SPARTA-GEMSTONE: A two-phase approach for efficient node placement in 3D WSNs under Q-Coverage and Q-Connectivity constraints

Vu Quang Truong, Trinh The Minh, Nguyen Thi Hanh, Trinh Van Chien, Huynh Thi Thanh Binh, Nguyen Xuan Thang

***Abstract*—Achieving target coverage and addressing connectivity issues pose significant hurdles within the realm of wireless sensor networks (WSNs). In real-world scenarios, when distinct targets necessitate varying degrees of priority, each individual target is allocated a value q which signifies both the number of sensors effectively covering the target and the quantity of node-disjoint paths essential for conveying the sensory data from itself to the base station. In instances where $q > 1$, the network can guarantee resilience against faults. These constraints are referred to as the Q-Coverage and Q-Connectivity. Because of the practicality of the problem, various pieces of research are attempted to tackle it. Nevertheless, existing works do not directly aim to minimize the number of nodes under these both constraints simultaneously in realistic 3D environment. Moreover, the type of connection mentioned in most of these works is the connection between sensors, which is not suitable for target-oriented networks. Since this NP-hard problem involves two constraints, this paper proposes a two-phase solution. In Phase I, we utilize a heuristic algorithm called SPARTA with two variances SPARTA-CC and SPARTA-CP to address one of the constraints. In Phase II, we employ a heuristic algorithm based on minimum spanning tree called GEMSTONE to tackle the other constraint. Additionally, we evaluate the performance of these algorithms using our proposed real-life dataset and compare them with baseline methods, namely GLA in Phase I, CMFA and FCSA in Phase II. The results indicate a significant improvement in various evaluation metrics when compared to the baseline methods. These findings are not only beneficial for future research on WSNs but also specifically contribute to the advancement of target coverage.**

***Index Terms*—Wireless sensor network, Target coverage, Node placement, Q-coverage, Q-Connectivity, Connected component, Minimum Spanning Tree**

I. Introduction

In recent years, the Internet of Things (IoT) has garnered significant attention from the general public and the technology community worldwide. One of the essential components of IoT is Wireless Sensor Networks (WSN). Devices need to be equipped with sensor chips to detect phenomena in the physical environment and convert them into data on the Internet for user processing and analysis [1], [2]. Therefore, wireless sensor networks contribute significantly to the success of IoT [3]. Sensors in the network gather data from a specific area and transmit monitoring data to a base station. Subsequently, processing units analyze the data and generate reports to provide meaningful information to users.

N. T. Hanh is with Phuong Dong University (email: hanhnt@dhpd.edu.vn). H. T. T. Binh, V. Q. Truong, T. T. Minh and T.V. Chien are with the School of Information and Communication Technology, Hanoi University of Science and Technology, Vietnam (email: binh.ht@soict.edu.vn, truong.vq194198@sis.hust.edu.vn, minh.tt225513@sis.hust.edu.vn, chien.trinhvan@hust.edu.vn)

WSN finds applications in various fields, including military, healthcare, environmental monitoring, agriculture, industry, transportation, and more [4], [5]. For instance, sensors can be embedded in the human body, on the skin, or on clothing items to monitor health conditions and measure vital signs such as blood pressure, heart rate, blood oxygen levels, body temperature, and glucose levels, enabling appropriate and timely treatment for patients. Additionally, WSN is used to monitor environmental conditions in healthcare facilities, such as temperature, humidity, light, dust, toxic gases, noise, etc. It helps maintain a safe and comfortable working environment and ensures optimal patient care. With the value that WSN brings, there is an increasing number of research and scientific works published to address challenges in sensor networks, such as network lifetime, coverage capacity, connectivity, fault tolerance, load balancing, and security.

Coverage issues can be categorized into three types: target coverage, area coverage, and barrier coverage. This paper focuses on addressing the target coverage problem (ensuring all targets are covered by at least one sensor) using the Binary Disk Coverage Model (BDCM) [6]. BDCM states that a sensor s with sensing range r_s covers an target t if and only if the Euclidean distance between t and s is less than r_s. The target coverage problem is divided into three subproblems: 1-coverage (simple coverage), k-coverage, and Q-coverage. In 1-coverage, each target must be covered by at least one sensor. However, in practice, sensors are prone to errors and failures. Hence, the k-coverage model is proposed to ensure fault tolerance in the network, where each target requires at least k sensors for coverage [7], [8]. Nevertheless, certain applications require different targets to have varying degrees of coverage priority, specifically, the minimum number of nodes needed to monitor each target may differ. This is referred to as Q-coverage [9], [10] , which is the focus of this research problem.

Network connectivity issues also have constraints similar to coverage, specifically, 1-connectivity, k-connectivity [11]–[14], and Q-connectivity. Connections are established to transmit data from targets to the base station through relay nodes. Each relay node has a certain communication range. The Q-connectivity constraint states that each target must have q non-intersecting paths to the base station, and the number of such paths may vary for different targets, similar to Q-coverage.

To the best of our knowledge, the Q-coverage problem in two-dimensional space has been addressed in various sensor network types such as wireless sensor networks, directional sensor networks [15]–[20]. However, authors consider that addressing this problem in a 2D deployment domain is not

suitable for several practical applications such as monitoring targets in smart cities, IoT systems, where smart devices equipped with sensors need to be monitored in real-time and placed in a 3D space.

Regarding coverage and connectivity issues in a 3D space, several studies have tackled this problem [21], [22]. They addressed the Q-coverage by vertex coloring algorithm and 1-connectivity problems by Breadth-first search.

