Neural Computing and Applications (2024) 36:8501–8517
https://doi.org/10.1007/s00521-024-09520-5

ORIGINAL ARTICLE

iCapS-MS: an improved Capuchin Search Algorithm-based mobile-sink sojourn location optimization and data collection scheme for Wireless Sensor Networks

Zaher Al Aghbari[1] · P V Pravija Raj[1] · Reham R. Mostafa[2,3] · Ahmed M. Khedr[1,4]

Received: 17 November 2022 / Accepted: 14 January 2024 / Published online: 23 February 2024

Abstract
Data collection using Mobile Sink (MS) is one of the best approaches to address the hot spot issue resulting from multihop data collection and extend the lifetime of Wireless Sensor Networks wherein the MS tours a few specific locations called sojourn locations that serve as data collecting points (DCPs). The best choice of these locations is an NP-hard problem, and the optimum or nearly optimum results can be achieved by applying meta-heuristic optimization methods. It is challenging to create an effective algorithm that allows MS for data collection irrespective of the network topology changes caused by node failures since these changes affect node coverage, data transmission, and network lifespan. Hence, an effort must be made to ensure a trade-off between the MS trajectory and the number of hops. Different MS-based techniques have been proposed; however, most of them fell short of addressing the above goals. With this inspiration, we propose iCapS-MS, which is an integrated approach that utilizes an improved Capuchin Search Algorithm (iCapSA) to determine the best set of DCPs and enhanced Ant Colony Optimization (e-ACO)-based MS trajectory design. Using iCapSA, the best DCPs are selected such that almost every node is served in one-hop communication with the shortest feasible hop distance and minimum coverage intersection between DCPs. The best trajectory for MS is established using e-ACO method. The results demonstrate that iCapS-MS outperforms existing methods based on several performance metrics.

Keywords Wireless Sensor Network (WSN) · Data Collection · Capuchin Search Algorithm (CapSA) · Ant Colony Optimization (ACO) · Mobile Sink (MS)

1 Introduction

In applications involving Wireless Sensor Networks (WSNs), data gathered by the sensor nodes is transmitted via single or multi-hop routes to a designated destination, which can be either a fixed sink or a Base Station (BS) [1, 2]. A major drawback of a multi-hop data delivery strategy is that nearby nodes to the BS must transfer greater data than far-off nodes, resulting in increased energy consumption by the nearby nodes compared to those at

✉ Zaher Al Aghbari
zaher@sharjah.ac.ae

P V Pravija Raj
pravijarajpv@gmail.com

Reham R. Mostafa
reldeiasti@sharjah.ac.ae

Ahmed M. Khedr
akhedr@sharjah.ac.ae

[1] Department of Computer Science, University of Sharjah, Sharjah 27272, United Arab Emirates

[2] Big Data Mining and Multimedia Research Group, Centre for Data Analytics and Cybersecurity (CDAC), Research Institute of Sciences and Engineering (RISE), University of Sharjah, Sharjah 27272, United Arab Emirates

[3] Information Systems Department, Faculty of Computers and Information Sciences, Mansoura University, Mansoura 35516, Egypt

[4] Department of Mathematics, Zagazig University, Zagazig, Egypt

longer distances from the BS [3]. This adverse situation (called hotspot problem) causes network partitioning and increases traffic dispersion, ultimately decreasing the WSN lifetime [4–6]. Several studies have shown that utilizing a Mobile Sink (MS) to address the hotspot issue is a viable method that evenly distributes the network's total energy usage [7, 8]. Additionally, it promotes reliable data acquisition and improves accessibility to isolated regions. Despite having numerous benefits, the task of designing an efficient trajectory for the MS and determining the optimal Data Collecting Points (DCPs) where the MS should pause for data collection presents significant challenges [9–12]. Researchers have suggested several Swarm Intelligence (SI)-based techniques to improve the DCPs' selection and quickly discover near-optimal MS trajectory [13, 14]. This includes Ant Colony Optimization (ACO), Harmony Search Algorithm (HSA), Artificial Bee Colony (ABC), Moth Flame Optimization (MFO), Particle Swarm Optimization (PSO), Butterfly Optimization Algorithm (BOA), Cuckoo Search Algorithm (CSA), and others [15–19].

WSN performance greatly relies on the proper selection of DCPs and efficient MS trajectory design, impacting coverage, data transmission, and network lifetime [20]. A higher number of DCPs can increase the trajectory size. Hence, it is recommended to shorten MS trajectory to enhance data transfer speed. However, a shorter trajectory necessitates greater multi-hop communications, resulting in increased hop counts and multi-hop paths. Consequently, nodes consume more energy. This implies that the best data collection strategy may not always involve the shortest MS trajectory. Conversely, as the trajectory size increases, hop counts decrease, and multi-hop paths decrease. However, this can lead to an increase in data delivery latency. This implies that consideration must be given while constructing the MS trajectory so that there will be a trade-off between the multi-hop paths and the MS trajectory. The best choice of the locations of DCPs is an NP-hard problem, and the optimum or nearly optimum solutions can be generated by applying meta-heuristic optimization algorithms. Current techniques in MS-based WSNs are concentrated on developing trajectories that minimize energy usage and minimize data acquisition time. However, most of the strategies fell short of addressing the adaptability toward the changes in the aspect of network topology caused by node failures that have an impact on the coverage of nodes, network lifetime, and data collection performance. To effectively reduce the latency and enhance the lifetime performance of WSN, the number and location of DCPs as well as the trajectory must be adaptable to changes in the network.

This motivates us to design an effective approach, called iCapS-MS, which is a hybrid approach that utilizes an improved Capuchin Search Algorithm (iCapSA) to determine the best set of DCPs and enhanced Ant Colony Optimization (e-ACO)-based MS trajectory design method for MS to achieve the above goals. The number and location of DCPs adapt to network changes caused by node failures, optimizing the MS path for decreased data delivery latency and increased WSN lifetime. This method allows most nodes to employ single-hop communication, with only a few outliers resorting to multi-hop transmission. While the original Capuchin Search Algorithm (CapSA) [21] excels in maintaining a balance between exploration and exploitation phases through the lifespan factor and exhibits enhanced performance with efficient global search capabilities than other recent SI methods, it encounters premature convergence and local optima challenges, like other meta-heuristic algorithms. We propose an improved version (called iCapSA) to tackle the limitations of the CapSA and enhance its performance. iCapSA incorporates three mutation strategies (Gaussian, Cauchy, and Levy) into the basic CapSA, promoting population diversity and guiding search agents toward the best solutions. In addition, we propose the e-ACO algorithm, an enhanced version of the original ACO. This enhancement focuses on refining the existing ACO's operation by maximizing the utilization of global information through an enhanced pheromone update mechanism. When the algorithm fails to identify a superior solution, the proposed approach allows the pheromones to be maintained as fixed rather than updated.

The following summarizes the key contributions of this paper:

- We propose a novel scheme called iCapS-MS for reducing the data collection time and enhancing the data delivery performance of WSN.
- We propose an improved Capuchin Search Algorithm (iCapSA) to tackle the limitations of the original CapSA algorithm and improve its search abilities.
- We develop an iCapSA-based method for selecting the best DCPs that enhances the coverage of nodes with minimized intersecting coverage among the DCPs for efficient data collection in WSN by allowing the nodes to communicate the data with MS with the least feasible hop distance.
- An e-ACO algorithm is presented for identifying the best MS trajectory through the DCPs.
- iCapS-MS enables the dynamic adjustment of DCP count and positions in response to node failures, leading to a shorter and more efficient MS trajectory for decreased data delivery latency and increased WSN lifetime.

The remainder of this paper is divided into the following sections. Section 2 provides the related research, while the background details are discussed in Sect. 3. Section 4 gives the system model and problem statement. In Sect. 5, we present the proposed iCapS-MS in detail. Simulation results are discussed in Sect. 6. Section 7 concludes the work. Table 1 summarizes the main abbreviations and notations used in this paper.

2 Related research

Numerous research strategies have recently been put forward to lower energy use and data gathering latency by utilizing MS in WSNs [2, 3]. Existing research indicates that there are two types of MS mobility: uncontrolled movement and controlled mobility [1, 4]. The MS travels erratically in the sensing field while in uncontrolled mobility to collect data. The nodes communicate data to the MS while the MS travels within their respective transmission range. Due to the mobility pattern's randomness, the latency in data acquisition, which is crucial in many applications, cannot be guaranteed. In contrast, controllable mobility optimizes MS trajectory to maximize the network efficiency, by allowing for a reduction in data collection delay. MS needs to visit a few specific locations, called sojourn (rendezvous) locations, for data collection [11]. Here, we review the advantages and disadvantages of relative methods in DCP selection as well as planning of MS trajectory.

