EE-MSWSN: Energy-Efficient Mobile Sink Scheduling in Wireless Sensor Networks

Morteza Biabani, Nasser Yazdani, and Hossein Fotouhi

***Abstract*—Data gathering using mobile sink (MS) based on rendezvous points (RPs) is a need in several Internet of Things (IoT) applications. However, devising energy-efficient and reliable tour planning strategies for MS is a challenging issue, considering that sensors have finite buffer space and disparate sensing rates. This is even more challenging in delay-tolerant networks, where it is more desirable to select the shortest traveling path. There exist several algorithms on MS scheduling, which are based on hierarchical protocols for data forwarding and data collection. These algorithms are lacking efficient tradeoff between the Quality-of-Service (QoS) requirements in terms of energy efficiency, reliability, and computational cost. Besides, these algorithms have shown high packet losses while jointly performing MS tour planning and buffer overflow management. To address these limitations, we propose EE-MSWSN, an energy-efficient MS wireless sensor network that reliably collects data by implementing efficient buffer management. It forms novel clustered tree-based structures to cover all the network, and select each RP based on 1) hop count; 2) number of accumulated data in each clustered tree; and 3) distance to the stationary sink. The extensive simulation results verify that the EE-MSWSN minimizes tour length for various network configurations and incurs less energy consumption while reliably gathering data without packet losses as compared with existing protocols.**

***Index Terms*—Energy efficiency, mobile sink (MS), path planning, wireless sensor network (WSN).**

I. Introduction

IOT PARADIGM is empowered by a virtual layer of wireless sensor networks (WSNs) that contains a vast number of resource-constrained devices, namely, sensors, uniformly, or randomly deployed in the sensing field [1]. WSNs are the main building blocks of the current Internet of Things (IoT) applications, enabling sensing as well as low-power wireless communication. To form pervasive smart environments, WSNs are integrated as an important pillar of IoT-based collaborative framework and operate as an interface for data gathering from the physical environment to the IoT [2]. Due to the use of battery-driven sensors (or IoT devices), energy efficiency is one of the main Quality-of-Service (QoS) requirements in WSNs, where sensors may be employed in delay-tolerant networks [3] that require collecting data in time-bounded periods, e.g., in fire-detection systems and military perimeter monitoring [4], [5]. In WSN-assisted IoT applications, collected data by sensors are transmitted to a base station (BS) or a stationary sink (also called the IoT Hub) through single- or multi-hop [6]. In such applications, the stationary sink acts as an access point (like a bridge) between the Internet and traditional WSN so that the measured data is commonly shared with the Internet or cloud-based infrastructures for data analytics [7], [8].

A crucial disadvantage of a multihop data forwarding model is that surrounding sensors to a stationary sink must relay more data than distant nodes, leading to the use up of their batteries sooner than other nodes. This well-known situation is called *hotspot* or *energy hole problem*, leading to network partitioning and further increasing traffic dispersion, thereby reducing WSN lifetime [9], [10].

Mobile sink (MS) was raised first in [11] and adopted as a widely accepted solution to many of the challenges caused by a stationary sink in resource-constrained WSNs. Moreover, many studies in WSN-assisted IoT applications have proven that using an MS to solve the hotspot issue is an effective solution as it leads to changing the surrounding sensors to the stationary sink frequently. The MS (e.g., unmanned aerial vehicle [12]) uniformly disperses the energy consumption in the entire network. In addition, it ensures reliable data collection by reducing the number of transmitted packets and enhances the accessibility in isolated environments [13]. On the other hand, in some scenarios, using MS can be inescapable, e.g., health monitoring application is composed of mobilized patients carrying MSs that constantly report health data [11], [14]. Apparently, in spite of many advantages, the MS brings new challenges to IoT networks.

Tour planning is the periodic travels of the MS on a predetermined route to visit stationary sensors to collect their data [3], [13]. In this regard, there is no need for immediate data transmission through multihop, instead of waiting for the MS to collect the buffered data, and thus enhancing power efficiency by reducing communication [15]. One challenging issue is the scheduling of the shortest traveling path of the MS, especially for hard real-time applications [16]. Intuitively, if the MS travels the periphery of each node, the issue becomes the NP-complete traveling salesman problem (TSP) [17]. It becomes more difficult in large-scale WSNs when the MS is liable to breach the data delivery delay constraint due to the journey

Manuscript received 23 September 2021; revised 6 January 2022 and 19 February 2022; accepted 14 March 2022. Date of publication 17 March 2022; date of current version 23 September 2022. This work was supported by the Swedish Research Council (Vetenskapsrådet) through the MobiFog starting grant, and Vinnova by the GREENER Project. *(Corresponding author: Morteza Biabani.)*

Morteza Biabani and Nasser Yazdani are with the Router Laboratory, School of Electrical and Computer Engineering, College of Engineering, University of Tehran, Tehran 14395515, Iran (e-mail: mbiabani@ut.ac.ir; yazdani@ut.ac.ir).

Hossein Fotouhi is with the School of Innovation, Design and Engineering, Mälardalen University, 721 23 Västerås, Sweden (e-mail: hossein.fotouhi@mdh.se).

Digital Object Identifier 10.1109/JIOT.2022.3160377

in a longer path, leading this solution to be impractical [18]. Existing literature has proposed identifying a certain number of sensors as rendezvous points (RPs) to bound the MS trajectory [10], [13]. However, electing the proper RPs is a challenging problem as NP-hard. In particular, the regular nodes (non-RPs) carry sensory data via multihops to nearby RPs and the MS only communicates with RPs to gather all data [19]. Consequently, one critical issue becomes how to seek proper RPs not only uniformly disperse the energy consumption due to a multihop manner (from non-RPs to RPs) but also minimize the length of the MS's trajectory.

To the best of our knowledge, existing methods in WSNs with an MS are focused on designing problem-solving approaches with fixed assumptions that lead to infeasible and impractical solutions [15], [20]. By relaxing the fixed assumptions of sensors to 1) limited buffer space and 2) disparate sensing rates, these studies will lead to packet losses at several RPs due to the occurrence of buffer overflow, and thereby failing to support real-world applications, e.g., after detecting abnormal events in some WSN-assisted IoT applications, the stationary sink may request some sensors to speed up their sensing rates once [21]. Since the sensors have a limited buffer capacity due to their tiny design, a large space require to compress data to mitigate transmissions due to the temporary and spatial correlation of sensing reports [18]. Thus, the compression methods are expected to be simple to execute in common sensor devices.

On the other hand, data aggregation processes (from non-RP sensors to RPs) in WSNs with an MS mostly contain two techniques: 1) tree-based and 2) clustering, which are known as client–server-based methods [5], [22], where the stationary sink behaves as the server and the sensor nodes as the clients. In the tree-based technique, selected relay nodes (tier nodes) usually forward sensed data from leaves to the root. In the clustering technique, sensors are divided into several clusters. Each cluster member (CM) communicates with its cluster head (CH) to perform the data aggregation process [23]. Although client–server-based methods prolong the network lifetime to some extent, they result in further problems including low reliability (considering data priority) and high data transmission delay. Existing tree-based data-aggregation protocols mainly suffer from high energy consumption but partially reduce packet losses. In contrast, clustering techniques generally improve network lifetime but inject high overhead costs to the network, leading to packet losses [4], [24]. Due to the existing tradeoff between the QoS requirements, in terms of reliability, energy efficiency, and computation time, these techniques generally concentrate on one of these metrics, while neglecting the others, leading to an unstable balance in QoS requirements. Besides, in most of the existing techniques, there may be a long distance between non-RP nodes and their RP, also known as a chain problem, thus being exposed to hotspot on high relay nodes (top-tier) of each RP [6], [25], [26]. Essentially, the number of non-RPs belonging to each RP also depends on the amount of collected data from them. Therefore, hop-constrained communications between non-RPs and their RP should be designated to the data gathering mechanism, which is ideal for delay-tolerant networks.

To overcome the restrictions of the existing works, we propose EE-MSWSN, an energy-efficient MS WSN that leverages advantageous of a combination of tree-based and clustering techniques and reduces both the length of MS traveling path and energy consumption, by considering that sensors have finite buffer space and disparate sensing rates. By varying sensors' sensing rates, EE-MSWSN can adaptively recompute new tours. In turn, enhancing the buffer utilization of sensors throughout the network by hop-constrained communications. The objectives are to 1) optimize the efficient MS trajectory scheduling (reduce path length); 2) minimize the energy consumption of sensors; and 3) prevent the occurrence of buffer overflow at RPs. To do so, the EE-MSWSN algorithm constructs novel clustered shortest-path (SP) tree structures to cover the entire network then by starting from proper RP candidate selection, the initial MS traveling path is formed. Afterward, by starting from leaf nodes in each partition, it progressively finds finalized RPs according to the amount of relay data, hop count, and distance to the stationary sink. Thus, each RP can have enough buffer size to cache all forwarding data and its own data, before the MS comes to retrieve them. Our centralized protocol achieves efficiency in terms of total energy consumption, buffer utilization and its standard deviation, path length, allowable computation time, and dropped packet counts, as compared with state-of-the-art protocols. Major contributions of this work are summarized as follows.

1) We propose an energy-efficient method for MS tour planning that jointly minimizes the length of the MS's trajectory and avoids packet loss by varying sensing rate of sensors and their finite buffer size.
2) We leverage a combination of tree-based and clustering techniques, where the sensing field is divided into different partitions. This efficiently alleviates the hotspot issue on the high relay nodes of each clustered tree.
3) We implement the RPs replacement algorithm in the data gathering process to further reduce tour length by bounding relay-hop value. Moreover, computed new tours adjusting with sensing rates based on expected computational cost.
4) We perform extensive simulations on various network configurations to show the significance of the proposed scheme that yields to both uniform energy consumption and reliable data gathering. This attribute conditions in the network to efficient data dissemination that comply with the QoS requirements of WSNs.

The remainder of this article is organized as follows. The related works are discussed in Section II. The network model and problem formulation of the proposed work are discussed in Section III. The proposed MS scheduling scheme for improving reliability and energy efficiency is described in Section IV. The performance evaluation of the proposed work is presented in Section V. Finally, a brief conclusion and future scopes are discussed in Section VI.

II. Related Works

The concept of using MS in tour planning for WSNs is considered in the existing literature. There are two general categories of application scenarios developed by researchers [11], [13]. First, using MS that will move throughout the sensing field to collect information from the sensors, including random mobility pattern and predetermined (controlled) [10], [27]. Second, using MS travels within the sensing field to cover the uncovered area [5], [28]. In this section, we focus on existing methods for reliable data collection to schedule the MS's trajectory.

Tunca *et al.* [9] presented a virtual infrastructure-based MS routing protocol with a random pattern for WSNs, which is called Ring routing. In their work, a number of sensors are selected as ring nodes that save the recent position of the MS and share it with other sensors. The MS determines a neighbor sensor as the anchor which is responsible for sending the position of the sink to the ring nodes. When the current anchor is unavailable, the MS determines a fresh anchor. In the data collection phase, first, the sensor sends a request to the ring nodes. After receiving the recent position of the MS from the ring nodes, it forwards sensed data to the MS by the greedy geographic routing protocol. In Ring routing, when the scale of the network increases, the algorithm forms one ring in each state that causes high traffic to sensors in the vicinity of the ring node and thereby increasing the probability of the hotspot occurrence. Moreover, advertising the MS position and exchanging messages adds up to the overall overhead and cost.

Nested routing [29] creates several virtual rings to overcome the weaknesses of the Ring routing protocol [9]. In this random pattern, the number of rings is regarded as a function of network scale. The proposed algorithm consists of several nested rings, including normal nodes and router nodes. The goal of these nested rings is to reduce the number of routing hops to save and update the latest MS position. When a sensor intends to transmit data to the MS, the nearest router in the nearest ring is responsible for replying with the latest MS position. This method brings high computational complexity due to several virtual rings and is considered impractical and inefficient if the router nodes are not available.

Wang and Chen [15] proposed an efficient path planning algorithm called EARTH for delay-sensitive applications in WSNs. They state that these applications are vulnerable to the energy hole problem due to unbalanced energy consumption. The EARTH algorithm benefits a predetermined mobility pattern for the MS to visit the RPs. However, their work assumes that the sensors have various sensing rates and fixed buffer sizes. The RPs are responsible for gathering information from other sensors, which can effectively overcome the hotspot problem. This algorithm forms an SP tree to join all nodes and find RPs based on the amount of cumulative data and the number of hops to the sink. Then, MS moves periodically across the sensing field to gather data from the RPs and returns to the stationary sink for data delivery. Nevertheless, in the data collection phase, EARTH may encounter the chain problem in subtrees under each RP.

Gutam *et al.* [32] have proposed a tree-based clustering approach to select the optimal MS trajectory planning. Their method first forms a virtual infrastructure based on tree-based clustering to find the optimal RPs and then the MS benefits computational geometric method in order to construct MS trajectory among all RPs. To further reduce energy consumption, they also use RP re-election and virtual RP election. In their work, the authors have constructed the shortest MS path along with the improved network lifetime, but their method neglects buffer overflow management at RPs.

Jain *et al.* [27] proposed a hierarchical virtual infrastructure-based MS routing for query-driven scenarios called QWRP. In query-driven scenarios, the MS imposes queries into the sensing field. In answer to the query, sensors belonging to the sensing field forward their sensed data to the MS. In some IoT applications, query injection and data collection location are constantly changing that can severely affect network performance, including network lifetime and delivery delay. A common method in such cases is to completely distribute the sink position that increases network overhead. The QWRP method tries to minimize the data delivery delay as well as energy consumption by suggesting a query in virtual infrastructure such as wheel shape. Moreover, QWRP has used an angle-based data transmission mechanism in wheels via the shortest path algorithm. Also, this method claims that the energy of the nodes is uniformly consumed and the delay of deliveries to MS is reduced. In practice, this method will encounter packet loss due to not considering buffer management.

