Received August 4, 2021, accepted August 20, 2021, date of publication August 27, 2021, date of current version September 13, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3108156

QIH: An Efficient Q-Learning Inspired Hole-Bypassing Routing Protocol for WSNs

PHI LE NGUYEN[1], NANG HUNG NGUYEN[1], TUAN ANH NGUYEN DINH[1], KHANH LE[1], THANH HUNG NGUYEN[1], AND KIEN NGUYEN[2], (Senior Member, IEEE)
[1]School of Information and Communication Technology, Hanoi University of Science and Technology, Hanoi 11615, Vietnam
[2]Graduate School of Engineering, Chiba University, Chiba 263-8522, Japan

Corresponding author: Phi Le Nguyen (lenp@soict.hust.edu.vn)

This work was supported in part by the Research Organization of Information and Systems (ROIS) National Institute of Informatics (NII) Open Collaborative Research under Grant 2021-21S1201, in part by the Japan Society for the Promotion of Science (JSPS) KAKENHI under Grant 19K20251 and Grant 20H04174, and in part by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under Grant 102.01-2019.302.

ABSTRACT This paper addresses the *local minimum phenomenon*, routing path enlargement, and load imbalance problems of geographic routing in wireless sensor networks (WSNs) with holes. These issues may degrade the network lifetime of WSNs since they cause a long detour path and a traffic concentration around the hole boundary. Aiming to solve these problems, in this work, we propose a novel geographic routing protocol for WSNs, namely, Q-learning Inspired Hole bypassing (QIH), which is lightweight and efficient. QIH's conceptual idea is to leverage Q-learning to estimate the distance from a node to the holes. QIH makes routing decisions following the nodes' residual energy, their estimated distance to the holes, and their distance to the destination. We first confirm the effectiveness of QIH by theoretical analysis. Then, we conduct extensive simulations of QIH in comparison to state-of-the-art protocols. The simulation results show that QIH outperforms the other protocols in terms of network lifetime, packet latency, and energy consumption.

INDEX TERMS Geographic routing, network lifetime, Q-learning, hole bypassing, energy efficiency.

I. INTRODUCTION

A wireless sensor network (WSN) has sensor nodes deployed over a region of interest with various applications, such as disaster management or agriculture monitoring [1]–[4]. Each sensor node senses its surrounding environment and sends the data via multiple hops to a destination (i.e., the sink). Many WSN applications require full sensing coverage, where the death of a single node may cause inefficacious network operations. Moreover, the time from the beginning until the first node's death (i.e., the earliest energy depletion) is defined as the network lifetime. Hence, one of the most critical problems in WSNs is maximizing the network lifetime. A sensor node typically consumes its energy for computation and transmission tasks. The former energy amount is insignificant compared to the latter [5], [6]. Specifically, the energy needed for transmitting a single bit is approximately the same amount for processing a thousand operations [6]. Therefore, an energy-efficient routing protocol is essential to conserve energy.

The associate editor coordinating the review of this manuscript and approving it for publication was Peng-Yong Kong.

Since sensor nodes have limited resources, the routing protocols for WSNs should be simple and energy-efficient. In this aspect, geographic routing has been one of the most popular approaches. The geographic routing protocol is stateless and does not use routing tables. Instead, it exploits the location information of 1-hop neighbors to make the routing decision. Typically, the protocol starts with a greedy strategy that selects the next forwarder among a node's neighbors. The chosen node has the smallest Euclidean distance to the destination, which should be shorter than the current node's distance. It is worth noting that in geographic routing, it is usually assumed that every node knows the location of its 1-hop neighbors, and the source node knows the location of the destination (thus, the location of the destination is inserted into the packet header). Therefore, to determine the next node, the current node only needs to search its neighbor table and find the node with the smallest distance to the destination. The computational complexity for determining the next node is only $O(m)$, where m is the number of the current node's 1-hop neighbors. Geographic routing performs well in dense WSNs but suffers from a severe drawback called the *local minimum phenomenon* with the occurrence of holes [7]–[11]. A hole is a region without working sensors that may be formed by either geographical obstacles or sensor node failure (i.e., due to battery depletion or external impacts such as fires or earthquakes). In such a case, the forwarding process is stopped at the hole boundary because there is no neighboring node closer to the destination than the current node.

The *hole-aware* approach determines and broadcasts the hole location in advance to the sensor nodes. When a node wants to send a packet, it exploits the aware information to create a routing path that avoids all the holes [12], [13]. In [14]–[19], the authors proposed a variation of hole-awareness by introducing a forbidden area for routing packets around every hole, which is a static region with a simple shape such as circle [14], [15], ellipse [16], hexagon [17], or convex hull [18], [19]. The forbidden area information is then disseminated to nodes to establish hole awareness. Although this approach can alleviate the local minimum problem, it suffers from many additional severe issues. First, this approach incurs a significant overhead caused by dissemination. Second, in most of the protocols [14]–[17], the hole bypassing paths tend to go around the forbidden areas' boundaries. This causes heavy traffic concentration on the nodes surrounding the areas, resulting in quick energy depletion. Several protocols, such as [20]–[23], can mitigate this traffic concentration phenomenon at the trade-off of high complexity in determining the routing path, hence suffering from a high packet delay.

The *heuristic* approach exploits heuristic algorithms to maintain the load balance between the nodes to extend the network lifetime. The selection of next forwarders depends on information such as the residual energy [24], [25] and the distance to the destination [24], [26]. Recent works [27], [28] show the applicability of reinforcement learning (i.e., Q-learning) for geographic routing in WSNs. By carefully designing the reward function, Q-learning can help to advance geographic routing. In [27], the reward function is a combination of the distance-to-destination and the link delay. Meanwhile, in [28], the reward function is contributed by the nodes' residual energy. Unlike the *hole aware* approach, the *heuristic* approach can eliminate the overhead of determining and disseminating hole information. In addition, this approach does not require a source node to calculate the destination's full path, hence potentially shortening the packet delay. Despite this improvement, few works in the *heuristic* approach solve the hole problem thoroughly. Most of the protocols in this approach still have issues such as high packet drop ratios, especially for packets initiated at nodes staying inside the holes' concave regions.

This work aims to take advantage of both approaches mentioned above in a lightweight, efficient routing protocol. In addition to the local minimum phenomenon, we have also addressed two related problems: routing path enlargement and load imbalance. We then propose the Q-learning inspired hole bypassing (QIH) protocol for solving the problem. In the QIH design, we observe that the local minimum problem only occurs when the packet arrives at a node that stays inside the convex hull of a hole. QIH alleviates the phenomenon by determining the hole boundary as in the *hole aware* approach but without broadcasting the information to the whole network. QIH limits the disseminating region to the inside of the convex hull. For the nodes on the outside, QIH uses Q-learning to estimate their distances to the hole. QIH also adopts Q-learning to avoid routing path enlargement and to balance traffic among the nodes. To this end, QIH newly includes the reward function, which is designed to combine the residual energy information, the distance to destination, and the estimated distance to the hole. To the best of our knowledge, among the related works utilizing Q-learning in WSN routing, this work is the first to consider the distance to the hole in the reward function. The main contributions of our paper are as follows.

- We propose QIH, a Q-learning-based routing protocol for bypassing holes. QIH has a novel reward function that considers three factors: residual energy, the distance to the destination, and the distance to the hole.
- We newly introduce a lightweight hole information dissemination that only needs to broadcast hole information within the scope of the so-called hole's convex hull. Moreover, we design a feedback mechanism that helps a node outside a hole's convex hull estimate the distance to the hole.
- We perform a theoretical analysis to prove the loop-free property of QIH.
- We conduct extensive experiments to evaluate the impacts of key parameters in QIH. We then provide a guideline for choosing the optimal parameters.
- We compare the performance of QIH to existing approaches. QIH outperforms the others in terms of network lifetime, packet latency, and energy consumption.

The remainder of the paper is organized as follows. Section II provides preliminaries that help to better understand the paper. Section III introduces the proposed QIH protocol. The theoretical analysis is in Section V. Section VI shows the evaluation results. The related works are introduced in Section VII. Finally, Section VIII concludes the paper.

II. PRELIMINARIES

This section first introduces the definitions and notations used throughout this paper. Then, it presents the general ideas used to achieve protocol design goals and a Q-learning introduction.

A. DEFINITIONS AND NOTATIONS

We assume that each node knows its position and its 1-hop neighbors (e.g., by using positioning services [29] and the neighbor notification packets, respectively). In addition, the source node knows the position of the destination node. This assumption is legitimate in geographic routing [8], [14]–[16], [18], [20], [21], [25], [30] We illustrate the definitions in Fig. 1.

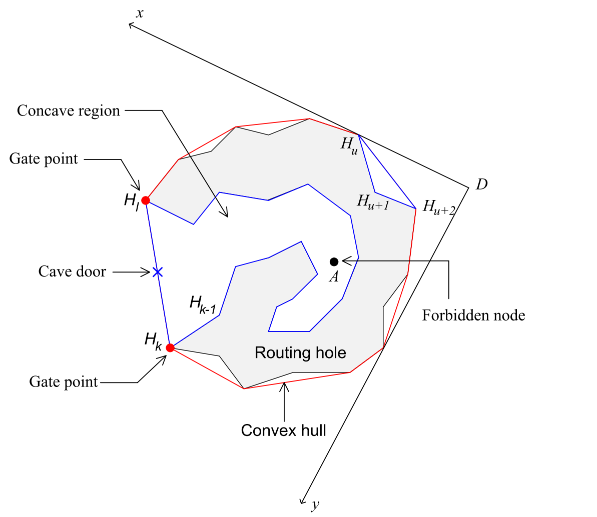


FIGURE 1. Definition illustration (the red and blue lines represent the convex hull boundary and the concave boundary, respectively). The two red points indicate the gate, while the blue points indicate the cave door).

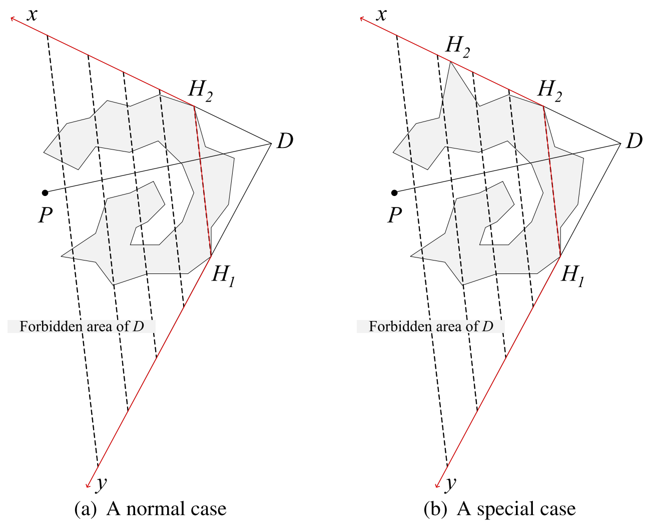


FIGURE 2. Illustration of the definition of forbidden area.