Some of the existing algorithms have applied clustering-based methods of selecting the data collection points to tackle the problem. In [22], an enhanced Artificial Bee Colony (ABC) method is employed to establish a clustering strategy, and ACO is used to discover the MS trajectory visiting the cluster heads (CHs) for data collection. Park et al. [23] uses a different iterative clustering approach to address the "hotspots" issue, which arises when nodes are placed more densely in some areas than others. Based on node density and remaining energy, MS determines the optimal number of clusters to visit. An online suboptimal approach is devised in [24], where the ideal position for the sink is determined by using a primal-dual approach with limited background information. In [25], a CH selection mechanism using nature inspired firefly method with improved results than existing models is discussed. In [20], the MS trajectory is designed using ACO covering an appropriate set of rendezvous points. Rendezvous points are selected so that each rendezvous point can serve the greatest number of CHs through single-hop communication. The technique performs better in terms of data acquisition and power consumption. A decentralized data gathering method employing MS for WSNs is recommended in [9], where each link's data flow varies depending on the source's rate of data generation. They used the computationally intensive and NP-hard Traveling Salesman Problem (TSP) to tackle the trajectory formation of MS. A lightweight solution is always recommended for

Table 1 Summary of main abbreviations and notations

Notation	Description
CapSA	Capuchin Search Algorithm
ACO	Ant Colony Optimization
DCP	Data Collecting Points
MS	Mobile Sink
iCapSA	Improved Capuchin Search Algorithm
e-ACO	Enhanced Ant Colony Optimization
F_{j}	Food location in the *j*th dimension
P_{ef}	Elasticity probability of the capuchin motion on the ground
P_{bf}	Balancing probability or balance factor
ζ	Random number generated uniformly in [0, 1]
τ	Lifespan factor of capuchins
P_{r}	Random search probability of capuchins
Hcn	Average hop count
Hdis	Average hop distance
D	Set of DCPs
$\mathrm{OH}_{\mathrm{co}}$	Coverage rate of DCP
$\mathrm{OH}_{\mathrm{int}}$	Intersecting coverage of DCP
H_{d}	Hop distance
H_{c}	Hop count

the constrained sensor nodes since distributed solutions cause the nodes' power consumption to increase quickly. An MS-based method is suggested by Gharaei et al. [26] to extend the network lifetime by balancing the CHs' power consumption. In order to gather the data, the MS awaits at every cluster for a reasonable amount of time. A fuzzy clustering-based technique is proposed by Verma et al. [10], which takes into account the energy of the nodes to increase WSN lifespan. It's an improvement on the LEACH method that uses fuzzy clustering rather than k-means. A Neural-Fuzzy-based method is given in [27]. In [15], a minimal spanning tree approach for choosing rendezvous points is suggested. In this study, a computational geometric technique is used to establish the path planning for MS. Miao et al. [28] considers the reduction of the data collecting delay utilizing MS in multi-hop networks. By transforming the problem into a TSP, the solution is found. Although the clustering-based approach accomplishes a set of goals, it also puts additional overheads on the nodes throughout the clustering and data collection processes. Additionally, the energy usage for packet relaying through nodes toward the MS continues to increase, which has an impact on the WSN lifespan. We focus on eliminating this overhead in our proposed iCapS-MS paper and enable the majority of nodes to deliver data directly to the MS with just one hop.

A directed spanning tree-based rendezvous technique is proposed in [16] considering non-uniform data restrictions, and an improved version is presented in [14]. The findings indicate improvements in network lifetime and data collection latency. The authors studied data collection and path formation employing MS in three-dimensional WSN and presented them in [29]. The research in [30] offered an ambient crop field evaluation for enhancing context-based agriculture utilizing WSNs. By doing so, the MS sent the freshly received data from the field to the BS with a primary focus on it. The energy-aware path construction method suggested by [31] employs an MS for collecting and communicating data from nodes to BS. Depending on the count of data packets and the separation between two adjacent rendezvous points, the initial rendezvous points selection is made. For MS trajectory planning, convex polygons are used, and the convex-hull technique is used to design the route. [32] presented a density-aware data gathering by building the MS trajectory that takes into account the nodes' remaining energy. DEDC divides the region initially into a number of grids, the size of which is based on the MS travel distance. Prior to refining the path to take into account the uneven grids, DEDC builds a normal path in order to evenly distribute relaying loads and increase WSN lifetime. The outcomes demonstrate reduced energy usage and WSN lifespan. Unfortunately, the technique proved ineffective as it only collected a small amount of data when traveling through sparse grids with few or no sensor nodes. [33] presented an information-gathering technique that enables the MS to get data from fixed nodes with a shorter route. This method not only ensures that all the nodes are covered for data gathering, but it also reduces the path length by determining the transmission rate. However, the algorithms discussed above guarantee the minimized latency requirement of applications by offering a shorter trajectory for MS, the expended energy is comparatively greater than one-hop data communication owing to multi-hop communication. As an alternative to these methods, we use enhanced ACO to plan the MS trajectory via the designated set of DCPs that makes a trade-off between the multi-hop paths and the MS trajectory, where the planned trajectory is adaptive to node failures to produce a shorter trajectory with improved coverage and less latency.

Some of the current algorithms facilitate single-hop communication for data forwarding to reduce the energy usage of the nodes. In order to minimize the time needed for data collection, the shortening of the trajectory of the MS is carried out. An MS trajectory planning method with moth flame optimization for WSNs is discussed in [13]. In [11], an optimal rendezvous selection and MS trajectory design is provided. With this method, the rendezvous selection and MS trajectory are effectively identified.

In [34], rendezvous nodes are chosen to use a distributed technique following a game theoretic approach, and the MS trajectory is designed using ACO. Three different types of path planning are introduced in [35], including reduced energy, reduced delay, and delay-bound path, respectively. An effective sparrow search-based data collection method for WSNs is suggested in [36]. Comparing the findings to previous approaches, the method exhibits improved coverage of nodes. The authors of [12] present an MS path planning strategy, where the rendezvous points are notated by their position and communication range of the MS. MS-based data collection in [37] is achieved by traversing a set of network points, where a larger number of nodes are addressed for simultaneous data gathering. Multi-objective PSO is used in [17] to balance the data loads at rendezvous points and shorten the MS path. Instead of using node locations, the rendezvous locations are chosen from within the transmission range overlaps among the nodes. A route encoding technique that chooses an arbitrary number of rendezvous points is developed in order to construct the MS path. The ultimate result is increased performance in terms of energy usage, delay, and WSN lifespan. Additionally, in [38], the rendezvous points are chosen based on the nodes' locations, communication overlaps, and data accessibility. The intersection sets that are produced by communication overlap between nodes are seen as possible visiting faces by the MS.

We can notice that different heuristic techniques have been used to overcome MS-based data collection issues [18]. As demonstrated in past research, fewer DCPs are preferable since they result in shorter MS paths and require less time for data collection. The downside of this is that it could leave a number of nodes exposed and uncovered at a DCP, compelling the rest of the nodes to perform multi-hop transmission to relay data toward MS. This increases the expended energy of the network. Furthermore, the previously mentioned MS-based methods do not preserve node coverage for the MS by the respective DCPs, particularly after the first node dies (FND), since they do not take any coverage adaptive criteria into consideration. The designed path is also not resilient to node failures. Hence, we are motivated to design an effective approach, called iCapS-MS, for selecting the best DCPs that enhance the node coverage by the respective DCPs with minimized intersection of coverage among the DCPs, and constructing the MS trajectory for WSNs that allows for reduced data collection time and enhanced data delivery performance. Although some heuristic techniques have minimized the problem complexity, they do not provide the best results in regard to both energy utilization and delay since they do not emphasize node coverage of the DCPs and do not update the designed MS trajectory in reference to node failures for improved coverage.

3 Background

In this section, we provide the background details of the different methods we have utilized in the proposed work.

3.1 Capuchin Search Algorithm (CapSA)

CapSA is a novel optimization technique inspired by the wandering and hunting behaviors of capuchin monkeys across riverbanks and trees in forests in search of food [21]. CapSA can be modeled based on three motion mechanisms, as follows:

Jumping motion: Capuchins traverse extensive distances over trees to locate food resources, resembling a global search mechanism. In this context, the capuchins' movement between trees reflects the principles of projectile motion, which can be described by Newton's third law of motion, as below [21]:

$$x = x_0 + v_0 t + \frac{1}{2}\alpha t^2 \tag{1}$$

where x, x_0, v_0, α, and t are the capuchin's new location, original location, original velocity, acceleration, and time span, respectively. This equation can be modified as follows [21]:

$$x = x_0 + \frac{{v_0}^2 \sin(2\theta_0)}{g} \tag{2}$$

where g and θ_0 are the acceleration of gravity and the leaping angle, respectively.

Wavering motion Capuchins sway on the branches of trees during the foraging process, a behavior identical to pendulum motion. This motion emulates the local search phase and can be represented as below [21]:

$$x = L\sin(\theta) \tag{3}$$

where L is the tail length and θ denotes the wavering angle.

Climbing motion Capuchins climb trees in their search for food, a motion that mimics the local search process in an optimization method. The climbing motion can be represented as below [21]:

$$x = x_0 + v_0 t + 0.5(v_f - v_0)t^2 \tag{4}$$

where v_f is the capuchin's ultimate velocity and t is the current iteration.

CapSA population CapSA's first step is to create a swarm of n capuchins in an d-dimensional search space using upper and lower limits (ub and lb), as shown below [21]:

$$x_i = ub_j + \zeta \times (ub_j - lb_j) \tag{5}$$

where $\zeta \in [0, 1]$ is a random number, ub_j and lb_j are the respective upper and lower bounds of ith capuchin in the jth dimension. The population is split into two groups: group leaders (alpha) and followers. The leader jumps between the branches to locate the food resource, and this can be described as follows [21]:

$$x_{ij} = F_j + \frac{P_{bf}(v_{ij})^2 \sin(2\theta)}{g}, \quad i < \frac{n}{2}, \quad 0 \le \zeta < 0.2 \tag{6}$$

where F_j is the food location in the jth search space, P_{bf} is the balancing probability provided by the capuchins tail, v_{ij} is the ith capuchin velocity in the jth search space, θ is the jumping angle and n is the size of the population, respectively.

CapSA possesses the benefit of balancing the exploration and exploitation stages using the lifespan factor τ given by [21]:

$$\tau = \beta_0 e^{-\beta_1 \left(\frac{t}{T}\right)^{\beta_2}} \tag{7}$$

where β_0, β_1, and β_2 are experimentally set constants with values 2, 21, and 2, and t and T are the current and maximal iterations, respectively. The velocity of ith capuchin corresponding to jth search space while tree jumping is calculated as [21]:

$$v_{ij} = \rho v_{ij} + \tau\alpha_1(x_{bestij} - x_{ij})r_1 + \tau\alpha_2(F_j - x_{ij})r_2 \tag{8}$$

where ρ is the inertia factor, x_{bestij} is the best location of the capuchin, r_1 and $r_2 \in [0, 1]$ are random numbers, F_j is the food source location, and α_1 and α_2 are two positive values taken as unity.