Almiáni *et al.* [30] proposed a cluster-based (CB) method to find RPs. In this way, some nodes are randomly assigned as clusters and then each node joins the cluster based on its proximity to one of the clusters. An RP is then selected from each cluster. If the path length used to visit all RPs is shorter than the threshold of a predefined maximum length, given as an input to the algorithm, this method adds more CHs and repeats the steps. Since clusters are selected randomly, some clusters may have more nodes that cause their cluster nodes to consume more energy in communication. In [10], the weighted RP (WRP) selection based on the tree structure is proposed for mobile WSNs. Each weight is a product of the number of packets sent by the node to the RP multiplied by the number of hops between the node and its nearest RP. Thus, the WRP selects the maximum weight as the RP executes the TSP to schedule an appointment with the RPs. Repeat these steps until the path does not exceed a threshold. Although WRP can efficiently reduce path length, it inflexibly recomputes new paths when the sensing rate of sensors is changed.

Redhu and Hegde [20] presented an energy-efficient Q-learning cluster-based (Q-CB) algorithm for optimizing MS scheduling while considering dynamic buffer management. Due to the benefits of the clustering technique, a significant improvement in network lifespan is observed. In Q-CB, MS collects sensor data sequentially from all clusters. The reinforcement learning framework for intelligent modeling of MS trajectory is implemented based on the decision to move to the next cluster or stay in the current cluster. This article operates by minimizing data loss based on learning how to prevent buffer overflow. The performance evaluation of Q-CB shows that MS is able to learn network behavior after more rounds for a fixed number of nodes. Note that with increasing network size, this period of learning will increase, which brings performance challenges, including both

TABLE I
COMPARISON OF SOME PREVIOUS WORKS

Protocol	Sink mobility pattern	Multi-sink support	Delivery Delay	Energy Efficacy	Virtual infrastructure type	Computation time/overhead	Buffer management	Sensing Rate	Adaptively Compute Path	Possibility of hotspot
CB [30]	Predetermined	N/A	High	Low	Simple Clustering	High	No	Same rate	Inflexible	Low
WRP [10]	Predetermined	No	Low	High	Weighted RPs in Trees	Very High	No	Same rate	Inflexible	High
Ring [9]	Random	No	Medium	Medium	Virtual Ring	Medium	No	Same rate	Inflexible	Medium
Nested [29]	Random	Yes	Low	High	Several Nested Rings	High	No	Same rate	Inflexible	Low
QWRP [27]	Random	No	Low	High	Angle based Forwarding	High	No	Same rate	Inflexible	Medium
EARTH [15]	Predetermined	No	Medium	Medium	Shortest paths in Trees	Low	Medium	Diverse rate	High	High
Q-CB [20]	Random	N/A	Medium	High	Q-learning in Clusters	High	Low	Diverse rate	Low	Low
DGOB [31]	Predetermined	N/A	Medium	Medium	Artificial intelligence in Clusters	Medium	No	Same rate	Inflexible	Medium
ORPSTC [32]	Predetermined	Yes	Low	High	Minimum spanning tree-based clustering	Medium	No	Same rate	Inflexible	Low
HACDC [33]	Predetermined	No	Medium	High	Unsupervised learning-based clustering	Low	No	Same rate	Inflexible	Low
ACO-MSPD [34]	Predetermined	No	Medium	Medium	Ant Colony for directed spanning tree	Low	Medium	Diverse rate	Medium	Low
eACO-MSPD [35]	Predetermined	No	Low	High	Ant Colony for directed spanning tree	Low	Medium	Diverse rate	Medium	Low
Tuft [36]	Random	No	Low	High	Cells in Trees	Medium	No	Same rate	Inflexible	Low
EE-MSWSN	Predetermined	Yes	Very Low	Very High	Clustered tree-based structures	Medium	High	Diverse rate	Medium	Very Low

high data delivery delay and high computational overhead. However, depending on growing the number of sensors, measuring machine learning states for sensors leads to complicate MS scheduling, thereby reducing WSN lifetime.

Najjar-Ghabel *et al.* [31] proposed a data gathering algorithm in the face of obstacles (namely, DGOB) for mobile WSN. DGOB includes the two main stages of cluster construction (first phase) and tour planning (second phase). In the first phase, it uses the obstacle-aware cluster construction (OCC) and obstacle-aware cluster updating (OCU) methods. The OCC method benefits the ant colony algorithm to select the clusters. Further, the genetic algorithm is used to update the clusters selected by the OCC. In the second phase, the obstacle-aware MS tour construction (OMTC) method is used to build reliable path planning. Their work tries to reduce the effects of obstacles in the process of data collection by artificial intelligence techniques. In order to use artificial intelligence methods, Donta *et al.* [33] have used unsupervised learning-based clustering strategies in 3-D WSNs. Their work has led to the optimal RP selection along with reduced complexity. However, its approach best suites for 3-D WSNs, not for the 2-D WSNs.

Busaileh *et al.* [36] present Tuft, a new tree-based structure with a random mobility pattern that is able to avoid overhead costs by providing a new sink location in the sensing field while keeping a uniform dispersion of concentrated data traffic. Tuft leverages the sink mobility to its advantage to decrease energy consumption on communication. Tuft with a spanning tree covers the entire network, and then each tree node is composed of a cell. Hence, each cell is a group of sensor nodes belonging to a specific area. Advertising the sink position is achieved by traveling MS through this structure. Furthermore, the top tier nodes in the spanning tree change periodically according to the sink position which brings low overhead costs and a load balance of both energy efficiency and the concentrated data traffic at the top tier nodes. The sink node selects an access node to operate as its deputy, and the source nodes obtain the position of the access node with easy access to the spanning tree. In large-scale scenarios, the length of MS's trajectory may increase by varying sensors count. Accordingly, the computational overhead becomes much larger by varying Tuft's cell numbers.

Kumar *et al.* [34] have proposed an ant colony optimization (ACO)-based MS trajectory planning to find the shortest MS tour for minimizing data gathering delay under nonuniform data constraints. In addition, Donta *et al.* [35] have introduced an extended ACO-based MS trajectory planning to increase further network performance for event-driven WSNs. The ACO metaheuristic algorithm used in [34] has also proposed a new probability function to find the optimal RPs in [35]. It has been effective in finding the shortest MS tour and preventing data delivery delay, but these methods are not suited for real-world applications, considering that sensors have finite buffer space.

We have summarized the characteristics and performance of the related works in Table I. Although all these works have addressed tour planning, their objectives were limited to either reducing energy consumption or packet losses. Our work addresses designing energy-efficient hierarchical protocol while providing a reliable data gathering process by considering efficient buffer management, which has been neglected in the literature. The present work obtains less length of MS traveling path to prolong sensors' lifetime on communications, prevent packet losses at RPs due to buffer overflow, and aid the MS to collect RPs data efficiently.

III. PROBLEM DEFINITION

We first present the network architecture, followed by the radio-energy dissipation model. Then, we depict the tour planning problem and formulation.

A. Network Model and Assumptions

We describe the network model by the stationary sink S_∞ and a set of N sensors S_N, where $S_N = \{s_1, \ldots, s_N\}$. Sensors are homogeneous and deployed randomly using a uniform distribution in the sensing field. They have identical initial battery power, limited buffer capacity, and communication range. Additionally, each sensor s_i has a sensing rate sr_i (number of packets generated per unit time) and it can store at most $\Im$ packets of sensing data in its buffer. A node $s_i \in S_N$ has coordinates (x_i, y_i) and the distance between two sensors s_i and s_j is calculated by the Euclidean distance $\hat{D}(s_i, s_j)$. At the initialization phase, the sink S_∞ gathers the information of all nodes in the network to perform the construction of the virtual structure by distributing different designated roles for sensor nodes, which is done during *the bootstrapping phase* [26] and will be explained in the next section. Then, there is an MS

TABLE II
SUMMARY OF COMMON NOTATIONS

Notation	Description
S_N	Set of sensor nodes, where $\|S_N\| = N$
S_∞	The sink for offloading data by MS
$\Im$	Buffer capacity
Υ	Solution set of RPs
$\mathbb{C}$	Set of RP candidates
Θ	Set to save the RP candidates status
O	A set for saving new RP candidate after tree pruning
ST_j	Set of sensors that transmit their packets to sensor s_j
ω	Height of each tree
δ	Chain threshold
ℓ	A set for checking sensor nodes
Δ	Set of distance to the sink of all nodes
B_i	Packet count generated by sensor s_i in a round
acc_i	Amount of relay data
$V\hat{R}_i$	Voronoi resign for generator RP_i in the sensing field
$\hat{H}(s_i, s_j)$	Hop count between two nodes
$\hat{D}(s_i, s_j)$	Euclidean distance between two nodes
cr_i	Communication range of sensor s_i
sr_i	Sensing rate of sensor s_i
VR	Voronoi Region

node with a constant speed of ν that periodically selects a tour throughout the network to collect data from RPs. Finally, the MS has to come back to S_∞ to offload the collected data. The most used notations are listed in Table II.

We outline some of our assumptions based on [10], [15], and [36]. First, the sensors are aware of their geographic position since they are employing a GPS device or utilizing a localization method such as [37], and thus they are aware of their neighbors' position using neighbor discovery techniques, such as [38]. Hence, the MS can be informed of sensor nodes by exchanging control messages between sensors and the MS when it is moving in the sensing field. Second, unlike the MS's traveling time, the MS's communication time with sensors is insignificant, so there is no need to consider the sojourn time of the MS at RPs. Third, all data are collected by RPs from non-RPs before the MS arrives, so the MS's traveling time merely depends on the length of the MS traveling path. Fourth, the MS travels with a constant velocity ν in the connected WSN. Finally, based on [15], we also ignore the effect of two additional communication overheads for the sensors, including 1) RP-announcement (S_∞ has to announce sensors of the elected RPs set) and 2) rate-adjustment (changing sr_i of sensors based on the requirement of applications by S_∞ messages), because these overhead messages are unavoidable in any tour selection solution. Instead, we compute sensors' energy consumption for sending/receiving the measured data due to the RP selection by variant solutions.

B. Energy Model

In sensor networks, the destination of all sensory data is forwarded to the sink by one or multiple hops (through relay nodes). According to the first-order radio model [39], the amount of power spent to send and receive κ bits of data packet from sensor s_i to sensor s_j with the Euclidean physical distance $\hat{D}(s_i, s_j)$ is calculated by (1) and (2), respectively, where ε_{efs} (multipath fading channel modeling), ε_{amp} (energy consumed in the free space by the amplifier), and $\varepsilon_{\text{elec}}$ (energy of transmitter electronics) have constant amounts [40]

$$Tx_{(i,j,\kappa)} = \begin{cases} (\kappa.\varepsilon_{\text{elec}}) + \left(\kappa.\varepsilon_{efs}.\hat{D}_{i,j}^2\right), & \hat{D}_{i,j} < \sqrt{\frac{\varepsilon_{efs}}{\varepsilon_{\text{amp}}}} \\ (\kappa.\varepsilon_{\text{elec}}) + \left(\kappa.\varepsilon_{\text{amp}} \cdot \hat{D}_{i,j}^4\right), & \hat{D}_{i,j} \geq \sqrt{\frac{\varepsilon_{efs}}{\varepsilon_{\text{amp}}}} \end{cases} \quad (1)$$

$$Rx_{(j,\kappa)} = \kappa \cdot \varepsilon_{\text{elec}}. \quad (2)$$

Suppose $\mathbb{P}_{(s_s,s_d)}$ denotes a path including one or more hops between sensor source s_s and sensor destination s_d. Total energy spent by traveling along $\mathbb{P}_{(s_s,s_d)}$ can be expressed in (3), where $\hat{H}$ is the relay hop count between two nodes

$$E_{\text{Path}} = \sum_{\forall i,j \in \mathbb{P}}^{\hat{H}} E_{\text{hop}(i,j)}. \quad (3)$$

Herein, the energy consumption in each hop (E_{hop}) of E_{Path} to send and receive κ bits of data packet is defined in

$$E_{\text{hop}(i,j)} = Rx_{(j,\kappa)} + Tx_{(i,j,\kappa)} = \begin{cases} (2\kappa.\varepsilon_{\text{elec}}) + \left(\kappa.\varepsilon_{efs}.\hat{D}_{i,j}^2\right), & \hat{D}_{i,j} < \sqrt{\frac{\varepsilon_{efs}}{\varepsilon_{\text{amp}}}} \\ (2\kappa.\varepsilon_{\text{elec}}) + \left(\kappa.\varepsilon_{\text{amp}} \cdot \hat{D}_{i,j}^4\right), & \hat{D}_{i,j} \geq \sqrt{\frac{\varepsilon_{efs}}{\varepsilon_{\text{amp}}}}. \end{cases} \quad (4)$$

C. Problem Formulation

In this section, we formulate the tour planning problem as an optimization problem into a mixed-integer programming (MIP). Given S_∞ and S_N, tour planning seeks to pick a set of RPs Υ (a proper subset) from a set S_N and decides to form a traveling path for the MS starting from S_∞, then visits each node of solution set Υ once and finally revert to S_∞, in order to 1) optimize the efficient MS trajectory scheduling (reduce path length); 2) minimize the energy consumption of sensors; and 3) prevent the occurrence of buffer overflow at RPs. Note that to satisfy the energy efficiency objective, non-RPs in set S_N need to send their data packets to the nearby RPs. Fig. 1 demonstrates an example of an RP-based scheme, where each node generates one data packet in the sensing field. As shown in Fig. 1(a), to reduce the length of MS traveling path while saving energy in communications, sensors s_4, s_6, and s_{10} are considered as RPs. For data gathering, in each round, the MS moves from S_∞, and it visits RPs once and goes back to S_∞. We assume that each sensor can store five packets in its cache. In Fig. 1(b), sensor s_8 produces three packets due to the occurrence of the abnormal events in the sensing field and its buffer capacity reaches four packets (accumulating with one packet of s_7). In this case, RP s_{10} has to discard one packet with the original path. On the other hand, due to the long distance between non-RPs and their RP (chain problem), the hotspot problem may re-emerge as the subtree of RP s_4 in Fig. 1(b). Hence, we can change the MS traveling path from $S_\infty \rightarrow s_4 \rightarrow s_6 \rightarrow s_{10} \rightarrow S_\infty$ to $S_\infty \rightarrow s_3 \rightarrow s_6 \rightarrow s_{10} \rightarrow s_8 \rightarrow S_\infty$ by increasing the length of the original traveling path in order to avoid packet losses at RP s_{10}, and reduce the effect of the hotspot on the top-tier s_3 of subtree at RP s_4 as seen in Fig. 1(c).