Definition 1 (Routing hole): A routing hole is defined as a nonself-intersecting polygon that has all vertices as sensor nodes (i.e., called hole boundary nodes*) with two following conditions. First, its interior does not contain any sensor nodes. Second, the Euclidean distance between two consecutive vertices is within their transmission range.*

Definition 2 (Convex hull): The convex hull of hole $\mathcal{H}$ is defined as a convex polygon that has all vertices as vertices of $\mathcal{H}$.

Definition 3 (Concave region): Let H_k, H_l be two consecutive vertices of the convex hull of $\mathcal{H}$ such that $k < l - 1$. The polygon $H_k H_{k+1} \dots H_l H_k$ is defined as a $\mathcal{H}$'s concave region*, of which H_k, H_l are called* gate-points*. The* cave-door *of a concave region is the midpoint of the segment connecting the two gate points.*

Definition 4 (Forbidden node): A node is called a forbidden node *of a hole if it stays inside the convex hull of the hole. Otherwise, the node is a non-forbidden node.*

Definition 5 (Forbidden area): Given a destination D and a hole H, the forbidden area of D with respect to H is the area containing all points P such that DP intersects H's interior.

The above forbidden area of D can be determined as follows. Let H_1 and H_2 be two vertices of H staying closest to D such that $\overrightarrow{DH_1}$ and $\overrightarrow{DH_2}$ create the smallest and largest angles with the x-axis; then, the forbidden area of D with respect to H is the plane delimited by the rays DH_1, DH_2 and H_1H_2. Figure 2 illustrates the definition of a forbidden area. In Fig. 2(b), DH_2 goes through another boundary node H_3. As H_2 is closer to D than H_3, the forbidden areas are defined by the rays DH_1, DH_2, and H_1H_2.

Definition 6 (Bypassing hole): When a packet is being forwarded by either HOLE-DETOUR or the HOLE-ESCAPE mode concerning a hole H, it is said to bypass *H.*

Definition 7 (Hole bypassed): A packet is said to bypass *a hole H if it has been bypassing H, and currently, it is not bypassing H.*

B. PROTOCOL DESIGN CONCEPTS

We aim to design a protocol to address three problems: the local minimum phenomenon, routing path enlargement, and load imbalance. We observe that the local minimum phenomenon occurs when the packet arrives at a node staying in a concave region. Our idea to alleviate this phenomenon can be described as follows. Initially, we consider two forwarding packet cases based on node position (i.e., in a concave region of a hole and outside of all holes' concave regions). The packet is first sent to a nearby cave-door in the former. It will then be targeted to the destination. In the latter, the packet is forwarded to avoid the vicinity of all the holes. We then consider a packet initiated at a node staying outside of all holes' concave regions. The packet is forwarded to avoid the vicinity of all the holes. We realize the forwarding strategies in those cases by introducing three forwarding modes (i.e., HOLE-ESCAPE, HOLE-DETOUR, and Q-GREEDY). HOLE-ESCAPE is applied when a packet is at a node inside a concave region of a hole. The objective of this mode is to forward the packet toward the cave-door. Therefore, the next forwarder in this mode tends to be the neighbor that is closest to the cave-door. HOLE-DETOUR is applied at a node that stays outside of all concave regions and near the hole's convex hull. This means that the node using this mode may have neighbors staying inside a concave region. Therefore, this mode aims to prevent the packet from approaching the hole and alleviating the routing loop. Q-GREEDY is used when the node holding the packet stays far from all the holes. In this mode, the packet is always forwarded to a node closer to the destination than the current node. The details of the forwarding modes are detailed in Section III. Despite the mode used, the routing decision is always made based on Q-learning techniques. More specifically, it relies on the Q-values of the neighbors. Q-learning's reward function is also designed to achieve our last design goals: alleviating routing path enlargement and balancing traffic over the network.

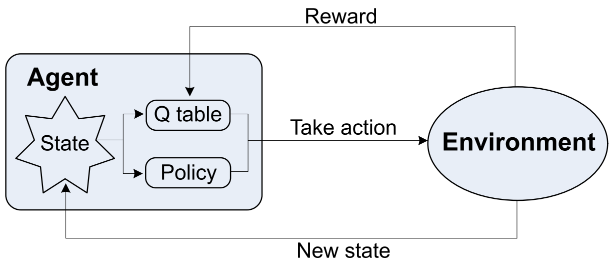


FIGURE 3. Q-learning framework.

C. Q-LEARNING TECHNIQUE

Q-learning is a reinforcement learning technique that lets a system learn to achieve a specific goal based on experience. Figure 3 illustrates the Q-learning framework, including an environment, an agent, states, and actions. Each possible action is assigned a Q-value at every state, representing the action's approximate goodness concerning the agent's goal. An agent chooses actions according to their Q-value and a policy. After performing an action, the agent modifies its policy to attain its goal. The Q-value is updated using the following equation: $Q(S_t, A_t) \leftarrow (1-\alpha)Q(S_t, A_t) + \alpha\ [R_t + \gamma\ \underset{a}{max}\ Q(S_{t+1}, a)]$ where $Q(S_t, A_t)$ is the Q-value of the current state S_t when action A_t is selected at time t. R_t represents the reward of performing action A_t at state S_t, and $\underset{a}{max}\ Q(S_{t+1}, a)$ is the maximum possible Q-value in the next state S_{t+1} for all possible actions a. α and γ are the learning rate and the future reward discount factor, respectively. Their values are set between 0 and 1.

In the routing context, a packet is considered an agent. Moreover, a sensor node N represents a state in the state set. A neighbor node B of N is considered a transitional state of node N. An action a defines the packet transmission from node N to node B. The reward function is critical to Q-learning, which decides the performance of the agent. Our routing protocol aims to alleviate the local minimum problem, routing path enlargement, and load imbalance. Therefore, we design a reward function such that a node staying further from the holes, close to the destination, and having more residual energy will tend to receive a higher reward. The details of the reward function are outlined in Section IV.

III. Q-LEARNING INSPIRED HOLE BYPASSING PROTOCOL

A. PROTOCOL OVERVIEW

We initially denote N and D as the current node and the destination, respectively. When N wants to forward a packet, it first determines a list of forwarder candidates, a subset of its 1-hop neighbors. Then, among all the candidates, the one with the highest Q-value is chosen as the next forwarder. Moreover, the forwarder candidate selection depends on the packet's forwarding mode (i.e., Q-GREEDY, HOLE-DETOUR, or HOLE-ESCAPE). In Q-GREEDY mode, the candidates are all 1-hop neighbors closer to the destination than the current node. When the current node is located near the hole, it may be a stuck node (i.e., without neighbors with a shorter distance to the destination), or the shorter-distance neighbors are stuck nodes. In such a case, the HOLE-DETOUR mode is applied to alleviate the local minimum problem. The next forwarder candidate has to satisfy two conditions. First, it stays outside of all concave regions of the hole. Second, it resides on the same half plane with N's 1-neighbor that is outside of all the concave regions and closest to D, with respect to $\overrightarrow{ND}$. Figure 4 illustrates the next forwarder candidate selection algorithm in HOLE-DETOUR mode.

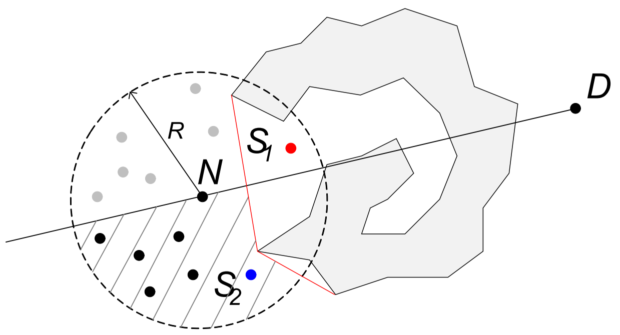


FIGURE 4. This illustrates the next forwarder candidate selection scheme in the HOLE-DETOUR mode. S_2 is the 1-hop neighbor of N, staying outside of the concave region and being closest to the destination D. The red nodes are not candidates because they remain inside the concave region. The gray nodes are not candidtaes since they do not stay on the same side with S_2. The black nodes are the next forwarder candidates.

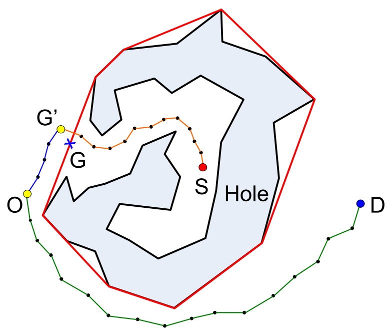


FIGURE 5. An example of the data forwarding scheme.

The HOLE-ESCAPE mode is applied when the current node stays inside a concave region. In this case, the next forwarder candidates are either closer to the cave-door than the current node or stay outside of the concave region. Figure 5 shows an example of routing a data packet from a source node S to a destination node D. As S stays inside the concave region, S uses the HOLE-ESCAPE mode and forwards the packet towards the cave-door (i.e., G). When the packet arrives at a sensor node G' staying outside of the concave region, the forwarding mode is switched to HOLE-DETOUR. As the neighbor of G' that is closet to the destination is on the right side of the vector $\overrightarrow{SD}$, the forwarder candidates must also stay on the right side of $\overrightarrow{SD}$. Accordingly, the packet is routed along the blue path until it arrives at node O, which has nonstuck neighbors that are closer to the destination. O switches the forwarding mode to Q-GREEDY and greedily forwards the packet towards the destination D.

To support the data forwarding scheme, we need a control plane consisting of two steps. The first is to identify the holes' boundaries, and the second is to disseminate the holes' information. We broadcast the hole information only for nodes inside the hole's convex hull to reduce the control overhead. Note that the number of nodes staying inside the holes' convex hulls is insignificant compared to the nodes' total number. Moreover, we propose a mechanism for a node outside of a hole's convex hull to estimate the distance to the hole. This mechanism is detailed later.

B. HOLE DETERMINATION

All nodes determine whether they are on the boundary of a hole using the TENT rule[1] described in [8]. Each hole boundary node creates a hole boundary determination (HBD) packet and sends it to its left neighboring node. These HBD packets are then forwarded by the right-hand rule described in [8] to determine the hole boundary. Suppose the current hole boundary node is t_i and its previous node is t_{i-1}; then, the right-hand rule determines node t_{i+1} as follows. Draw a ray l with direction $t_i t_{i-1}$ and sweep it around t_i counterclockwise; then, t_{i+1} is the first 1-hop neighbor of t_i hit by l. Figure 6 illustrates the right-hand rule.