Capuchin motion The leader and the other capuchins' motions can be represented as follows:

- In the event of a capuchin jumping on the ground, the new location is given by [21]:

$$x_{ij} = F_j + \frac{P_{ef} P_{bf} (v_{ij})^2 \sin(2\theta)}{g}, \quad i < \frac{n}{2}, 0.2 \leq \zeta < 0.3 \tag{9}$$

  where P_{ef} and P_{bf} are the elasticity probability of the capuchin motion on the ground and the balancing probability, respectively.
- The motion of the alpha leader on the ground can be represented as below [21]:

$$x_{ij} = x_{ij} + v_{ij}, \quad i < \frac{n}{2}, 0.3 \leq \zeta < 0.5 \tag{10}$$

- During the wavering phase, alpha, and other capuchins employ local foraging strategies, and their respective locations are given by [21]:

$$x_{ij} = F_j + \tau P_{bf} \sin(2\theta), \quad i < \frac{n}{2}, 0.5 \leq \zeta < 0.75 \tag{11}$$

- The capuchin location in the climbing phase is given by [21]:

$$x_{ij} = F_j + \tau P_{bf} (v_{ij} - v_{i(j-1)}), \quad i < \frac{n}{2}, 0.75 \leq \zeta < 1.0 \tag{12}$$

  where $v_{i(j-1)}$ denotes the prior velocity of ith capuchin in the jth search space.
- Capuchins also explore in random directions for better food, which can be described as [21]:

$$x_{ij} = \tau \times (lb_j + \zeta(ub_j - lb_j)), \quad i < \frac{n}{2}, \zeta \leq P_r \tag{13}$$

  where P_r is the random search probability assigned to 0.1. This setting improves CapSA's global search capability and prevents it from falling into the local minimum.
- The followers adjust their positions based on alpha's new locations as follows [21]:

$$x_{ij} = \frac{1}{2}(x'_{ij} + x_{(i-1)j}), \frac{n}{2} \leq i \leq n \tag{14}$$

  where x'_{ij} is the alpha capuchin's current location in jth search space, and $x_{(i-1)j}$ is the followers' prior locations vector in jth search space.

3.2 Gaussian mutation

Gaussian mutation has been demonstrated to enhance the search efficiency of various meta-heuristic techniques in the literature [39, 40]. It enhances population diversity, improves search speed, and optimizes convergence. The Gaussian mutation density function is given by [39]:

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} \tag{15}$$

where σ represents the standard deviation of the Gaussian distribution. This equation simplifies to generate a random number from a Gaussian distribution with mean 0 and variance 1.

3.3 Cauchy mutation

Cauchy mutation method is based on the Cauchy distribution [41]. Cauchy distribution has a density function similar to the Gaussian distribution, however, the primary difference lies in the fact that the Cauchy distribution is narrower vertically, whereas the Gaussian distribution is broader horizontally. The Cauchy distribution density function is defined as follows [41, 42]:

$$C(y) = \frac{1}{2} + \frac{1}{\pi} \arctan\left(\frac{y}{g}\right) \tag{16}$$

where g is the ratio parameter with a fixed value of 1, and y is a number with uniform distribution $\in [0, 1]$.

3.4 Levy mutation

Lévy flight, inspired by the foraging behavior of various species in nature [43], is an optimal random search strategy. Unlike the uniform Gaussian distribution, Lévy flight employs the Levy distribution to generate step sizes. Lévy mutation approach enables the optimization algorithm to explore its surroundings in shorter steps while also incorporating occasional large jumps, preventing it from getting stuck in local minima. The Lévy flight follows a simple power law equation, $L(S) \sim |S|^{-1-\beta}$ in which S is step size and $0 < \beta \leq 2$. The Lévy distribution can be defined as follows [43]:

$$L(S) = \frac{u}{|\, v \,|^{1/\beta}} \tag{17}$$

where u and v follow the standard normal distribution: $u \sim N(0, \sigma_u^2)$, $v \sim N(0, \sigma_v^2)$,

$$\sigma_u = \left(\frac{\Gamma(1+\beta) * \sin(\frac{\pi\beta}{2})}{\Gamma[\frac{1+\beta}{2}] * \beta * 2^{\frac{\beta-1}{2}}} \right)^{\frac{1}{\beta}}, \sigma_v = 1 \tag{18}$$

where Γ represents the standard gamma function.

3.5 Ant colony optimization algorithm

ACO is a population-based meta-heuristic technique for tackling challenging combinatorial optimization issues, such as TSP [19]. In ACO, each individual in the population is an artificial agent (or artificial ant) that produces a solution to the issue gradually and stochastically [34]. The artificial pheromones are used by the ants to form a probabilistic model that is employed to decide throughout the solution construction stages. In order to boost the likelihood that subsequent agents will produce better answers, the agents continuously enhance the probabilistic model. This technique is continued until a termination condition is met.

4 System model and problem statement

We consider a WSN comprised of randomly distributed nodes in a square region and a single MS. The MS passes through the sensor field at a constant speed and has a sufficient power supply. The nodes are deployed during the setup procedure to cover the complete region [34, 36]. The nodes are conscious of their geographic location using a GPS device or a localization mechanism [11, 13], and hence they know their neighbors. Nodes communicate among themselves when they are in their transmission range and use a wireless link to do so. We define DCPs as distinct geographical locations where the MS pauses to gather data. We consider that the MS traverses the network on a regular basis to collect data, and that nodes send the data to the MS if it is available at their respective DCP. MS possesses the same communication range as that of the nodes, and it uses its trajectory to collect data from the nodes. A node is considered covered if at least one DCP is within its communication range. Nodes upload their data through a single hop when the MS is within the range. Any node that is not inside the MS's range must rely on nearby nodes. The nodes are powered by batteries, which have a limited capacity. When their power runs out, they are considered inoperable.

We assume the same energy model as adopted in [34, 35]. Consider E_{elc} and E_{am} represent the energy expended by the electronic circuitry of the transmitter/receiver and the transmitter amplifier, respectively. The energy required to send a packet of size p bits over a distance of k is given by:

$$E_{\text{trn}}(p,k) = E_{\text{elc}} * p + E_{\text{am}} * k^2 * p \tag{19}$$

The energy required to receive a packet of size p-bits over a distance of k is given by:

$$E_{\text{rec}}(p) = E_{\text{elc}} * p \tag{20}$$

The objective of our work is to find the trajectory for MS that passes through a set of selected DCPs for data collection. To enhance the data collection performance, an improved Capuchin Search Algorithm (iCapSA)-based MS sojourn location optimization and enhanced Ant Colony Optimization (e-ACO)-based MS trajectory design is proposed. We determine the potential set of DCPs so that most of the nodes are covered in one-hop communication with the shortest feasible hop distance and hop count. The range of at least one DCP must be able to reach every node in order to achieve our objective of having a sufficient number of DCPs. Because if a node is within the coverage range of a DCP, it can communicate with the MS directly in one hop. In addition, both the average communication distance and hops at any DCP should be kept minimum. When the MS comes to a halt at a DCP location, the range of DCP mimics the communication range of MS. Hence, any node within the DCP range can deliver data in single-hop. However, we must keep in mind that the MS visits each DCP during data collection; so, the greater the DCPs count, the greater the data-gathering latency. As a result, the best position to get the most out of each DCP would be the one that encompasses as many nodes as feasible while minimizing any intersecting coverage among DCPs. Therefore, we interpret this issue as a coverage problem, where a node is considered covered if at least one DCP is within its communication range. To achieve the highest number of data uploads in single hop communication, we need every node to be covered by at least one DCP. At the same time, to prevent wasting resources by repetitively covering the same nodes, we must also reduce the coverage intersection among DCPs. In this way, the proposed approach enables the majority of the nodes to use single hop connection, with only a few outlier nodes adopting multi-hop transmission. DCPs are assigned by prioritizing one-hop coverage of nodes. Intersecting coverage among DCPs gets minimized and at the same time ensures maxima node coverage. Also, we emphasize resilience in response to node failure conditions in addition to the objective of reducing network energy utilization and decreasing data communication latency. Those nodes that fail because of the lack of energy or other circumstances have to be considered while scheduling the trajectory. To minimize overall trajectory length and data gathering latency, these nodes have to be removed from the planned trajectory. Let $N = \{s_1, s_2, \ldots, s_i, \ldots, s_n\}$ be the number of functional nodes in the network, $D = \{d_1, d_2, \ldots d_k, \ldots d_m\}$ where $d_k \in D$, D denotes the set of DCPs and m is the number of selected DCPs. A node is considered allotted to a DCP according to Eq. 23. We transfer the problem into mixed integer programming as follows:

$$\text{Maximize } \{\text{OHc}\} \;\&\&\; \text{Minimize } \{\text{OHin}, \text{ Hcn}, \text{ Hdis}\} \tag{21}$$

where OHc represents the number of single hop covered nodes at DCPs, OHin represents the coverage intersection among DCPs, and Hcn and *Hdis* represent the average hop count and hop distance at DCPs, respectively, subject to the following constraints:

$$\sum_{k=1}^{m} \text{allot}(s_j, d_k) = 1, \forall s_j \in N \tag{22}$$

given,

$$\text{allot}(s_j, d_k) = \begin{cases} 1, & s_j \in N \quad \text{is allotted to} \quad d_k \in D, \\ 0, & \text{otherwise} \end{cases} \tag{23}$$

