A Novel Shortcut Addition Algorithm with Particle Swarm for Multi-sink Internet of Things

***Abstract*—The Internet of Things (IoT) integrates a large number of distributed nodes to collect or transmit data. When the network scale increases, individuals use multiple sink nodes to construct the network. This increases the complexity of the network and leads to significant challenges in terms of the existing methods with respect to the aspect of data forwarding and collection. In order to address the issue, this paper proposes a Shortcut Addition strategy based on the Particle Swarm algorithm for multi-sink network (SAPS). It constructs network topology with multiple sinks based on a small-world network. In SAPS, we create a fitness function by combining the average path length and load of the sink node to evaluate the quality of a particle. Subsequently, crossover and mutation are used to update particles to determine the optimal solution. The simulation results indicate that SAPS is superior to the GMSW and LM-GAS in terms of the average path length, load balance, and number of added shortcuts.**

***Keywords*—Internet of Things, small-world network, particle swarm, multi-sink network**

I. Introduction

INTERNET of Things (IoT) [1]–[4] is used in military affairs, environmental monitoring, agriculture production and many other fields [5], [6]. It contains a large number of distributed nodes (sensor nodes and sink nodes) with sensing, and computing and communication capabilities [7], [8]. In IoT, sensor nodes transmit the perceived temperature, humidity, and other environmental information to sink nodes through wireless communication [9]. Given the limited communication distance of sensor nodes, it is typically necessary to pass multiple hops to transfer data from the sensor nodes to sink nodes. Additionally, the energy of the sensor node is limited, and multi-hop transmission consumes a large amount of energy in the node. This easily leads to the failure of the node. Therefore, it is necessary to construct a network topology to reduce the number of hops in the multi-hop transmission path and the energy consumption of the node.

The small-world network exhibits the characteristics of shorter path length [10], [11]. Therefore, we introduce the small-world phenomenon in the complex network theory to construct network topology. The small-world phenomenon was first studied by Milgram in a scientific way via the mail delivery experiment, and he suggested that any individual can establish contact with only five intermediate individuals on an average [12]. Subsequently, Watts and Strogatz [13] proposed to start randomly reconnecting edges from the rule graph, and this led to the features of short average path length and high clustering coefficient. This presented a small-world phenomenon. Newman and Watts [14] used randomization to add edges instead of previous random reconnections to make the network exhibit small-world features and avoid the destruction of the original structure of a network. In this paper, we construct a small-world model based on the theory of Newman and Watts by adding a small amount of remote links (shortcuts) to the original topology. In order to enable nodes to add remote links, it is necessary to equip more powerful hardware devices on the nodes for remote communication. We term the node with stronger communication capacity as a super node [15]. Given the high cost of super-nodes, only a small number of super-nodes are deployed in the network. Most of the nodes correspond to ordinary nodes equipped with common hardware devices. This paper is based on the super nodes to establish shortcuts.

When only one sink node exists in the network, the nodes around the sink consume their energy quickly [16]–[18]. Given special data transmission characteristics in IoT (i.e., the sink node denotes the end and start point of data transmission), more traffic is routed through the nodes around the sink [19], [20]. Additionally, when the sensor node is far away from the sink node, it is necessary to forward the message via several hops to reach the sink node, and the data transmission time is extended. The aforementioned problems are more serious when the network size increases. In order to overcome the problems, we use multiple sink nodes to construct the network to extend the lifetime and reduce the number of hops for message forwarding [9]. In the multi-sink network, each sensor node selects a sink node as the end of the data transmission. The sensor nodes that select the same sink node form a cluster. Each sink is responsible for receiving data gathered from sensor nodes in its cluster. Additionally, the probability of having a sink in the proximity of a sensor node increases when the number of sink nodes increases. Therefore, the number of hops that reach the sink node decreases.

However, in a network with multiple sinks, the load among the sink nodes is unbalanced when a large difference exists in the number of nodes among clusters. With respect to solving the problem, in the existing algorithm [21], constraints are set to balance the load of the sink while adding shortcuts although a certain degree of randomness exists when selecting the node pair as the shortcut endpoint. Therefore, we attempt to use the particle swarm algorithm to determine the appropriate node pairs as the endpoints of the shortcuts.

In this paper, we examine the modeling problem of a small-world network with multiple sinks and propose a novel shortcut addition algorithm with particle swarm for multi-sink Internet of Things (SAPS). The main contributions of this paper are as follows:

- We propose a shortcut adding algorithm for multi-sink Internet of Things based on the particle swarm algorithm (SAPS). In SAPS, the fitness function is constructed, and thus the network average shortest path length is minimized and the load of the sink node is balanced.

- We introduce novel crossover operation and mutation operation to replace the particle update method in the traditional particle swarm algorithm to achieve individual evolution
- We compare SAPS with existing methods [21], [22] in terms of the average path length, standard deviation of cluster nodes, and number of shortcuts. The simulation results indicate that SAPS outperforms GMSW and LM-GAS.

The rest of this paper is organized as follows. In Section II, a briefly related work is introduced. Section III describes the modeling process of SAPS with small-world characteristics. Section IV details the specific algorithm implementation. We present the evaluation results of the performance of SAPS, GMSW, and LM-GAS in Section V. Finally, in Section VI, we summarize the main results and discuss our future work.

II. RELATED WORK

Several studies explore the application of the small-world theory in a single-sink network. Helmy et al. [23] introduced the idea of small world to wireless network for the first time wherein several shortcuts were randomly added to the network to significantly reduce its average path length and ensure that the wireless network exhibits the characteristics of small world. Jiang et al. [24] used mobile router nodes (data mules) to construct variable length shortcuts. Data are loaded onto the mule at the source node and dismounted at the destination node. The path between the two nodes is termed as a shortcut. These data could be transmitted between nodes that do not communicate directly through data mules. Additionally, it is proved that the addition of a small number of data mules can significantly reduce the average path length of the network. Guidoni et al. [15], [25] proposed the directed angulation toward the sink node (DASM) model. They introduced super nodes and chose super nodes as the endpoints of shortcuts. Furthermore, given the special communication mode of IoT, the addition of a shortcut is allowed when the deflection angle of the shortcut and the sink node satisfies certain conditions. The addition of a small number of shortcuts reduces the communication delay of nodes and improves the reliability of the network. Huang et al. [26] used the narrow-band search space model to construct a small-world model. Simultaneously, the endpoints of the shortcuts are periodically replaced to reduce their energy consumption. Subsequently, the network exhibits the characteristics of small world and robustness. Asif et al. [27] calculated the betweenness of each node in the network by the neighbor avoiding walk (NAW) method and subsequently selected the nodes with large betweenness as the endpoints of the shortcuts. Zheng et al. [28] proposed the power control algorithm (PCS) to construct a small world theory for airborne health management. They constructed a shortcut between a power amplifier node and this node's neighbor node. Kong et al. [29] optimized topology based on a small-world network. They deployed a small number of super nodes in the network, and each super node was connected to other super nodes with a probability of p to form a long connection. After optimizing the topology, the efficiency of the network improved and the lifetime of the network was prolonged. Qiu et al. [22] proposed the greedy model with small world (GMSW). In GMSW, the neighbor nodes with high local importance are selected to add the shortcut, and thus the network exhibits better small-world characteristics and a certain robustness in terms of the node failure.