Note that multiple HBD packets may be created for each hole; thus, to avoid overhead, we eliminate the late-coming, redundant packets. Consequently, for each hole, there is only one HBD packet that can go around the hole without being dropped by intermediate nodes. We denote the creator of this HBD packet as H_0. Then, H_0 has the location information about all nodes on the hole boundary. After the HBD packet comes back to its creator, i.e., H_0, we obtain a list of all the hole's coordinate vertices. H_0 then determines the convex hull of the hole and triggers the next step. In addition, H_0 identifies the boundary nodes belonging to concave regions. The information of all concave regions and the convex hull is then embedded into the HBD message. We call this updated HBD message the hole cave announcing (HCA) message.

C. HOLE INFORMATION DISSEMINATION

As the information of all the holes is disseminated in the same way, in the following, we describe the dissemination algorithm of one hole. H_0 broadcasts the HCA message to its one-hop neighbors. Upon receiving an HCA message, a node performs the following tasks. If the node has already received an HCA message or stays outside of the hole's convex hull, then it simply drops the message. Otherwise, the node must stay inside or on the boundary of a concave region. Then, it stores information of the hole's convex hull and the concave region (i.e., the location of all nodes on the boundary of the concave region) contained before broadcasting an HCA message to its neighbors. After that, all the forbidden nodes and their 1-hop neighbors obtain the hole's convex hull information. The nodes inside the hole's convex hull have

[1]p is a hole boundary node if it has two adjacent neighbors u, v such that the center of the circumcircle of triangle puv is out of p's transmission range.

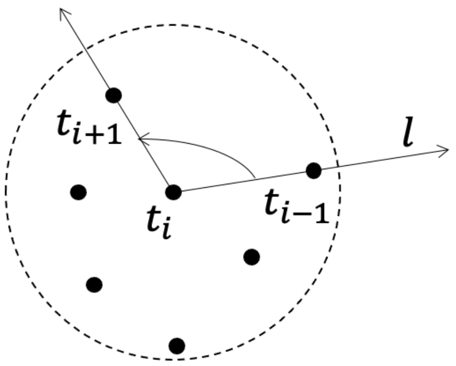


FIGURE 6. This illustrates the right-hand rule in the hole determination algorithm. In this figure, the black dots represent sensor nodes. Suppose that t_{i-1} and t_i are the previous and the current boundary nodes; then, the next boundary node t_{i+1} is the 1-hop neighbor of t_i such that $\overrightarrow{t_i t_{i+1}}$ makes the smallest angle with $\overrightarrow{t_i t_{i-1}}$ in a counterclockwise direction.

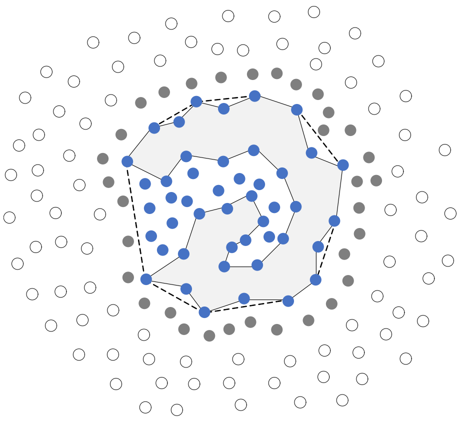


FIGURE 7. In this illustration of the hole information dissemination scheme, the blue nodes receive and store information of hole boundary nodes. The gray nodes receive and store the information of the hole's convex hull vertices. The blank nodes do not receive any information.

concave region information. This information is used to help packets escape the hole in the HOLE-ESCAPE mode. Meanwhile, the convex hull information guides the packet in the HOLE-DETOUR mode. The hole information is only disseminated to nodes inside the hole's convex hull instead of the whole network in our approach. The nodes outside the hole's convex hull estimate the distance to the hole by updating the parameter h in the local memory when receiving a feedback message from a neighbor. In this way, we can significantly reduce the number of broadcasting packets, as the number of nodes inside the convex hull is much smaller than those outside. Although this approach requires an updating parameter h, it is very lightweight, as its complexity is only $O(1)$. Moreover, according to [5], [31], the computing energy is much smaller than the communication energy.

Figure 7 presents an example of the hole information dissemination algorithm.

D. DATA FORWARDING

Our data forwarding algorithms consist of three key points below. First, we introduce three forwarding modes to cope with the local minimum problem. Second, we propose a mechanism to avoid the routing loop. Third, we exploit Q-learning to balance the traffic over the network and alleviate the routing path enlargement. In the following, we describe the forwarding modes and then present the Q-value determination algorithm.

1) PACKET HEADER

The packet header contains the following fields.

- *dest* is the coordinates of the destination. This information is initiated by the source node and remains unchanged during the routing process.
- *direction* is either *left* or *right*. This field indicates the direction of the routing path in the HOLE-DETOUR mode. The forwarding direction is assigned when the packet is switched to the HOLE-DETOUR mode and remains unchanged after that.
- *view_point_list* contains the coordinates of viewpoints of all the holes the packet has bypassed. Based on *view_point_list*, an intermediate node can determine the forbidden areas of the bypassed holes and thus alleviate forwarding the packet into the forbidden areas. This field helps to assure the loop-free property of the routing scheme.
- *convex_hull_vertices* consists of coordinates of all vertices of the convex polygon of the hole the packet is bypassing. Intermediate nodes use this field to check whether the packet stays inside the forbidden area.

2) FORWARDING MODES

Let N be an intermediate sensor node and D be the destination of a packet to which N wants to forward. If N stays inside of a concave region of a hole, then it uses the HOLE-ESCAPE mode to escape the hole. The term *escape the hole* means that the packet is forwarded to a nonforbidden node. When N is outside of all concave regions, it considers the neighbors satisfying conditions **G-1**, **G-2**, and **G-3** described below. If N has neighbors satisfying all the conditions, the Q-GREEDY mode is applied. Otherwise, N uses the HOLE-DETOUR mode. Algorithm 1 represents the pseudocode to select the forwarding mode. After determining the forwarding mode, N applies the algorithm described in the following to choose the next forwarder.

HOLE-ESCAPE mode: The current node N with this mode determines all of its neighbors whose shortest path to the concave region's cave-door is less than its path. If such neighbors exist, then N forwards the packet to the one with the highest Q-value. By doing so, the packet can gradually escape the concave region. If there is no such neighbor, then N is closer to the cave-door than all of its neighbors. It must have a neighbor staying outside of the concave region. In this case, N forwards the packet to the neighbor with the highest Q-value outside the concave region.

Q-GREEDY mode: When the current node stays outside of all concave regions, it determines whether any 1-hop neighbor satisfies the following conditions.

Algorithm 1 Forwarding Mode Determination

```
Input: packet $p$
Output: packet $p$ with determined forwarding mode
$p \leftarrow$ the packet
$N \leftarrow$ the current node; $D \leftarrow$ the destination;
if $N$ stays inside a concave region then
    $p.forwarding\_mode$ = HOLE-ESCAPE
else
    $\mathcal{L} \leftarrow$ null for $B \in N.neighbor\_table$ do
        if $B$ is closer to the $D$ than $N$ then
            $flag_1 = true$
        if $B$ is not a forbidden node then
            $flag_2 = true$
        for $(V_1, V_2) \in p.view\_point\_list$ do
            if $B \in \angle V_1 D V_2$ then
                $flag_3 = false$
                break
        if $flag_1$ and $flag_2$ and $flag_3$ then
            $\mathcal{L}$.append($B$)
    if $\mathcal{L}$ is not null then
        $p.forwarding\_mode$ = Q-GREEDY
    else
        $p.forwarding\_mode$ = HOLE-DETOUR
        $p.convex\_hull\_vertices \leftarrow$ current hole's convex hull
Return $p$
```

- **G-1.** They are closer to the destination than the current node.
- **G-2.** They are not forbidden nodes. This condition can be checked by using the *convex_hull_vertices* field in the packet header. Specifically, the nodes staying inside the convex hull defined by vertices in *convex_hull_vertices* are considered forbidden nodes.
- **G-3.** They do not belong to the forbidden areas of the holes the packet has bypassed. This condition is checked by using the *view_point_list* field in the packet header.

If such neighbors exist, the packet is forwarded by the Q-GREEDY mode, where the next forwarder is the one with the highest Q-value. Otherwise, N must have information on the hole (as will be proven in Proposition 1). Therefore, N determines the hole's convex hull, inserts into the *convex_hull_vertices* field, and switches the forwarding mode to the HOLE-DETOUR mode.

HOLE-DETOUR mode: In the HOLE-DETOUR mode, to avoid a routing loop, a packet is always forwarded in a fixed direction (i.e., concerning the line connecting the current node and the destination). In the following, we call this direction the packet direction. The packet direction (stored in the packet header's *direction* field) is defined when the packet is either initiated with or switched to the HOLE-DETOUR mode. If the packet has bypassed other holes before, then the viewpoints of all the bypassed holes are stored in the *view_point_list* field. The current node then checks whether the line connecting it and the destination intersect any previous bypassed hole. If it does, then the packet direction is kept at the same value when bypassing the last hole. Otherwise, the current node determines an anchor for setting the direction of the forwarding packet. A similar process is applied for the packet bypassing the first hole (i.e., the *view_point_list* field is NULL). The anchor is the neighbor outside of all concave regions of the current hole and closest to the destination. The packet's direction is set to the anchor's direction concerning the vector connecting the current node and the destination. Expressly, the packet direction is set to *right* if the closest neighbor stays on the right side of the vector and is set to *left* otherwise. The next forwarder candidates are all the neighbors satisfying the following conditions.

- **D-1.** They stay outside of all the concave regions of the current hole.
- **D-2.** They reside on the same side as the packet direction concerning the vector that connects the current node and the destination.
- **D-3.** They stay outside of the forbidden areas of all the holes that the packet has bypassed.

The current node then selects the node with the highest Q-value among the next forwarder candidates and forwards the packet. The details of the forwarding modes are presented in Algorithm 2.