And,

$$\text{dis}(s_j, d_k) \times \text{allot}(s_j, d_k) \le c_r, \quad \forall s_j \in N, \forall d_k \in D \tag{24}$$

$$\text{dis}(s_j, s_p) \times \text{allot}(s_p, d_k) \le c_r, \quad \forall s_j, s_p \in N, j \ne p, \forall d_k \in D \tag{25}$$

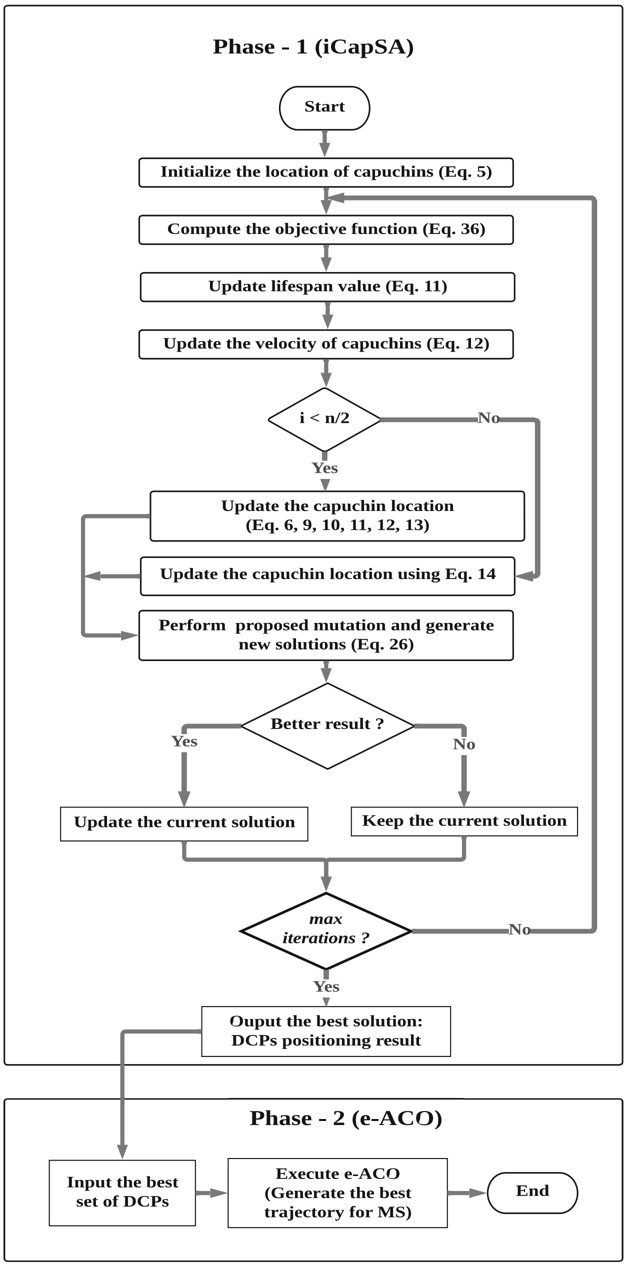
The constraint in Eq. 22 specifies that every node $s_j \in N$ should be assigned to exactly one DCP $d_k \in D$. Constraint in Eq. 24 specifies that all the nodes $s_j \in N$ which comes under the communication range (c_r) of $d_k \in D$ are allotted to it. Furthermore, the constraint in Eq. 25 guarantees that the nodes that are not within the transmission range of any DCP but are within the transmission range of some other node that is allotted to a DCP are covered. That is, if a node s_j is not covered by any DCP but is within the range of a node s_p that is allotted to d_k, then s_j also gets allotted to d_k as well.

5 Proposed iCapS-MS Algorithm

In this section, we provide a detailed discussion of the proposed iCapS-MS algorithm. It operates in two phases. Phase 1: Improved Capuchin Search Algorithm-based selection of DCPs (iCapSA) and Phase 2: Enhanced ACO-based MS trajectory design (e-ACO). Figure 1 gives the flowchart of the proposed iCapS-MS algorithm.

5.1 Phase 1: Improved Capuchin Search Algorithm-based selection of DCPs (iCapSA)

We provide the proposed technique for selecting relevant DCPs where the MS pauses for data collection from the nodes. The capuchin population is used to represent the placement of DCPs, and each capuchin represents a full solution for DCPs' selection. Like other meta-heuristic algorithms, CapSA faces convergence issues or tends to fall into a local minimum. iCapSA is introduced to tackle the limitations of the original CapSA algorithm and improve its search abilities. The main idea of iCapSA is to use different mutation strategies to improve the solution diversity obtained by the optimization algorithm and to overcome the problem of premature convergence and being trapped in the local minima. To mutate the population in a number of ways, mixed mutation operators are implemented. In the proposed iCapSA, three strategies are integrated into the original CapSA: Gaussian, Cauchy, and Levy mutation operation. As a result, this technique drives



Phase - 1 (iCapSA)
Start
Initialize the location of capuchins (Eq. 5)
Compute the objective function (Eq. 36)
Update lifespan value (Eq. 11)
Update the velocity of capuchins (Eq. 12)
i < n/2
No
Yes
Update the capuchin location (Eq. 6, 9, 10, 11, 12, 13)
Update the capuchin location using Eq. 14
Perform proposed mutation and generate new solutions (Eq. 26)
Better result ?
Yes
No
Update the current solution
Keep the current solution
max iterations ?
No
Yes
Ouput the best solution: DCPs positioning result
Phase - 2 (e-ACO)
Input the best set of DCPs
Execute e-ACO (Generate the best trajectory for MS)
End

Fig. 1 iCapS-MS flowchart

search agents to strive for the best result, and the mixed mutation operators are greatly proficient in enhancing population diversity. The mutation technique is an optimization approach that acts on the initial position vector with a random value that follows the normal distribution to create a new result. The proposed iCapSA is coupled with a randomized mutation, where in every iteration, n search agents are mutated as given below:

$$x_i^{'} = x_i \times (1 + \delta) \tag{26}$$

where x_i and $x_i^{'}$ represent the original and the new position of a search agent after mutation, respectively, and δ is a mutation operator. In the iCapSA method, three mutation operations are taken into consideration as follows:

1. Gaussian-based mutated solution $G(x_i^{'})$ is calculated as follows:
$$G(x_i^{'}) = x_i \times (1 + G(\alpha)) \tag{27}$$
where $G(\alpha)$ denotes a random number generated from Gaussian distribution using Eq. 15.
2. Cauchy-based mutated solution $C(x_i^{'})$ is calculated as follows:
$$C(x_i^{'}) = x_i \times (1 + C(y)) \tag{28}$$
where $C(y)$ denotes a random number generated from Cauchy distribution using Eq. 16.
3. Levy-based mutated solution $L(x_i^{'})$ is calculated as follows:
$$L(x_i^{'}) = x_i \times (1 + L(S)) \tag{29}$$

where L(S) denotes a random number generated from Levy distribution using Eq. 17.

The fitness value of $x_i^{'}$, $G(x_i^{'})$, $C(x_i^{'})$, $L(x_i^{'})$ are compared at the end of every iteration, and the best solution from them is selected.

The goal of the proposed iCapSA-based algorithm in this phase is to select the best set of DCPs which allows almost every node to pursue one-hop transmission for uploading data to MS, with only a few outlier nodes making use of multi-hop transmission. Due to the fixed dimensional characteristics of the algorithm, we should first figure out the number of desired DCPs based on the network setting. For this, we use the formula $nD = \lceil * \rceil \frac{A}{\pi c_r^2} + \psi$, where nD denotes the number of DCPs, A is the area of the network, c_r is the nodes' communication range, and ψ is a constant that adjusts for the given topology of the network by taking into account the count of nodes and their positions, respectively. Next, we define the objective function for the proposed iCapSA algorithm. We utilize Eqs. 30 and 31 to define the coverage rate and coverage intersection rate, respectively. The nodes within the communication range of the MS, when it is at $d_k \in D$, are called the neighbors of that d_k.

$$\mathrm{OH_{co}} = \frac{\sum_{j=1}^{N} \mathrm{ohc}_j}{N} \tag{30}$$

$$\mathrm{OH_{int}} = \frac{\sum_{j=1}^{N} \mathrm{int}_j}{\sum_{j=1}^{N} \mathrm{ohc}_j} \tag{31}$$

where N is the number of functional nodes, ohc_j and int_j are defined in Eqs. 32 and 33, respectively.

$$\mathrm{ohc}_j = \begin{cases} 1, & \text{if } s_j \text{lies within the radius of} \\ & \text{atleast one } d_k \in D, \\ 0, & \text{otherwise} \end{cases} \tag{32}$$

$$\mathrm{int}_j = \begin{cases} 1, & \text{if node } s_j \text{lies within the radius of} \\ & \text{more than one } d_k \in D, \\ 0, & \text{otherwise} \end{cases} \tag{33}$$

In order to cover all the nodes with a minimum possible count of DCPs, each of them should cover as many nodes as possible. So, we prefer to maximize the $\mathrm{OH_{co}}$ result. At the same time, we must reduce the intersecting node coverage among DCPs by minimizing the $\mathrm{OH_{int}}$ result. In order to ensure the minimum possible hop count and hop distance at each DCP, we also consider the following changes in the aspect of network topology caused by node failures that affect the coverage of nodes, data transmission, and WSN lifetime. Considering these factors can effectively reduce the latency and enhance the lifetime performance of WSN since this reduction signifies that the expended energy due to the intermediate forwarding is minimized. Let T_d denotes the transmission range of a node s_i which is provided by the Euclidean distance to either the next-hop node or the adjacent DCP through which s_i sends its data to MS, $H_\mathrm{d}(d_k)$ is the average transmission distance of the nodes at a given $d_k \in D$, H_c denotes the hop count of nodes, and $H_\mathrm{c}(d_k)$ denotes the average hop counts at a given $d_k \in D$ of the nodes allotted to d_k through which the data from s_i reaches MS at its corresponding DCP, respectively.

$$H_\mathrm{d}(d_k) = \frac{\sum_i^N \mathrm{allot}(s_i, d_k) \times T_\mathrm{d}(s_i)}{\sum_i^N \mathrm{allot}(s_i, d_k)} \tag{34}$$

$$H_\mathrm{c}(d_k) = \frac{\sum_i^N \mathrm{allot}(s_i, d_k) \times H_\mathrm{c}(s_i)}{\sum_i^N \mathrm{allot}(s_i, d_k)} \tag{35}$$

The average hop distance and hop count of the DCPs in the set D are represented by $A(H_\mathrm{d}(D))$ and $A(H_\mathrm{c}(D))$ and their normalized values are denoted by nH_d and nH_c, respectively.