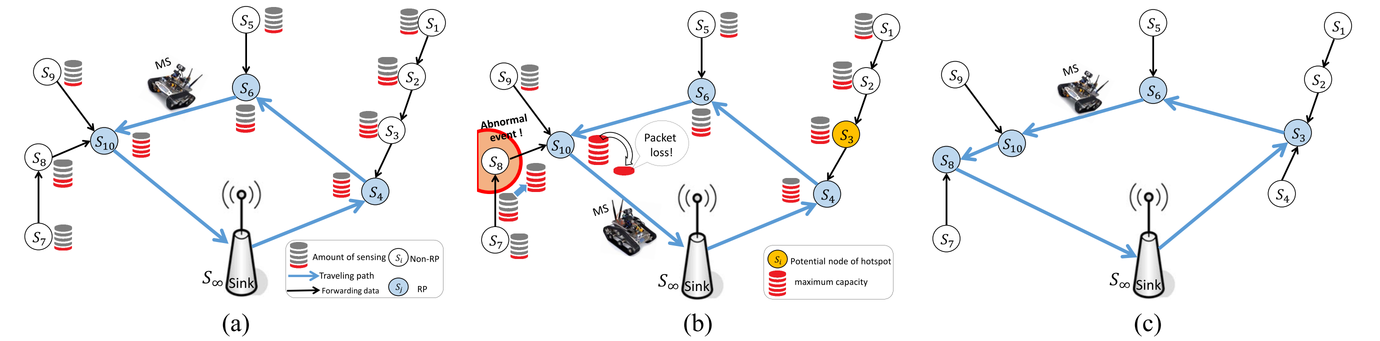


Fig. 1. Illustration of the problem statement with $\Im = 5$: (a) results after RP selection, where all sensors generate one packet; (b) due to the occurrence of the abnormal event surrounding sensor s_8, this sensor generates three packets, thereby RP s_{10} will drop one packet based on its caching capacity. Besides, the possibility of hotspot at s_3 is high due to the long distance between nodes of s_1 and s_4; (c) greedy solution selects both s_3 and s_8 as RPs, which leads to longer path for MS.

Motivated by the above observation, the shortest tour length L_{obj} required by the MS to reduce packet losses is mathematically formulated by defining binary decision variables as follows.

1) $\psi_{i,j} \in \{0, 1\}$, one means that the tour includes link(i, j).
2) $\Upsilon_i \in \{0, 1\}$, one means that the tour includes sensor s_i as RP.
3) $\Gamma_{i,j} \in \{0, 1\}$, one means that a feasible flow routes on link(i, j).

Obtaining three variables, becoming the near-optimal/optimal solutions to our problem, where the following objective function is satisfied:

$$L_{\text{obj}} = \min \left\{ \sum_{i=0}^{N} \sum_{j=0}^{N} \psi_{i,j} \hat{D}(s_i, s_j) \right\} \tag{5}$$

$$\text{subject to} \quad \sum_{j \in L_{\Im}(s_i)} \Upsilon_i \geq 1 \tag{6}$$

$$T = T_{\text{sojourn}} + T_{\text{moving}} \quad (T_{\text{sojourn}} = 0) \tag{7}$$

$$L_{\Im}(s_i) \geq \lfloor sr_i \times T_{\max} \rfloor \tag{8}$$

$$\hat{D}(MS, s_i) \leq cr_i \ \ (\forall s_i \in S_N) \tag{9}$$

$$\sum_{j=0} \psi_{i,j} = \Upsilon_i, \quad \sum_{i=0} \psi_{i,j} = \Upsilon_j \tag{10}$$

$$\sum_{i=1}^{N} \Gamma_{j,i} - \sum_{k=1}^{N} \Gamma_{k,j} = \Upsilon_j, \quad \sum_{j=1}^{N} \Gamma_{j,0} = \sum_{j=1}^{N} \Upsilon_j \tag{11}$$

$$\Gamma_{i,j} \leq |S_N| \psi_{i,j}. \tag{12}$$

Herein, $i \in \{0, 1, \ldots, N\}$, $j \in \{0, 1, \ldots, N\}$, while index 0 is referred to S_∞. In the above formulation, the objective function (5) minimizes the tour length. Constraint (6) ensures that each sensor is covered by at least one RP node based on buffer limitation $\Im$. Let T denote the overall time needed for MS's traveling path as expressed in Constraint (7). Moreover, let T_{sojourn} as the time interval that MS needs to visits each RP, where we assume $T_{\text{sojourn}} = 0$. Let T_{moving} show the time interval that MS is moving and collecting data. Note that each sensor produces at most $B_i = \lfloor sr_i \times T_{\max} \rfloor$ packets in a round, where B_i is smaller than $\Im$; otherwise, buffer overflow must occur at some sensors, where Constraint (8) denotes these condition. To determine round time $T_{\max}$ based on [10] and [15], we first set the maximum path length $L_{\max}$ for MS can travels, where $L_{\max}$ is a given input of algorithm. Then, if MS moves at a constant speed ν, the default value of T can be set to $L_{\max}/\nu$. The communication constraint is satisfied by (9) that restricts the MS to pass through the communication range cr$_i$ of each sensor for data gathering. Constraints in (10) ensure that MS collaborates with each selected RP once per tour. Additionally, there is an input and output flow for each selected RP, while the output flow is one unit more than the input flow. Constraint (11) specifies that the input flow to S_∞ is equal to the number of selected RP nodes in the tour. Finally, Constraint (12) states that link(i, j) can only have flow if it is included in the tour.

IV. EE-MSWSN Algorithm

We propose EE-MSWSN, a novel hierarchical scheme for MS-based data collecting in WSNs. The main idea of EE-MSWSN is to construct a virtual backbone composes of clustered tree-based structures in the entire network and use it to rapidly find potential RPs based on the distance to the sink, hop count, and the amount of collected data in each clustered tree. In other words, these structures encapsulate a group of sensors contingent on the buffer and hop-count constraints. We first elect RP candidates by referring to sensors' performance parameters. By starting to partition the sensing field based on the distance of sensors to RP candidates, we build an SP tree rooted at their RP candidate in each partition. Afterward, this procedure is repeated until each sensor identifies its RPs. Finally, the desired path for MS traveling is created by connecting the finalized RPs.

As mentioned earlier, the EE-MSWSN protocol leverages a combination of both tree-based and clustering techniques to take advantages of both approaches. Fig. 2 illustrates an example of the experimental result of these techniques for energy consumption while varying the number of rounds, where a fixed number of IoT devices with unlimited buffer size is used to efficiently gather data [5], [22]. In this snapshot, we observe that the energy consumption reported by these techniques is quite intersected at $[R(t), E(R)]$ point. By increasing the number of rounds, the tree-based technique shows a steeper slope than the clustering technique, experiencing more

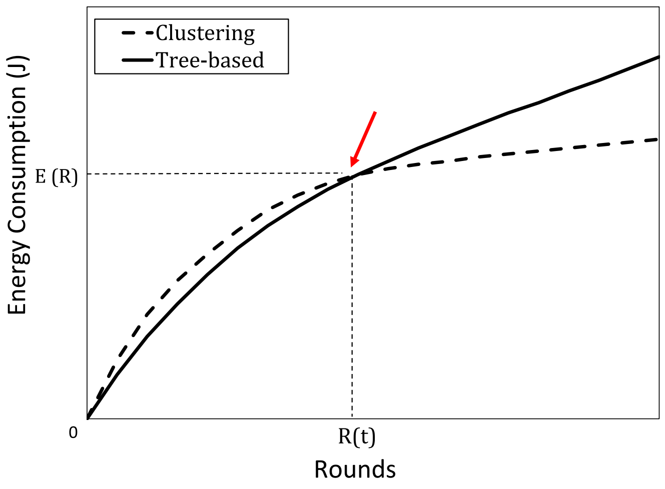


Fig. 2. Varying the number of rounds over energy consumption with a fixed number of nodes and unlimited buffer size.

energy consumption. One of the reasons is that in large-scale networks, the clustering technique requires the less number of hops compared to the tree-based technique to forward data to the sink, resulting in lower energy consumption. Further, the probability of occurrence of hotspots in the tree-based technique is higher. Besides, the clustering technique suffers from high computational cost in order to construct clusters. Note that the above outcomes have been observed in several works in [24] and [41]. This is the main reason for employing clustered tree-based structures in the backbone of EE-MSWSN to design a hybrid scheme, which is rarely studied in similar works. Algorithm 1 further enumerates EE-MSWSN's orders for RPs selection and MS scheduling for data collection.

The details of the EE-MSWSN algorithm can be described as follows, which contains three phases.

1) *Virtual Infrastructure Construction:* We first construct a virtual backbone to connect all nodes and elect RP candidates, which includes three steps: a) candidate RPs finding; b) Voronoi area generation; and c) tree construction.
2) *RP Selection:* We pick a solution set of RPs from set $\mathbb{C}$ based on buffer limitation and try to minimize MS scheduling cost, which includes two steps: a) network refinement and b) Voronoi regions (VRs) consolidation.
3) *Tour Planning:* We finally employ a local-search TSP scheme to find the shortest traveling path and try to further reduce the tour length, which includes two steps: a) RPs replacement and b) MS scheduling.

A. Phase 1—Virtual Infrastructure Construction

In this section, we construct clustered trees rooted at their RPs, which can be done by the Voronoi tessellation and Dijkstra's algorithm. After deploying the network with respect to *the bootstrapping process*, the information of sensors is collated in S_∞. The MS gathers the network information and returns to S_∞ to upload these data. The monitoring center (located either at the sink node or a remote controller) makes the decision to tour planning for MS and performs role allocation to sensors in a round-by-round manner [26], [36]. Fig. 3 gives an example to show EE-MSWSN's run processes in the connected WSN with an MS, where $\Im = 10$, and the number nearby to each sensor's circle means the number of packets that were generated B_i. The details of this figure will be completed over different processes of EE-MSWSN.

Algorithm 1 Algorithm for RPs Selection and MS Scheduling for Data Collection

```
Input: The sensing field information
Output: The collection path
1:  procedure EE-MSWSN
2:      The Bootstrapping Process
3:      Candidate RPs Finding                    ▷ Algorithm 2
4:      Voronoi Area Generation                  ▷ Algorithm 3
5:      Tree Construction                        ▷ Algorithm 4
6:      if $\ell$ in each VR is empty? then
7:          if O in each VR is empty? then
8:              if Check all nodes in $\mathbb{C}$? then
9:                  Mobile Sink Scheduling
10:                 Return Path and Go to Next Cycle
11:             else
12:                 RPs Replacement              ▷ Algorithm 9
13:                 Go to Line 8
14:             end if
15:         else
16:             VRs Consolidation                ▷ Algorithm 8
17:             Go to Line 7
18:         end if
19:     else
20:         Network Refinement                   ▷ Algorithm 5
21:         Go to Line 6
22:     end if
23: end procedure
```

1) Candidate RPs Finding: S_∞ elects RP candidates from its adjacency nodes in the sensing field. This procedure is done by considering five metrics, including distance, power, latency, packet queue, and sensors' capacity. In general, an RP (e.g., CH selection) is selected based on the metrics, including thresholds of distance, power, and latency. However, EE-MSWSN is developed especially for reliable data gathering mechanism to adaptively electing RPs based on limited buffer capacity and varied sensing rate. Thus, in order to improve efficiency, two further metrics, such as packet queue and sensors' capacity, are considered. The objective function of RP candidate selection is obtained using (13)–(15), where the distance metric is the Euclidean distance, and coefficients (α, β) are constant values

$$F_{\text{obj1}} = f_{\text{power}}\big(1/f_{\text{capacity}}\big) + f_{\text{power}}\big(1/f_{\text{queuing}}\big) \tag{13}$$
$$F_{\text{obj2}} = \alpha(1/f_{\text{distance}}) + (1-\beta)F_{\text{obj1}} \tag{14}$$
$$F_{\text{obj3}} = \alpha\big(F_{\text{obj2}}\big) + (1-\alpha)\big(1/f_{\text{latency}}\big). \tag{15}$$