Algorithm 2 Next Forwarder Selection

```
Input: packet $p$
Output: the next node to forward packet $p$
$N \leftarrow$ the current node; $D \leftarrow$ the destination;
$p \leftarrow$ the packet; $\mathcal{L} \leftarrow$ null;
if $p.forwarding\_mode = HOLE\text{-}ESCAPE$ then
    for $B \in N.neighbor\_table$ do
        if $shortest\_path(B, cave\text{-}door) < shortest\_path(N, cave\text{-}door)$ then
            $\mathcal{L}$.append($B$)
    if $\mathcal{L}$ is empty then
        for $B \in N.neighbor\_table$ do
            if $B$ stays outside of concave regions then
                $\mathcal{L}$.append($B$)
else if $p.forwarding\_mode = HOLE\text{-}DETOUR$ then
    if $p.direction = null$ then
        $Anchor \leftarrow$ the neighbor of $N$, which stays outside of all concave regions and is closest to the destination;
        $p.direction \leftarrow$ the direction of $Anchor$ with respect to $\overrightarrow{ND}$;
    for $B \in N.neighbor\_table$ do
        for $(V_1, V_2) \in p.view\_point\_list$ do
            if $B \in \angle V_1DV_2$ then
                $flag_1 = false$
                break
        if $B$ stays outside of all concave regions of the current hole then
            $flag_1 = true$
        if $B$ stay on the same side as $p.direction$ concerning $\overrightarrow{ND}$ then
            $flag_2 = true$
        if $flag_1$ and $flag_2$ and $flag_3$ then
            $\mathcal{L}$.append($B$)
else
    for $B \in N.neighbor\_table$ do
        if $B$ is closer to $D$ than $N$ then
            $flag_1 = true$
        for $H \in p.convex\_hull\_vertices$ do
            if $B \in H$ then
                $flag_2 = false$
                break
        for $(V_1, V_2) \in p.view\_point\_list$ do
            if $B \in \angle V_1DV_2$ then
                $flag_3 = false$
                break
        if $flag_1$ and $flag_2$ and $flag_3$ then
            $\mathcal{L}$.append($B$)
if $\mathcal{L}$ is empty then
    Return $-1$
Return item of $\mathcal{L}$ whose Q-value is the highest
```

IV. Q-VALUE DETERMINATION

Every node maintains a Q-table that stores information about the Q-values of its neighbors. Let us denote by N a sensor node and B a neighbor of N. N determines the Q-value of B as follows. First, the Q-value of B is initialized at 0. Then, every time after N sends a packet to B, N updates B's Q-value based on the feedback information received from B. We describe the feedback mechanism below. After that, we detail the Q-value updating algorithm.

Feedback mechanism: After N sends a data packet to B, it may receive a feedback message from B. N can use feedback to update its neighbors' status and modify its policy in choosing the next forwarder. Generally, the more frequently the neighbor feeds to N, the more accurately N can update. However, sending and receiving more feedback messages consumes more energy. In our protocol, after receiving the packet from N, the neighbor B only sends a feedback message to N if either of the following conditions is satisfied.

- **F-1.** B cannot find any next hop to forward the packet, and the packet has to be dropped at B.
- **F-2.** B is the destination.
- **F-3.** The residual energy of B is below a threshold. Let e_c, e_p be B's current residual energy and the one at the moment when B sent the last feedback message to N. Then, the condition for sending the feedback message is $e_c < \delta_e * e_p$, where δ_e is a parameter ranging in (0, 1).

The feedback message contains the estimated distance from B to the nearest hole, h_B, and the maximum Q-value of B's neighbors, Q_B. In addition, for the first two conditions, the message feedback includes a binary value indicating whether the packet was dropped at B or B is the destination. In the last case, the message consists of B's residual energy, denoted as $E_r(B)$.

Q-value updating: After sending a packet to B, N performs the following tasks to update the Q-value of B (denoted as $Q_N(B)$).

- If N receives feedback from B, it first checks the information of h_B to update its distance to the nearest hole. Then, it uses information in the feedback message to update the Q-value of B. Specifically, let h_N be the distance from N to the nearest hole, which is estimated so far (and stored in the local memory of N); then, h_N is replaced by $h_B + 1$ if $h_B + 1 < h_N$. Note that the distance to the nearest hole of all sensors is initially initiated by a very large positive number. After the hole information dissemination phase, the distance to the nearest hole of all hole boundary nodes is updated to 0.
- Otherwise, N uses the latest information stored in its local memory to update the Q-value of B.

Below is the formula for updating Q-value of B:

$$Q_N(B) \leftarrow (1-\alpha)Q_N(B) + \alpha\,[R_N(B) + \gamma Q_B]$$

Based on the analysis presented in Section II-C, we propose a novel reward function as follows:

$$R_N(B) = \begin{cases} r_{stuck}, & \text{if the packet is dropped at } B. \quad (1)\\ r_{dest}, & \text{if } B \text{ is the destination.} \quad (2)\\ -\alpha_1\left(1-\frac{E_r(B)}{E_i(B)}\right)-\alpha_2\frac{d(B)}{s}-\alpha_3\frac{1}{1+e^{h(B)}}, & \text{otherwise.} \quad (3)\end{cases}$$

r_{stuck} is the minimum reward, defined by a small negative constant. As the Q-learning algorithm always chooses the action with the highest Q-value, this reward prevents forwarding packets to the stuck nodes. r_{dest} is the maximum reward achieved when B is the destination. r_{dest} is defined by a large positive constant. (3) is contributed by B's residual energy, the estimated distance from B to the nearest hole and the distance from B to the destination. Specifically, $E_r(B)$ and $E_i(B)$ are the residual and initial energy of B, so the first term in (3) (i.e., $1 - \frac{E_r(B)}{E_i(B)}$) describes how much energy N has consumed so far. Therefore, by applying a negative factor $-\alpha_1$ to the first item, the neighbor with higher residual energy will have a higher reward and have more chance to forward the packet. $d(B)$ is defined by the distance from B to the cave-door if the forwarding mode is HOLE-ESCAPE and the distance to the final destination otherwise. Therefore, the second term leads packets to be relayed from the source to the destination by a path that is as short as possible. s is the total distance from all the neighbors of the current node to the destination. $h(B)$ estimates how far B stay from the holes. Therefore, by introducing $h(B)$ to the reward function, our routing protocol can alleviate the nodes surrounding holes and thus avoid the local minimum problem and the traffic concentration around the hole boundary.

α_1, α_2, and α_3 are the weight factors, which define the impact of the residual energy, the distance to the destination, and the distance to the hole to the reward. α_1, α_2, and α_3 are normalized such that $\alpha_1 + \alpha_2 + \alpha_3 = 1$. Intuitively, increasing α_1 will weight more on the energy balancing; hence, the energy consumption of the nodes tends to be more even. When α_2 increases, the routing path length factor will be more prioritized (e.g., the routing path length tends to be reduced). α_3 is the weight of the distance to the hole. When increasing α_3, routing paths tend to stay further from holes. This helps alleviate the local minimum phenomenon and the traffic concentration around the hole boundary. However, this may result in longer routing paths. According to the guidelines and the experiments given in Section VI, we set $\alpha_1 = 0.3$, $\alpha_2 = 0.45$, and $\alpha_3 = 0.25$.

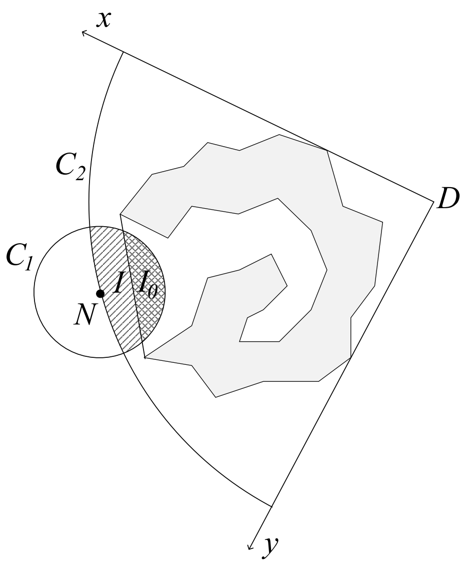


FIGURE 8. Illustration of the proof of Proposition 1.

V. THEORETICAL ANALYSIS

A. LOOP-FREE PROPERTY

In this theoretical analysis, we assume that *the network is sufficiently dense such that there are sensors everywhere apart from the considered holes*$^{(*)}$. Given such an ideal situation, we prove that every packet is forwarded to the destination by our proposed protocol. We also denote N as the current node and D as the destination.

Proposition 1: If N stays outside of concave regions and all of its neighbors do not satisfy at least one of the three conditions G-1, G-2, G-3 described in Section III-D2, then N must stay inside the forbidden area of a hole. Moreover, N has the information of the convex hull of that hole.

Proof 1: We illustrate the proof of Proposition 1 in Fig. 8. In the figure, C_1 is a R radius-circle, whose center is N. Moreover, the C_2 circle, which is centered at D, goes through N. Let I be the intersection of C_1 and C_2 that stays outside of the hole. It is obvious that all the nodes staying in I are closer to the destination than N. Note that if the entire I belongs to a hole's forbidden area, N also stays inside the area. Therefore, there must exist a portion of I, say I_0, that does not belong to the forbidden areas of all the bypassed holes. If I_0 does not intersect with the current hole's concave regions, then the nodes inside I_0 satisfy all three constraints **G-1**, **G-2**, and **G-3**. Therefore, I_0 must intersect with a concave region of the current hole. According to the hole information dissemination scheme, the nodes whose locations are inside the intersection of I_0 and the concave region must be involved in the dissemination process. Moreover, such nodes have broadcast hole information to N.

Proposition 2: If a packet has bypassed all the holes, it will be forwarded to the destination using the Q-GREEDY mode.

Proof 2: Let D be the destination and N be the current node. According to our routing mechanism, the next-hop forwarder never falls into the forbidden area of a bypassed hole. Therefore, after bypassing all holes, packets are forwarded to nodes staying outside of all holes' forbidden areas. Because the sensors are located everywhere apart from the considered holes, a neighbor of the current node satisfying conditions **G-1**, **G-2**, **G-3** is described in Section III-D2. Consequently, the packet is forwarded by the Q-GREEDY mode to a neighbor of N that is closer to the destination than the current node. The distance to the destination is decreased gradually. Thus, the packet eventually arrives at the destination.

Proposition 3: Suppose that N bypasses hole H; then, N will bypass H without a loop.

Proof 3: Without loss of generality, we assume that the packet's forwarding direction is *right*. The packet is then forwarded to nodes on the right side of the array connecting the current node and the destination. An illustration of Proposition 3 is shown in Fig. 9. Let S_0 be the first node in the HOLE-DETOUR mode and $S_1, \ldots, S_n$ be the next node. Let H_1 and H_2 be the two view-points D with respect to H, and suppose that H_1 stays on the left side of the array $\overrightarrow{DH_2}$. Let S_i be a node inside the forbidden area of hole H. We will prove that the next node S_{i+1} creates a greater angle with $\overrightarrow{DH_2}$ than S_i does. In this way, we show that the angles created by the forwarding nodes in the HOLE-DETOUR mode with $\overrightarrow{DH_2}$ gradually increase. Thus, the packet is eventually forwarded to a node outside of the forbidden are of the hole H. As S_i is located inside the forbidden area of H, $\overrightarrow{DS_i}$ must stay on the left side of $\overrightarrow{DH_2}$ (1). Moreover, as S_{i+1} stays on the right side of $\overrightarrow{S_iD}$, $\overrightarrow{DS_i}$ must be located on the right side of $\overrightarrow{DS_{i+1}}$ (2). From (1) and (2), we deduce that $\overrightarrow{DS_i}$ stays between $\overrightarrow{DH_2}$ and $\overrightarrow{DS_{i+1}}$. Therefore, $\angle H_2DS_{i+1} > \angle H_2DS_i$. Therefore, the packet will eventually be forwarded to the node outside of H_1DH_2.

Proposition 4: If a packet has bypassed hole H, it will never be forwarded to a node inside the forbidden area of H.