When compared to the network with a single-sink node, a few studies explored the application of the small-world theory to the multi-sink network. Verma et al. [21] proposed a load-balanced multi-gateway aware long link addition strategy (LM-GAS). In this, each node selects a gateway closest to itself based on the number of hops. The nodes that select the same gateway form a cluster. The shortcut addition includes the addition of shortcuts within the cluster and addition of shortcuts between the clusters. With respect to adding intra-cluster shortcuts, two nodes are randomly selected in the same cluster and the constraints are verified to add a shortcut between them. If the conditions are satisfied, a connection is established between the two nodes. With respect to adding inter-cluster shortcuts, two nodes that are in different clusters are randomly selected. Additionally, it is verified as to whether the two nodes satisfy the constraints, and if so, a connection is established between them. The results indicate that the average path length reduces and the load of the gateway is balanced.

In order to reflect the superiority of SAPS, we extend GMSW to the multi-sink network based on the idea of LM-GAS and compare SAPS with GMSW. First, the nodes in the network are divided into multiple clusters, and each cluster is equivalent to a single-sink network. While adding intra-cluster shortcuts, a shortcut is added between certain nodes based on the local importance. The method in GMSW is no longer applicable to add shortcuts among clusters, we adopt the strategy in LM-GAS when adding inter-cluster shortcut.

III. SAPS

The particle swarm algorithm is an evolutionary algorithm. It begins from a random solution and determines the optimal solution through iteration. The particle swarm algorithm e-valuates the quality of the solution through fitness value and obtains the optimal solution by tracking individual extremum and population extremum in the solution space. Given its simple and easy implement characteristics, the particle swarm algorithm is used to solve the problem of establishing a small-world model in the multi-sink Internet of Things.

In this section, we first introduce some preliminary concept-s. Subsequently we detail the modeling process of SAPS. The processes include particle encoding, construction of the fitness function, initialization of particles, and updation of particles.

A. Preliminaries

- Euclidean distance denotes the distance between two nodes. The Euclidean distance of node v and node s is as follows:

$$d(v,s)=\sqrt{(v_x-s_x)^2+(v_y-s_y)^2} \quad (1)$$

where v_x and v_y denote the coordinates of node v, and s_x and s_y denote the coordinates of node s.

- Neighbor node set: the maximum communication radius of the ordinary node is r, the maximum communication radius of the super node is R, and $R > r$. We consider the super node v as an example. Specifically, P'' denotes a collection of short-range neighbor nodes of the super node v. The Euclidean distance between the neighbor nodes and super node v is $\leq r$. Thus, $\forall s \in P'', d(v,s) \leq r$. Additionally, P' denotes a set of long-range neighbor nodes of super node v, which are super nodes or the sink nodes. The Euclidean distance between the neighbor nodes and the super node v is $> r$ and $\leq R$, $\forall s \in P', r < d(v,s) \leq R$. Therefore, the set of super node's neighbor nodes v is $P = P'' + P'$.
- Next hop node set: we use the long-range neighbor node set P' of each super node as its next hop node set.
- Average path length: It refers to the average number of hops from sensor nodes to the sink nodes.
- Standard deviation of number of nodes among clusters: it shows how the total number of sensors in the network is divided among sinks. The formula is as follows:

$$S = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(N_i - \frac{N}{m})^2} \tag{2}$$

where m denotes the number of sink nodes, N_i denotes the number of sensor nodes in the cluster of sink i (each node selects a sink closest to itself based on the number of hops). It is observed that a decrease in the standard deviation value makes the sink node load more balanced.

B. Encoding

In SAPS, each particle represents a shortcut addition scheme. In this paper, the particle is coded in terms of integers, and this corresponds to an integer array with a length of $|V|*d$. Specifically, $|V|$ represents the number of super nodes in the network, and d represents a sufficiently high number. The location of the $(i-1)*d$ of the array represents the super node V_i's ID. The $(i-1)*d+1 \sim i*d$ positions indicate the IDs of nodes that are connected to the super node V_i. It shows that shortcuts are added between the super node V_i and the nodes. Each particle encoding is shown in Fig. 1.

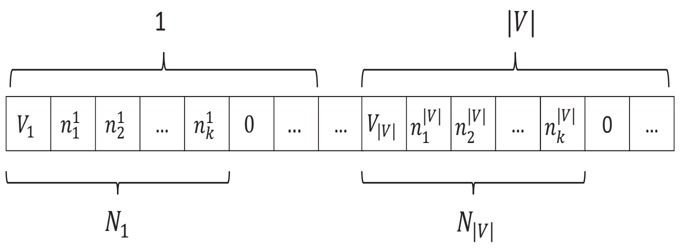


Fig. 1. Individual encoding

In Fig. 1, n_j^i denotes the ID of the jth node connected to the super node V_i. When the number N_i of nodes connected to V_i is lower than $d-1$, we set the values of $(i-1)*d+n_i+2 \sim i*d$ positions to 0. Additionally, if $n_j^i < 0$, it indicates that the jth node connected to the super node V_i is sink node.

C. Fitness Function

Each individual in the population is associated with a fitness value that is calculated from the fitness function. During the update iteration of the algorithm, we use a fitness value to represent the relative quality of the solution of the particle.

A few clusters are overloaded with sensor nodes when only the average path length is given while establishing a shortcut in a multi-sink network, and this results in an unbalanced load. This increases the pressure on the neighbor nodes of the sink node with a large load and significantly impacts the network. Thus, we combine the two factors of path length and sink load to construct the fitness function. This is expressed as follows:

$$Fitness = \gamma * \frac{L}{L_{initial}} + (1-\gamma) * D \tag{3}$$

$L_{initial}$ denotes the path length from sensor nodes to the sink node before adding shortcuts, L denotes the path length from sensor nodes to the sink node after adding shortcuts, and γ denotes the sliding factor for the interval $[0,1]$. Additionally, D denotes a variance function to evaluate the load among multiple sinks and is expressed as follows:

$$D = \frac{1}{m}\sum_{i=1}^{m}(\frac{N_i - N_{avg}}{N_{avg}})^2 \tag{4}$$

where m denotes the number of sink nodes, N_i denotes the number of sensor nodes in the sink i's cluster, and N_{avg} denotes the number of sensor nodes that should be contained in each cluster under ideal circumstances. This is expressed as follows:

$$N_{avg} = \frac{N}{m} \tag{5}$$

where N denotes the number of sensor nodes in the network. As shown in Eq. 4, when the number of sensor nodes in each cluster is closer to N_{avg}, a decrease in the value of D makes the load of sink more balanced. When the difference between the number of nodes in each cluster and N_{avg} is high, an increase in the value of D makes the load of sink more unbalanced. Therefore, as shown in the fitness function, a decrease in the fitness value of the particle improves the solution.

D. Initial Population

In SAPS, particles are randomly generated and each particle is a feasible solution. It is necessary to select node pairs in the communication range as endpoints for shortcuts to ensure the feasibility of the solution. Therefore, when a particle is initialized, each super node randomly selects a node from P' and connects with it. Thus, the connected nodes are within the communication range of the super node.