These criteria are presented as the objective function of the algorithm. We transform the Formula given in 21 as the objective function and our goal is the minimization of this function F_o given by Eq. 36, where $\mathrm{OH_{ci}} = (\mathrm{OH_{int}}/\mathrm{OH_{co}})$, and nH_d and nH_c represent the normalized values of hop distances and hop count at the DCPs, respectively.

$$F_\mathrm{o} = \frac{\mathrm{OH_{ci}}}{nH_\mathrm{d} \times nH_\mathrm{c}} \tag{36}$$

Accordingly, the algorithm returns the best positioning of the DCPs. Sometimes, a few outlier nodes may not be covered, so they must transfer their data to a covered neighbor node, which will subsequently relay that data to the MS at the corresponding DCP. Most of the nodes adopt one-hop communication and only a few nodes, especially the outliers, inevitably utilize multi-hop communication. However, because these outliers are not often, multi-hop communication is rarely employed over the network's lifespan. The algorithm for Phase 1 is provided in Algorithm 1.

Algorithm 1 Phase 1: iCapSA-based selection of DCPs

```
Input: Node locations.
Output: Best location of DCPs
 1: Initialize the location of capuchins using Eq. 5.
 2: while $iteration < Max_{iter}$ do
 3:     Compute the objective function using Eq. 36.
 4:     Update the lifetime value using Eq. 7.
 5:     Update the velocity of capuchins using Eq. 8.
 6:     for $i < n/2$ do
 7:         Update the capuchin location following Eq. 6, Eq. 9 - 13 accordingly.
 8:     end for
 9:     for $i \geq n/2$ & $i \leq n$ do
10:         Update the capuchin location following Eq. 14.
11:     end for
12:     Perform the proposed mutation strategy and generate new solutions based on Eq. 26
13:     Evaluate the fitness and update the current solution if the new result is better than the previous one.
14:     $iteration = iteration + 1$
15: end while
16: Output the best solution (DCPs positioning result).
```

5.2 Phase 2: Enhanced ACO-based MS trajectory design (e-ACO)

The best positions of DCPs are recognized when Phase 1 is completed. The next step is to construct the shortest trajectory for the MS visiting the DCPs, which is Phase 2 of the algorithm. An enhanced ACO-based MS trajectory design is executed in this phase. The MS will visit each DCP following the constructed trajectory to collect data from the nodes. The nodes will communicate the sensed data as well as their energy levels. MS then returns to the BS to deliver the collected data, and if necessary, obtain a revised trajectory for the next round. ACO is a well-known method suited for TSP, which identifies the fastest closed path that passes through a specified collection of data points [19, 34]. At the same time, the selection of hyper-parameters has a direct influence on search process performance, including search process diversification and intensification. Therefore, we utilize the enhanced ACO algorithm for improving the results. To enhance the performance, we make modifications to the working of the existing ACO, where we strive to more effectively utilize the global information by implementing an enhanced pheromone update mechanism. If the algorithm cannot identify a better solution, the proposed approach allows the pheromones to be maintained fixed rather than updated. We describe the collection of DCPs as a completely connected undirected graph denoted by $\mathrm{Gv} = \langle \mathrm{Vr}, \mathrm{Ed} \rangle$ in which Vr is the collection of vertices representing the DCP positions and Ed is the collection of edges linking them. The Euclidean distance between the linked vertices is used to weight these edges. The main steps of Phase-2 are encapsulated in Algorithm 2, which is carried out in the subsequent steps:

1. Set the m ants at random to start at any of the $D = \{d_1, d_2, \ldots d_k, \ldots d_n\}$ available DCPs. For each edge (i, j), set the pheromone concentration $\tau_{ij}(t)$ to a small positive constant c and set $\Delta\tau_{ij}$ to zero.
2. According to the Euclidean distance and pheromone concentration of the following edge, decide for each ant which DCPs it will go next. Ants are only allowed to visit a DCPs that has not yet been visited. The selection probability for the next destination is computed according to Eq. 37, where $p_{ij}^k(t)$ is the probability that the k-th ant will select the path from d_i to d_j

in the present iteration. τ_{ij} denotes the pheromone concentration on the path from d_i to d_j. η_{ij} is the reciprocal of the distance between d_i to d_j, and it denotes the visibility of the destination for the current ant. α and β are parameters that control the importance of the pheromone concentration and the visibility, respectively. Meanwhile, the set allowed$_k$ specifies the collection of DCPs that the k-th ant hasn't yet accessed.

$$p_{ij}^k(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^\alpha [\eta_{ij}]^\beta}{\sum_{k \in \text{allowed}_k} [\tau_{ik}(t)]^\alpha [\eta_{ik}]^\beta} & \text{if } j \in \text{allowed}_k \\ 0 & \text{otherwise} \end{cases} \tag{37}$$

3. Update the tour length so far after moving to the destination using Eq. 38, where l_{ij} is the distance between d_i to d_j. b_{ij} is described in Eq. 39.

$$L_k = \sum_{i,j \in D, i \neq j} l_{ij} b_{ij} \tag{38}$$

$$b_{ij} = \begin{cases} 1 & \text{if } k\text{-th ant toured the edge}(i,j) \\ 0 & \text{otherwise} \end{cases} \tag{39}$$

4. Upon tour completion, update pheromone concentration on all edges (i, j) toured by the k-th ant as shown in Eqs. 40-42. To enhance the performance, we make modifications to the working of the existing ACO as follows. If the algorithm cannot identify a better solution, the proposed approach allows the pheromones to be maintained fixed rather than updated. The parameter ρ is a coefficient that controls the evaporation rate of the pheromones on the trail, Q is a constant, and $\Delta\tau_{ij}^k$ is the pheromone concentration placed on edge (i, j) by k-th ant in the interval t and $t + 1$.

$$\tau_{ij}(t+1) = \begin{cases} \rho\tau_{ij}(t) + \Delta\tau_{ij} & \text{if a better result is obtained} \\ \tau_{ij}(t) & \text{otherwise} \end{cases} \tag{40}$$

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^k \tag{41}$$

$$\Delta\tau_{ij}^k = \begin{cases} \dfrac{Q}{L_k} & \text{if} k\text{-th ant toured}(i,j) in interval t and t+1 \\ 0 & \text{otherwise} \end{cases} \tag{42}$$

5. Repeat the process until the maximum iterations are reached.

Following the completion of the aforementioned procedure, BS sends the information to the MS for its next tour.

Algorithm 2 Phase 2: e-ACO-based MS trajectory design

```
Input: The locations of DCPs.
Output: Best MS trajectory.
1: while trajectory is being built do
2:     Set the ant positions at any of the DCPs $d_i \in D$ (Initialization).
3:     for every ant do
4:         Find the next $d_j \in D$ to go
5:         Revise the current tour duration.
6:         Repeat until the ant has accomplished a tour.
7:     end for
8:     Update the pheromone concentration.
9:     if maximum number of iterations executed then
10:        Break loop.
11:    end if
12: end while
```

5.3 Complexity Analysis

The complexity of the iCapS-MS is the combination of Phase 1 (iCapSA-based DCP selection) and Phase 2 (e-ACO-based MS trajectory design). The complexity of Phase 1 can be determined mainly based on the following steps: initialization, fitness evaluation, and population updating using mutation strategies. Assuming that the algorithm's population size is p, the maximum number of iterations is i, the cost of the objective function is c, the number of evaluation experiments is v and the number of DCPs is k, the time complexity of Phase 1 can be deduced to $O(v.p.i.k)$. During Phase 2, e-ACO-based MS trajectory design through the DCPs is executed. The rate at which the pheromone evaporates has a substantial impact on the algorithm's time complexity, and as a result, the pheromone update value in each iteration relies on the outcomes of previous iterations. In the worst-case scenario, with a given number of ants (a), the time complexity can be expressed as $O(1/\rho(a.k.e.\log k))$, where e, k, and ρ represent the number of edges, DCPs, and the pheromone evaporation rate, respectively. Therefore, the time complexity for Phase 2 can be described as $O(1/\rho(a.k.e.\log k))$. Consequently, the overall time complexity of iCapS-MS is estimated as $O(v.p.i.k) + O(1/\rho(a.k.e.\log k))$.

6 Simulation analysis

In this section, we examine the performance of iCapS-MS, and the results are compared with CTOS [38], TRPMC [12], ESRP-MP [36] and ORPSTC [15]. The simulations are conducted in MATLAB. The simulation environment is considered as a 400 x 400 *m* square area with 100–300 nodes deployed randomly to monitor and collect data. The nodes' sensing and communication radii are set at 25 *m* and 50 *m*, respectively. The initial energy of nodes is 2 *J*. MS is used to collect data from nodes, and its communication range is identical to that of the nodes. MS moves to visit the DCPs at a consistent speed of 2 m/s, waiting 5 s at each DCP to acquire data. We use the same network model to execute and evaluate the performance of all compared algorithms. In this study, the evaluation of energy consumption is based on the radio energy dissipation model [12, 34–36], which accounts for the distance between the transmitter and receiver. The energy and communication models employed in this research are similar to those outlined in [12, 35, 36]. Specifically, the energy expended by the electronic circuitry of both the transmitter (E_{elc}) and the receiver is quantified at 50 nJ/bit. Meanwhile, the transmitter amplifier (E_{am}) consumes 100 pJ/bit/m^2. To ensure a meaningful comparison, we have adopted other parameters consistent with those found in previous works [12, 15, 36, 38]. Table 2 discusses the main parameters employed in this research.