Proof 4: We consider the Q-GREEDY and HOLE-DETOUR modes of packet forwarding. If the packet is in the former mode, then it is not forwarded to a forbidden area of any hole (1). If it is in the latter, the packet is not forwarded to its bypassed hole's forbidden area (2). From (1) and (2), we deduce that the proposition holds if the packet is in Q-GREEDY mode or HOLE-DETOUR mode (3). If the packet is in the HOLE-ESCAPE mode, it is in the convex hull of the current hole, say hole H_1. It must then be initiated by a node staying inside the convex hull. The proposition holds. If not, the packet must enter the convex hull from a node,

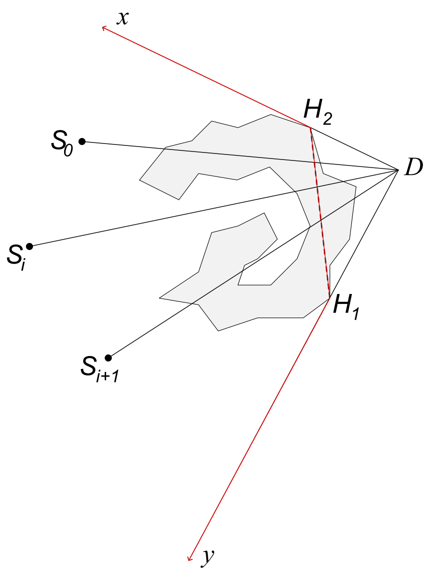


FIGURE 9. Illustration of the proof of Proposition 3.

say N_0, that belongs to the forbidden area of H_1 and stays outside the convex hull. As N_0 stays outside of the convex hull, it uses either the HOLE-DETOUR or Q-GREEDY mode to forward the packet. Therefore, the next hop cannot stay inside a convex hull of a hole.

Theorem 1: Every packet is forwarded to the destination.

Proof 5: According to Propositions 3 and 4, every packet in QIH will eventually bypass all the holes. Moreover, according to Proposition 2, once the packet has bypassed all the holes, it will be forwarded to the destination by the Q-GREEDY mode.

B. COMPUTATIONAL COMPLEXITY

In this section, we analyze the computational complexity of our proposed routing algorithm. We denote by M the maximum number of 1-hop neighbors of a sensor node.

1) Q-VALUE UPDATING

The Q-value is updated using the Hellman equation as follows:

$$Q_N(B) \leftarrow (1-\alpha)Q_N(B) + \alpha\left[R_N(B) + \gamma Q_B\right].$$

$Q_N(B)$ is retrieved from the Q-table; thus, the computational complexity for this action is $O(1)$. Q_B is the highest Q-value of B, which is included in the feedback message from node B. Therefore, the computational complexity of determining Q_B is $O(1)$. $R_N(B)$ is the reward calculated from three factors: the *energy consumption*, i.e., $\frac{E_r(B)}{E_i(B)}$, the *distance to destination*, i.e., $\frac{d(B)}{s}$, and the *distance to the hole*, $h(B)$. To determine the first factor, we need information about the initial energy, i.e., $E_i(B)$, and the remaining energy, i.e., $E_r(B)$ of B. As this information is obtained from the feedback message, this operation requires the complexity of $O(1)$. The distance to the destination is derived from two components: the Euclidean distance from node B to the destination, i.e., $d(B)$, and the total sum of all neighbors' distances to the destination, i.e., s. The computational complexity of the former and the latter are $O(1)$ and $O(m)$, respectively, where m is the number of the current node's 1-hop neighbors. Accordingly, the total computational complexity for determining the *distance to the destination* factor is $O(m)$. Finally, the *distance to the hole* factor is decided by using the estimated distance from B to the nearest hole, i.e., h_B. As h_B is contained in the feedback message, the computational complexity for determining the *distance to the hole* factor is $O(1)$.

In conclusion, the total computational complexity for updating a Q-value is given by $O(1) + O(m) + O(1) = O(m)$. This computational complexity is at most equal to $O(M)$, where M is the maximum number of neighbors of a sensor node.

2) HOLE BOUNDARY DETERMINATION

Let us consider a hole h that has b boundary nodes. We denote by M the maximum number of 1-hop neighbors of a sensor. According to our hole determination algorithm, every boundary node initiates an *HBD* packet and sends this packet around the hole boundary by using the right-hand rule. The computational complexity of the right-hand rule is as follows. Let t_i be the current boundary node that receives the *HBD* packet and t_{i-1} be the previous boundary node. Then, t_i searches among its neighbors a node t_{i+1} such that $\overrightarrow{t_it_{i+1}}$ makes the smallest angle with $\overrightarrow{t_it_{i-1}}$. The computational complexity for determining t_{i+1} equals the number of t_i's neighbors. As the number of t_i's neighbors does not exceed M, the computational complexity for the right-hand rule is upper bounded by $O(M)$. Moreover, as the *HBD* packet travels through at most m nodes, the computational complexity of the hole boundary determining algorithm does not exceed $O(m \times M)$.

VI. EVALUATION RESULTS

In this section, we conduct two main experiments. The first experiment evaluates the impacts of parameters $\alpha_1, \alpha_2, \alpha_3$, α, and γ on the performance of **Q**-learning **I**nspired **H**ole-bypassing (QIH). Based on the first experimental results, we determine the parameters' optimal values and perform a second experiment that compares QIH to three existing works (i.e., CBMH [21] and EDGR [25], and BSMH [23]). CBMH [21] and BSMH follows the hole-aware approach, while EDGR [25] uses a heuristic routing algorithm. Table 1 describes the parameters of QIH. We run experiments with network topologies that are generated as follows. First, from Google Earth data, we extract maps, including obstacles. Each map is then embedded into a $1000 \times 1000\ m^2$ network area. We randomly scatter approximately 4000 nodes within the area by dividing the network into a 63×63 square grid. Finally, we remove all sensor nodes staying inside the obstacles. The details of the topologies are represented in Table 2. Figure 10 shows Google Earth's real images, from which we create five network topologies, as shown in Fig. 11. In the topologies, we use the most common communication model of WSNs that contains one destination and multiple sources.

TABLE 1. Parameters of QIH.

Parameter	*Value*
Learning rate α	0.5
Discount factor γ	0.5
α_1	0.3
α_2	0.45
α_3	0.25
δ_e	0.7
r_{stuck}	-20
r_{dest}	50

TABLE 2. The details of simulated network topologies.

Topology	1	2	3	4	5
Number of nodes	3906	3674	3724	3809	3768
Number of sources	150	150	150	151	150
Number of destinations	1	1	1	1	1

TABLE 3. Parameters of a sensor node.

Factor	*Value*
MAC type	CSMA/CA
Interface queue model	DropTail
Transmission of radio	TwoRayGround
Antenna type	OmniAntenna
Node initial energy	30 J
Queue length	50 packets
Transmission range	40 m
Node idle power	9.6 mW
Node receive power	45 mW
Node transmit power	88.5 mW
Packet sending interval	10 s
Data packet size	50 bytes

The destination is placed near the boundary of the network. The sources are randomly chosen such that 1) the line connecting each source with the destination intersects the hole and 2) the sources contain both nodes inside and outside of the hole's convex hull. We set the number of sources in each topology at 150. The simulation time is 1500 seconds. In the following, the plotted values are the average of 10 simulation runs along with a 95% confidence interval. The experiments are conducted using the NS-2 simulator and on a computer with an Intel Core i5-4570 3.2 GHz x 4 CPU and 8 GB of RAM running Ubuntu 14.04 64-bit. To study the protocols' energy consumption, we used the energy model suggested by Shnayder *et al.* [32]. The sensor nodes' parameters are listed in Table 3.

We investigate the following metrics.

- *Network lifetime*: the network lifetime is defined as the time duration until the first node dies.
- *Average end-to-end delay*: the average time to route from sources to the destination of all data packets that successfully arrive at the destination.
- *Packet delivery ratio*: the delivery ratio is the ratio of the number of data packets successfully arriving at the destination to the total number of data packets sent by the sources.
- *Average energy consumption per packet*: the average consumed energy is the ratio of all nodes' total energy consumption to the total number of packets successfully delivered.

(a) Topology 1 (b) Topology 2 (c) Topology 3 (d) Topology 4 (e) Topology 5

FIGURE 10. Real maps obtained from the Google Earth.

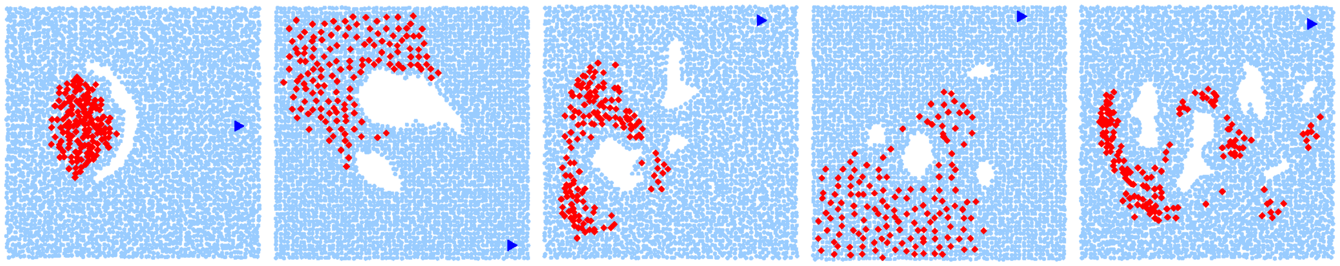
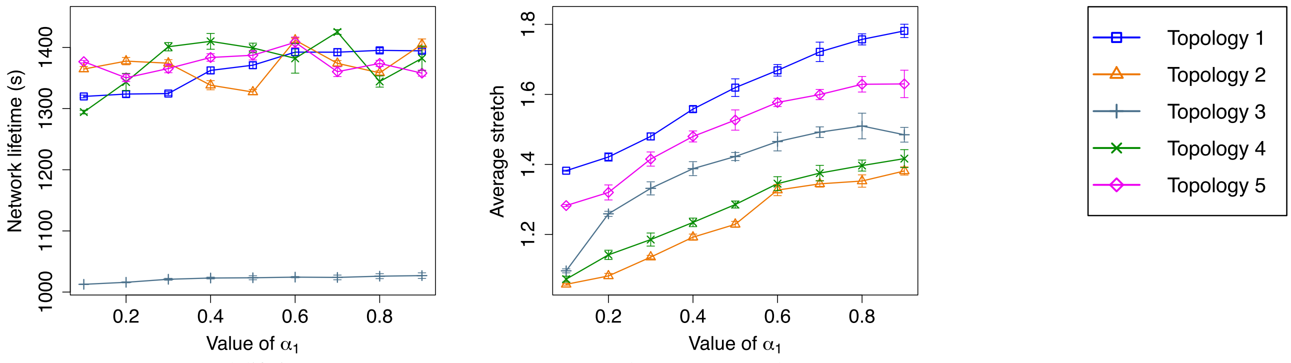

(a) Topology 1 (b) Topology 2 (c) Topology 3 (d) Topology 4 (e) Topology 5

FIGURE 11. Network topologies (the blue circles represent the sensors. The red diamonds represent the sources, and the blue triangle represents the destination).