E. Particle Updates in SAPS

The particles in the SAPS are encoded into discrete forms, and thus the particle update formula in the standard particle swarm algorithm is no longer applicable. In order to update the particles, we introduce crossover and mutation in the genetic algorithm. A part of the information on the current particle and its optimal particle is saved by the crossover, and new individuals are generated through mutation.

*1) **Crossover**:* In this paper, the current particle is adjusted based on the population optimal particle. First, α different super nodes $v_1, v_2 \cdots v_\alpha$ are randomly selected from all $|V|$ super nodes. Specifically, $Lis_1, Lis_2 \cdots Lis_\alpha$ denote shortcut lists of the α super nodes in the current particle. The shortcut lists of the α super nodes in the optimal particle are as follows: $Ls_1, Ls_2 \cdots Ls_\alpha$. Next, the following operations are performed on the α super nodes. We disconnect the shortcuts between the super node v_i and the nodes in the list Lis_i. Subsequently, the connections between the super node v_i and nodes in the list Ls_i are established. A new particle is formed after completing the above operations of α super nodes. As shown in Fig. 2, an example is used to illustrate the operation.

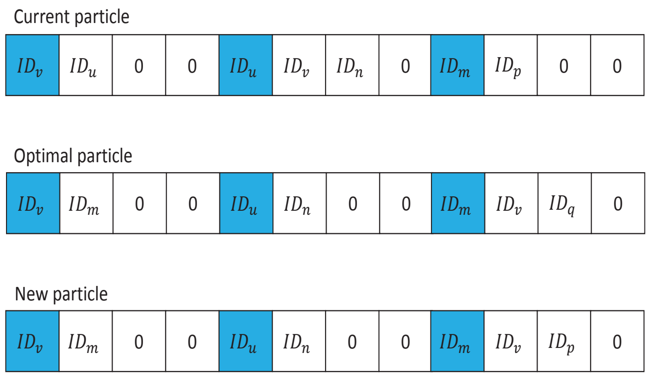

Current particle
ID_v ID_u 0 0 ID_u ID_v ID_n 0 ID_m ID_p 0 0
Optimal particle
ID_v ID_m 0 0 ID_u ID_n 0 0 ID_m ID_v ID_q 0
New particle
ID_v ID_m 0 0 ID_u ID_n 0 0 ID_m ID_v ID_p 0

Fig. 2. An example of crossover operation

In Fig. 2, the position denoted in blue stores the ID of the super node, and three positions are reserved behind each super node to store the node IDs that establish connections with the super node. We refer to the three positions as the super node's associated location. And the ID value of node i is expressed as ID_i. We only list the connections of a few super nodes. We assume that the super node v is selected as a cross node. The ID of the node v is ID_v. The super node v in the current particle is connected to node u, and the super node v in the optimal particle is connected to node m. While crossing, we disconnect the connection between nodes u and v. We find ID_u from the associated location of super node v and delete it. If $ID_u < 0$, it means that node u is a sink node. There is no associated position of the sink node in the particle, we don't need to perform subsequent delete operations. If $ID_u > 0$, we find ID_v from the associated location of super node u and delete it. We add a connection between nodes v and m. We add ID_m in the association position of super node v. And if $ID_m > 0$, we add ID_v in the association position of super node m. Finally, we obtain a new particle.

*2) **Mutation**:* In this paper, the mutation operation for each individual is as follows. We randomly select β different super nodes $(v_1, v_2 \cdots v_\beta)$ as the mutated nodes. Each super node is connected to the $n_1, n_2 \cdots n_\beta$ nodes in the communication range. Specifically, $Lis_1, Lis_2 \cdots Lis_\beta$ correspond to the shortcut lists of β super nodes. Subsequently, the following operations are performed on the β super nodes. The previous connection between v_i and the nodes in the list Lis_i is disconnected. We then randomly select n_i nodes (the nodes make up a list Lst_i) from the next hop list $P_i^{'}$ of v_i and $Lis_i \neq Lst_i$. The connections between the super node v_i and nodes in the list Lst_i are established. A new particle is formed after completing the above operations of β super nodes. An example of a mutation operation is given in Fig. 3.

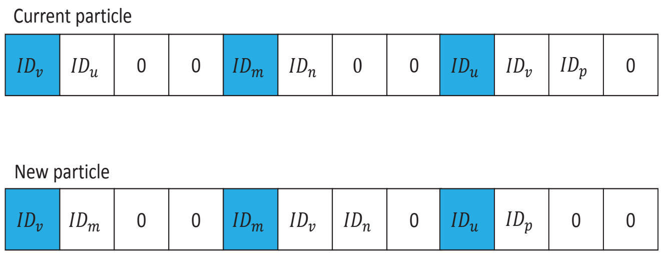

Current particle
ID_v ID_u 0 0 ID_m ID_n 0 0 ID_u ID_v ID_p 0
New particle
ID_v ID_m 0 0 ID_m ID_v ID_n 0 ID_u ID_p 0 0

Fig. 3. An example of mutation operation

It is assumed that the super node v is selected as a mutation node. The ID of the node v is ID_v. In the current particle, super node v is connected to node u. When mutating, we disconnect the connection between nodes v and u. We find ID_u from the associated location of super node v and delete it. If $ID_u > 0$, we find ID_v from the associated location of super node u and delete it. Subsequently, we randomly select a node m from the next hop node list and add a connection between v and m. We add ID_m in the association position of super node ID_v. And if $ID_u > 0$, we add ID_v in the association position of super node m. Finally, we obtain a new particle.

In the algorithm, we first initialize the particle, and each particle represents a scheme for adding shortcuts. After obtaining the initial solution, the fitness value of each particle in the population is calculated, and the population optimal particle is updated based on the fitness value. Subsequently, the crossover operation and mutation operation of each individual are performed, the fitness value of each particle is recalculated, and the population optimal particle is updated. Finally, whether the termination condition of the algorithm is reached is determined. If the termination condition is reached, the optimal particle of the population represents the final shortcut addition scheme; otherwise, the next iteration is performed.

IV. Algorithm Design

In this section, we implement the SAPS initialization algorithm, crossover operation algorithm, mutation operation algorithm, and SAPS algorithm. The variables used in all algorithms are shown in Table I:

Algorithm 1 describes the SAPS initialization process. Algorithm 1 works as follows. Before initializing the particle, the list of each super node's next hop is first obtained (lines 2–4). While initializing a particle, the shortcut lists of all super nodes of the particle are set as empty. (lines 6–8). Subsequently, each super node selects a node v_k from its next hop node list $P_i^{'}$ and connects with it, and the shortcut lists of the two nodes are separately updated. (lines 9–13). Finally, we obtain a new particle based on the shortcut lists of all super nodes (lines 14–18). After all particles are initialized, we obtain S different particle individuals.