Note that these functions have been successfully tested in several works, such as in [14], and we are inspired by these works. Algorithm 2 further indicates the initial RP selection stage, where in Fig. 3(a), the resulting network of this stage is illustrated. As opposed to many studies [1], another advantage

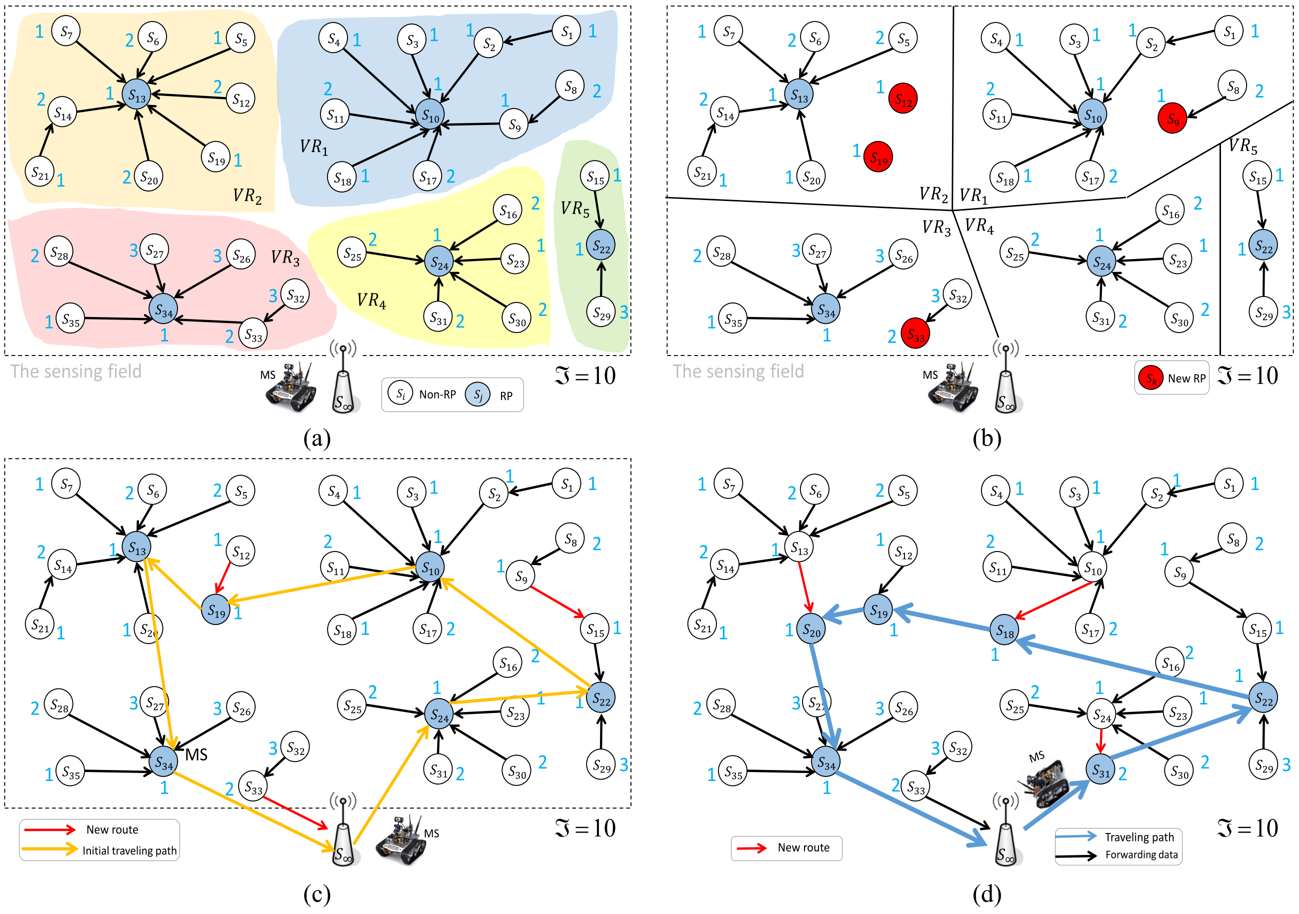


Fig. 3. Example of data gathering steps in EE-MSWSN: (a) results after the virtual infrastructure construction phase, including candidate finding, VRs production, and tree construction; (b) pruning trees based on buffer limitation under the network refinement stage; (c) initial traveling tour for the MS is formed by connecting RP candidates after the VRs consolidation stage; and (d) results after the tour planning phase with $\delta = 3$, where the desired tour for the MS is constructed under RPs replacement.

Algorithm 2 RP Candidate Selection

```
Input: $S_N$, the sensing field information
Output: $\mathbb{C}$, $\Delta$
 1: procedure RPCANDIDATE
 2:     Get all sensors $S_N$ and assign a false flag to them
 3:     Obtain $S_\infty$ position
 4:     for each sensor $s_i \in S_N$ in the sensing field do
 5:         Compute objectives using (13), (14), (15)
 6:         Get $\hat{D}(s_i, S_\infty)$ and add them to set $\Delta$
 7:     end for
 8:     Select RP candidates and add them to set $\mathbb{C}$
 9:     Inform RP positions and assign a true flag to RPs
10: end procedure
```

of EE-MSWSN is that the RP selection in a diverse number helps to increase the scalability for different network sizes. As illustrated in Algorithm 2, at the first phase, the system will elect RP candidates, then add them to a set $\mathbb{C}$ as the candidate set, further S_∞ broadcasts an identification message of RPs. As illustrated in Algorithm 2, let us denote by Δ the set of distance to the sink of all nodes is derived the Euclidean distance, where each member of Δ is a two-tuple $(s_k, \hat{D}_{(s_k,S_\infty)})$. Afterward, all sensors of S_N are attached with a checked flag. A "true" flag shows the RP candidate, while a "false" flag shows normal nodes.

2) Voronoi Area Generation: In 2-D mathematics, a Voronoi diagram is a division of the surface (e.g., the sensing field) into convex regions based on the distance to a set of points (e.g., $\mathbb{C}$). Each RP candidate position $RP_{(x,y)}$ is used as a Voronoi generator, and each VR contains only one RP candidate. The sensors in VR are closer to the RP candidate than any other sensors. To do so, given a set $\mathbb{C}$ belonging to a set S_N, the VR_i corresponding to $RP_i \in \mathbb{C}$ is defined by

$$\widehat{\text{VR}}_i = \Big\{ s \in S_N | \hat{D}(s, RP_i) < \hat{D}\big(s, RP_j\big) \quad \text{for } j \in \{1, \ldots, |\mathbb{C}|\}, j \neq i \Big\}. \tag{16}$$

Herein, a node $s \in S_N$ has coordinates (x, y) in the sensing field. By applying the above equation, this stage is repeated until each sensor in S_N is covered by VRs encapsulating a cluster of sensors surrounded by RP candidates as seen in Fig. 3(a). To reduce the cost of computing distances, we use the divide and conquer algorithm to construct Voronoi polygons. Moreover, to minimize the cost of scheduling between VRs and thereby reduce the MS' traveling tour, we seek to consolidate VRs. Further detail on VRs' consolidation is

Algorithm 3 VRs Generation

```
Input: $\mathbb{C}$, $S_N$
Output: VRs
1: procedure VRsGENERATION
2:     Get the positions of RP candidates in $\mathbb{C}$
3:     Get the positions of sensors in $S_N$
4:     for each $RP_i \in \mathbb{C}$ do              ▷ As seen in Figure 3(a)
5:         for each $s \in S_N$ do
6:             Compute Cell $VR_i$ by equation (16)
7:         end for
8:     end for
9:     Generate Voronoi diagram by the Divide and Conquer
10: end procedure
```

Algorithm 4 SP Tree Construction

```
1: Input: VRs information, set $\mathbb{C}$
2: Output: SP trees, set $\ell$
3: procedure TREECONSTRUCTION
4:     Get VRs nodes in each VR
5:     Define a set $\ell$ for pruning trees
6:     for each $VR_i$ do              ▷ As seen in Figure 3(a)
7:         Generate SP rooted at $RP_i$ by Dijkstra's Algorithm
8:         Inform SPs and assign new roles (child and parent)
9:         Add the leaf nodes of each SP to set $\ell$
10:    end for
11: end procedure
```

explained in Phase 2. Algorithm 3 illustrates the Voronoi area generation stage, where in Fig. 3(a), the partitioning of the sensing field into VRs is evident. We use the Voronoi diagram for backbone construction due to two reasons. First, the traffic load can be dispersed to achieve a superior balance in terms of energy consumption, thereby diminishing the energy hole problem in the whole network. Second, to enhance the coverage ratio in the presence of coverage holes over the sensing field, the use of the Voronoi diagram has been successfully tested in many studies [28], [42].

3) Tree Construction: After VRs' generation, we construct SP trees rooted at RP candidates in each VR, which can be done by Dijkstra's algorithm. In fact, since each VR contains only one RP candidate, all the sensory data transmission paths of each VR node will form a forest graph, where each SP tree has its roots in the RP candidate. Algorithm 4 further illustrates the tree construction stage, and Fig. 3(a) depicts the SP tree of each VR. Afterward, each VR including an SP tree will be checked via a bottom-up way. To do so, we define a set ℓ to check the sensors, where each element of set ℓ is a two-tuple (s_i,VR$_i$). Hence, those elements must satisfy one of two conditions: 1) being from the leaf nodes of an SP tree in each VR or 2) each child of s_i's having a true flag. So far, ℓ includes only leaf nodes in stage 1, as the flags of other nodes are false "0". Additionally, we assume that S_∞ has a true flag and must include in the solution set of RPs Υ, thereby we have $\Upsilon = \{S_\infty\}$ in the beginning.

Thus, we use an SP tree in our backbone due to two reasons. First, using the SP tree helps the sensors transmit data to the

Algorithm 5 Network Refinement

```
1: Input: SP trees, VRs, $\Im$, $\ell$
2: Output: Update sets $\Theta$, $\mathbb{C}$, and O
3: procedure NETREFINEMENT
4:     Define set $\Theta$ for RP candidate information
5:     Get all members in $\ell$
6:     for each $VR_i$ do              ▷ Using bottom-up fashion
7:         traverse tree by starting from a node $s_i$ in $\ell$
8:         if $acc_j! = \Im$ then              ▷ By (17)
9:             if $s_j$ has a maximum of one child then
10:                STATEI()              ▷ Algorithm 6
11:            end if
12:            if $s_j$ has more than one child then
13:                STATEII()              ▷ Algorithm 7
14:            end if
15:        end if
16:        Add $s_j$ to sets $\mathbb{C}$ and O, and its subtree to set $\Theta$
17:    end for
18: end procedure
```

sink with the minimum number of hops, and thereby saving their energy on communication. Second, since the SP tree is used in a number of TSP-based schemes, it can help to further reduce path length by determining the relationship between the ancestor node and descendant node in each subtree belonging to its RP.

B. Phase 2—RP Selection

After the virtual structure construction phase, since each SP tree keeps the shortest path between normal nodes and each RP candidate, there may encounter buffer overflow in the data aggregation process due to disparate sensing rates and finite buffer capacity of sensors. Thus, the network refinement stage will compensate SP trees in order to avert packet loss and satisfy our reliability objective in the EE-MSWSN algorithm.

1) Network Refinement: Starting from a node in set ℓ, we move toward its RP and accumulate the number of packets produced by the visited nodes (traversal bottom-up manner), which is denoted by acc$_j$. When we visit a node s_j such that acc$_j$ $= \Im$ (i.e., achieve maximum buffer size), s_j will need refinement. In other words, s_j along with its subtrees will be a new SP tree rooted at new RP candidate s_j. Thus, s_j is added to set $\mathbb{C}$. Algorithm 5 further illustrates the network refinement stage, where in the case of Fig. 3(b) will proceed to the refinement of the network under the constraint of buffer size.

As illustrated in Algorithm 5, we define a set Θ to save the RP candidate information, where each element of Θ is a two-tuple (s_j, ST_j). Let ST_j represent the set of all sensors checked by the above procedure (except s_j). In fact, STj demonstrates the subtree rooted at s_j under the constraint of the amount of relay data using (17). Also, we define a set O to save new RP candidate, where each element of O is a two-tuple (s_j, $s_j \in$ VR$_k$). The next stage is based on this set. Thus, we define acc$_j$ to check buffer size based on the above procedure as follows:

$$\text{acc}_j = B_j + \sum_{s_i \in \text{ST}_j} B_i. \tag{17}$$

Algorithm 6 First Special Case in Pruning Trees

```
1: Input: SP trees, $\Im$, $\ell$
2: Output: Update sets $\Theta$, $\mathbb{C}$, and O
3: procedure STATEI
4:     if $acc_j < \Im$ then
5:         Check parent of $s_i$
6:         if $acc_j \geq \Im$ or its parent has a true flag then
7:             $s_j$ will be new RP candidate
8:             Add $s_j$ to sets $\mathbb{C}$ and O and its subtree to $\Theta$
9:         end if
10:    end if
11: end procedure
```

Algorithm 7 Second Special Case in Pruning Trees

```
1: Input: SP trees, $\Im$, $\ell$
2: Output: Update sets $\Theta$, $\mathbb{C}$, and O
3: procedure STATEII
4:     Use DFS to traverse subtress
5:     if the operation terminates in any descending child of $s_j$ then
6:         $s_j$ will be new RP candidate
7:         Add $s_j$ to sets $\mathbb{C}$ and O, and its subtree to $\Theta$
8:     end if
9:     if checking all descendant children but $acc_j < \Im$ then
10:        STATEI()
11:    end if
12:    if each child of $s_j$ has $acc_j \geq \Im$ then
13:        Stop at $s_j$
14:    end if
15: end procedure
```

When encountering a node s_j, two states may happen. *State I:* s_j has a maximum of one child. If $\text{acc}_j < \Im$, we visit s_j's parent. As seen in Algorithm 6, when either its parent holds a true flag or $\text{acc}_j = \Im$, we elect s_j as an RP candidate. Thus, we add s_j to sets $\mathbb{C}$ and O, and its subtree to set Θ.