We discuss the results of the simulation experiments under different performance metrics as follows:

1. Number of selected DCPs, which gives the count of DCPs selected for planning the MS trajectory under different algorithms.
2. Percentage of node coverage, which measures the covered nodes by the DCPs during the operation of WSN.
3. Intersecting coverage among DCPs, which measures the overlapping or redundant coverage among DCPs.
4. Trajectory length, which is the total distance traveled by the MS.
5. Tour time of MS, which measures the time taken by the MS to finish its trip.
6. Energy consumption, which is the total energy utilized by nodes in communication.
7. Network lifetime, which gives the lifetime of the network under different algorithms (first node dead).

6.1 Number of data collecting points (DCPs)

Figure 2 shows the number of DCPs encountered in the MS tour to accomplish data gathering. A reduced number is usually preferred given that most of the nodes are covered since minimal stopping points result in a shorter travel path and less delay in data collection. During simulation, the initial number of nodes is fixed to $n = 200$. We evaluate the performance of the algorithms in terms of the number of DCPs with respect to varying percent of failed nodes. TRPMC, ESRP-MP, ORPSTC and CTOS have a greater number of DCPs than our planned iCapS-MS. The number of DCPs in TRPMC is preset during network construction and remains consistent throughout the network's lifespan. Meanwhile, ESRP-MP, ORPSTC, CTOS, and iCapS-MS involve a dynamic number of DCPs that vary according to the network's node percentage. The number of DCPs is shown to decrease as the percentage of nodes decreases. The number of DCPs drops consistently as the percentage of nodes in the network decreases for both techniques. ESRP-MP has the second lowest number of DCPs. CTOS, on the other hand, generates several more DCPs than the other algorithms considering the same number of deployed nodes owing to the fact that CTOS identifies DCPs based on the intersection of the communication ranges of the nodes. Moreover, there is no option for updating DCP positioning, and thus more DCPs are necessary to continuously cover most of the nodes.

6.2 Percentage of node coverage by DCPs

This metric assesses the percentage of nodes that the DCPs have covered. The results are depicted in Fig. 3. The amount of energy expended by nodes for communication directly relates to the number of nodes covered by DCPs. A node can use single-hop communication to submit its data directly to the MS if it is covered by a DCP. Multi-hop communications increase network energy consumption since the nodes that are not covered by a DCP must depend on a neighbor node to relay their data to the MS instead of being able to upload data directly. During simulation, the initial number of nodes is fixed to n = 200. We evaluate the performance of the algorithms in terms of the percentage of node coverage by DCPs with respect to varying percent of failed nodes. With the rise in the percent of node failures, the percentage of node coverage by DCPs is found to become more reduced in ORPSTC and TRPMC than iCapS-MS, ESRP-MP, and CTOS. As can be seen in the previous result, TRPMC and CTOS use more DCPs than the proposed iCapS-MS in order to cover the nodes. ORPSTC has the least coverage among the algorithms. It utilizes a spanning tree-based approach and the disadvantage is the combination of multi-hop transmission. If at all feasible, CTOS, ESRP-MP, and iCapS-MS locate DCPs to provide coverage of all nodes with high significance to facilitate one-hop communication. In terms of node coverage, their performance is essentially comparable. However, as has been previously observed, more DCPs are considered (or approximately twice as many DCPs (by CTOS)) to obtain the same output, and this has an adverse effect on MS trajectory length and data collection time, as demonstrated in the results that follow.

Table 2 Parameters and their values

Parameter	Value
Network dimension	$400 \times 400\,\text{m}^2$
Nodes count	100–300
Sensing range of nodes	25 m
Node's communication range	50 m
Communication range of MS	50 m
Initial energy of nodes	2 J
Energy consumption of transmitter circuit (E_{elc})	50 nJ/bit
Amplifier parameter for free-space model (ϵ_{f})	10 pJ/bit/m^2
Amplifier parameter for multi-path model (ϵ_{m})	0.0013 pJ/bit/m^4
Velocity of MS	2 ms^{-1}
Data packet size	500 bits
MC wait time at each $d_i \in D$	5 s
iCapSA population size	100
β_0, β_1, β_2	2, 21, and 2
P_r	0.1
Ants count m in e-ACO	40
Pheromone concentration control factor α in e-ACO	1
Control factor β in e-ACO	2
Evaporation co-efficient ρ in e-ACO	0.5

6.3 Intersecting node coverage among DCPs

This metric gives an indication of DCP coverage that is redundant or overlapping. Figure 4 illustrates the intersecting node coverage among DCPs. The best DCP locations are those that can offer complete coverage of all nodes in the network while minimizing DCP redundant coverage. This is because the DCPs are the locations toured by the MS for data collection from the nodes. If the node coverage among DCPs intersects, their positions are not the best since the nodes' coverage is redundant. MS has to visit each DCP during data collection. Consequently, as the count of DCPs increases, the data-gathering latency also increases due to the increased trajectory length. According to Fig. 4, TRPMC has the lowest percentage of redundant coverage, followed by iCapS-MS and ESRP-MP. However, there is a trade-off between node coverage and intersecting coverage of DCPs. Node coverage suffers as a result of emphasizing redundancy reduction in TRPMC, as shown in Fig. 3. For ESRP-MP, redundant coverage percentage about 26%. Meanwhile, ORPSTC and CTOS do not consider DCP redundant coverage at all, and the percentage of coverage intersection goes as high as in the range of 61.6–82.5%, which is highly redundant. However, iCapS-MS emphasizes minimizing the intersecting coverage while guaranteeing full node coverage. Therefore, the redundant coverage among DCPs in iCapS-MS may go up

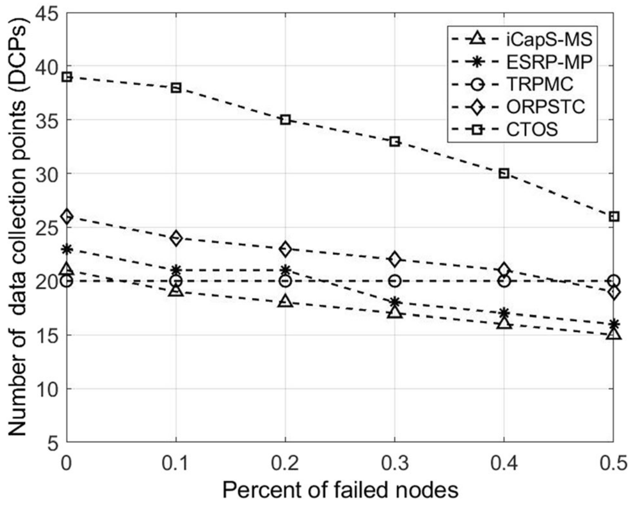


Fig. 2 Number of data collecting points

to only 24.6% depending on the percentage of nodes in the network.

6.4 MS trajectory length

This is the overall length of the MS trajectory for a varying percentage of sensor nodes. During simulation, the initial number of nodes is fixed to n = 200. We evaluate the performance of the algorithms in terms of MS trajectory length with respect to the change in the percent of failed nodes. Figure 5 illustrates the overall distance traveled by the MS to complete its tour under varying percentages of failed nodes. It is critical to decrease the length of the projected trajectory whenever feasible in order to reduce data transmission delay. TRPMC generates the path just once during network initialization, hence the path length stays unchanged irrespective of the number of nodes in the network. ORPSTC readjusts the path length by adopting a re-selection strategy. CTOS, ESRP-MP, and iCapS-MS select a trajectory depending on the count and location of the nodes. We can observe that iCapS-MS outperforms all the other methods, followed by ESRP-MP and ORPSTC, and later, CTOS. Furthermore, when the percent of failing nodes in the network gets larger, ORPSTC and CTOS are unable to plot their path across the changing topology. The number of DCPs will drop as the percent of failed nodes increases, resulting in lesser node coverage and increased multi-hop communication in them, which is also evident from the previous figures. This implies that the scheduled trajectory cannot be optimized to reach the maximum number of nodes, and overall network energy usage will rise as a result of the higher hop count and hop lengths involved in multi-hop communications. In contrast, the proposed iCapS-MS maintains a favorable trajectory that allows for data gathering in a more efficient manner by prioritizing maximum coverage of nodes, reduced hops, and reduced intersecting coverage of DCPs.

6.5 MS tour time

Figure 6 depicts the entire tour time of the MS for the algorithms. It is influenced by two key factors: the length of the MS trajectory and the number of visited DCPs (Figs. 5 and 2, respectively). Moreover, MS traverses the entire distance at a constant speed and pauses at each DCP for data upload by the nodes. During simulation, the initial number of nodes is fixed to n = 200. We evaluate the performance of the algorithms in terms of MS tour time with respect to varying percent of failed nodes. When compared to others, iCapS-MS has a shorter travel distance. Furthermore, when the percentage of nodes decreases in iCapS-MS, the path length and number of DCPs lowers. As the percentage of failing nodes increases, the overall tour time for iCapS-MS lowers, and the data transmission latency decreases. TRPMC's overall tour time remains constant during the network's operation. ESRP-MP has a shorter tour time than CTOS and ORPSTC. When the number of DCPs is large, the MS must wait for data at each of them for the defined interval, and as a result, the data delivery latency for existing methods will worsen.