Algorithm 2 describes the SAPS crossover operation process. Algorithm 2 works as follows. In the cross operation of a particle, α super nodes are first randomly selected from the list L. The contents of shortcut lists of the α super nodes are reorganized to generate a new individual (line 2). Subsequently, based on the population optimal particle and the current particle, two shortcut lists Ls_i and Lis_i of a super node v_i in Lc are obtained (lines 4 and 5), respectively. Next, the following operations are performed on the α super nodes. We disconnect the shortcuts between the super node v_i and the nodes in the list Lis_i (lines 6–8). Subsequently, the connections (shortcuts) between the super node v_i and nodes in the list Ls_i are established (lines 9–11). After performing the above operations on the α super nodes, the original particles recombine to form a new particle (lines 2–13).

TABLE I
PARAMETERS IN ALGORITHMS

Symbol	Description
S	The population size
L	List of all super nodes
$p_i^{'}$	list of node v_i's next hop node
Q	A population consisting of S particles
Lis_i	The shortcut list of node v_i in current particle
n	The number of elements in list Lis_i
num	Number of super nodes
d	A large enough constant
$particle$	Current particle
α	Number of cross super nodes
$opt\_particle$	Optimal particle in the population
Ls_i	Shortcut list of node v_i in optimal particle
Lc	List of cross super nodes
β	Number of the mutated super nodes
Lst_i	Shortcut list of node v_i after mutation
N	Number of elements in list $p_i^{'}$
Gen	Maximum number of iterations
Fit	List of fitness values for the population
$Gbest$	Record the best particle in the population

Algorithm 3 describes the SAPS mutation process. Algorithm 3 works as follows. First, we set a variable i with an initial value of 0 to indicate the number of super nodes that are conducted on the following evolution operations (line 2). We perform the following operations when i is less than β (line 3). In the mutation operation of a particle, we first randomly select a super node v_i from L that is not selected in this round (line 5). Subsequently, the shortcut list Lis_i of node v_i is obtained from the current particle, and the length of the Lis_i is calculated (lines 6–7). Next, the list $P_i^{'}$

Algorithm 1 Initialization

```
Input: $S$, $L$, $num$, $d$
Output: $Q$
1: procedure $Initialization()$
2:     for all $v_i \in L$ do
3:         $p_i^{'} \leftarrow getNextHopNodeList(v_i)$
4:     end for
5:     for $i = 1 \rightarrow S$ do
6:         for all $v_j \in L$ do
7:             $Lis_i \leftarrow \varnothing$
8:         end for
9:         for all $v_j \in L$ do
10:            Randomly select a node $v_k$ from $p_j^{'}$
11:            $Lis_j \leftarrow Lis_j \cup v_k$
12:            $Lis_k \leftarrow Lis_k \cup v_j$
13:        end for
14:        for $j = 1 \rightarrow num$ do
15:            $Q[i, (j-1) * d + 1] = v_j$
16:            $n \leftarrow getLength(Lis_j)$
17:            $Q[i, (j-1) * d + 2 \cdots (j-1) * d + n + 1] = Lis_j[1 \cdots n]$
18:        end for
19:    end for
20:    return $Q$
21: end procedure
```

Algorithm 2 Crossover Operator

```
Input: $\alpha$, $num$, $particle$, $opt\_particle$, $L$
Output: $particle$
1: procedure $Cross()$
2:     $Lc \leftarrow getNodeList(L, \alpha, num)$
3:     for all $v_i \in Lc$ do
4:         $Ls_i \leftarrow getShortcutList(opt\_particle, v_i)$
5:         $Lis_i \leftarrow getShortcutList(particle, v_i)$
6:         for all $v_k \in Lis_i$ do
7:             Delete the link from $v_k$ to $v_i$
8:         end for
9:         for all $v_k \in Ls_i$ do
10:            Add the link from $v_k$ to $v_i$
11:        end for
12:    end for
13:    return $particle$
14: end procedure
```

of the super node v_i's next hop node is obtained, and the length of the $P_i^{'}$ is calculated (lines 8–9). When shortcuts are established between v_i and all nodes in list $P_i^{'}$, or no shortcuts are established between v_i and nodes in list $P_i^{'}$, we select a super node again to perform the mutation (10–12 lines). Otherwise, we randomly select n nodes from the list $P_i^{'}$ to form the list Lst_i. In order to generate a new particle, we select $Lst_i \neq Lis_i$ (lines 13–15). We disconnect the shortcuts between the super node v_i and nodes in the list Lis_i (lines 16–18). The connections (shortcuts) between the super node v_i and nodes in the list Lst_i are established (lines 19–21). After completing the mutation operation of super node v_i,

Algorithm 3 Mutation Operator

```
Input: $\beta$, $num$, $particle$, $L$
Output: $particle$
procedure $Mutation()$
    $i \leftarrow 0$
    while $i < \beta$ do
        $v_i \leftarrow getNode(L, num)$
        $Lis_i \leftarrow getShortcutList(particle, v_i)$
        $n \leftarrow getLength(Lis_i)$
        $p_i^{'} \leftarrow getNextHopNodeList(v_i)$
        $N \leftarrow getLength(p_i^{'})$
        if $n == N \,\|\, n == 0$ then
            continue
        end if
        do
            Randomly select $n$ nodes from $p_i^{'}$, the $n$ nodes constitute the $Lst_i$
        while $Lst_i \neq Lis_i$
        for all $v_k \in Lis_i$ do
            Delete the link from $v_k$ to $v_i$
        end for
        for all $v_k \in Lst_i$ do
            Add the link from $v_k$ to $v_i$
        end for
        $i \leftarrow i + 1$
    end while
    return $particle$
end procedure
```

we set the value of variable i to increase by 1 (line 22). After performing the aforementioned operations on the β super nodes, the original particle mutates to form a new particle (lines 2–24).

Algorithm 4 describes the detailed process of the SAPS. The algorithm 4 works as follows. First, different particle individuals are obtained (line 2) through the initialization operation. Before the optimization operation, the initial fitness value for each particle is calculated, and the particle with the smallest fitness value is recorded as the population optimal particle (lines 3–6). Subsequently, the SAPS begins. The crossover operation and mutation operation are performed for each particle in the population in turn. A new particle is obtained by crossing the particle with the population optimal particle. If the fitness value of the new particle is smaller than the fitness value of the old particle, then the old particle is replaced by the new particle; otherwise, it remains unchanged (lines 9-13). A new particle is obtained by the particle self-mutation. Similarly, if the new particle's fitness value is lower than the old particle's fitness value, the old particle is replaced by the new particle; otherwise, it remains unchanged (lines 14–18). At the end of each iteration, the population optimal particle is updated based on the fitness value (line 20). The process continues until the number of iterations reaches Gen.