State II: s_j has more than one child. We utilize the depth-first search (DFS) algorithm to traverse and check subtrees of s_j. As seen in Algorithm 7, three substates are created. 1) if the algorithm terminates in any descending child of s_j (with a false flag) due to $\text{acc}_j \geq \Im$, we elect s_j as an RP candidate; 2) if any acc_j's child of s_j attains $\Im$, we terminate the algorithm at s_j; and 3) when we checked all the descending children of s_j but still $\text{acc}_j < \Im$, we continue the procedure using Algorithm 6.

After running the above algorithms, the new roles (i.e., new RP and etc.) toward some nodes may be defined by pruning trees due to buffer limitation. These nodes in each VR lead to maximizing the MS scheduling cost. In other words, when the number of RP increases, it means that the length of the MS traveling path is becoming larger. To reduce the effect of this problem, we try to consolidate VRs.

2) VRs Consolidation: This section is used to check the elements of set O that are generated during the network refinement stage. In other words, the VRs consolidation stage is not performed if set O is empty (null). We use VRs consolidation to find the minimum cost of scheduling between VRs. To do

Algorithm 8 VRs Consolidation

```
1: Input: VRs, $\Theta$, $\mathbb{C}$, O, $S_N$, $\Delta$
2: Output: Pushing data
3: Obtain Table VR by Floyd algorithm
4: procedure VRSCONSOLIDATION(O)                          ▷ by Table VRs
5:     Find $VR_{max}$ and $VR_{min}$                            ▷ By (18)
6:     if $VR_{min}$ is neighbor of $VR_{max}$ then
7:         Try to tree construction between elements of O $\in$ $VR_{max}$ and $VR_{min}$
8:         Perform data forwarding from $VR_{max}$ to $VR_{min}$
9:         if $acc_k < \Im$ then
10:            Push the scheduling method for MS's stack
11:            Update sets
12:        else                                            ▷ no consolidation
13:            Try to tree construction rooted at $s_k \in$ O with minimum $\Delta$ in $VR_{max}$
14:            Update sets
15:        end if
16:    end if
17:    if Not finding a neighbor then                      ▷ no consolidation
18:        Try to tree construction in each VR for set O
19:        Update sets
20:    end if
21:    if Sink is neighbor to VRs then                     ▷ by a true flag
22:        Push data from set O to $S_\infty$ as final RP
23:        Update sets
24:    end if
25: end procedure
```

so, we assume one VR to be the entire sensing field as a basic unit of scheduling. Further, we compute the mean allocation number of Mean_S using

$$\text{Mean}_S = (\text{Sensors Count})/(\text{VRs Count}). \quad (18)$$

Algorithm 8 further illustrates the VRs consolidation stage, where in the case of Fig. 3(c), it depicts this consolidation.

Inspired by the idea of mean allocation, the VR whose sensor count is higher than Mean_S (called VR_{max}) assigns sensor data to other VRs whose sensor count is less than Mean_S (called VR_{min}). In other words, we aim to minimize the number of RPs by assigning sensor data of the elements of O from VR_{max} to VR_{min}. That is why we use equation Mean_S in this section. As illustrated in Algorithm 8, we first find VR_{max} among VRs and then obtain both the smallest and the nearest VR_i as VR_{min} in the table of adjacency matrix between VRs using the Floyd algorithm, which means that we can assign a scheduling method from VR_{max} to VR_{min}. After connecting the elements of O in VR_{max} to an SP tree of VR_{min}, we construct an SP tree rooted at the RP candidate of VR_{min}. These connected nodes are removed from O, and sets $\mathbb{C}$ and Θ are updated. Besides, if we cannot consolidate any element of O to other VRs, we will try to construct a new SP tree rooted at $s_k \in \text{O}$ with minimum Δ in their VR. Afterward, s_k is removed from set O, and then this RP and their subtree are included in set $\mathbb{C}$ and Θ, respectively. Additionally, the elements of set O located in neighboring of S_∞ can forward their data to the sink. Thus, these elements and their subtrees are removed

Algorithm 9 RPs Replacement

```
1: **Input:** $\Theta$, $\mathbb{C}$, $\delta$, $\Delta$
2: **Output:** Update sets of $\Theta$ and $\mathbb{C}$
3: **procedure** REPLACEMENT
4:     Define path=ancestor(parent)–descendant(child)
5:     **for** each $RP_i \in \mathbb{C}$ **do**
6:         Compute $\omega$ of $RP_i$ in set $\Theta$
7:         **if** $\delta > \omega$ **then**                ▷ e=residual energy
8:             Obtain vacant hop to move by $k = \delta - \omega$
9:             Sort (1 to k)-hop children (neighbors) of $RP_i$
10:            **for** each k-hop child in sorting **do**
11:                **if** $\Delta_{(k)-hop\,child} < \Delta_{RP_i}$ **then**
12:                    **if** each $e_{path(child,parent)} \geq e_{RP_i}$ **then**
13:                        Changing the roles (flags)
14:                        Update sets $\mathbb{C}$ and $\Theta$
15:                    **end if**
16:                **end if**
17:            **end for**
18:        **end if**
19:        **if** $\delta \leq \omega$ or $RP_i$ has the smallest $\Delta$ **then**
20:            Prevent the growth of RPs count.
21:        **end if**
22:    **end for**
23: **end procedure**
```

from sets $\mathbb{C}$, O, and Θ, thereby the possibility of reducing the tour length increases. On the other hand, these nodes are not included in the solution set Υ, so they do not participate in the MS trajectory. The above procedure is repeated until both buffer limitation in the tree construction is not violated and we check all element of O. Fig. 3(c) gives an example to further illustrate the VRs consolidation process.

C. Phase 3—Tour Planning

In this section, we select the finalized RPs based on a set of $\mathbb{C}$ for MS tour scheduling. Suppose the initial traveling path of MS is derived by visiting the sequence of each element in set $\mathbb{C}$ as shown in Fig. 3(c). However, we check whether it is possible to replace RPs to further reduce the MS's tour length.

1) RPs Replacement: We bounded the relay hop count between non-RPs and RPs due to three reasons. First, the energy-efficient objective can be achieved by limiting the number of packets forwarded from non-RPs to RPs. Second, the packet loss ratio is enhanced by transmitting data over multiple relay hops. Third, since each sensor has a finite buffer size, a vast number of non-RPs cannot be assigned to one RP. Bounded relay hop is one of the system parameters that can be adjusted based on the priority of IoT applications over energy conserving and data delivery delay [16], [25]. For example, a small value for this parameter is considered in delay-sensitive applications due to energy efficiency.

We describe the RPs replacement stage based on Algorithm 9 to locate appropriate RPs to minimize MS's tour length, and all elements of $\mathbb{C}$ are also checked, where in the case of Fig. 3(d), it depicts this stage. To do so, for each element in set $\mathbb{C}$, we first compute the maximum hop count in their trees to reach them and call it the RPs' weight ω. In other words, we define the height of each tree rooted at each element of $\mathbb{C}$ as its weight. Let us denote by δ the maximum height of each tree as the chain threshold (the relay hop count) where the chain threshold is a given input of the algorithm. Then, If ω of each clustered tree is more than δ, we consider it for the relocation update process. Let us consider an instance of two different cases as seen in Fig. 4, in the first case sensor s_3 is an RP and its weight is 2, while in the second case sensor s_5 is an RP and has a weight of 4. It can be seen that the amount of data collection in RPs s_5 and s_3 is identical, but in the first case, the probability of occurrence of the hotspot is less than in the second case. Due to the tradeoff between the QoS requirements of WSNs, the first case is an allowable structure that satisfies our objectives in this article. That is why we take the chain threshold δ in this section. Suppose we consider $\delta = 3$ in Fig. 3(d), so variable hop of replacement with a maximum length 3 has occurred in some RPs. In addition, we assume that the sensor nodes have vacant residual energy in order to RP replacement in Fig. 3.

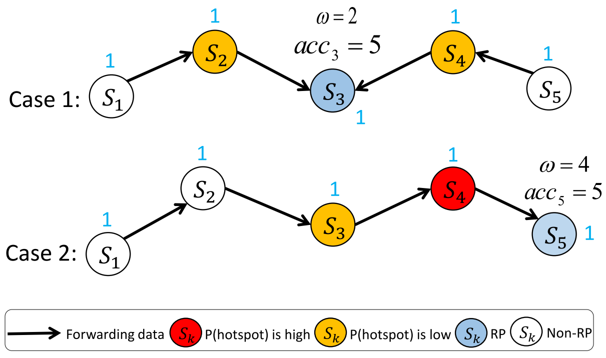


Fig. 4. Illustration of different cases in RPs replacement.

As illustrated in Algorithm 9, we first compute the weight of each $RP_i \in \mathbb{C}$ according to its subtree in set Θ. For each RP candidate, If the condition of $\delta > \omega$ is satisfied, in the next step, we determine the difference between δ and ω as the number of hops, k, to check the children of parent RP ($k = 1, \ldots, \omega$). By sorting the k-hops children of RP based on the minimum Δ, we consider the child s_i as a deputy for RP replacement that has the minimum Δ toward RP candidate. Besides, to reduce the possibility of the hotspot occurrence, the residual energy (e) of each node located in the ancestor(RP)-descendant(k-hop child) relationship of considered path must be greater than and equal to the residual energy of original RP candidate. In that case, the role of RP and that k-hop child (child s_i) changes. In other words, s_i will be included in set $\mathbb{C}$ instead of its parent in this round of EE-MSWSN. Otherwise, the next k-hop child of the RP candidate is checked in sorting. The above procedure is repeated until all elements of $\mathbb{C}$ are checked and the termination conditions of $\delta \leq \omega$ are reached.

2) MS Scheduling: We adopt a local-search TSP method to find efficiently the MS trajectory so that the MS visits all RPs in set Υ. This heuristic whose details can be derived from [43], takes the time complexity of $O(|\Upsilon|^3)$. We give a complete example in Fig. 3. Since RP annihilation with higher priority information violates reliability more than RP annihilation with lower priority information. Hence, we are considering the same priority for each RP in this example, thereby we will perform the MS scheduling according to the local search TSP. The result of applying this heuristic to the solution set of Υ is depicted in Fig. 3(d). The local-search scheme needs a primary tour to get started and its efficiency depends on the primary tour quality. That is why we construct the initial path in phase 2 of the EE-MSWSN algorithm as shown in Fig. 3(c). It can be seen that there are differences distinguish between the initial path and the desired path constructed in Fig. 3(c) and (d), thereby EE-MSWSN was able to further reduce the tour length. As shown in Fig. 3(d), the example has several iterations.

Iteration (1): All RP candidates are included in set $\mathbb{C}$. So far, we have $\mathbb{C} = \{s_{31}, s_{22}, s_{18}, s_{20}, s_{34}, s_{19}\}$. In this case, we exclude randomly one element of set $\mathbb{C}$ and add it to set solution of RPs Υ. We take s_{31} and remove it from set $\mathbb{C}$ as well. Thus, we have $\Upsilon = \{S_\infty, s_{31}\}$.

Iteration (2): By repeating Iteration 1, another element (s_{34}) of $\mathbb{C}$ is added to Υ. Thus, the MS can find the desired traveling path among elements of $\Upsilon = \{S_\infty, s_{31}, s_{34}\}$ based on the local-search TSP method. In this case, if RP s_{34} has a higher priority (e.g., having emergency data) than the other RPs, it should always be visited at the beginning of the MS scheduling in the desired tour.

The Last Iteration: By repeating the previous procedures until all elements of $\mathbb{C}$ are checked, the final solution set of Υ will be $\Upsilon = \{s_{31}, s_{22}, s_{18}, s_{19}, s_{20}, s_{34}\}$ and the desired path without any packet dropping based on local-search TSP is $S_\infty \to s_{31} \to s_{22} \to s_{18} \to s_{19} \to s_{20} \to s_{34} \to S_\infty$.

Next Cycle of Data Aggregation: When all RPs are traversed in the tour, the MS again begins the data collection process in the next round.

D. Analysis

In this section, we first prove that the EE-MSWSN algorithm is NP-complete by Theorem 1 and then by Theorem 2 we analyze the time complexity of our scheme.

Theorem 1: The proposed scheme belongs to the class of NP-complete.

Proof: We consider two cases for sensors in tour planning: 1) without buffer limitation and 2) with limited buffer capacity. In the first case, we consider cr_s as the coverage metric for sensors. All sensors can become inaccessible to each other if their cr_s is below a special range. In this case, the decision version of TSP known as an NP-complete problem can be reduced to our problem in polynomial time. In other words, if cr_s are below a special range (e.g., zero range), the MS must visit all nodes as RPs to schedule the traveling path, which is "Is there a TSP tour of length at most $L_{\max}$?". Thus, this case belongs to the NP-complete class.