Based on the existing research, we realize that studying the problem of minimizing the number of deployed nodes to satisfy Q-coverage and Q-connectivity constraints simultaneously in a 3D environment is highly suitable for numerous optimized IoT system deployments. Currently, no research has tackled this specific problem. Therefore, this paper focuses on addressing the problem of finding the positions of sensor nodes to satisfy Q-coverage and Q-connectivity constraints with a minimal number of deployed nodes in a 3D terrain, aiming to achieve high scientific and practical significance.

A. Related work

The challenges of coverage and connectivity is a familiarizing concept in network deployment. They are the basis of data exchange and consequently reliable network performance. Thus, there have been various studies on this area of network deployment. This section investigates recent work on WSNs in terms of coverage and connectivity constraints.

1) Coverage: This part presents a concise overview of related works on coverage constraint in WSNs, including 1-Coverage, K-Coverage, and Q-Coverage.

1**-Coverage.** Arivudainambi et al. [21] presented a vertex coloring based sensor deployment (VC-SD) algorithm to determine sensor requirement and optimal spot placement and to obtain 100% target coverage in a 3D region. Mini et al. [23] treated the 3D deployment problem as a clustering problem to be solved with an artificial bee colony algorithm to determine optimal spot placement for minimum sensing range sensors. Additionally, the variation in sensing range and whether node locations are near optimal are studied through the sensitivity analysis test. Chen et al. [24] proposed an improved ant lion optimizer for maximum network coverage by combining Cuckoo Search, Cauchy mutation and differential evolution, which is then shown to have higher convergence speed and accuracy as well as ability to skip local optima. Song et al. [25] proposed a solution using the change step of fruit fly optimization algorithm along with construction of two mathematical network coverage models. Njoya et al. [26] solved the problem of simple coverage with the targetive of minimizing the number of nodes deployed using a new method involving virtual sensors. Virtual Sensor is basically a concept where a sensor can be active or inactive during the process of determining the location of nodes before the actual deployment itself. Virtual sensors can move, merge, recombine or explode, again, during the course of the algorithm, not after the deployment of nodes. Two works [12], [14] introduce an approach for 1-coverage at any point in the surveillance area by utilizing regular pattern.

K**-Coverage.** Kim et al. [27] proposed three regular patterns to ensure p-coverage at any point in an area of interest. A lowerbound of p is determined by the type of pattern used. Wang et al. [28] tackled K-coverage different constraints for wireless sensor networks in 3D underwater regions by a K-Equivalent Radius enhanced Virtual Force Algorithm which can satisfy diverse K-coverage requirements for practical application. Ozdag et al. [29] proposed Maximum Area Detection Algorithm based on Whale Optimization Algorithm to for the K-coverage in 2D problem, which notably outperformed previously suggested Maximum Area Detection Algorithm based on Electromagnetism-Like Algorithm in terms of energy consumption. Gupta et al. [30] inspected the problem of placing sensor nodes in pre-fixed potential positions so that k-coverage and m-connectivity are fulfilled and the number of positions selected is minimized. They proposed a metaheuristic approach based on GA. The proposed algorithm was shown to have far better time complexity and yielded better results than other GA-based approaches. Hanh et al. [31] proposed a heuristic approach to tackle K-coverage in 2D domain. Their proposal is proved to be superior to the prior GA approach.

Q**-Coverage.** Balaji et al. [22] proposed solving the problem obliging Q-Coverage constraint in 3D for deterministic deployment with an evolutionary algorithm called particle swarm optimization algorithm. This approach has accomplished simulation results significantly outperforming random deployment. Afizudeen et al. [32] proposed the Grundy Number based approach for the coverage and connectivity constraints of the Q-Coverage problem in 3D terrain. This method uses Grundy Number based Deterministic Sensor Deployment and provides further improvements in optimality using Grundy Number based Random Deployment algorithm, Grundy Number based Cuckoo Search algorithm and Grundy Number based Genetic algorithm. Singh et al. [33] investigated the optimal deployment for Q-coverage through a matheuristic approach and used a genetic algorithm within a column generation framework. This approach targets basic Q-coverage problem as well as that with variable energy consumption and that with base station connectivity requirement. Hanh et al. [20] solved Q-coverage problem in 2D space by a heuristic approach based on MIP (mixed interger programming). The result is superior to other baseline methods but has high time complexity.

2) Connectivity: This part introduces our survey on existing works regarding connectivity constraint, consisting of 1-Connectivity, K-Connectivity, and Q-Connectivity. Note that $1, K, Q$ are just the notations, and different paper may use other expressions with the same concept.

1**-Connectivity.** Arivudainambi et al. [21] utilized Breadth-first search (BFS) to verify the connectivity between sensors in the network. Afzudeen et al. [32] did the same thing but proposed adding the relay nodes if the connectivity is not achieved. Both of these works operate on 3D environment.

K**-Connectivity.** Most of the multi-connectivity research studies focus on K-connectivity problem, in which each node is connected to K other nodes. These nodes can be sensors or relay nodes. One approach is using Genetic Algorithm (GA) to optimize the number of nodes [30], [34]–[36]. The nodes are placed in predefined positions, which is encoded in the chromosomes. Some other works solve the problem of finding optimal regular pattern providing K-connectivity with some

TABLE I: Comparison of related works on target coverage and connectivity problems