(a) Network lifetime (b) Average stretch

FIGURE 12. Impacts of α_1 on network performance.

A. IMPACTS OF WEIGHTS ON REWARD FUNCTION

This section studies the impacts of α_1, α_2, and α_3 on the reward functions in three scenarios. In the first one, we vary the value of α_1 from 0.1 to 0.9. With each value of α_1, we set $\alpha_2 = \alpha_3 = \frac{1-\alpha_1}{2}$. Similarly, in the two others, we vary the values of α_2 and α_3 from 0.1 to 0.9. We perform experiments on the five topologies shown in Fig. 11 to evaluate the impacts. In the evaluation, we consider two metrics: the network lifetime and the average stretch. We omit the observation concerning the packet delivery ratio, as it has been theoretically proven to be 100% in Section V. We also eliminate the end-to-end delay and the average energy consumption per packet since they can be derived from the average stretch.

1) IMPACTS OF α_1

Let us reiterate that α_1 defines how significantly the sensors' residual energy impacts the routing decision. In other words, the greater the value of α_1 is, the more priority the residual energy gains. Therefore, when α_1 increases, the neighbors with higher levels of residual energy have a higher probability of being selected as the next forwarder. This will make the energy consumption of all the nodes more even. As a result, the network lifetime tends to be extended when increasing the value of α_1. As in Fig. 12(a), the network lifetime has increased in all the investigated topologies. However, the α_1 increment frequently changes on the routing paths, which are often not the shortest paths. Therefore, the average stretch is proportional to the increase in α_1. This observation can be clearly seen in Fig. 12(b). From the evaluation results, we can see that α_1 should be set to moderate values from 0.2 to 0.4.

2) IMPACTS OF α_2

The α_2 value reflects the path length's weight in making the routing decision of QIH. When we increase the value of α_2, the shorter paths to the destination are prioritized. The average stretch values in relation to α_2 are shown in Fig. 13(b).

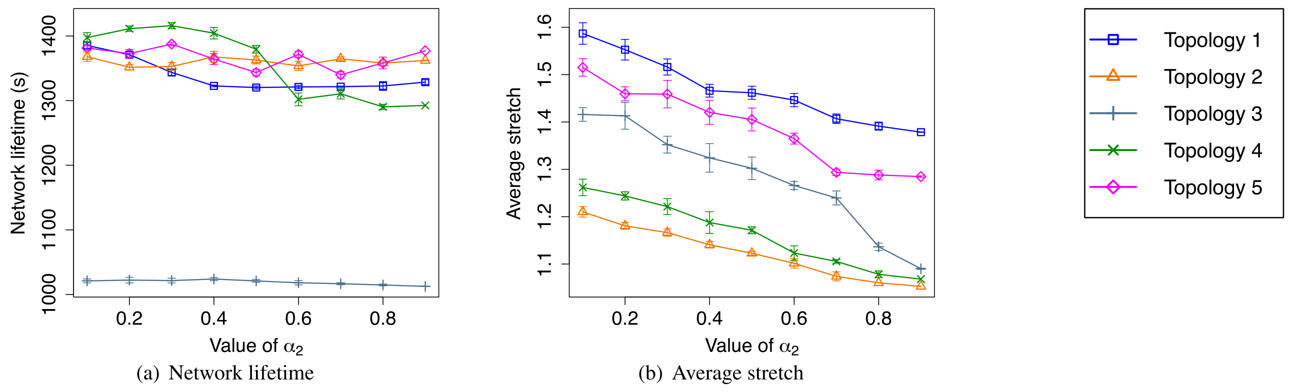


FIGURE 13. **Impacts of α_2 on network performance.**

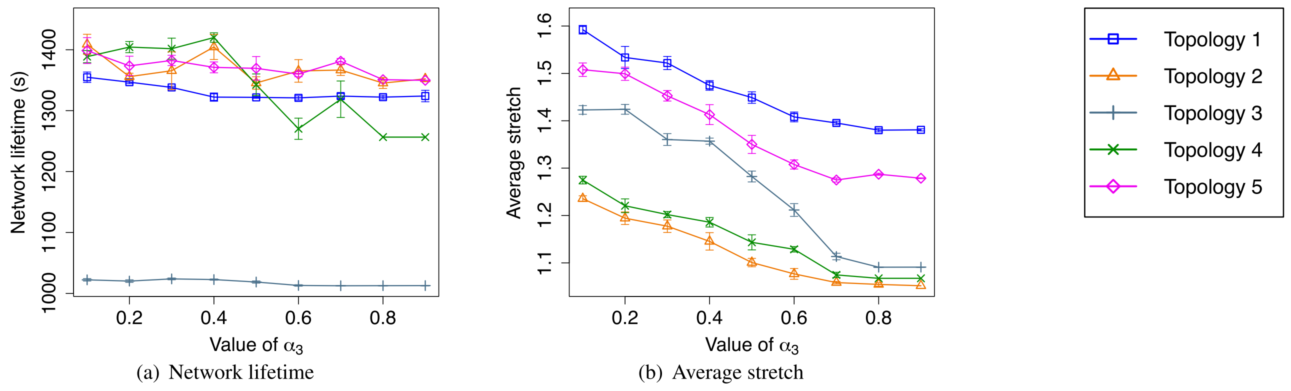


FIGURE 14. **Impacts of α_3 on network performance.**

We can see that the average stretch gradually decreases when the value of α_2 increases. Fig. 13(a) presents the impacts of α_2 on the network lifetime. With a higher value of α_2, the network lifetime tends to decrease. This is because QIH prefers to choose shorter paths when making routing decisions. As a result, the traffic load tends to concentrate on the nodes along several specific paths. When we consider the trade-off between the network lifetime and the average stretch, the suitable value of α_2 should be in a range from 0.2 to 0.5.

3) IMPACTS OF α_3

In the case of α_3, its value is associated with the priority of the distance to the hole. The impacts of α_3 on the network lifetime and average stretch are shown in Fig. 14. In Fig. 14(b), the average stretch decreases when α_3 increases. This is because when the value of α_3 becomes larger, the packets bypass the hole earlier. In other words, when the α_3 value is high, the packets will be forwarded along the routing paths that stay far from the hole. These routing paths are also far from the shortest paths because the shortest path goes through the hole's convex hull. Fig. 14(a) illustrates the variation of the network lifetime when we change the value of α_3. The network lifetime tends to decrease when α_3 is large. The reason is that the large value of α_3 causes longer routing paths and thus imposes more traffic on sensors. From the experimental results, α_3 should be set to a moderate value from 0.2 to 0.4.

In summary, we deduce that

- The network lifetime increases if we increase α_1 or decrease α_2, α_3,
- The average stretch decreases if we decrease α_1 or increase α_2, α_3.

From the results described above, we choose the following values of α_1, α_2, and α_3 to perform experiments to compare the performance of the proposed protocol with the other benchmarks, $\alpha_1 = 0.3$, $\alpha_2 = 0.45$, and $\alpha_3 = 0.25$.

B. COMPARISON WITH OTHER APPROACHES

1) NETWORK LIFETIME

Figure 15 shows the network lifetime achieved by the protocols. As shown, QIH significantly improves the network lifetime compared with CBMH, EDGR, and BSMH. Specifically, QIH extends the network lifetime more than 16%, 12%, 15% in comparison with EDGR, CBMH, and BSMH, respectively. In most of the cases, CBMH has the second-best performance, and EDGR achieves the worst performance. The reasons for QIH's improvement over the hole-aware protocols (i.e., CBMH and BSMH) are as follows. First,

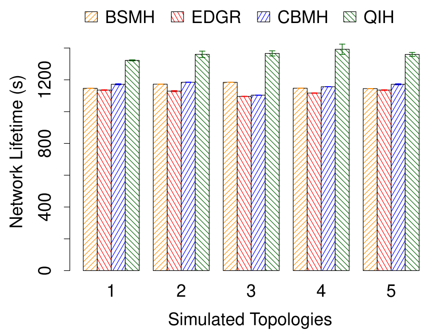


FIGURE 15. Network Lifetime.

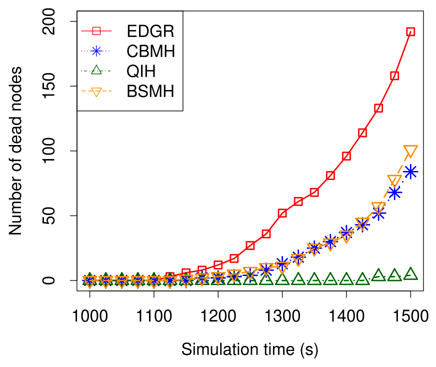


FIGURE 16. Number of dead nodes.

QIH disseminates hole information for only nodes inside the concave regions. Therefore, QIH can reduce the overhead caused by the dissemination phase. Second, thanks to the proposed reward function, QIH can balance the energy consumption between nodes and alleviate long routing paths. On the other hand, although both EDGR and QIH consider residual energy in making a routing decision, QIH is better. This is because QIH optimizes the global energy cost over the whole routing path, while EDGR only optimizes the local energy consumption at each node on the routing path.

To thoroughly evaluate how well our protocol can balance the energy consumption of nodes, we plot the number of dead nodes over simulation time in Fig. 16. Figure 16 depicts the number of dead nodes from 1000 s to the end of the simulation concerning Topology 4. It can be seen that the number of dead nodes caused by EDGR, CBMH, and BSMH increases rapidly upon 1200 s. In contrast, QIH results in a very small number of dead nodes. Specifically, until the simulation ends (i.e., 1500 s), only 4 nodes die when using our routing protocols. The total number of dead nodes when using EDGR, CBMH, and BSMH are 192, 84 and 98, respectively. This result again proves that our protocol can balance traffic much better than the other protocols and thus can prolong the network lifetime.

2) ROUTING PATH STRETCH

The average stretch of routing paths is depicted in Fig. 17. BSMH attains the best performance in all cases, followed by CBMH and EDGR. CBMH attains the second best performance in topology 1, 3 and 5, QIH attains the second best performance in topology 2 and 4, and EDGR does not attain the best performance in any case. As shown in Figure 11, the source nodes in topology 1, 3, and 5 are concentrated in small regions with high density, and the routing path stretch of QIH is also longer than that of other topology. This is for the purpose of load balancing, i.e., packets route farther paths to reduce the network load at the source node concentration regions. In addition, all protocols tend to have higher routing path stretching when the network topology is more complex, i.e., in topology 1 and 5, where the routing hole has a large concave region, or the network has multiple holes with various sizes.