Algorithm 4 SAPS

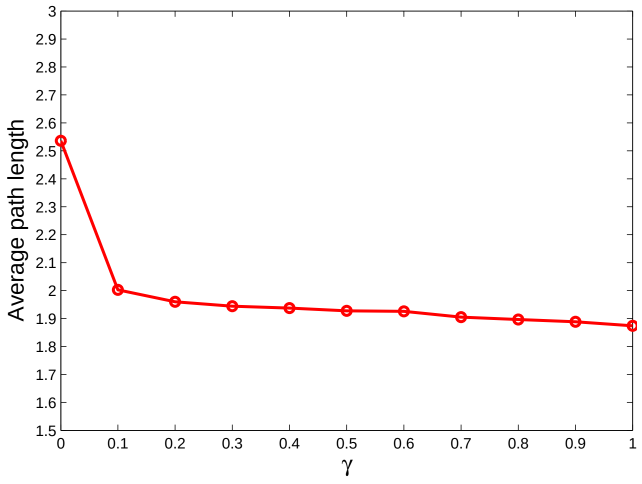
```
Input: $S$, $num$, $Gen$, $L$, $\alpha$, $\beta$
Output: $Gbest$
procedure $SAPS()$
    $Q \leftarrow Initialization()$
    for $i = 1 \rightarrow S$ do
        $Fit[i] \leftarrow getFitnessValue(Q[i])$
    end for
    Record individual $Gbest$ with the minimum fitness value
    for $i = 1 \rightarrow Gen$ do
        for $j = 1 \rightarrow S$ do
            $new\_particle \leftarrow Cross(num, \alpha, Q[j], Gbest, L)$
            if $Fit[j] > getFitnessValue(new\_particle)$ then
                $Fit[j] = getFitnessValue(new\_particle)$
                $Q[j] = new\_particle$
            end if
            $new\_particle \leftarrow Mutation(\beta, Q[j], num, L)$
            if $Fit[j] > getFitnessValue(new\_particle)$ then
                $Fit[j] = getFitnessValue(new\_particle)$
                $Q[j] = new\_particle$
            end if
        end for
        Record individual $Gbest$ with the minimum fitness value
    end for
    return $Gbest$
end procedure
```

V. Simulation results

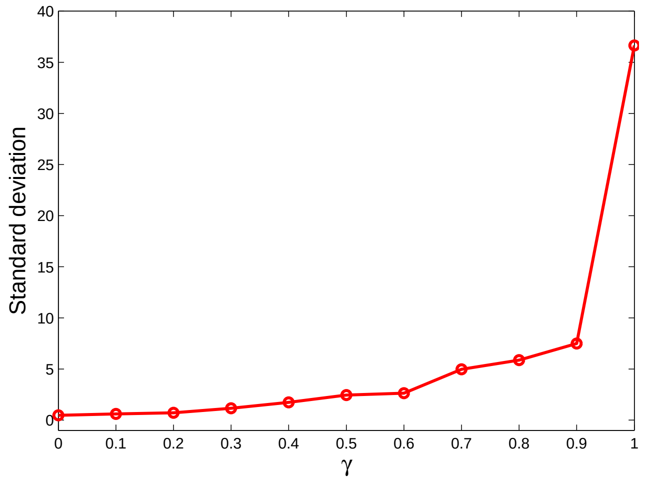
In this section, we first determine the value of γ through experiments. Subsequently, we compare our proposed method SAPS with the LM-GAS and GMSW from three aspects while randomly placing different sink numbers. The aspects include the average path length, degree of load of sink, and the number of added shortcuts. All algorithms are simulated in MATLAB. In the simulation experiment, we randomly deploy nodes in a 1000*1000 area. The nodes correspond to super nodes and ordinary nodes. The communication range of the super node is 450 m, and the communication range of the ordinary node is 150 m. The number of sink node in the topology increases from 2 to 6, and the sink nodes are randomly placed in the network. The results obtained from the experiments are taken from the average of 10 simulations.

A. Effect of fitness function coefficients

The coefficient γ of the fitness function is a sliding factor used to balance the average path length and sink node load. The optimal value of γ should be determined before conducting comparison experiments. Specifically, γ is in the range [0,1], and we select γ with an interval of 0.1. Prior to determining the value of γ, we obtain the number of crossover nodes $\alpha = 4$, number of mutation nodes $\beta = 1$, and number of particles $Num = 40$. We set the maximum number of



(a) Average path length



(b) Standard deviation

Fig. 4. The effect of change in γ on the average path length and standard deviation of number of nodes among clusters

iterations $Nmax$ to 400.

In order to analyze the effect of changes in γ on the average path length and the sink node load in further detail, we separately evaluate the average path length and standard deviation of number of nodes among the clusters (Eq. 2). We set the number of sensor nodes to 200, and the number of super nodes is 20%. Additionally, all experiments are performed under the same number of sink 3. Each sensor node selects a sink node as the end of the data transmission in terms of number of hops. If more than one such sink node with equal distance exists, it randomly picks one. As shown in Fig. 4, when the number of sink nodes in the network is constant, the value of the average path length decreases when γ increases, and the value of the standard deviation continues to increase. However, when the value of γ changes from 0.9 to 1, the value of the standard deviation increases significantly. We combine the conclusions in Fig. 4(a) and Fig. 4(b) and select γ =0.9 because the average path length is close to the minimum. Furthermore, the value of standard deviation is maintained at a relatively high level.

B. Shortcut addition diagram

Figure 5 illustrates the shortcuts created using LM-GAS, GMSW and SAPS respectively when the number of sink n is 3 and 5. In the above figure, the black nodes represent sensor nodes and the number is 200, and the red nodes represent the sink nodes. The grey edges indicate the connection relationships between the sensor nodes. The blue thick edges indicate shortcuts. As we can see in Figure 5(a), Figure 5(b), Figure 5(d) and Figure 5(e), LM-GAS and GMSW add constraints while adding inter-cluster shortcuts although a certain degree of randomness exists when selecting the node pair as the shortcut endpoint. In this case, they may add some useless edges. In SAPS (Figure 5(c) and Figure 5(f)), it searches the appropriate node pairs to add shortcuts by the way of iteration. It improves the quality of shortcuts and reduces the appearance of redundant edges.

C. Performance evaluation of SAPS

In this part, we analyze the average path length of the network, the number of added shortcuts and the standard deviation of the number of nodes among clusters change with the number of sink nodes when there are 200 sensor nodes in the network. And the number of super nodes accounts for 20%.

Figure 6 shows the average path length when the total number of sensor nodes is 200 and the number of sink nodes is 2–6 in the network. As shown in the figure, the average path length decreases when the number of sink nodes in the network increases. The total number of sensors in each cluster is reduced when the number of sink nodes is high. Therefore, only a small number of hops is required from the sensor node to the sink node. Additionally, in all the cases, the average path length of the SAPS is less than that of the LM-GAS and GMSW.