Authors	*Year*	*Dimensions*		*Coverage*			*Connectivity*		
		2D	**3D**	1	K	Q	1	K	Q
Arivudainambi et al. [21]	2020		✓	✓			✓		
Mini et al. [23]	2010		✓	✓					
Chen et al. [24]	2022	✓		✓					
Song et al. [25]	2018	✓		✓					
Njoya et al. [26]	2017	✓		✓					
Bai et al. [12]	2008	✓		✓				✓	
Yun et al. [14]	2010	✓		✓				✓	
Kim et al. [27]	2012	✓			✓			✓	
Wang et al. [28]	2019		✓		✓				
Ozdag et al. [29]	2018	✓			✓				
Gupta et al. [30]	2016	✓			✓			✓	
Hanh et al. [31]	2022	✓			✓			✓	
Balaji et al. [22]	2018		✓			✓			
Afizudeen et al. [32]	2023		✓			✓	✓		
Singh et al. [33]	2013	✓				✓	✓		
Hanh et al. [20]	2023	✓				✓			✓
Gupta et al. [34]	2016	✓			✓			✓	
Mini et al. [35]	2012	✓						✓	
Srivastava et al. [36]	2019	✓						✓	
Pu et al. [13]	2009	✓						✓	

small value of K [12], [14], [27]. A sensor deployment pattern is called regular if the Voronoi polygon generated by each sensor node has the same shape and size. A different definition of K-Connectivity is presented in these works, in which there are at least K disjoint paths between any node. With that concept, Pu et al. [13], proposed a partial K-connectivity repair algorithm, in which they minimized the number of nodes to add in a network so that it becomes partial K-connected (i.e, it guarantees K disjoint paths only between special pair of nodes). Hanh et al. [31] introduced an improved Djikstra algorithm to solve K-connectivity problem in 2D area with the aim to minimize the number of relay nodes.

Q-Connectivity. Our survey reveals that there is only one study solving this problem. Hanh et al. [20] proposed a novel Q-Connectivity concept which is target-centric. Each target t_i has q_i node-disjoint paths to the base station through the corresponding sensors covering it. The authors used a maximum flow approach combined with clustering algorithm to minimize the number of relay nodes.

The conclusion drawn is that though there are numerous solutions developed by scholars to deploys sensor nodes in diverse-scaled regions, most have been under ideal predefined environment behavior, thus making realistic adaptation challenging. Specifically, much of the research has been conducted on the 2D plane or only tackled 1-coverage and K-coverage whilst many real-world scenarios requires more complex 3D deployment to which 2D techniques cannot be extended to. Moreover, none of the related works solve the Q-Connectivity in 3D environment. Table I shows the summary of comparisons of the target coverage and connectivity problem with different initial constraints.

B. Motivations and contributions

In practical applications, the susceptibility of sensor nodes to failure necessitates the need for increased monitoring precision or a fault-tolerant system. Simple coverage is often insufficient, and thus k-coverage presents a viable solution. K-coverage requires each target to be monitored by a minimum of k sensor nodes, where k is a constant integer. If k sensors cover a target simultaneously, then the failure of $k-1$ nodes does not disrupt monitoring as the target is still monitored by at least one sensor. Similarly, if there exist k node-disjoint connection paths from a target to the base station, then the connectivity of the target remains intact even after the failure of $k-1$ nodes. Moreover, k-coverage aids in energy efficiency by enabling the scheduling of sensor nodes for activation and deactivation as required, thus extending the lifetime of the network. However, a significant drawback of k-coverage is that it does not take into account the varying levels of priority that different targets or areas may have in real-world settings. To address this, Q-coverage and Q-connectivity have been proposed, which involve assigning different minimum numbers of sensor nodes or node-disjoint paths to each target based on its level of importance.

Because of the high practical need of solving the above problems, many research studies have been conducted taking into account the multi coverage and connectivity constraints. However, these works mainly focus on extending the network lifetime or scheduling the sensor set with predefined locations of all the nodes. Moreover, when solving the Q-connectivity problem, most of the works address the multi-path connectivity among the sensors without considering the full paths from the targets to the base station, which directly serves the purpose of target-centric WSNs. While a small number of researches have been done solving multi coverage and multi connectivity at the same time [20], [31], these works only consider the ideal 2D environment, which can not be yielded in practice. Besides, the available researches in 3D environment only address the 1-Coverage problem [21] or Q-Coverage with given potential positions for placing sensors [22]. In realistic scenario, getting these positions is often infeasible. Hence, our work is the first to tackle the Q-Coverage and Q-Connectivity problems concurrently with the aim of minimizing the number of nodes. Our contribution is as follows:

- Formulating the node minimization problem under Q-Coverage and Q-Connectivity constraints in 3D environment.
- Proposing a two-phase graph-based solution with superior performance compared to the state of the art algorithms.
- Providing a dataset of targets in real-life 3D environment for the problem.

C. Paper outline and notations

The remaining sections of the paper are structured in the following manner. Section II formulates the problem. Section III presents our proposed method to solve the problem. Section IV elaborates the details of our extensive experiments. Ultimately, Section V provides a conclusion and presents our thoughts on possible directions for future research.

All of the notations in the paper are summarized in Table II.

TABLE II: Notation definition

Notation	Definition
$A(L\times W\times H)$	3D surveillance area with length L, width W, and height H
B	base station
n	number of targets
T	target set
m	number of sensors deployed in phase I
S	sensor set
p	number of relay nodes deployed in phase II
R	relay node set
r_s	sensing range
r_c	communication range
Q	priority vector with q_i is priority level of target t_i
$G(V, E)$	graph G with vertex set V and edge set E
O	sphere set with O_i is the sphere having center at t_i and radius r_s
I^{triad}	intersection points of three spheres
I^{pair}	midpoint of two spheres that intersect
I	$I^{triad} + I^{pair}$
p^{cover}	list of targets that $p \in I$ covers
p^{parent}	list of spheres in the triad or pair that creates p
t^{child}	list of point p such that $t \in p^{parent}$
t^{sensor}	list of sensors that cover target t
C	list of clusters

II. Problem formulation

The problem of Q-coverage and Q-connectivity in a 3D environment for wireless sensor networks can be defined as follows:

Given a known 3D environment with specified locations of targets and a base station, along with a vector Q where each element q_i represents the priority level of target t_i, the objective is to minimize the overall number of deployed sensor nodes and relay nodes. This deployment should satisfy two conditions: (1) target t_i must be covered by q_i sensor nodes, ensuring Q-coverage, and (2) target t_i must have q_i node-disjoint paths to the base station, ensuring Q-connectivity.