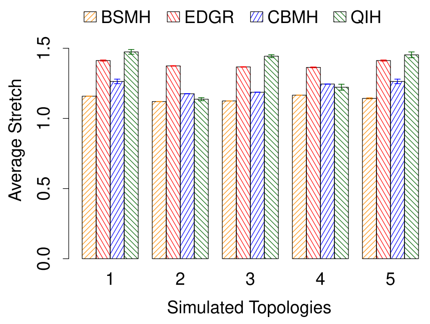


FIGURE 17. Average stretch.

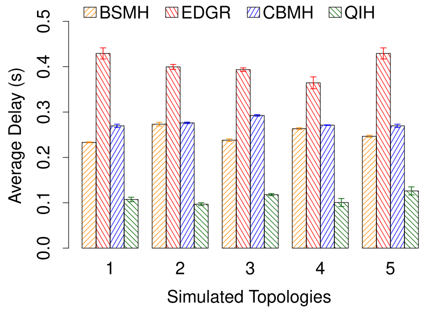


FIGURE 18. Average end-to-end delay.

3) AVERAGE END-TO-END DELAY

Figure 18 depicts the average end-to-end delay of all successfully delivered packets. As shown, the average end-to-end delay of QIH is much smaller than that of EDGR, CBMH and BSMH. The average end-to-end delay achieved by QIH is always smaller than 47% of that achieved by CBMH, 30% of that achieved by EDGR, and 51% of that achieved by BSMH. Interestingly, although BSMH outperforms QIH in terms of routing path stretch, its average delay is much more significant than that of QIH. The reasons for the improvement of QIH can be explained as follows. The packet delay is composed of two components. The first component is the computational time spent to determine the routing path, and the second component is the transmission time for transmitting the packet from the source to the destination. In EDGR, the routing decision is made mainly based on the energy

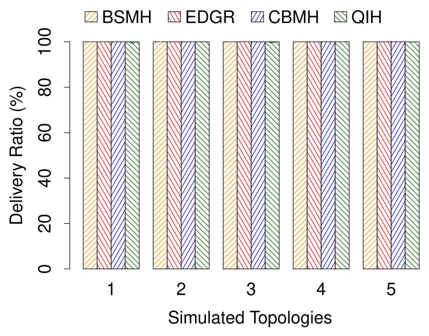


FIGURE 19. Delivery ratio.

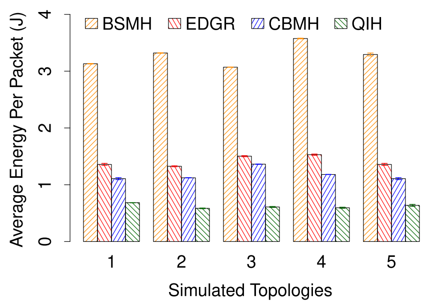


FIGURE 20. Average energy consumption per packet.

information of the nodes. This means that EDGR does not consider the routing path length when determining the routing path. Consequently, the routing path attained by EDGR may be very large and lead to a large delay. In contrast, although CBMH and BSMH can guarantee all routing paths' constant stretch, they require a complicated process for a source node to determine the routing path. Hence, CBMH and BSMH also suffers from a high packet delay. Thanks to the simplicity of the Q-learning technique, the computational time is reduced significantly in our protocol. Moreover, since our reward function is designed to prioritize the short routing path, QIH can improve the transmission time.

4) PACKET DELIVERY RATIO

The values of the packet delivery ratio achieved by the protocols are shown in Fig. 19. It can be seen that all of the four protocols achieve a delivery ratio of approximately equal to 100%. In EDGR, CBMH, and BSMH, the source node predetermines a Euclidean path, avoiding all the holes. Thus, they can alleviate the local minimum problem and successfully deliver most of the packets. In QIH, the local minimum phenomenon is solved by introducing the HOLE-ESCPAE and HOLE-DETOUR forwarding modes. The first mode helps a packet escape from the holes' concave regions, and the second mode prevents packets from approaching the hole boundaries.

5) AVERAGE ENERGY CONSUMPTION PER PACKET

Figure 20 shows the comparison of average energy consumed to deliver one packet. In all five topologies, QIH outperforms EDGR, CBMH, and BSMH by consuming significantly less power. More specifically, the average energy per packet consumed by QIH is always smaller than 62%, 51% and 21% of that consumed by CBMH, EDGR, and BSMH, respectively. As the packet delivery ratios are almost the same for all three protocols (as shown in Section VI-B4), the gap between the energy consumption of the protocols is decided by the total energy consumption. QIH outperforms the others because QIH has a small packet header size and a very lightweight routing path determination algorithm.

In summary, QIH shares the delivery ratio with the existing protocols, which approximately equals 100%. However, our protocol outperforms the existing protocols in terms of the network lifetime, the average end-to-end delay, and energy consumption.

VII. RELATED WORK

Initially, we discuss geographic routing protocols using the hole-aware approach. In [12], [14]–[18], [20]–[22], [33], the authors used the concept of the forbidden area around every hole from which all the packets are kept to stay away. Fucai Yu *et al.,* proposed a routing scheme wherein the forbidden area is a circle covering the hole [14]. The nodes on the hole boundary are identified using the bound hole algorithm [8], which creates a virtual circle exactly covering the hole. The circle's information is then disseminated to all nodes on the hole's boundary. When a source node S wants to communicate to a destination node D, it first sends the packet along the line SD. The node on the hole's boundary will inform S about the virtual circle upon receiving the data packet. Then, S calculates an anchor location, which is the intersection of two tangent lines from S and D to the virtual circle. S forwards the data packet to the node closest to the anchor location, and this node will forward the data packet to D. Similarly, the authors in [17] and [16] proposed using the forbidden area as a hole covering hexagons and ellipses, respectively. Currently, Muhammad A. *et al.* [34] focused on the hole avoidance problem in underwater WSNs. The authors proposed two algorithms to minimize the void hole occurrence ratio, thereby improving the packet delivery ratio. The main idea was to deploy fixed backup node deployment at different strategic locations. In [35], the authors studied how to bypass dynamic holes. They proposed a protocol that is a combination of a geographic routing protocol proposed in [36] and a dynamic hole detection algorithm proposed in [13].

The protocol in [37] can guarantee that the route stretch has an upper bound by $\Theta(c)$, where c is the shortest path length. In this protocol, the forbidden area is the convex hull of the hole, through which vertices the packets are routed from the source to the destination. The authors then extended the work to address the problem of routing between nodes inside the concave regions of the holes [18]. In [18], they described the hole by a polygon whose vertices are all the nodes staying on the hole boundary. The routing path is then determined by the graph, which is formed from the holes' vertices. The upper bound stretch was proven to be $\Theta\left(\frac{D}{\gamma}\right)$, where D is the diameter of the network and γ is the communication range of sensor nodes. Although these two protocols can provide the stretch upper bound, they still suffer from the same problem, i.e., traffic concentration around the forbidden area, as the other protocols described above. In [30], the proposed protocol used a dynamic forbidden area with a circle shape. The protocol can alleviate the traffic congestion around the forbidden area, but it still incurs the routing path enlargement problem. In our previous work [20], [21], [23], our routing protocol could alleviate both the traffic concentration and routing path enlargement problem. However, it suffers from a large control overhead in broadcasting information and high computational complexity in determining the routing path. Thus, it may result in high packet delay and high energy consumption.

Regarding the heuristic approach, Huang *et al.* exploited energy information in making routing decisions in EDGR [25]. Before sending data packets, each node sends two so-called burst packets towards the destination for hole-bypassing purposes. The first and second packets sequentially go along the right-hand and left-hand sides of the holes. These packets collect information on two anchor lists along the routing path. Upon arriving at the destination, the burst packets are pushed back to the source with the anchor lists embedded. When a source node sends a packet, it randomly chooses an anchor list and embeds the anchors' locations into the packet header. The packet is then gradually forwarded towards the anchors, where the next hop is chosen based on the neighbors and residual energy. EDGR incurs a high energy overhead, as it requires nodes to broadcast beacons to update the energy information periodically. Yu *et al.* [38] proposed a scheme to avoid the local minimum problem by identifying the nodes staying inside the concave areas and preventing them from participating in data delivery. Although these schemes can shorten the routing path, they continue to suffer from traffic congestion surrounding the hole. In [24], the next forwarder node was chosen based on a so-called forwarding factor. This factor is proportional to the residual energy and inversely proportional to the distance to the neighbors' destinations. Accordingly, neighbors with higher residual energy and the shortest distance to the destination are more likely to be chosen. The authors in [27], [28] proposed routing protocols based on the Q-learning technique. In [27], the reward was decided by the so-called packet travel speed, which is the ratio of advancement to the destination to the packet travel time. In [28], the reward combined the nodes' residual energy and the distance to the destination. Unfortunately, none of the heuristic routing algorithms proposed so far target the hole problem. We make an effort to solve this problem thoroughly. An earlier version of this work has appeared in [39]. In this version, we have newly added the theoretical analysis section and new evaluation results.

VIII. CONCLUSION

In this paper, we studied how to prolong the network lifetime under hole occurrence. We exploited the Q-learning technique to propose QIH, which is a lightweight routing protocol for WSNs. Our proposed protocol simultaneously solved three critical problems that shorten the network lifetime: 1) the local minimum phenomenon; 2) routing path enlargement; and 3) load imbalance. We performed theoretical analysis and proved that every packet is forwarded to the destination by QIH. We conducted extensive experiments to study the impacts of the parameters on QIH performance. Moreover, we provided a guideline for choosing the optimal parameters. We also performed extensive experiments to compare QIH with the state-of-the-art methods. The evaluation results showed that QIH outperforms the existing methods in terms of the network lifetime, packet latency, and energy consumption.

REFERENCES

[1] G. Han, X. Yang, L. Liu, W. Zhang, and M. Guizani, "A disaster management-oriented path planning for mobile anchor node-based localization in wireless sensor networks," *IEEE Trans. Emerg. Topics Comput.*, vol. 8, no. 1, pp. 115–125, Jan. 2020.

[2] T. Ojha, S. Misra, and N. S. Raghuwanshi, "Wireless sensor networks for agriculture: The state-of-the-art in practice and future challenges," *Comput. Electron. Agricult.*, vol. 118, pp. 66–84, Oct. 2015.

[3] N. T. Hanh, P. T. H. Hanh, H. T. T. Binh, and N. D. Nghia, "Heuristic algorithm for target coverage with connectivity fault-tolerance problem in wireless sensor networks," in *Proc. Conf. Technol. Appl. Artif. Intell. (TAAI)*, Nov. 2016, pp. 235–240.

[4] N. T. Hanh, N. T. Hai, L. Q. Tung, H. T. T. Binh, and E. Kurniawan, "Connectivity optimization problem in vehicular mobile wireless sensor networks," in *Proc. Int. Conf. Comput. Intell. Cybern.*, Nov. 2016, pp. 55–61.

[5] V. Raghunathan, C. Schurgers, S. Park, and M. B. Srivastava, "Energy-aware wireless microsensor networks," *IEEE Signal Process. Mag.*, vol. 19, no. 2, pp. 40–50, Mar. 2002.