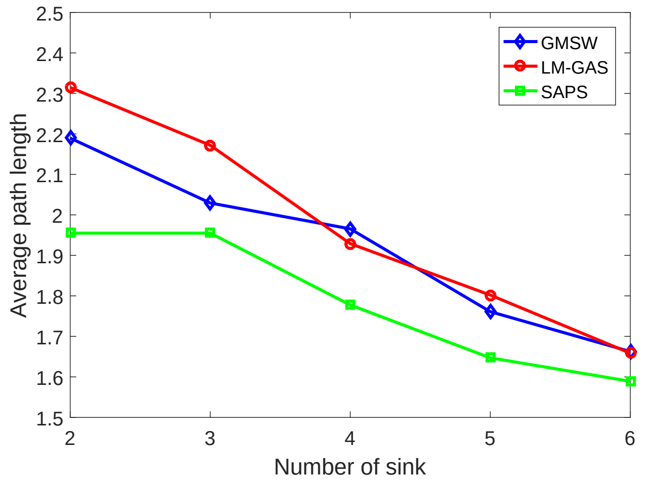
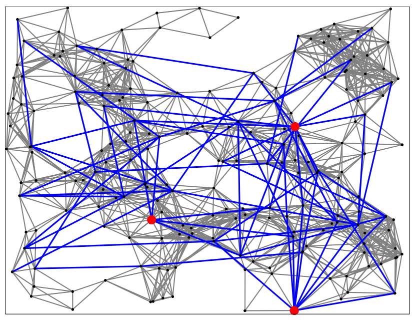
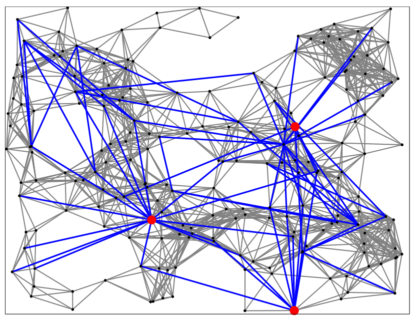
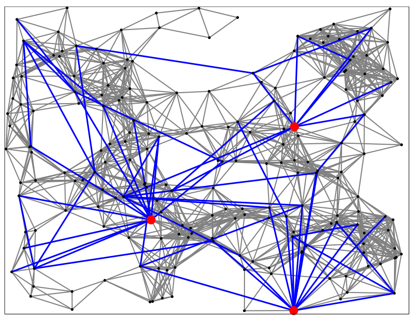


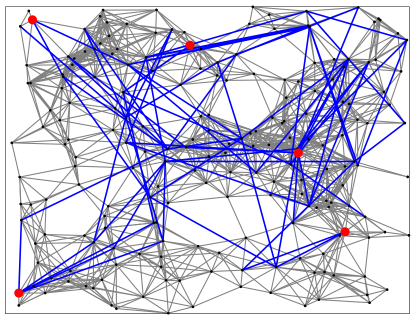
Fig. 6. Average path length with different number of sinks

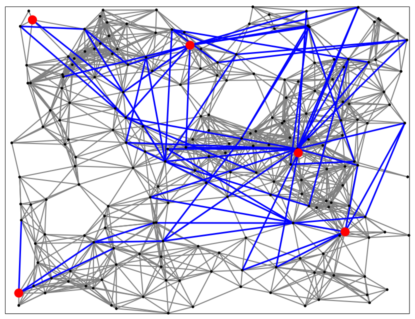
Figure 7 shows the standard deviation when the total number of sensor nodes is 200 and the number of sink nodes is 2–6 in


(a) LM-GAS(n=3)


(b) GMSW(n=3)


(c) SAPS(n=3)


(d) LM-GAS(n=5)


(e) GMSW(n=5)

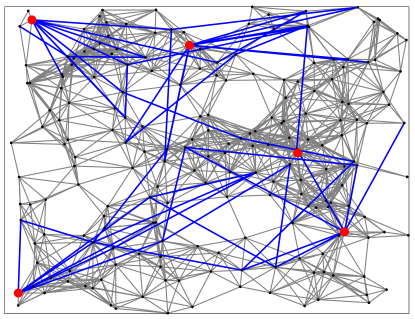

(f) SAPS(n=5)

Fig. 5. Creation of shortcuts when the number of sink changes in LM-GAS, GMSW and SAPS

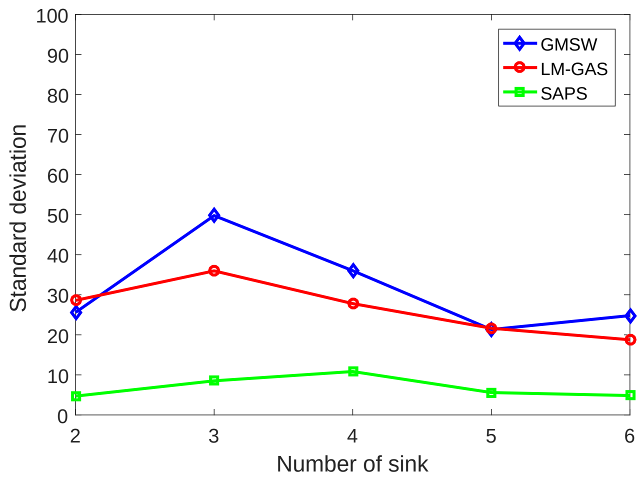


Fig. 7. Standard deviation with different number of sinks

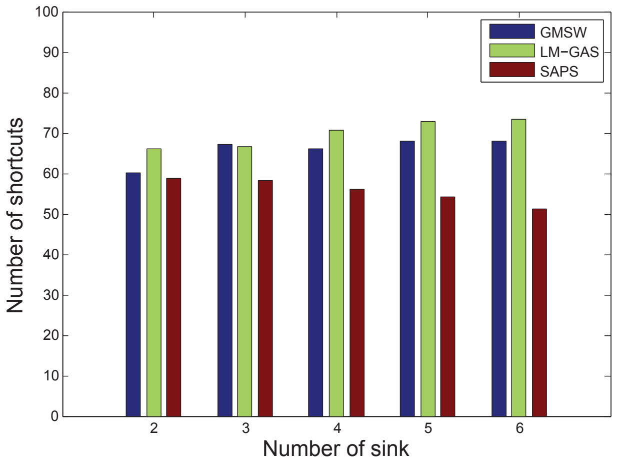


Fig. 8. Number of shortcuts with different number of sinks

the network. As shown in the graph, the standard deviation of the SAPS is lower than that of LM-GAS and GMSW. Thus, in SAPS, the number of sensor nodes in each cluster is more evenly distributed. This aids in balancing the load of the sink node to a considerable extent.

Figure 8 shows the number of shortcuts when the total number of sensor nodes is 200 and the number of sink nodes is 2–6 in the network. As shown in the figure, the number of shortcuts added by SAPS is lower than that of LM-GAS and GMSW. Additionally, when the number of sink nodes corresponds to 2 and 3, the number of shortcuts added by SAPS does not significantly differ from that of LM-GAS and GMSW. When the number of sink nodes is high, the number of shortcuts added by the SAPS is significantly lower than that of LM-GAS and GMSW.

D. The scalability of SAPS

In this section, we study the scalability of SAPS. For this, we use 300 and 500 sensor nodes to conduct experiments, in which the number of super nodes accounted for 20%. We analyze the average path length of the network, the number of shortcuts added and the standard deviation of the number of nodes among clusters as the number of sink nodes changes.

When there are 300 nodes and 500 nodes in the network respectively, the number of shortcuts added by different models is shown in Figure 9 and Figure 10. It can be seen that the number of shortcuts added by SAPS is still lower compared to GMSW and LM-GAS.