In the second case, we reduce the minimum cost Hamiltonian circuit (MCHC) problem known as NP-complete to our scheme in polynomial time. By considering N vertex in the graph, the decision problem of MCHC asks whether there is a circuit with length L that visits each vertex once. Similarly, given N sensors, EE-MSWSN asks whether there is a tour with traveling length L that visits each RP node once under satisfying both packet loss PL and energy consumption EC. Suppose that all nodes generate X packet except sensors s_a and s_b, which generate $X/2$ packet. The decision version of our problem will find a tour with length L whose starting from S_∞ and go back to it such that its total consumed energy is $EC = (E_{Rx} \times X/2\ ) + ((N-1) \times E_{Tx} \times X)$ and there is no buffer overflow at RPs. Assume that s_b is near to S_∞ than s_a in terms of hop count. If all nodes will be RPs and node s_a forward its $X/2$ data to s_b, then we will not packet loss at each node. Thus, the decision version of EE-MSWSN will ask to find a Hamiltonian cycle with length $L = T \times \nu$ to visit N sensors. The above reduction takes polynomial time, so the second case is also NP-complete. ■

Theorem 2: By considering N sensor in set S_N over the sensing field, the worst time complexity of the EE-MSWSN algorithm is $O(N^3)$.

Proof: Our scheme has three main phases, including 1) virtual structure construction; 2) RP selection; and 3) tour planning. In the first phase, at the beginning of the algorithm, five performance metrics of sensors are utilized to elect a proper set of RP candidates, they are included to set $\mathbb{C}$, in which the maximum time of this stage takes $|S_N| \times O(N) = O(N^2)$. Then, we partition the sensing field into $|\mathbb{C}|$ convex polygons called VRs based on the Voronoi diagram. In this way, we benefit the divide and conquer algorithm with the worst time complexity $O(N \times \log N)$. Finally, Dijkstra's algorithm is used to construct SP trees rooted at each member of $\mathbb{C}$ in each VR, which spends $|\mathbb{C}| \times O(N^2)$ time. In the second phase, we begin from each member in set ℓ to traverse SP trees in each VR in a bottom-up manner, until $\mathrm{acc}_j \geq \Im$. The worst-case scenario occurs when N sensors in set S_N have generated one packet and we use each of them as the starting point of the traveling tree. Thus, the worst time taken by pruning trees is $|\mathbb{C}| \times |S_N| \times O(\min\{\Im, N\}) = |\mathbb{C}| \times O(N \times \min\{\Im, N\})$. Afterward, to collect node data obtained from tree pruning and reduce the number of RPs, we seek to consolidate VRs. In this way, we find the adjacency list of VRs by the Floyd algorithm, which spends $O(|\mathbb{C}|^3)$ time. Also, comparisons between VRs take time complexity of $O(N^2)$ in the worst case. In the last phase, the EE-MSWSN algorithm first finds the weight of each member in $\mathbb{C}$, it takes $O(|\mathbb{C}| \times |\Theta|)$ time and then schedules the traveling path with the finalized RPs in set Υ. At the end, the worst time taken by path planning is $O(|\Upsilon|^3)$, Υ has every node in $\mathbb{C}$ except for the only node linked to S_∞. Thus, this modified local-search TSP algorithm spends $O(N^3)$ time in compared to existing TSP methods with the worst time complexity of $O(N^5)$.

To sum up above phases, since $|\mathbb{C}| << |S_N|$, the overall time complexity of the EE-MSWSN algorithm is $2 \times O(N^2) + O(N^3)$. We can simplify the worst time equation to $O(N^3)$, in which Theorem 2 is proved by this complexity. ■

V. Simulation Study

We use MATLAB to evaluate the performance of the EE-MSWSN algorithm. To obtain a fair comparison, the

TABLE III
DEFAULT PARAMETERS

Parameter	Value
Sensing Field Area	$200m \times 200m$
Network topology	Random
Sensor count	10-200
Buffer size	3-15 packets
Initial Energy	5 joule
MAC protocol	BoX-MAC
Sink count	1 MS
MS speed	5 m/s
Communication Range radious	20 m
Data packet size	134 bytes
ε_{elec}	50 nJ/bit
ε_{amp}	0.0013 $pJ/bit/m^4$
ε_{efs}	10 $pJ/bit/m^2$

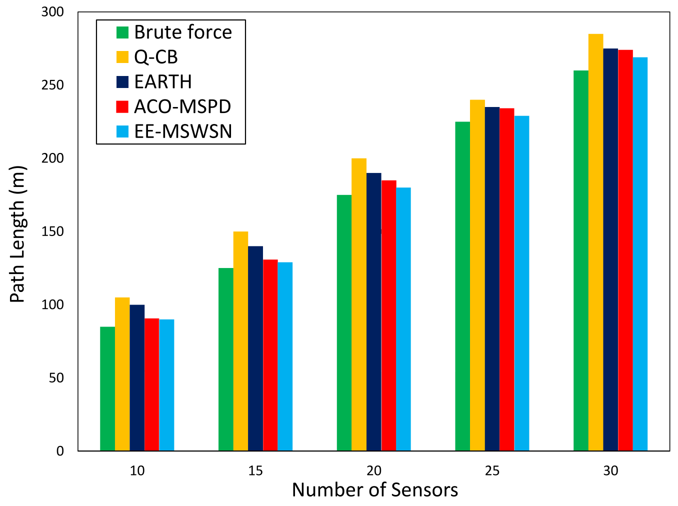


Fig. 5. Comparison of the average tour length in a round.

simulation parameters are derived from previous and similar MS-based data collection algorithms. The area of the sensing field is considered square with dimension 200 m × 200 m, where the communication range radius and initial energy of the nodes are set to 20 m and 5 J, respectively. In simulation, 10–200 sensors are randomly scattered [44] and one MS travels throughout the sensing field to collect the accumulative data from each RP once in a round-by-round manner. Also, different node deployment scenarios can be used [44]. Since the network is homogeneous, the buffer capacity and the communication range radius of sensor nodes are identical. All sensors run the BOX-MAC protocol [45]. To meet asynchronous BOX-MAC requirements, each sensor sets up to 5 s as a random beginning point for duty cycle initializing. In this WSN with random topology, each sensor can produce B_i packets between $[1, \alpha]$ randomly in each simulation round with packet size 134 bytes. In other words, we encounter buffer overflow at some sensors when $\Im < \alpha$. The related parameters to the energy model are derived from [39], where $\varepsilon_{\text{elec}} = 50$ nJ/bit, $\varepsilon_{efs} = 10$ pJ/bit/m^2, and $\varepsilon_{\text{amp}} = 0.0013$ pJ/bit/m^4. Each simulation scenario is run ten times and the results are then averaged and plotted. To measure the buffer efficiency, we use the following two metrics, which are derived from [15]:

$$\Im_{\text{mean}} = \sum_{r_i \in \Upsilon} \min\{\alpha_i, B\}/(|\Upsilon| \times B) \tag{19}$$

$$\Im_{\text{SD}} = \sqrt{{}^{1}/_{|\Upsilon|} \sum_{r_i \in \Upsilon} (\min\{\alpha_i, B\} - \Im_{\text{mean}})^2} \tag{20}$$

where $\Im_{\text{mean}}$ is the mean ratio of buffer usage and $\Im_{\text{SD}}$ is the standard deviation of number of packets stored in each buffer. Note that when $\alpha_i > B$, an RP_i can store a maximum of B packets in its buffer and other packets are dropped, which is why using the term $\min\{\alpha_i, B\}$ in (19) and (20). The network parameters considered in this simulation are summarized in Table III.

A. Experimental Results

To verify the efficiency, the results of the EE-MSWSN algorithm are compared with three MS-based data gathering methods, which are called ACO-MSPD [34] (employs ACO for directed spanning tree), Q-CB [20] (employs reinforcement learning in clustering considering buffer management), and EARTH [15] (employs a spanning tree rooted at the sink to cover all nodes by considering buffer management). EE-MSWSN incorporates the combination of clustering and SP tree to its benefit. We derive the experimental results by measuring path lengths and then evaluating different numbers of sensor nodes and buffer sizes. Performance metrics for comparison are comprised of buffer utilization ($\Im_{\text{mean}}$), standard deviation ($\Im_{\text{SD}}$), total energy consumption, number of dropped packets, and computation time.

1) Path Length: Fig. 5 shows the result of measurement of path length, where we have considered $\Im = 5$ and $B_i = [1, (\alpha = 5)]$. We first compared the MS tour length obtained by the EE-MSWSN algorithm with Q-CB, EARTH, ACO-MSPD, and the optimal solution. Hence, we used brute force as the optimal solution to search for all possible combinations between RPs, and then selected the best one that had a minimum path length and no buffer overflow. Since the optimal method is very long to obtain, we performed the experiment on a small-scale network with 10–30 sensors. As shown in Fig. 5, on average, the EE-MSWSN algorithm increases the path length around 5.5% compared to the optimal solution, which verifies that the EE-MSWSN algorithm can find the shortest traveling tour without dropping packets. The Q-CB, EARTH, and ACO-MSPD methods reported an average around 8.4%, 6.3%, and 6% of path length against brute force, which means that EE-MSWSN is approximately 52%, 15%, and 10% better than the above-mentioned methods, respectively. As seen in Fig. 5, as the number of sensors increases, we need to select more RPs to prevent buffer overflow. Thus, the length of the MS trajectory increases accordingly.

2) Varying the Number of Sensors: In this section, we conduct an experiment based on increasing the number of sensors from 50 to 200 by adjusting $\Im = 5$ and $B_i = [1, (\alpha = 5)]$ in simulation.

Total Energy Consumption: Fig. 6(a) displays the total energy consumption by the considered methods over the

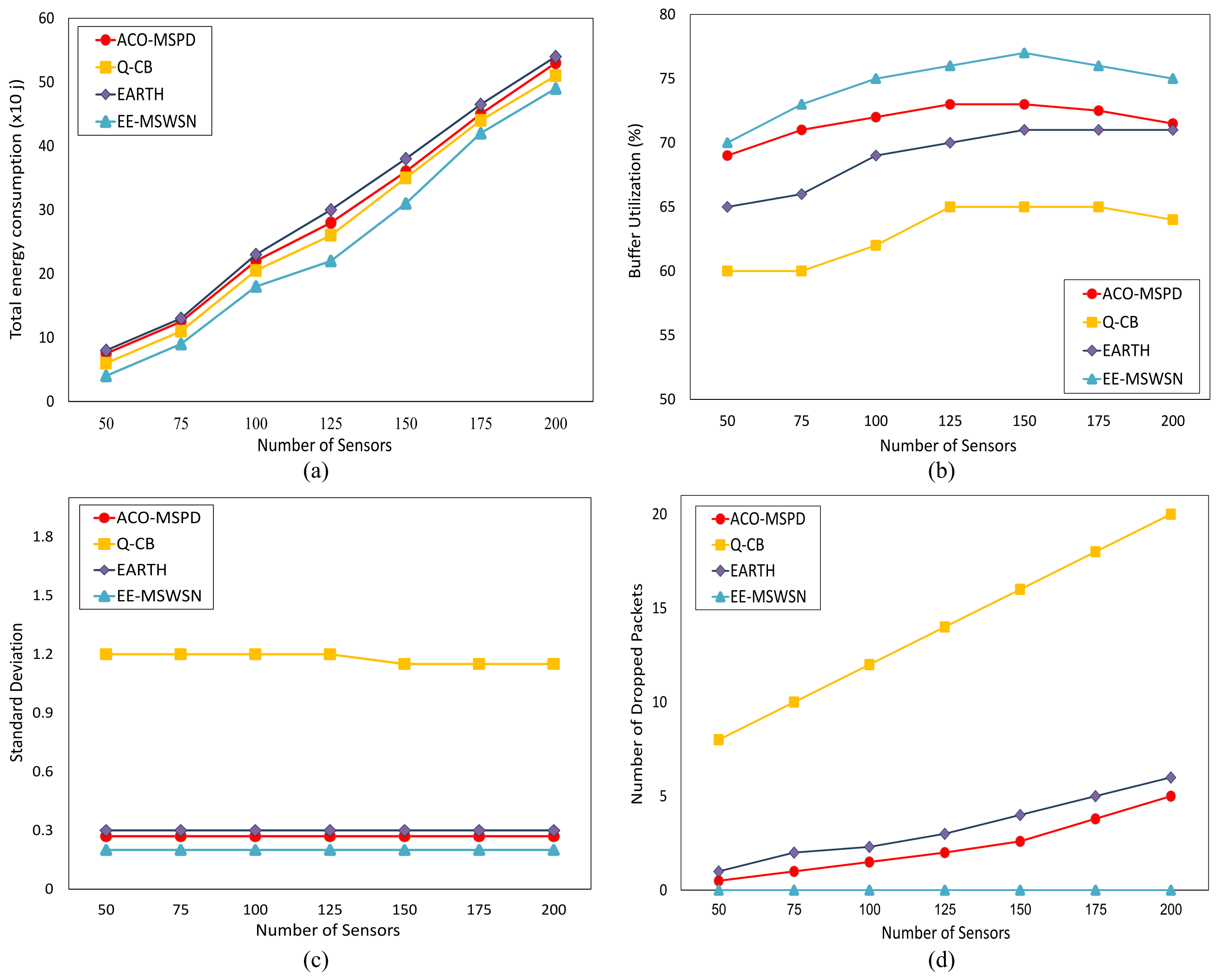


Fig. 6. Simulation results for varying the number of sensors of competitive algorithms in a round: (a) total energy consummation; (b) buffer utilization; (c) standard deviation of buffering; and (d) number of packet dropping.

number of sensors. It shows that by a growing number of sensors, some sensors need to pass from many hops to forward their data to RPs. Thus, minimizing hop counts can help more to diminish the amount of energy consumed. On the other hand, we need to elect more RPs to avert packet loss by growing the number of sensors. Accordingly, the uniform distribution of RPs across the network becomes more significant because some RPs may have collected more data from non-RP nodes. Hence, the reason for the increase in total energy consumption can be explained by increasing the number of sensors. As shown in Fig. 6(a), EE-MSWSN has high energy efficiency compared to the competitive schemes. According to this Figure, Q-CB as a clustering approach in total energy consumption is quite close to our protocol. On the other hand, since the convergence rate of the ACO-MSPD is low, this algorithm is initially involved in local optimum, but with an increasing number of sensors and simulation rounds, it shows better energy saving than the EARTH.