Let us consider a surveillance region A with a set of n targets, denoted as $T = \{t_i \mid i \in \{1, ..., n\}\}$, and a base station B whose locations are known. Location of target t is a triplet (t_x, t_y, t_z) where $0 \leq t_x \leq L, 0 \leq t_y \leq W, 0 \leq t_z \leq H$. The same spatial restrictions hold for the base station B, which is located at (B_x, B_y, B_z) where $0 \leq B_x \leq L, 0 \leq B_y \leq W, 0 \leq B_z \leq H$.

Additionally, we have a priority vector $Q = \{q_j \mid j \in \{1, ..., n\}\}$ associated with the targets. Within this context, each sensor node possesses a sensing range r_s and each relay node has a communication range r_c.

In this paper, the Binary Coverage Disk Model (BCDM) [6] is used, which means the detection probability of a sensor is 1 within the sensing range, otherwise, the probability is 0. Let $d(s, t)$ denote the Euclidean distance between sensor s and target t, then:

$$P_d(s,t) = \begin{cases} 1, & \text{if } d(s,t) \leq r_s, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

In addition, the connection between two relay nodes or between a sensor and a relay nodes is also determined by the binary disk model as follow:

$$P_c(s,s') = \begin{cases} 1, & \text{if } d(s,s') \leq r_c, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

In order to formulate the Q-coverage and Q-connectivity constraints, we introduce binary variables b_{ij} and c_i as follows:

$$b_{ij} = \begin{cases} 1, & \text{if sensor } s_j \text{ covers target } t_i, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

$$c_i = \begin{cases} 1, & \text{if target } t_i \text{ has } q_i \text{ node-disjoint paths to } B, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

Finally, the mathematical formulation of the problem is presented as follows:

$$\textbf{Minimize} \quad |S| + |R| \tag{5}$$

$$\textbf{Subject to} \quad \sum_{j=1}^{m} b_{ij} \geq q_i \qquad \forall i = 1, 2, \ldots, n, \tag{6}$$

$$\sum_{i=1}^{n} c_i \geq n, \tag{7}$$

where:

- Equation (5) describes the objective of the problem which is minimizing the total number of nodes.
- Equation (6) formulates the Q-coverage constraint, ensuring that target t_i is covered by at least q_i sensors.
- Equation (7) presents the formulation of the Q-connectivity constraint, which guarantees that target t_i possesses q_i node-disjoint paths leading to the base station.

In the following part, we prove that the above problem is NP-hard.

Theorem 1. *The problem of minimizing the number of nodes satisfying Q-Coverage & Q-connectivity constraint is NP-hard.*

Proof. Consider a specific scenario where we focus exclusively on the Q-Coverage problem, with the assumption that $q_i = 1$ for all $i = 1, 2, \ldots, n$, and the set $\mathcal{F}$ representing the feasible positions of sensors is finite. In this case, the task of minimizing the number of sensors required to cover all discrete targets can be equivalently mapped to the well-known *set-covering problem* in its canonical form. It is worth noting that the decision version of this problem, which determines whether a subset of $\mathcal{F}$ with a given size k can cover all the targets, has been proven to be NP-complete [37]. Consequently, it follows that the Q-Coverage problem is NP-hard, and thus the problem formulated in this paper is also classified as NP-hard. □

III. Proposed methods

To handle the formulated problem, we propose a two-phase graph-based approach. In the first phase, the sensor locations are optimized to satisfy Q-Coverage with an algorithm named **S**phere **P**oint **A**nd **R**emoval **T**arget **A**lgorithm (SPARTA). Then, in the second phase, the relay node coordinates are optimized to satisfy Q-Connectivity with an algorithm named **G**raph-based **E**fficient **M**inimum **S**panning **T**ree **O**ptimization for **N**etwork **E**stablishment (GEMSTONE). These two phases are solved independently with two different algorithms. The flow chart of the approach is presented in Figure 1. Our solution for two phases is elaborated below.

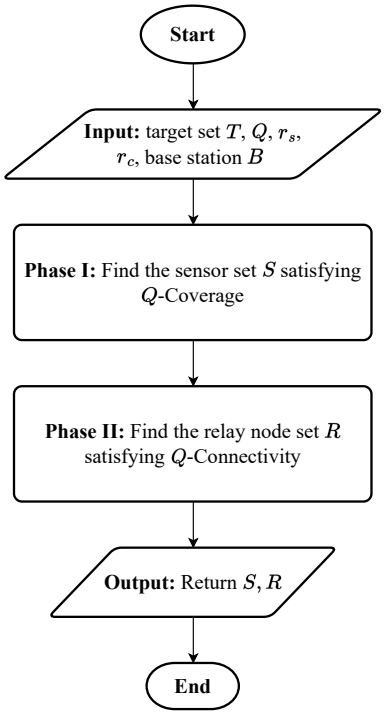
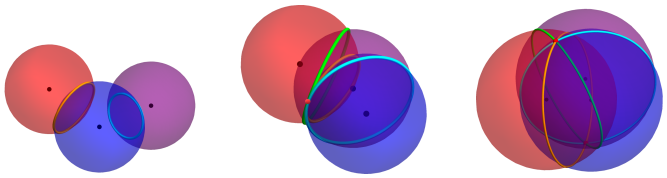


Fig. 1: Flow chart of the two-phase approach



(a) No intersection. (b) 1 intersection. (c) 2 intersections.