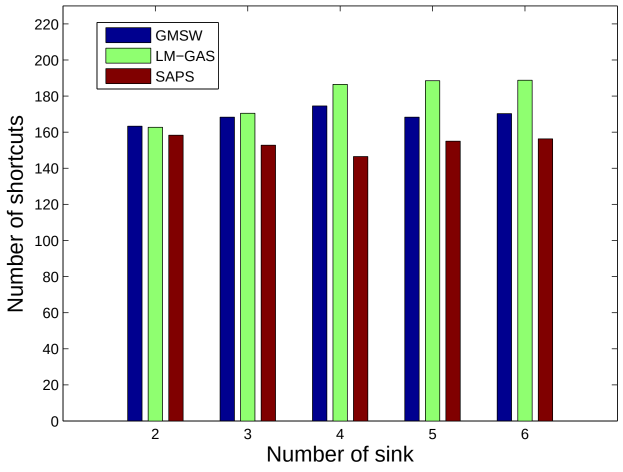


Fig. 9. Number of shortcuts with different number of sinks(the number of nodes is 300)

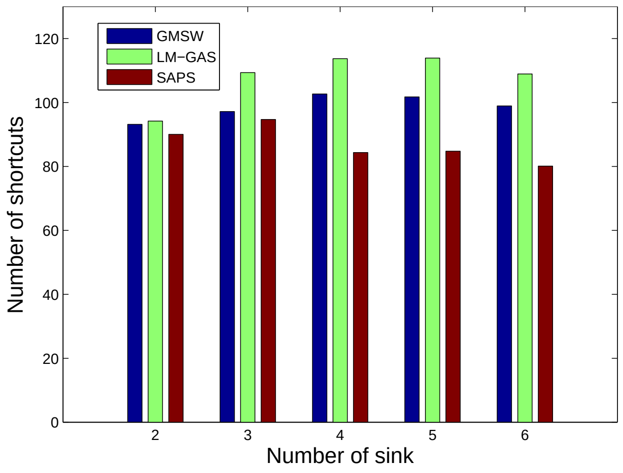


Fig. 10. Number of shortcuts with different number of sinks(the number of nodes is 500)

Figure 11 and Figure 12 show the average path length when the total number of sensor nodes in the network is 300 and 500, respectively. As can be seen from the figures, SAPS can still maintain the average path length of the network at a reduced level and is better than LM-GAS and GMSW.

Figure 13 and Figure 14 show the standard deviation of the number of nodes between clusters when the total number of sensor nodes in the network is 300 and 500, respectively. It can be seen from the figures that compared with LM-GAS and GMSW, the value of the standard deviation of the proposed method is relatively small, which can largely balance the load of the sink node.

VI. Conclusion

In this paper, we proposed a novel shortcut addition algorithm with particle swarm for multi-sink Internet of Things.

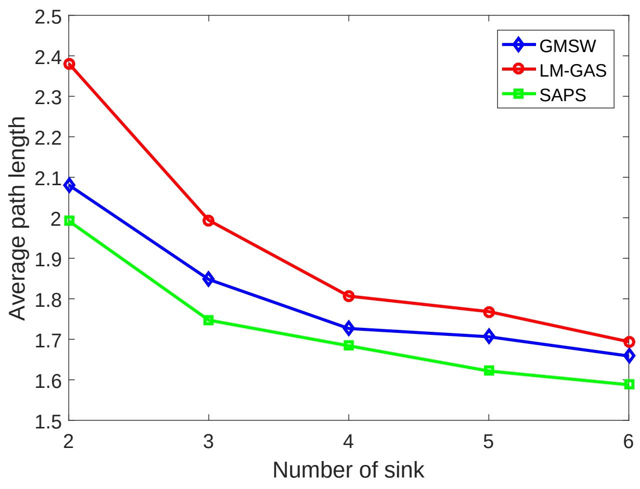


Fig. 11. Average path length with different number of sinks(the number of nodes is 300)

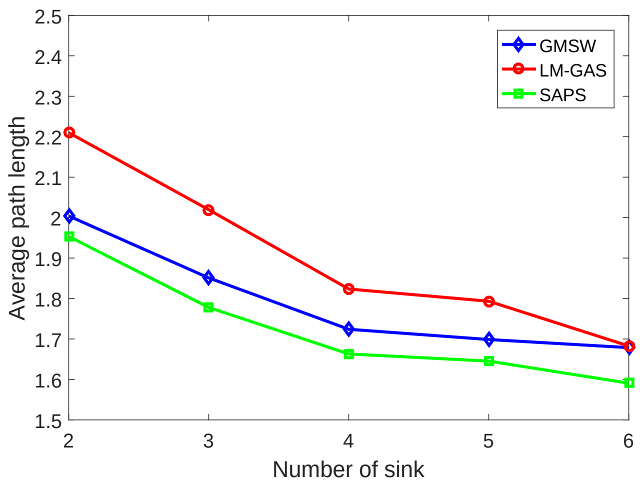


Fig. 12. Average path length with different number of sinks(the number of nodes is 500)

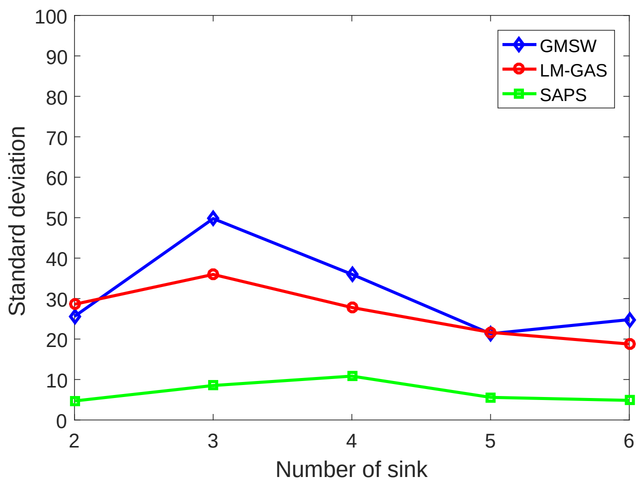


Fig. 13. Standard deviation with different number of sinks(the number of nodes is 300)

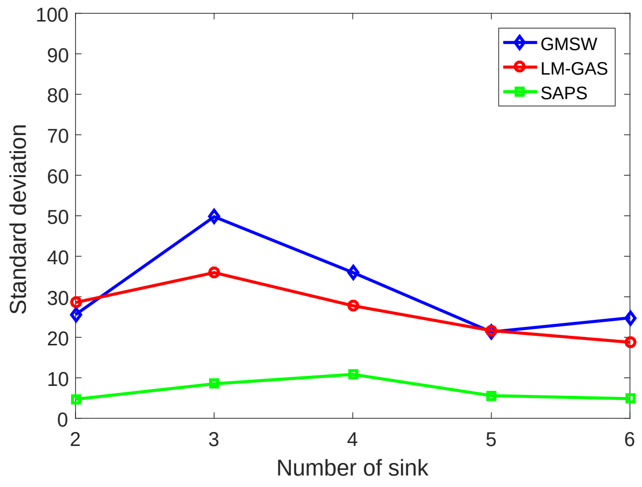


Fig. 14. Standard deviation with different number of sinks(the number of nodes is 500)

While adding shortcuts, the aim involved minimizing the average shortest path length of the network and balancing the load of sink nodes. Additionally, we introduced crossover and mutation operations. The optimal solution was obtained by searching for the solution space through crossing and particle self-mutation. Finally, we compared our algorithm with the two existing algorithms in terms of the average path length, number of added shortcuts, and standard deviation of cluster nodes. The experimental results indicated that when the number of sink nodes in the network was identical, the algorithm added fewer shortcuts and ensured that the entire network exhibited a shorter average path length. Simultaneously, when compared with GMSW and LM-GAS, the SAPS balanced the load among sink nodes in a better manner and maintained the standard deviation of cluster nodes at a relatively small level.

