the School of Computer Science and Technology, LAMDA Group, Nanjing University, Nanjing, China, advised by Zhi-Hua Zhou.

Currently, he is a Senior Researcher with Peng Cheng Laboratory, Shenzhen, China. He visited as a Research Assistant with LARC Laboratory, Singapore Management University, Singapore, supervised by Prof. Ee-Peng Lim and Prof. Feida Zhu from 2015 to 2018. His research interests include machine learning and data mining. He is currently working on ensemble methods, stream mining, and data governance.

Jinghui Zhong (Senior Member, IEEE) received the Ph.D. degree in computer application technology from the School of Information Science and Technology, Sun Yat-sen University, Guangzhou, China, in 2012.

Currently, he is a Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou. From 2013 to 2016, he worked as a Postdoctoral Research Fellow with the School of Computer Engineering, Nanyang Technological University, Singapore. His research interests include evolutionary computation, machine learning, and agent-based modeling. He has published more than 100 international journal and conference papers, including over 30 IEEE/ACM TRANSACTIONS journal papers.

Dr. Zhong won the 2023 IEEE Transactions on Emerging Topics in Computational Intelligence Outstanding Paper Award for his research work on evolutionary multitasking. He currently serves as an Editorial Board Member for *Memetic Computing* and *ICT Express*.