6.6 Total energy consumption

It is the overall energy expended by the communication of nodes. Figure 7 depicts the total energy utilized by nodes for communication. The covered nodes can connect to the MS directly and communicate in a single hop. This conserves energy and increases the lifetime of the WSN. During simulation, the initial number of nodes is fixed to n = 200. We evaluate the performance of the algorithms in terms of total energy consumption with respect to the change in the percent of failed nodes. ORPSTC has the greatest total energy usage, since many nodes are forced to undertake multi-hop communication owing to a lack of DCP coverage. Meanwhile, ESRP-MP and CTOS provide higher coverage and indeed consume less energy than

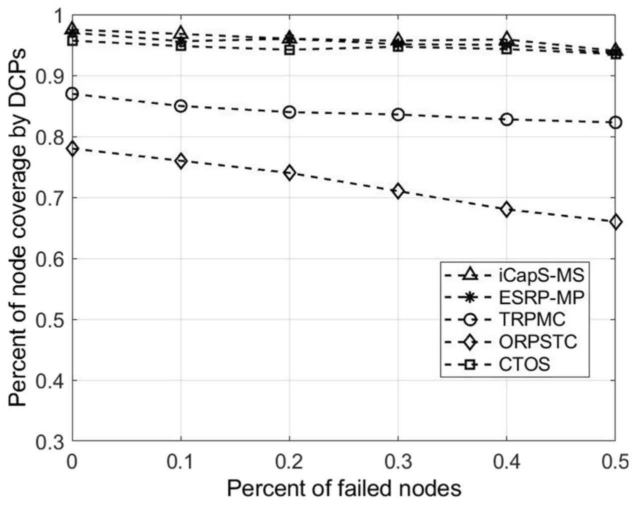


Fig. 3 Percentage of node coverage by DCPs

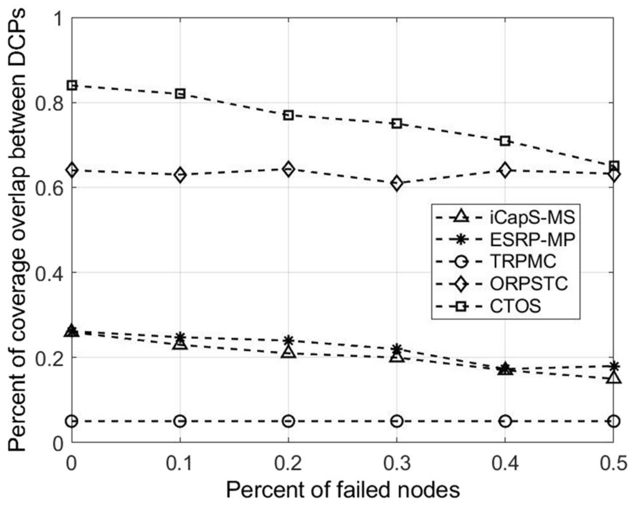


Fig. 4 Percentage of intersecting node coverage among DCPs

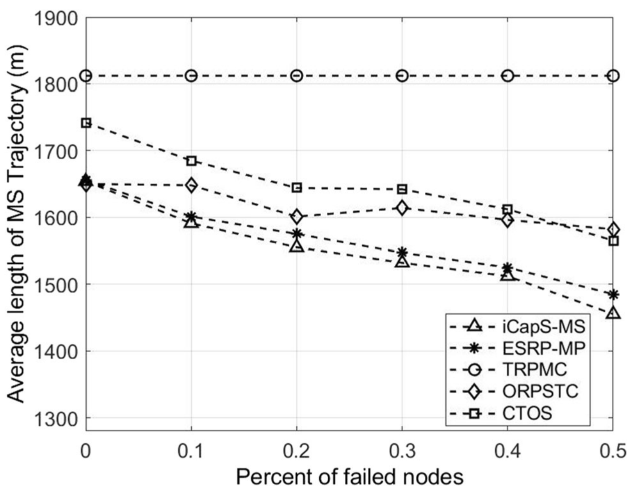


Fig. 5 MS trajectory length

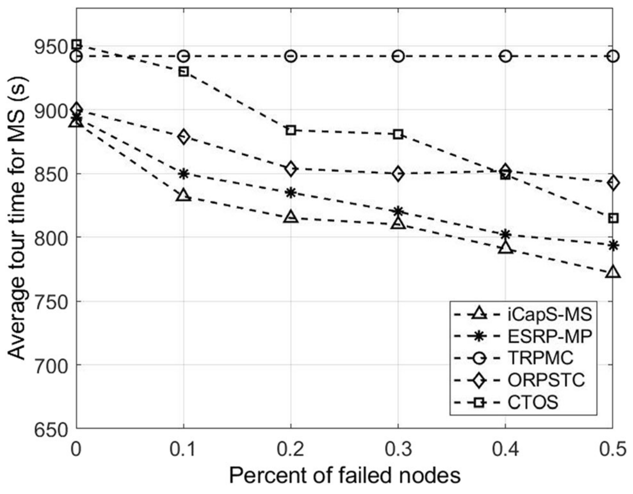


Fig. 6 MS tour time

ORPSTC. TRPMC also has greater transmission energy usage despite ensuring one-hop connectivity. However, based on the above results, we can infer that iCapS-MS is more energy efficient due to enhanced DCP selection and MS trajectory planning, which significantly enhances the data communication efficiency when compared to previous algorithms.

6.7 Network lifetime

The performance benefit of the methods on network lifetime is demonstrated in Fig. 8. The graph depicts how much time it takes for the first node to die (FND) in the WSN when each method is used. Because of the appropriate selection of DCPs and MS trajectory planning, the network remains functional for a longer period of time under the proposed iCapS-MS method.

6.8 Overall analysis

Path planning for WSN data collection using MS is a crucial and complex issue. The network's overall performance heavily depends on the chosen path. If the planned path is too, the MS will take a considerable amount of time to traverse it, leading to significant network latency. Conversely, if the path is too short, many sensors may not have the opportunity to directly interact with the mobile collector, resulting in inefficient energy utilization for data forwarding. Striking a balance between a short path and optimal coverage is, therefore, a challenging task. The proposed iCapS-MS maintains a favorable trajectory that allows for data gathering in a more efficient manner by prioritizing maximum coverage of nodes, reduced hops, and reduced intersecting coverage of DCPs. It not only reduces energy consumption and latency, but also demonstrates robustness and adaptability in the face of network topology changes caused by node failures. In TRP-MC, the number of DCPs is predetermined at the initiation of the network and remains constant throughout the network's operation. In contrast, ORPSTC, ESRP-MP, CTOS and iCapS-MS approaches dynamically adjust the number of DCPs based on the number of nodes present in the field, allowing for adaptability as the network evolves. CTOS generates considerably more DCPs than others because CTOS chooses DCPs in terms of intersecting communication ranges of the nodes, and requires more DCPs for coverage of the nodes. ORPSTC and CTOS do not consider DCP redundant coverage at all, and have high redundant coverage of more than 60%. However, the redundant coverage among DCPs in iCapS-MS is only upto 24.6% depending on the percentage of nodes in the network. CTOS, ESRP-MP, and iCapS-MS strategically position DCPs to ensure comprehensive node coverage, prioritizing one-hop communication. In terms of node coverage, their performance is quite similar. However, it's noteworthy that, as observed previously, a relatively larger number of DCPs (approximately twice by CTOS) is employed to achieve the same level of coverage. This increased DCP count has a negative impact on both MS trajectory length and data collection time, as evidenced in the experimental results. iCapS-MS is more energy efficient due to enhanced DCP selection and MS trajectory planning, which significantly enhances the data communication efficiency when compared to others, and the network remains functional for a longer period of time with a considerable extension of WSN lifetime by 3.56–43.6% compared to existing schemes.

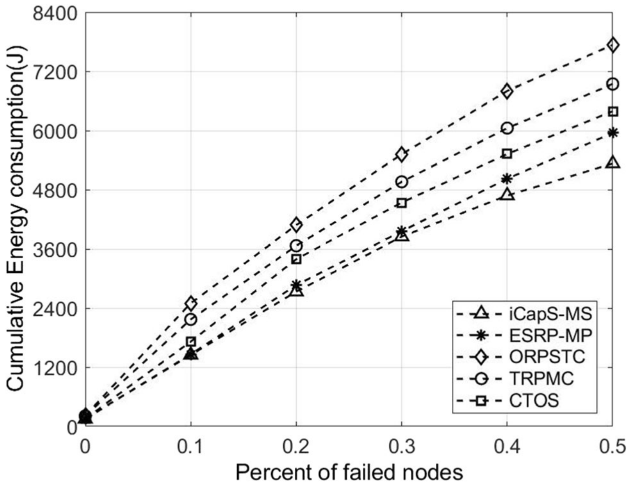


Fig. 7 Total energy consumption by nodes in communication

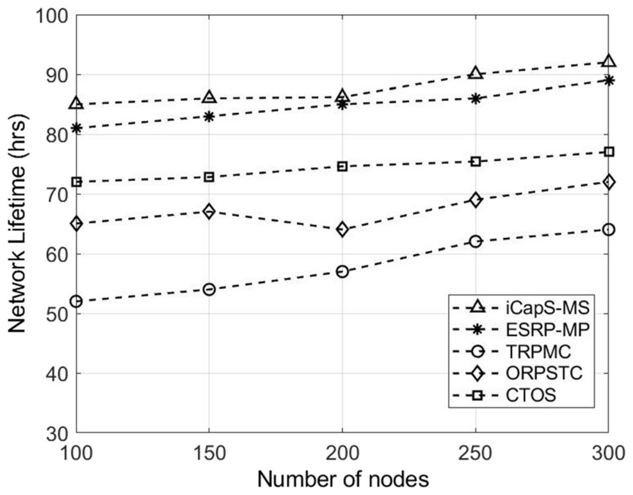


Fig. 8 Network lifetime (FND)

7 Conclusion

The performance of MS-based WSN is significantly influenced by effective trajectory design. In this paper, an effective algorithm, called iCapS-MS, is proposed based on two approaches, viz. an improved Capuchin Search Algorithm (iCapSA)-based MS sojourn location optimization and enhanced Ant Colony Optimization (e-ACO)-based MS trajectory design. An iCapSA-based algorithm is utilized in the first phase for selecting the best DCPs such that almost every node can communicate data in single-hop with the least feasible hop distance and coverage intersection between DCPs is minimal. The trajectory for MS is determined in the second phase using an enhanced ACO method. The results demonstrate that iCapS-MS outperforms existing methods based on several performance metrics. The redundant coverage among DCPs in iCapS-MS is only 24.6% whereas in the majority of existing algorithms, it exceeds 60%. On average, iCapS-MS demonstrates notable improvements, including about 4.7–12.67% reduction in MS tour time and a considerable extension of network lifetime by 3.56–43.6% compared to existing schemes.