In the future work, we will further explore the construction of topological structure based on complex networks in the actual environment. We plan to construct a high-performance network topology by considering various factors in the actual environment.

REFERENCES

[1] R. Amin, S. H. Islam, G. Biswas, and M. S. Obaidat, "A robust mutual authentication protocol for wsn with multiple base-stations," *Ad Hoc Networks*, vol. 75, pp. 1–18, 2018.

[2] D. Zhang, S. Zhao, L. T. Yang, M. Chen, Y. Wang, and H. Liu, "Nextme: Localization using cellular traces in internet of things." *IEEE Trans. Industrial Informatics*, vol. 11, no. 2, pp. 302–312, 2015.

[3] M. Hammoudeh, F. Al-Fayez, H. Lloyd, R. Newman, B. Adebisi, A. Bounceur, and A. Abuarqoub, "A wireless sensor network border monitoring system: Deployment issues and routing protocols," *IEEE Sensors Journal*, vol. 17, no. 8, pp. 2572–2582, 2017.

[4] T. Qiu, R. Qiao, and D. O. Wu, "Eabs: An event-aware backpressure scheduling scheme for emergency internet of things," *IEEE Transactions on Mobile Computing*, no. 1, pp. 72–84, 2018.

[5] D. Cullar, D. Estrin, M. Strvastava *et al.*, "Overview of sensor network," *Computer*, vol. 37, no. 8, pp. 41–49, 2004.

[6] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of things: Vision, applications and research challenges," *Ad hoc networks*, vol. 10, no. 7, pp. 1497–1516, 2012.

[7] T. Qiu, A. Zhao, F. Xia, W. Si, D. O. Wu, T. Qiu, A. Zhao, F. Xia, W. Si, and D. O. Wu, "Rose: Robustness strategy for scale-free wireless sensor networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 25, no. 5, pp. 2944–2959, 2017.

[8] J. Liu, K. Huang, and G. Zhang, "An efficient distributed compressed sensing algorithm for decentralized sensor network," *Sensors*, vol. 17, no. 4, p. 907, 2017.

[9] S. Saginbekov and A. Jhumka, "Many-to-many data aggregation scheduling in wireless sensor networks with two sinks," *Computer Networks*, vol. 123, pp. 184–199, 2017.

[10] C. Li, L. Li, and X. Wang, "An optimal clustering routing algorithm for wireless sensor networks with small-world property," *Wireless Personal Communications*, vol. 96, no. 2, pp. 2983–2998, 2017.

[11] P. Geng, Y. Liu, and J. Yang, "A dynamic energy balance method for wireless sensor networks with small world characteristics," in *Proceedings of the Fifth International Conference on Network, Communication and Computing*. ACM, 2016, pp. 287–291.

[12] W. Shu and Y.-H. Chuang, "The perceived benefits of six-degree-separation social networks," *Internet Research*, vol. 21, no. 1, pp. 26–45, 2011.

[13] D. J. Watts and S. H. Strogatz, "Collective dynamics of small-worldnetworks," *nature*, vol. 393, no. 6684, p. 440, 1998.

[14] M. E. Newman and D. J. Watts, "Renormalization group analysis of the small-world network model," *Physics Letters A*, vol. 263, no. 4-6, pp. 341–346, 1999.

[15] D. L. Guidoni, A. Boukerche, L. A. Villas, F. S. Souza, R. A. Mini, and A. A. Loureiro, "A framework based on small world features to design hsns topologies with qos," in *IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2012, pp. 732–737.

[16] H. M. Ammari, "Investigating the energy sink-hole problem in connected k-covered wireless sensor networks," *IEEE Transactions on Computers*, vol. 63, no. 11, pp. 2729–2742, 2014.

[17] S.-Y. Choi, J.-S. Kim, S.-J. Han, J.-H. Choi, K.-W. Rim, and J.-H. Lee, "Dynamic routing for mitigating the energy hole based on heuristic mobile sink in wireless sensor networks," in *Advances in Computer Science and Information Technology*. Springer, 2010, pp. 159–174.

[18] T. Nagamalar and T. Rangaswamy, "Energy efficient cluster based approach for data collection in wireless sensor networks with multiple mobile sink," in *International Conference on Industrial Instrumentation and Control (ICIC)*. IEEE, 2015, pp. 348–353.

[19] M. Liu, J. Zhang, M. Lyu, and Y. Bo, "A novel solution for energy hole of wireless sensor network," in *Chinese Control Conference (CCC)*. IEEE, 2014, pp. 456–460.

[20] M. Rajesh and J. Gnanasekar, "Congestion control in heterogeneous wanet using frcc," *Journal of Chemical and Pharmaceutical Sciences (ISSN)*, vol. 974, p. 2115, 2015.

[21] A. Verma, C. K. Verma, B. R. Tamma, and B. Manoj, "New link addition strategies for multi-gateway small world wireless mesh networks," in *International Symposium on Advanced Networks and Telecommunication Systems (ANTS)*. IEEE, 2010, pp. 31–33.

[22] T. Qiu, D. Luo, F. Xia, N. Deonauth, W. Si, and A. Tolba, "A greedy model with small world for improving the robustness of heterogeneous internet of things," *Computer Networks*, vol. 101, pp. 127–143, 2016.

[23] A. Helmy, "Small worlds in wireless networks," *IEEE Communications Letters*, vol. 7, no. 10, pp. 490–492, 2003.

[24] C.-J. Jiang, C. Chen, J.-W. Chang, R.-H. Jan, and T. C. Chiang, "Construct small worlds in wireless networks using data mules," in *IEEE International Conference on Sensor Networks, Ubiquitous and Trustworthy Computing*. IEEE, 2008, pp. 28–35.

[25] D. L. Guidoni, R. A. Mini, and A. A. Loureiro, "On the design of resilient heterogeneous wireless sensor networks based on small world concepts," *Computer Networks*, vol. 54, no. 8, pp. 1266–1281, 2010.

[26] J. Huang, Q. Xie, and B. Huang, "Creating small-world model for homogeneous wireless sensor networks," in *International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM)*. IEEE, 2012, pp. 1–4.

[27] W. Asif, H. K. Qureshi, and M. Rajarajan, "Variable rate adaptive modulation (vram) for introducing small-world model into wsns," in *Annual Conference on Information Sciences and Systems (CISS)*. IEEE, 2013, pp. 1–6.

[28] W. Zheng and D. Luo, "Small world-based wireless sensor network power control algorithm for airborne phm," in *Advanced Technologies in Ad Hoc and Sensor Networks*. Springer, 2014, pp. 177–186.

[29] P. Kong, G. Fang, C. He, and Z. Liu, "Topology optimization of port wireless sensor network based on small-world network," in *International Conference on Circuits, System and Simulation (ICCSS)*. IEEE, 2017, pp. 157–161.