Fig. 2: Possible number of intersection points of 3 spheres. Intersection points are marked with red.

A. SPARTA: Sphere Point And Removal Target Algorithm

1) Vanilla SPARTA: In phase I, we propose a algorithm with the main idea of gradually placing sensors into good overlap regions and remove satisfied targets until there are no targets left.

Firstly, we find the good overlap regions through examining the intersection points of spheres in the sphere set O. In this paper, to determine such points, we take 3 spheres at once and calculate the intersection of them. This is also known as the famous Trilateration problem in geopositioning. The solution can be easily calculated [38]. Accordingly, there can be $0, 1$ or 2 intersection points between 3 spheres (see Figure 2). Such points constitute the set I^{triad}.

In order to take advantage of more overlap regions, we also calculate the midpoint of each target pair if their two corresponding spheres intersect. Those points constitute the set I^{pair}. Then we can obtain the set $I = I^{triad} + I^{pair}$.

Note that with each point p found during the above process, p^{cover}, p^{parent} are computed. Besides, t^{child} and t^{sensor} are also calculated for each target t.

After constructing the intersection set I, we perform the following loop until there are no targets left:

- Select 2 points I_i and I_j in I which cover the same and largest set of targets.
- Place minimum sensor between I_i and I_j so that at least the coverage constraint of one target in I_i^{cover} is satisfied.
- Remove the targets whose coverage constraint is satisfied and update the coverage constraint of other targets in I_i^{cover}.
- If $I = \emptyset$, exit the loop. At this point, the remaining targets do not have any intersections with each other. Place the remaining necessary number of sensors inside each corresponding sphere and end the algorithm.

The pseudo code for SPARTA is given in 1.

Algorithm 1: SPARTA: **S**phere **P**oint **A**nd **R**emoval **T**arget **A**lgorithm

Input : n: number of targets
$T = \{t_1, t_2, \ldots, t_n\}$: list of targets
Vector Q: q_i: priority level of target $i, i = 1..n$
r_s: sensing range
$A = L \times W \times H$: surveillance region

Output : Location of the minimum number of sensors needed to satisfy Q-Coverage $S = [(x_1, y_1, z_1), (x_2, y_2, z_2), ..., (x_m, y_m, z_m)]$.

```
Build sphere set O.
I = FindIntersection(O)
while T ≠ ∅ do
    Update p^cover for all p ∈ I.
    if I ≠ ∅ then
        Select 2 point I_i, I_j in I which have the largest
          |I_l^cover| and I_i^cover = I_j^cover.
        q_min = min_{t ∈ I_i^cover} q_t
        Place q_min sensors S_tmp between I_i and I_j.
        for t in I_i^cover do
            t^sensor += S_tmp
            q_t -= q_min
            if q_t = 0 then
                Remove t.
                Remove t^child from I.
            end
        end
    end
    else
        for t ∈ T do
            Place q_t random sensors S_tmp in O_t.
            t^sensor += S_tmp
            Remove t from T.
        end
    end
end
```

Theorem 2. *Assuming that $n >> q_{max}$, the time complexity of SPARTA is $O(n^5)$.*

Proof. Firstly, finding the set $I = I^{triad} + I^{pair}$ takes $O(n^3)$ since we have to loop through all triads and pairs, of which

the quantities are $O(n^3)$ and $O(n^2)$, respectively.

Afterwards, we perform iterations until $T = \emptyset$. In the worst case, we need to loop n times. Within each loop, we first update p^{cover} for all $p \in I$. This takes $O(n^4)$ as we have to loop through all remaining targets (which is $O(n)$) and all remaining elements in I (which is $O(n^3)$), and each append operation takes only $O(1)$.

Next, we consider the worst case where $I \neq \emptyset$. In this case, we need to sort the set I, which contains the intersection of triplets and pairs of spheres. The cardinality of I is $O(n^3)$. Therefore, the sorting step takes $O(n^3 \log n)$.

The step of finding $q_{min} = \min_{t \in I_i^{cover}} q_t$ takes $O(n)$ since I_i^{cover} has $O(n)$ elements.

The placement of q_{min} sensors takes $O(q_{max})$.

The loop over the elements in I_i^{cover} takes about $O(n(n^3 + n))$ because I_i^{cover} has $O(n)$ elements, and the removal of t takes $O(n)$ time, while the removal of t_{child} takes approximately $O(n^3)$ time.

Overall, the total process takes $O(n^3 + n(n^4 + n^3 \log n + q_{max} + n + n(n^3 + n))) = O(n^3 + n(n^4 + q_{max})) = O(n^5)$ as $n >> q_{max}$. □

Note that the above theorem only provides a loose upper bound of the time complexity of SPARTA. In fact, we can efficiently reduce this upper bound by a simple technique. Suppose that we can divide the target set into k subsets and process each set independently. If the initial T has n elements then the complexity of the algorithm is $O(n^5)$. Hence if T is divided into k subsets, each has approximately $\frac{n}{k}$ elements, then the total complexity for executing SPARTA in k subsets is $k * O((\frac{n}{k})^5) = O(\frac{n^5}{k^4})$, which means the upper bound of complexity is reduced roughly k^4 times.

Hence we introduce two innovative strategies for partitioning the target set, which are clique partitioning (CP) and connected component (CC). In both strategies, a graph $G(V, E)$ is constructed with V is the target set. There is an edge between 2 vertexes if and only if the Euclidean distance between them is not greater than $2r_s$ (i.e., two corresponding spheres intersect). The details are presented in the following sections.

*2) SPARTA-CC: **SPARTA** with **C**onnected **C**omponent:* In SPARTA-CC, we find all the connected components in the graph with DFS. Finally, SPARTA is executed on each component. Since there are no intersections between two different components, SPARTA-CC gives the same result as SPARTA but in significantly less time.