Buffer Utilization and Standard Deviation: Both Fig. 6(b) and (c) show the utilization $\Im_{\text{mean}}$ and standard deviation $\Im_{\text{SD}}$ of buffers based on (19) and (20), respectively. The small $\Im_{\text{SD}}$ indicates the uniform utilization of RPs' buffer, while the large $\Im_{\text{SD}}$ demonstrates that the buffer utilization rates of RPs vary considerably. On the other hand, the high value of $\Im_{\text{mean}}$ implies that each RP has used its buffer better to data gathering. Thus, the largest $\Im_{\text{mean}}$ along with the smallest $\Im_{\text{SD}}$ can diminish the probability of the buffer overflow ar RPs considerably. Since the Q-CB method has divided the sensors into various sizes, it will have the highest $\Im_{\text{SD}}$, which leads to packet loss at some RPs easily. Additionally, EARTH is a tree-based structure and elects RPs based on hop count and the amount of accumulative data, thereby it has better $\Im_{\text{mean}}$ and $\Im_{\text{SD}}$ values than clustering method. Also, as the number of sensors increases, the possibility of falling into global optimum for the ACO-MSPD decreases, so without considering the heuristic metric for buffer overflow management in the ACO function, the buffer utilization gradually decreases. EE-MSWSN, due to the exploitation of advantages of both clustering and tree methods, always has high performance, in terms of $\Im_{\text{mean}}$ and $\Im_{\text{SD}}$ values against various numbers of sensors.

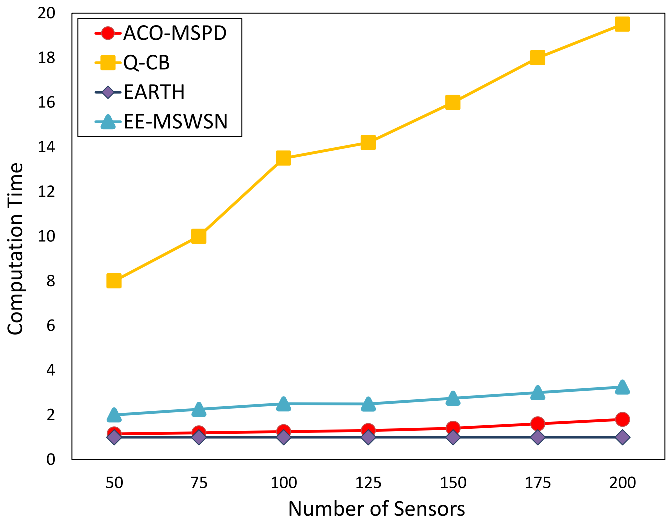


Fig. 7. Simulation results for the computation time of considered works in a round.

Number of Dropped Packets: Fig. 6(d) measures the dropped packet count against increasing the sensor count. As mentioned earlier, there is a direct effect between the performance metrics $\Im_{\text{mean}}$ and $\Im_{\text{SD}}$ on the packet loss ratio. As the ACO-MSPD method does not consider buffer overflow management, it loses more packets than the EE-MSWSN. Obviously, ACO-MSPD has been able to significantly surpass EARTH and Q-CB by using ACO-based MS path determination. Additionally, EARTH may encounter packet losses at some sensors due to the chain building problems in subtrees. The Q-CB has the highest rate in this metric based on low $\Im_{\text{mean}}$ and high $\Im_{\text{SD}}$. Conversely, our protocol schedules the MS trajectory after checking the buffer capacity, thereby it has no packet losses.

Computation Time: The computation time of the considered protocols by growing the number of sensors is depicted in Fig. 7. Due to the lowest computation overhead, we consider the EARTH time as one unit for 50–200 sensors. Because the Q-CB sensors are clustered repeatedly and find RPs accordingly, it averagely experiences about 14 times more computation overhead than the EARTH protocol. It can be seen that the machine learning approach injects extra overhead on the Q-CB method. Since EARTH uses a tree structure, it experiences less complexity time than the competitive methods. However, EE-MSWSN adopted a modified TSP for path scheduling like EARTH, but by dividing the sending field into VRs and selecting RP candidates based on the performance metric, it adheres to an appropriate balance between energy efficiency and computation time. For ACO-MSPD, constructing a directed spanning tree with n nodes leads to $O(n)$ complexity. This time along with using ACO show less computational complexity than the EE-MSWSN. Hence, the computation cost of EE-MSWSN is exploiting the intelligent hierarchical tools compared to EARTH. Thus, EE-MSWSN can find a new tour with allowable computation time in the dynamic sensing rate of the network behavior.

3) Varying the Buffer Size: In this section, by adjusting $B_i = [1, (\alpha = 7)]$, the fixed sensor count 100, and the different buffer size between [3, 15] in each round, we measure the effect of growing buffer size $\Im$ on the performance metrics by each competitive method.

Total Energy Consumption: Fig. 8(a) depicts the effect of growing the buffer capacity on the total energy consumption for each method. According to this figure, by increasing the buffer size and the fixed number of sensors, RPs may gather extra accumulative data from non-RP nodes. Hence, we may encounter a reduction in RP count. In this case, non-RP nodes have to pass more hops to forward their data to RPs. That is why the competitive algorithms consume more the total energy by growing buffer size as seen in Fig. 8(a). EE-MSWSN saves a little more energy than Q-CB but stores around 10% more energy than EARTH and ACO-MSPD, which indicates its high energy efficiency. Also, as seen in Fig. 6(a), the superiority of EE-MSWSN compared to in terms of total energy consumption is due to the inclusion of hop count and energy-aware metrics in the clustered tree-based structure.

Buffer Utilization and Standard Deviation: Fig. 8(b) and (c) displays $\Im_{\text{mean}}$ and $\Im_{\text{SD}}$ of the finalized RPs under different size of $\Im$, respectively. Since each RP can have more space to store accumulative data by growing $\Im$, some RPs may exhaust more buffer capacity, which also increments $\Im_{\text{SD}}$. More precisely, this phenomenon occurs when $\Im > \text{Max}(B_i) = 7$ as seen in Fig. 8(b) and (c). It can be seen that EE-MSWSN can obtain better values of $\Im_{\text{mean}}$ and $\Im_{\text{SD}}$ compared to the considered algorithms, which result of experiment verify that EE-MSWSN can enhance the buffer yield and also equipoise the RP workloads.

Number of Dropped Packets: Fig. 8(d) demonstrates the dropped packets count by each competitive method against growing $\Im$. As mentioned earlier, these methods are encountered packet loss when $\Im < \alpha$. When this equation changes to $\Im \geq \alpha$, the probability of buffer overflow at some RPs declines. It can be seen that EE-MSWSN does not discard any packets in $\Im \geq 7$ with 100 sensors due to finding more RPs for storing more packets. Moreover, the ACO-MSPD shows slightly fewer packet drops compared to EARTH as it benefits ACO strategy into its tree-based structure. Since the Q-CB as a clustering method in selecting RPs does not consider the buffer management issue, it encounters more packet losses even by growing $\Im$. However, Q-CB, by learning network behavior, can diminish this performance gap compared to tree-based methods.

VI. Conclusion

We proposed EE-MSWSN, an energy-efficient MS WSN. EE-MSWSN's clustered tree-based structure contributes to avert packet losses by considering nodes with disparate sensing rates and finite buffer sizes. EE-MSWSN, along with a novel hierarchical structure, adheres to the requirements of resource-constrained WSNs and seeks to find the shortest tour for the MS. The proposed algorithm finds RP candidates based on performance metrics and partitions the sensing field into VRs considering the locations of RP candidates. Afterward, it constructs an SP tree in each VR to connect sensors by Dijkstra's algorithm. Finally, EE-MSWSN elects the finalized RPs based

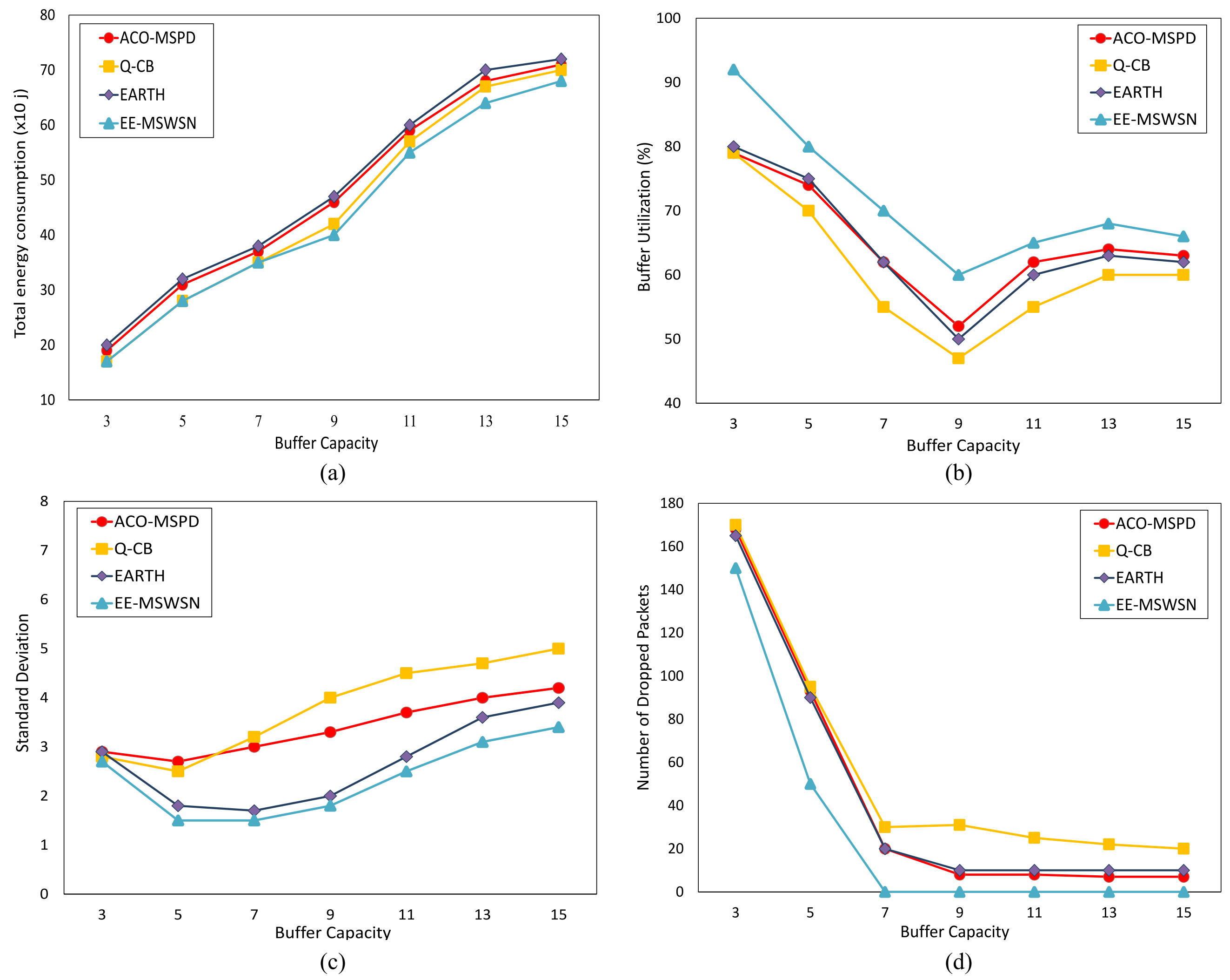


Fig. 8. Simulation results for varying the buffer size of different algorithms in a round: (a) total energy consummation; (b) buffer utilization; (c) standard deviation of buffering; and (d) number of packet dropping.

on the amount of relay data, hop count, and distance to the stationary sink through traveling along the trees in a bottom-up manner and consolidating RP candidates. Simulation results indicate that EE-MSWSN reduces the length of the MS tour by 52%, 15%, and 10% compared to Q-CB, EARTH, and ACO-MSPD, respectively. Hence, our method is an ideal choice for delay-tolerant applications and practical issues, where it is required to collect data in time-bounded periods. Besides, our results show that EE-MSWSN does not discard any packets and achieves 10% better energy saving in comparison to state-of-the-art protocols.

As future work, we plan to increase the performance of data gathering in realistic problems by considering the sojourn time of the MS at RPs. Further, considering the measured data with various delivery delay requirements (e.g., emergency data) in heterogeneous WSNs can improve our approach. Since we divide the large environment into several regions, each region can execute our algorithm. Hence, the proposed algorithm covers the use of multiple MS, but issues such as coordination and interference between MSs can be challenging. Also, we plan to reduce the time complexity by exploiting the property of the triangle, although it is very difficult to estimate the implementation requirements. We defer the analysis of such a method to our future work.

REFERENCES

[1] J. Yick, B. Mukherjee, and D. Ghosal, “Wireless sensor network survey,” *Comput. Netw.*, vol. 52, no. 12, pp. 2292–2330, 2008.