[6] G. J. Pottie and W. J. Kaiser, "Wireless integrated network sensors," *Commun. ACM*, vol. 43, no. 5, pp. 51–58, May 2000.

[7] N. Ahmed, S. S. Kanhere, and J. Sanjay, "The holes problem in wireless sensor networks: A survey," *ACM SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 9, no. 2, pp. 4–18, Apr. 2005.

[8] Q. Fang, J. Gao, and L. J. Guibas, "Locating and bypassing routing holes in sensor networks," in *Proc. IEEE INFOCOM*, Mar. 2004, pp. 2458–2468.

[9] F. Hajjej, M. Hamdi, R. Ejbali, and M. Zaied, "A distributed coverage hole recovery approach based on reinforcement learning for wireless sensor networks," *Ad Hoc Netw.*, vol. 101, Apr. 2020, Art. no. 102082.

[10] C. Zygowski and A. Jaekel, "Optimal path planning strategies for monitoring coverage holes in wireless sensor networks," *Ad Hoc Netw.*, vol. 96, Jan. 2020, Art. no. 101990.

[11] P. Singh and Y.-C. Chen, "Sensing coverage hole identification and coverage hole healing methods for wireless sensor networks," *Wireless Netw.*, vol. 26, no. 3, pp. 2223–2239, Apr. 2020.

[12] P.-L. Nguyen, D.-T. Nguyen, and K.-V. Nguyen, "Load balanced routing with constant stretch for wireless sensor network with holes," in *Proc. IEEE ISSNIP*, Apr. 2014, pp. 1–7.

[13] N. D. Trong, N. P. Le, P. Van Hau, and N. Van Khanh, "A distributed protocol for detecting and updating hole boundary in wireless sensor networks," in *Proc. ACM SoICT*, Dec. 2015, pp. 171–178.

[14] F. Yu, S. Park, Y. Tian, M. Jin, and S.-H. Kim, "Efficient hole detour scheme for geographic routing in wireless sensor networks," in *Proc. IEEE VTC*, May 2008, pp. 153–157.

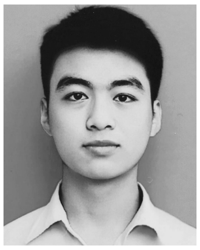
[15] S. Kim, C. Kim, H. Cho, Y. Yim, and S.-H. Kim, "Void avoidance scheme for real-time data dissemination in irregular wireless sensor networks," in *Proc. IEEE AINA*, Mar. 2016, pp. 438–443.

[16] Y. Tian, F. Yu, Y. Choi, S. Park, E. Lee, M. Jin, and S.-H. Kim, "Energy-efficient data dissemination protocol for detouring routing holes in wireless sensor networks," in *Proc. IEEE ICC*, 2008, pp. 2322–2326.

[17] H. Choo, M. Choi, M. Shon, and D. S. Kim, "Efficient hole bypass routing scheme using observer packets for geographic routing in wireless sensor networks," *ACM SIGAPP Appl. Comput. Rev.*, vol. 11, no. 4, pp. 7–16, Dec. 2011.

[18] M. Won and R. Stoleru, "A low-stretch-guaranteed and lightweight geographic routing protocol for large-scale wireless sensor networks," *ACM Trans. Sensor Netw.*, vol. 11, no. 1, pp. 18:1–18:22, Nov. 2014.
[19] C.-Y. Chang, C.-T. Chang, Y.-C. Chen, and S.-C. Lee, "Active route-guiding protocols for resisting obstacles in wireless sensor networks," *IEEE Trans. Veh. Technol.*, vol. 59, no. 9, pp. 4425–4442, Nov. 2010.
[20] P. L. Nguyen, Y. Ji, Z. Liu, H. Vu, and K. V. Nguyen, "Distributed hole-bypassing protocol in WSNs with constant stretch and load balancing," *Comput. Netw.*, vol. 129, pp. 232–250, Dec. 2017.
[21] P.-L. Nguyen, Y. Ji, N. T. Trung, and N. T. Hung, "Constant stretch and load balanced routing protocol for bypassing multiple holes in wireless sensor networks," in *Proc. IEEE NCA*, Oct. 2017, pp. 1–9.
[22] P. Le Nguyen, Y. Ji, K. Le, and T.-H. Nguyen, "Load balanced and constant stretch routing in the vicinity of holes in WSNs," in *Proc. IEEE CCNC*, Jan. 2018, pp. 1–6.
[23] P. L. Nguyen, T. H. Nguyen, and K. Nguyen, "A path-length efficient, low-overhead, load-balanced routing protocol for maximum network lifetime in wireless sensor networks with holes," *Sensors*, vol. 20, no. 9, p. 2506, Apr. 2020.
[24] N. Jan, A. R. Hameed, B. Ali, R. Ullah, K. Ullah, and N. Javaid, "A balanced energy consuming and hole alleviating algorithm for wireless sensor networks," in *Proc. IEEE AINA Workshops (WAINA)*, Mar. 2017, pp. 231–237.
[25] H. Huang, H. Yin, G. Min, J. Zhang, Y. Wu, and X. Zhang, "Energy-aware dual-path geographic routing to bypass routing holes in wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 17, no. 6, pp. 1339–1352, Jun. 2018.
[26] M. M. Lima, H. A. B. F. Oliveira, D. L. Guidoni, and A. A. F. Loureiro, "Geographic routing and hole bypass using long range sinks for wireless sensor networks," *Ad Hoc Netw.*, vol. 67, pp. 1–10, Dec. 2017.
[27] W.-S. Jung, J. Yim, and Y.-B. Ko, "QGeo: *Q*-learning-based geographic ad hoc routing protocol for unmanned robotic networks," *IEEE Commun. Lett.*, vol. 21, no. 10, pp. 2258–2261, Oct. 2017.
[28] T. Hu and Y. Fei, "QELAR: A machine-learning-based adaptive routing protocol for energy-efficient and lifetime-extended underwater sensor networks," *IEEE Trans. Mobile Comput.*, vol. 9, no. 6, pp. 796–809, Jun. 2010.
[29] N. Bulusu, J. Heidemann, and D. Estrin, "GPS-less low-cost outdoor localization for very small devices," *IEEE Pers. Commun.*, vol. 7, no. 5, pp. 28–34, Oct. 2000.
[30] F. Li, B. Zhang, and J. Zheng, "Geographic hole-bypassing forwarding protocol for wireless sensor networks," *IET Commun.*, vol. 5, no. 6, pp. 737–744, Apr. 2011.
[31] M. A. Razzaque and S. Dobson, "Energy-efficient sensing in wireless sensor networks using compressed sensing," *Sensors*, vol. 14, no. 2, pp. 2822–2859, 2014.
[32] V. Shnayder, M. Hempstead, B.-R. Chen, G. W. Allen, and M. Welsh, "Simulating the power consumption of large-scale sensor network applications," in *Proc. ACM SenSys*, 2004, pp. 188–200.
[33] P. L. Nguyen, Y. Ji, N. T. Trung, and N. T. Hung, "A delay-guaranteed geographic routing protocol with hole avoidance in WSNs," in *Proc. IEEE MASS*, Oct. 2017, pp. 135–143.
[34] M. Awais, I. Ali, T. A. Alghamdi, M. Ramzan, M. Tahir, M. Akbar, and N. Javaid, "Towards void hole alleviation: Enhanced geographic and opportunistic routing protocols in harsh underwater WSNs," *IEEE Access*, vol. 8, pp. 96592–96605, 2020.
[35] P. Hadikhani, M. Eslaminejad, M. Yari, and E. A. Mahani, "An energy-aware and load balanced distributed geographic routing algorithm for wireless sensor networks with dynamic hole," *Wireless Netw.*, vol. 26, no. 1, pp. 507–519, Jan. 2020.
[36] N. P. Le, N. T. Hieu, and N. K. Van, "ELBAR: Efficient load balanced routing scheme for wireless sensor networks with holes," in *Proc. 3rd Symp. Inf. Commun. Technol. (SoICT)*. New York, NY, USA: Association for Computing Machinery, 2012, pp. 190–199.
[37] M. Won, W. Zhang, and R. Stoleru, "GOAL: A parsimonious geographic routing protocol for large scale sensor networks," *Ad Hoc Netw.*, vol. 11, no. 1, pp. 453–472, Jan. 2013.
[38] F. Yu, S. Pan, and G. Hu, "Hole plastic scheme for geographic routing in wireless sensor networks," in *Proc. IEEE ICC*, Jun. 2015, pp. 6444–6449.
[39] K. Le, T. H. Nguyen, K. Nguyen, and P. L. Nguyen, "Exploiting *Q*-learning in extending the network lifetime of wireless sensor networks with holes," in *Proc. IEEE ICPADS*, Dec. 2019, pp. 602–609.

PHI LE NGUYEN received the B.E. and M.S. degrees from The University of Tokyo, in 2007 and 2010, respectively, and the Ph.D. degree in informatics from The Graduate University for Advanced Studies, SOKENDAI, Tokyo, Japan, in 2019. She is currently an Assistant Professor with the School of Information and Communication, Hanoi University of Science and Technology, Vietnam. Her research interests include network architecture, applied AI in networking, network simulation, and optimization.

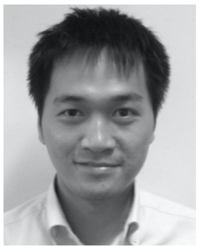


NANG HUNG NGUYEN is currently pursuing the bachelor's degree with the School of Information and Communication Technology, Hanoi University of Science and Technology (HUST). He is also a Research Assistant at the AIoT Laboratory, HUST. His research interests include reinforcement learning, optimization, and network architecture.

TUAN ANH NGUYEN DINH is currently pursuing the bachelor's degree with the School of Information and Communication Technology, Hanoi University of Science and Technology (HUST). He is a Research Assistant at the AIoT Laboratory, HUST. His research interests include mathematical modeling, optimization, and network architecture.

KHANH LE received the B.E. degree from Hanoi University of Science and Technology, in 2019. She is currently working as a Research Assistant with Hanoi University of Science and Technology. Her research interests include applied AI, wireless sensor networks, and optimization algorithms.

THANH HUNG NGUYEN received the Ph.D. degree in computer science from the University of Grenoble Alpes. He is currently an Assistant Professor with the School of Information and Communication, Hanoi University of Science and Technology, Vietnam. His research interests include modeling and verification of component-based systems.



KIEN NGUYEN (Senior Member, IEEE) received the B.E. degree in electronics and telecommunication from Hanoi University of Science and Technology (HUST), Vietnam, in 2004, and the Ph.D. degree in informatics from The Graduate University for Advanced Studies, SOKENDAI, Tokyo, Japan. He is currently working as an Assistant Professor with the Graduate School of Engineering, Chiba University. He has published more than 120 publications in peer-reviewed journals and conferences, three patents, and several internet drafts. His research interests include internet, the Internet of Things technologies, and wired and wireless communication. He is a member of IEICE. He also involves in IETF activities.