*3) SPARTA-CP: **SPARTA** with **C**lique **P**artition:* In the case that the network is dense enough, almost every target belongs to the same connected component, thus SPARTA-CC does not differ much from vanilla SPARTA. Therefore, we leverage the maximal clique partition [39] to split the target set. A clique is a subgraph G' of G so that G' is a complete graph. This strategy can still preserve the intersective relation between spheres but produce smaller thus more partitions compared to SPARTA-CC. This can lead to a reduction in computational time.

In Phase II, the sensor set S obtained in Phase I is used as input to determine the locations of relay nodes that satisfy Q-Connectivity. To achieve this, we introduce an algorithm in the following section that is based on the concept of minimum spanning tree (MST).

*B. GEMSTONE: **G**raph-based **E**fficient **M**inimum **S**panning **T**ree **O**ptimization for **N**etwork **E**stablishment*

The main idea of GEMSTONE is partitioning the sensor set S from Phase I into k disjoint sensor subsets $S_1, S_2, \ldots, S_k$ so that with each target t, all sensors in t^{sensor} belong to different subsets (*). Then with each subset S_i, we can define a complete graph $G_i(V_i, E_i)$ with $V_i = S_i + B$. In each graph, a minimum spanning tree can be formed, thus providing an unique path from each sensor to B. Accordingly, a target t which is covered by a set t^{sensor} of q_t sensors will have q_t node-disjoint connections to B, each corresponding to a different sensor subset.

With the above description, it can be deduced that the number of subsets k must not less than q_{max}. For the sake of simplicity, we choose $k = q_{max}$. The process of creating vertex sets $V_i (i \in \{1, 2, \ldots, k\})$ is illustrated in Figure 3 and Algorithm 2. Specifically, the details of it are elaborated as follows:

- In the first step, each column j corresponds to the set t_j^{sensor}.
- Then in column j, the sensors are reordered so that if a sensor s appears in column $j' < j$, it must appear in the same row as it does in column j' (see Figure 3).
- In each row, duplicated sensors are removed. Finally, add B to row i to form the set V_i.

It is trivial that the subsets created by the above process satisfy the condition (*). After finding the MST for each graph G_i, in the final step, relay nodes are placed along the tree edges. The whole process of the two-phase algorithm is illustrated in Figure 4, 5, and 6.

Theorem 3. *Denote l_{max} as the maximum length of an edge in any MST found by GEMSTONE. Assuming that $n >> l_{max}/r_c$, the time complexity of GEMSTONE is $O(q_{max}^2 n^2)$.*

Proof. Firstly, the sorting step takes $O(n \log n)$.

After that, we have three nested loops. Considering that t_k^{sensor} has $O(q_{max})$ elements and finding an element A in it takes $O(q_{max})$, the total time complexity of these loops is $O(n^2 q_{max}^2)$.

The next loop to form the set V_i takes $O(nq_{max})$.

Ultimately, since Prim's algorithm implemented with adjacency list and Fibonacci heap takes $O(E + V \log V)$ with E and V being the number of edges and vertexes in a graph, respectively. As $E = O(n^2)$ and $V = O(n)$ in our case, the time complexity of Prim's algorithm is $O(n^2)$. Placing relay nodes in $O(n)$ edges of the graph takes $O(nl_{max}/r_c)$, since there are $O(l_{max}/r_c)$ relay nodes on each edge. Hence, the complexity of the final loop is $O(q_{max}(n^2 + nl_{max}/r_c)) = O(q_{max} n^2)$.

Therefore, the time complexity of GEMSTONE is $O(n \log n + n^2 q_{max}^2 + nq_{max} + q_{max} n^2) = O(q_{max}^2 n^2)$. This represents a tight upper bound for the time complexity of GEMSTONE.

□

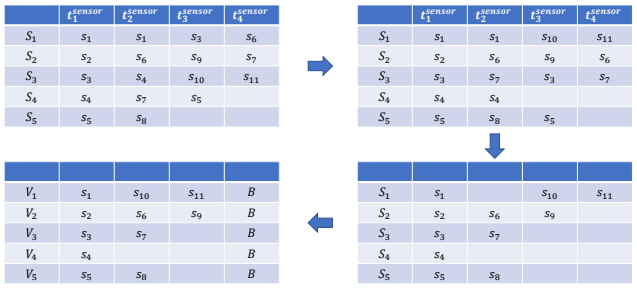


Fig. 3: Illustration of creating vertex sets $V_i (i \in \{1, 2, 3, 4, 5\})$. There are four targets with $q_{max} = 5$.

Algorithm 2: GEMSTONE: **G**raph-based **E**fficient **M**inimum **S**panning **T**ree **O**ptimization for **N**etwork **E**stablishment

Input : n: number of targets
$T = \{t_1, t_2, \ldots, t_n\}$: list of targets
Vector Q: q_i: priority level of target $i, i = 1..n$
r_c: communication range
$A = L \times W \times H$: surveillance region
B: base station

Output : Location of the minimum number of relay nodes needed to satisfy Q-Connectivity $R = [(x_1, y_1, z_1), (x_2, y_2, z_2), ..., (x_u, y_u, z_u)]$.