In the future, we aim to consider different swarm intelligence methods and also plan to test the proposed method for other QoS parameters. Moreover, we intend to extend the research to solve other real-world WSN data collection challenges.

Author Contributions All authors contributed to the study's conception and design. The final document was read and approved by all the authors.

Funding This research received no specific grant from any commercial, public, or non-profit funding source.

Availability of data and material N/A

Declarations

Conflicts of interest There are no conflicts of interest regarding the publishing of this research as stated by the authors.

References

1. Das R, Dash D (2022) A comprehensive survey on mobile sink-based data gathering schemes in WSNs. Adhoc Sen Wirel Netw 52:1–43
2. Khedr AM (2015) Effective data acquisition protocol for multi-hop heterogeneous wireless sensor networks using compressive sensing. Algorithms 8(4):910–928
3. Osamy W, El-Sawy AA, Khedr AM (2020) Effective TDMA scheduling for tree-based data collection using genetic algorithm in wireless sensor networks. Peer-to-Peer Netw Appl 13(3):796–815
4. Kamble AA, Patil B (2021) Systematic analysis and review of path optimization techniques in WSN with mobile sink. Comput Sci Rev 41:100412
5. Khedr AM, Ramadan H (2011) Effective sensor relocation technique in mobile sensor networks. Int J Comput Netw Commun (IJCNC) 3(1):204–217
6. Osamy W, Salim A, Khedr AM, El-Sawy AA (2021) IDCT: intelligent data collection technique for IoT-enabled heterogeneous wireless sensor networks in smart environments. IEEE Sens J 21(18):21099–21112
7. Kumar DP, Amgoth T, Annavarapu CSR (2019) Machine learning algorithms for wireless sensor networks: a survey. Inf Fusion 49:1–25
8. Khedr AM, Al Aghbari Z (2023) Raj PP (2022) MSSPP: modified sparrow search algorithm based mobile sink path planning for WSNs. Neural Comput Appl 35:1363–1378
9. Tao L, Zhang XM, Liang W (2019) Efficient algorithms for mobile sink aided data collection from dedicated and virtual aggregation nodes in energy harvesting wireless sensor networks. IEEE Trans Green Commun Netw 3(4):1058–1071

10. Verma A, Kumar S, Gautam PR, Rashid T, Kumar A (2020) Fuzzy logic based effective clustering of homogeneous wireless sensor networks for mobile sink. IEEE Sens J 20(10):5615–5623
11. Mehto A, Tapaswi S, Pattanaik K (2021) Optimal rendezvous points selection to reliably acquire data from wireless sensor networks using mobile sink. Computing 103(4):707–733
12. Gao Y, Wang J, Wu W, Sangaiah AK, Lim S-J (2019) Travel route planning with optimal coverage in difficult wireless sensor network environment. Sensors 19(8):1838
13. Sapre S, Mini S (2021) A differential moth flame optimization algorithm for mobile sink trajectory. Peer-to-Peer Netw Appl 14(1):44–57
14. Donta PK, Amgoth T, Annavarapu CSR (2020) An extended ACO-based mobile sink path determination in wireless sensor networks. J Ambient Intell Human Comput 10(11):3
15. Gutam BG, Donta PK, Annavarapu CSR, Hu Y-C (2021) Optimal rendezvous points selection and mobile sink trajectory construction for data collection in WSNs. J Ambient Intell Human Comput 14(14):1–12
16. Kumar P, Amgoth T, Annavarapu CSR (2018) ACO-based mobile sink path determination for wireless sensor networks under non-uniform data constraints. Appl Soft Comput 69:528–540
17. He X, Fu X, Yang Y (2019) Energy-efficient trajectory planning algorithm based on multi-objective PSO for the mobile sink in wireless sensor networks. IEEE Access 7:176204–176217
18. Al Aghbari Z, Khedr AM, Osamy W, Arif I, Agrawal DP (2020) Routing in wireless sensor networks using optimization techniques: a survey. Wirel Pers Commun 111(4):2407–2434
19. Dorigo M, Gambardella LM (1997) Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans Evol Comput 1(1):53–66
20. Roy S, Mazumdar N, Pamula R (2021) An optimal mobile sink sojourn location discovery approach for the energy-constrained and delay-sensitive wireless sensor network. J Ambient Intell Human Comput. https://doi.org/10.1007/s12652-020-02886-z
21. Braik M, Sheta A, Al-Hiary H (2021) A novel meta-heuristic search algorithm for solving optimization problems: capuchin search algorithm. Neural Comput Appl 33(7):2515–2547
22. Wang Z, Ding H, Li B, Bao L, Yang Z (2020) An energy efficient routing protocol based on improved artificial bee colony algorithm for wireless sensor networks. IEEE Access 8:133577–133596
23. Park J, Kim S, Youn J, Ahn S, Cho S (2020) Iterative sensor clustering and mobile sink trajectory optimization for wireless sensor network with nonuniform density. Wirel Commun Mobile Comput. https://doi.org/10.1155/2020/8853662
24. Deng R, He S, Chen J (2018) An online algorithm for data collection by multiple sinks in wireless-sensor networks. IEEE Trans Control Netw Syst 5(1):93–104
25. Chauhan V, Soni S (2020) Mobile sink-based energy efficient cluster head selection strategy for wireless sensor networks. J Ambient Intell Humaniz Comput 11(11):4453–4466
26. Gharaei N, Bakar KA, Hashim SZM, Pourasl AH (2019) Inter- and intra-cluster movement of mobile sink algorithms for cluster-based networks to enhance the network lifetime. Ad Hoc Netw 85:60–70
27. Verma A, Kumar S, Gautam PR, Kumar A (2021) Neural-fuzzy based effective clustering for large-scale wireless sensor networks with mobile sink. Peer-to-Peer Netw Appl 14(6):3518–3539
28. Miao Y, Sun Z, Wang N, Cao Y, Cruickshank H (2017) Time efficient data collection with mobile sink and vMIMO technique in wireless sensor networks. IEEE Syst J 12(1):639–647
29. Donta PK, Rao BSP, Amgoth T, Annavarapu CSR, Swain S (2019) Data collection and path determination strategies for mobile sink in 3d WSNs. IEEE Sens J 20(4):2224–2233
30. Khan T, Kumar DS (2020) Ambient crop field monitoring for improving context based agricultural by mobile sink in WSN. J Ambient Intell Humaniz Comput 11(4):1431–1439
31. Wen W, Zhao S, Shang C, Chang C-Y (2018) EAPC: energy-aware path construction for data collection using mobile sink in wireless sensor networks. IEEE Sens J 18(2):890–901
32. Wen W, Shang C, Chang C-Y, Roy DS (2020) DEDC: joint density-aware and energy-limited path construction for data collection using mobile sink in wsns. IEEE Access 8:78942–78955
33. Chang C-Y, Chen S-Y, Chang I-H, Yu G-J, Roy DS (2020) Multirate data collection using mobile sink in wireless sensor networks. IEEE Sens J 20(14):8173–8185
34. Raj PP, Khedr AM, Al Aghbari Z (2020) Data gathering via mobile sink in WSNs using game theory and enhanced ant colony optimization. Wireless Netw 26(4):2983–2998
35. Alsaafin A, Khedr AM, Al Aghbari Z (2018) Distributed trajectory design for data gathering using mobile sink in wireless sensor networks. AEU-Int J Electron Commun 96:1–12
36. Khedr AM, Al Aghbari Z, Raj PP (2022) An enhanced sparrow search based adaptive and robust data gathering scheme for WSNs. IEEE Sens J. https://doi.org/10.1109/JSEN.2022.3167515
37. Naghibi M, Barati H (2020) EGRPM: energy efficient geographic routing protocol based on mobile sink in wireless sensor networks. Sustain Comput: Inf Syst 25:100377
38. Dash D, Kumar N, Ray PP, Kumar N (2020) Reducing data gathering delay for energy efficient wireless data collection by jointly optimizing path and speed of mobile sink. IEEE Syst J. https://doi.org/10.1109/JSYST.2020.3019213
39. Bäck T, Schwefel H-P (1993) An overview of evolutionary algorithms for parameter optimization. Evol Comput 1(1):1–23
40. Song S, Wang P, Heidari AA, Wang M, Zhao X, Chen H, He W, Xu S (2021) Dimension decided Harris hawks optimization with gaussian mutation: balance analysis and diversity patterns. Knowl-Based Syst 215:106425
41. Gupta S, Deep K (2018) Cauchy grey wolf optimiser for continuous optimisation problems. J Exp Theor Artif Intell 30(6):1051–1075
42. Ali M, Pant M (2011) Improving the performance of differential evolution algorithm using cauchy mutation. Soft Comput 15:991–1007
43. Kamaruzaman AF, Zain AM, Yusuf SM, Udin A (2013) Levy flight algorithm for optimization problems-a literature review. Appl Mech Mater 421:496–501