[2] M. A. Al-Jarrah, M. A. Yaseen, A. Al-Dweik, O. A. Dobre, and E. Alsusa, “Decision fusion for IoT-based wireless sensor networks,” *IEEE Internet Things J.*, vol. 7, no. 2, pp. 1313–1326, Feb. 2020.

[3] R. Anwit, A. Tomar, and P. K. Jana, “Tour planning for multiple mobile sinks in wireless sensor networks: A shark smell optimization approach,” *Appl. Soft Comput.*, vol. 97, Dec. 2020, Art. no. 106802.

[4] A. Shahraki, A. Taherkordi, Ø. Haugen, and F. Eliassen, “A survey and future directions on clustering: From wsns to IoT and modern networking paradigms,” *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 2, pp. 2242–2274, Jun. 2021.

[5] B. Pourghebleh and N. J. Navimipour, “Data aggregation mechanisms in the Internet of Things: A systematic review of the literature and recommendations for future research,” *J. Netw. Comput. Appl.*, vol. 97, pp. 23–34, Nov. 2017.

[6] H. Fotouhi, M. Alves, M. Z. Zamalloa, and A. Koubâa, “Reliable and fast hand-offs in low-power wireless networks,” *IEEE Trans. Mobile Comput.*, vol. 13, no. 11, pp. 2620–2633, Nov. 2014.

[7] S. Jain, K. K. Pattanaik, R. K. Verma, S. Bharti, and A. Shukla, “Delay-aware green routing for mobile-sink-based wireless sensor networks,” *IEEE Internet Things J.*, vol. 8, no. 6, pp. 4882–4892, Mar. 2021.

[8] Z. Chu, F. Zhou, Z. Zhu, R. Q. Hu, and P. Xiao, “Wireless powered sensor networks for Internet of Things: Maximum throughput and optimal power allocation,” *IEEE Internet Things J.*, vol. 5, no. 1, pp. 310–321, Feb. 2018.

[9] C. Tunca, S. Isik, M. Y. Dönmez, and C. Ersoy, “Ring routing: An energy-efficient routing protocol for wireless sensor networks with a mobile sink,” *IEEE Trans. Mobile Comput.*, vol. 14, no. 9, pp. 1947–1960, Sep. 2015.

[10] H. Salarian, K.-W. Chin, and F. Naghdy, “An energy-efficient mobile-sink path selection strategy for wireless sensor networks,” *IEEE Trans. Veh. Technol.*, vol. 63, no. 5, pp. 2407–2419, Jun. 2014.

[11] R. C. Shah, S. Roy, S. Jain, and W. Brunette, “Data mules: Modeling and analysis of a three-tier architecture for sparse sensor networks,” *Ad Hoc Netw.*, vol. 1, nos. 2–3, pp. 215–233, 2003.

[12] M. A. Araghizadeh, P. Teymoori, N. Yazdani, and S. Safari, “An efficient medium access control protocol for WSN-UAV,” *Ad Hoc Netw.*, vol. 52, pp. 146–159, Dec. 2016.

[13] S. Yu, B. Zhang, C. Li, and H. T. Mouftah, “Routing protocols for wireless sensor networks with mobile sinks: A survey,” *IEEE Commun. Mag.*, vol. 52, no. 7, pp. 150–157, Jul. 2014.

[14] T. Han, L. Zhang, S. Pirbhulal, W. Wu, and V. H. C. de Albuquerque, “A novel cluster head selection technique for edge-computing based IoMT systems,” *Comput. Netw.*, vol. 158, pp. 114–122, Jul. 2019.

[15] Y.-C. Wang and K.-C. Chen, “Efficient path planning for a mobile sink to reliably gather data from sensors with diverse sensing rates and limited buffers,” *IEEE Trans. Mobile Comput.*, vol. 18, no. 7, pp. 1527–1540, Jul. 2019.

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

[17] D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook, *The Traveling Salesman Problem*. Princeton, NJ, USA: Princeton Univ. Press, 2011.

[18] Y.-C. Wang, Y.-Y. Hsieh, and Y.-C. Tseng, “Multiresolution spatial and temporal coding in a wireless sensor network for long-term monitoring applications,” *IEEE Trans. Comput.*, vol. 58, no. 6, pp. 827–838, Jun. 2009.

[19] G. Xing, T. Wang, Z. Xie, and W. Jia, “Rendezvous planning in wireless sensor networks with mobile elements,” *IEEE Trans. Mobile Comput.*, vol. 7, no. 12, pp. 1430–1443, Dec. 2008.

[20] S. Redhu and R. M. Hegde, “Cooperative network model for joint mobile sink scheduling and dynamic buffer management using Q-learning,” *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 3, pp. 1853–1864, Sep. 2020.

[21] Y.-C. Wang and G.-W. Chen, “Efficient data gathering and estimation for metropolitan air quality monitoring by using vehicular sensor networks,” *IEEE Trans. Veh. Technol.*, vol. 66, no. 8, pp. 7234–7248, Aug. 2017.

[22] D. Jiang, W. Wang, L. Shi, and H. Song, “A compressive sensing-based approach to end-to-end network traffic reconstruction,” *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 1, pp. 507–519, Jan.–Mar. 2020.

[23] M. Biabani, H. Fotouhi, and N. Yazdani, “An energy-efficient evolutionary clustering technique for disaster management in IoT networks,” *Sensors*, vol. 20, no. 9, p. 2647, 2020.

[24] C. Zhu, S. Wu, G. Han, L. Shu, and H. Wu, “A tree-cluster-based data-gathering algorithm for industrial wsns with a mobile sink,” *IEEE Access*, vol. 3, pp. 381–396, 2015.

[25] S. M. Ghoreyshi, A. Shahrabi, T. Boutaleb, and M. Khalily, “Mobile data gathering with hop-constrained clustering in underwater sensor networks,” *IEEE Access*, vol. 7, pp. 21118–21132, 2019.

[26] M. Biabani, N. Yazdani, and H. Fotouhi, “REFIT: Robustness enhancement against cascading failure in IoT networks,” *IEEE Access*, vol. 9, pp. 40768–40782, 2021.

[27] S. Jain, K. K. Pattanaik, and A. Shukla, “QWRP: Query-driven virtual wheel based routing protocol for wireless sensor networks with mobile sink,” *J. Netw. Comput. Appl.*, vol. 147, Dec. 2019, Art. no. 102430.

[28] T. Wang *et al.*, “Data collection from WSNs to the cloud based on mobile fog elements,” *Future Gener. Comput. Syst.*, vol. 105, pp. 864–872, Apr. 2020.

[29] R. Yarinezhad, “Reducing delay and prolonging the lifetime of wireless sensor network using efficient routing protocol based on mobile sink and virtual infrastructure,” *Ad Hoc Netw.*, vol. 84, pp. 42–55, Mar. 2019.

[30] K. Almiáni, A. Viglas, and L. Libman, “Energy-efficient data gathering with tour length-constrained mobile elements in wireless sensor networks,” in *Proc. IEEE Local Comput. Netw. Conf.*, 2010, pp. 582–589.

[31] S. Najjar-Ghabel, L. Farzinvash, and S. N. Razavi, “Mobile sink-based data gathering in wireless sensor networks with obstacles using artificial intelligence algorithms,” *Ad Hoc Netw.*, vol. 106, Sep. 2020, Art. no. 102243.

[32] B. G. Gutam, P. K. Donta, C. S. R. Annavarapu, and Y.-C. Hu, “Optimal rendezvous points selection and mobile sink trajectory construction for data collection in WSNs,” *J. Ambient Intell. Humanized Comput.*, vol. 13, pp. 1–12, Oct. 2021.

[33] P. K. Donta, B. S. P. Rao, T. Amgoth, C. S. R. Annavarapu, and S. Swain, “Data collection and path determination strategies for mobile sink in 3D WSNs,” *IEEE Sensors J.*, vol. 20, no. 4, pp. 2224–2233, Feb. 2020.

[34] P. Kumar, T. Amgoth, and C. S. R. Annavarapu, “ACO-based mobile sink path determination for wireless sensor networks under non-uniform data constraints,” *Appl. Soft Comput.*, vol. 69, pp. 528–540, Aug. 2018.

[35] P. K. Donta, T. Amgoth, and C. S. R. Annavarapu, “An extended ACO-based mobile sink path determination in wireless sensor networks,” *J. Ambient Intell. Humanized Comput.*, vol. 12, no. 10, pp. 8991–9006, 2021.

[36] O. Busaileh, A. Hawbani, X. Wang, P. Liu, L. Zhao, and A. Y. Al-Dubai, “Tuft: Tree based heuristic data dissemination for mobile sink wireless sensor networks,” *IEEE Trans. Mobile Comput.*, vol. 21, no. 4, pp. 1520–1536, Apr. 2022.

[37] Y. Zou, H. Liu, and Q. Wan, “Joint synchronization and localization in wireless sensor networks using semidefinite programming,” *IEEE Internet Things J.*, vol. 5, no. 1, pp. 199–205, Feb. 2018.

[38] L. Wei, Y. Chen, Y. Zhang, L. Zhao, and L. Chen, “PSPL: A generalized model to convert existing neighbor discovery algorithms to highly efficient asymmetric ones for heterogeneous IoT devices,” *IEEE Internet Things J.*, vol. 7, no. 8, pp. 7207–7219, Aug. 2020.

[39] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, “Energy-efficient communication protocol for wireless microsensor networks,” in *Proc. 33rd Annu. Hawaii Int. Conf. Syst. Sci.*, 2000, p. 10.

[40] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, “An application-specific protocol architecture for wireless microsensor networks,” *IEEE Trans. Wireless Commun.*, vol. 1, no. 4, pp. 660–670, Oct. 2002.

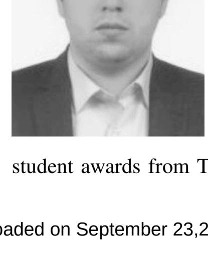
[41] S. Yousefi, F. Derakhshan, H. Karimipour, and H. S. Aghdasi, “An efficient route planning model for mobile agents on the Internet of Things using Markov decision process,” *Ad Hoc Netw.*, vol. 98, Mar. 2020, Art. no. 102053.

[42] N. Xia, C. Wang, Y. Yu, H. Du, C. Xu, and J. Zheng, “A path forming method for water surface mobile sink using voronoi diagram and dominating set,” *IEEE Trans. Veh. Technol.*, vol. 67, no. 8, pp. 7608–7619, Aug. 2018.

[43] W. Zhang, “Depth-first branch-and-bound versus local search: A case study,” in *Proc. AAAI/IAAI Conf.*, 2000, pp. 930–935.

[44] D. K. Sah, K. Cengiz, P. K. Donta, V. N. Inukollu, and T. Amgoth, “EDGF: Empirical dataset generation framework for wireless sensor networks,” *Comput. Commun.*, vol. 180, pp. 48–56, Dec. 2021.

[45] D. Moss and P. Levis, “BoX-MACs: Exploiting physical and link layer boundaries in low-power networking,” Comput. Syst. Lab., Stanford Univ., Stanford, CA, USA, Rep. SING-08-00, 2008.



Morteza Biabani received the B.S. and M.S. degrees in computer engineering from the University of Tabriz, Tabriz, Iran, in 2015 and 2017, respectively. He is currently pursuing the Ph.D. degree with the School of Electrical and Computer Engineering, University of Tehran, Tehran, Iran.

He is working as a Researcher with the Router Laboratory, University of Tehran. His current research interests include Internet of Things and wireless networking.

Mr. Biabani also received several distinguished student awards from Tabriz University.

Nasser Yazdani received the B.S. degree in computer engineering from Sharif University of Technology, Tehran, Iran, in 1985, and the Ph.D. degree in computer science and engineering from Case Western Reserve University, Cleveland, OH, USA, in 1997.

He is currently a Full Professor with the School of Electrical and Computer Engineering, University of Tehran, Tehran, Iran. He worked with Iran Telecommunication Research Center, Tehran, as a Consultant, a Researcher, and a Developer for few years. Then, he worked with different companies and research institutes in USA. He joined the ECE Department, University of Tehran in September 2000. Then, he initiated different research projects and labs in high-speed networking and systems. His research interests include networking, packet switching, access methods, operating systems, and database systems.

Hossein Fotouhi received the B.Sc. degree in electrical and electronics engineering from University of Guilan, Rasht, Iran in 2004, the M.Sc. degree in communication network engineering from University Putra, Seri Kembangan, Malaysia, in 2008, and the Ph.D. degree in electrical and computer engineering from University of Porto, Porto, Portugal, in 2015.

He is currently an Assistant Professor with the School of Innovation, Design, and Engineering, Mälardalen University, Västerås, Sweden. He performed his Ph.D. research with CISTER Research Unit, Porto, Portugal, from July 2009 to April 2015. His research interests are Internet of Things, sensor networks, wireless communication, and fog/edge computing.

Dr. Fotouhi has received a very competitive national grant, called VR starting grant, from the Swedish Research Council. He has been a Guest Editor of *Sensors* (MDPI), *Journal of Sensor and Actuator Networks* (MDPI), and Hindawi journals. He has publications in several top venues, including EWSN Conference, IEEE TRANSACTIONS ON MOBILE COMPUTING, *Ad-Hoc Networks* (Elsevier), *Computer Communications* (Elsevier), IEEE ACCESS, *Sensors* (MDPI), and *Journal of Sensor and Actuator Networks* (MDPI). For more information see http://www.es.mdh.se/staff/2992-Hossein_Fotouhi.