Sort T in descending order of t^{sensor} length.
for $i = 2$ *to* n **do**
  for $A \in t_i^{sensor}$ **do**
    for $k = 1$ *to* $i - 1$ **do**
      if $A \in t_k^{sensor}$ **then**
        $B = t_k^{sensor}$.index(A)
        t_i^{sensor}.swap(A, B)
      end
    end
  end
end
$V = \emptyset$
for $i = 1$ *to* q_{max} **do**
  for $j = 1$ *to* n **do**
    $V_i += \{t_j^{sensor}{}_i\}$
  end
  $V_i = \text{set}(V_i)$
  $V_i += \{B\}$
end
for $i = 1$ *to* q_{max} **do**
  $E_i =$ **MinimumSpanningTree**(V_i)
  Place relay nodes along the edges in E_i.
end

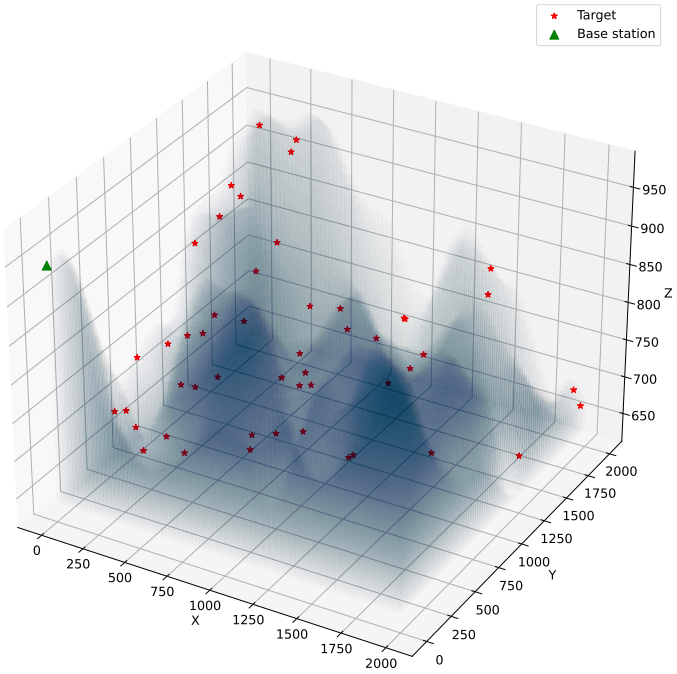


Fig. 4: Illustration of 50 targets in 3D environment with $q_{max} = 2$. The targets are located on the surface of a 2000×2000 mountainous area.

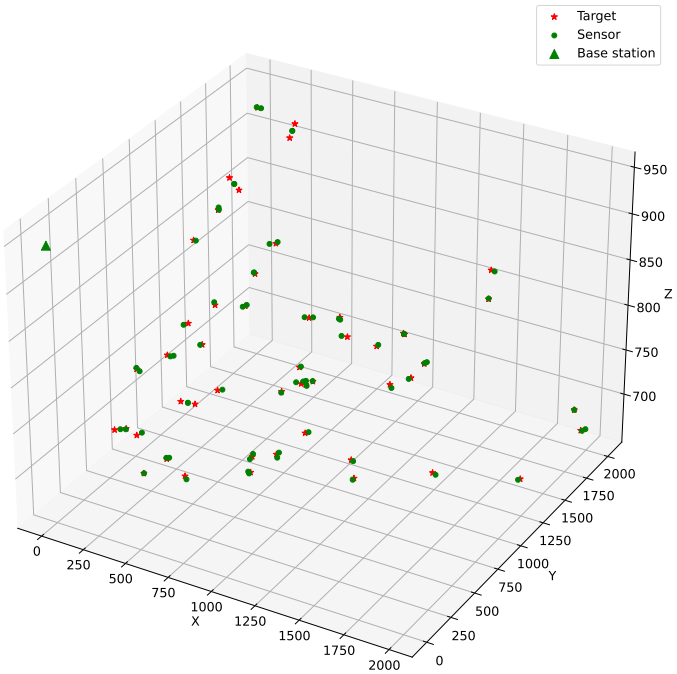


Fig. 5: Result illustration of phase I when $r_s = 40\ (m)$. Sensors are placed to satisfy Q-Coverage.

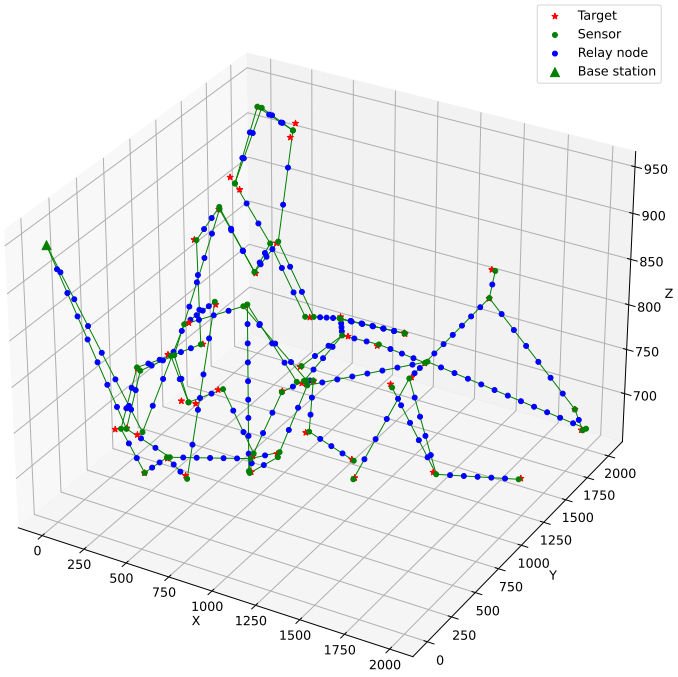
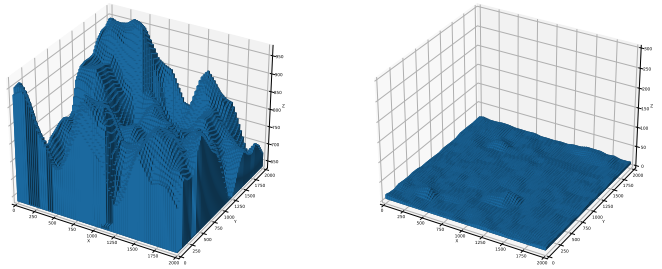


Fig. 6: Result illustration of phase II when $r_c = 80\ (m)$. Relay nodes are placed to satisfy Q-Connectivity. The connections between them and the base station are also plotted.



(a) AoI 1 (high variation in altitude).

(b) AoI 2 (low variation in altitude).

Fig. 7: Illustrations of two datasets.

IV. Experimental results

A. Dataset

Due to the absence of an accessible public dataset pertaining to the given problem, we develop two new distinct datasets derived from divergent geographical settings, namely AoI 1 and AoI 2. AoI 1 has high variation in altitude. Meanwhile, AoI 2 has low variation in altitude. Each dataset corresponds to a region measuring $2000 \times 2000(m^2)$. Both datasets comprise an altitude matrix of dimensions 80×80, where each element denotes the elevation of a $25 \times 25(m^2)$ square area. The illustrations of these datasets are shown in Figure 7, where each entry in the matrix is plotted as a bar with the corresponding height. The altitude matrix creates a surface on which the target locations are later generated randomly on. Note that in this research, obstacles in the environment are not considered. One example is shown in Figure 4.

TABLE III: Scenario 1.1 parameters.

n	$r_s(m)$	q_{max}	**Dataset**
100	40	10	AoI 1
250			
400			
550			
700			
850			

B. Parameter settings

The proposals are implemented in Python 3.9 and executed on AMD Ryzen 7 5800H with Radeon Graphics 3.20 GHz, RAM 16GB DDR4 3200MHz.

Besides, the default parameters and dataset used in the experiments are presented as follows. Note that the default values of r_s, r_c and q_{max} are adopted from [20].

- The default dataset is AoI 1 because AoI 2 does not have much difference with a 2D dataset (i.e., the variation in height is not significant).
- $n = 400$. n 3D target coordinates are randomly generated on the surface of the dataset.
- $r_s = 40(m)$
- $r_c = 80(m)$
- $q_{max} = 10$. The priority value of each target is randomly generated in the range $[1, q_{max}]$.

C. Experiment scenarios

In each phase, we conduct comprehensive experiment scenarios to assess the impact of each factor to the performance of the proposed algorithms, as well as compare them to the baselines. The details of the scenarios are elaborated below.

1) Q-Coverage: In phase I, we compare SPARTA, SPARTA-CC and SPARTA-CP with two other baselines. We keep the default setting and implementation in the original papers and run all the algorithms on our dataset for a fair comparison. The details of the baselines are introduced below.

- **GLA** [20] is a method utilizing the Mixed Integer Programming (MIP) to determine the number of sensors to place in the optimal regions found by the greedy algorithm. Since the original GLA aims to solve Q-Coverage on 2D environment, we apply the same process in SPARTA to find the optimal regions of GLA so that it can work on 3D environment.
- **GN-DSD** [32] applies the Vertex Coloring Algorithm to partition the target set into multiple groups, each of which can be covered by only one sensor. The algorithm also gradually remove satisfied targets and ultimately attempt to remove redundant sensors.

There are three scenarios assessing the impact of changing n, r_s, q_{max} and environment type, as follows:

- **Scenario 1.1.** This scenario evaluates the impact of changing n to the algorithm performance. The number of targets varies from 100 to 850 with stride of 150. There are six network instances involved, which are presented in Table III.
- **Scenario 1.2.** In this particular scenario, we assess the effect of modifying the sensing range (r_s) on the

TABLE IV: Scenario 1.2 parameters.

n	$r_s(m)$	q_{max}	**Dataset**
400	20	10	AoI 1
	35		
	50		
	65		
	80		
	95		

TABLE V: Scenario 1.3 parameters.

n	$r_s(m)$	q_{max}	**Dataset**
400	40	2	AoI 1
		4	
		6	
		8	
		10	

performance of the algorithm. The sensing range is adjusted in increments of $15(m)$, ranging from $20(m)$ to $95(m)$. This evaluation involves six different network instances, which are outlined in Table IV.

- **Scenario 1.3.** This scenario focuses on assessing the influence of the maximum priority value (q_{max}) on the algorithm's performance. We vary q_{max} in increments of 2, ranging from 2 to 10. The evaluation is conducted using five different network instances, as detailed in Table V.
- **Scenario 1.4.** In this particular scenario, we examine the influence of the environment on the performance of the algorithm. The experiments are conducted using two datasets while keeping other parameters fixed as default. This evaluation encompasses two network instances, which are detailed in Table VI.

For each scenario and network instance, we calculate the average number of sensors and runtime as the performance metrics. To obtain reliable results, we repeat the experiments 20 times for each algorithm and record the average values.

2) Q-Connectivity: In the second phase of our method, we conduct a comparative analysis between our proposed GEMSTONE algorithm and two other algorithms as introduced below:

- **CMFA** [20] constructs target clusters and establishes intra-connection within each cluster, as well as interconnection among clusters. To ensure a fair comparison, we adopt the best configuration settings as reported in the original paper, while maintaining the unchanged implementation for CMFA since it can be directly applied to 3D area.
- **FCSA**, as proposed in [20], is a straightforward method wherein every target generates q_i connections between itself and the base station using its corresponding set of sensors. Similar to CMFA, FCSA can be seamlessly implemented in a 3D setting without requiring any adjustments or modifications.

TABLE VI: Scenario 1.4 parameters.

n	$r_s(m)$	q_{max}	**Dataset**
400	40	10	AoI 1
			AoI 2

TABLE VII: Scenario 2.1 parameters.

n	$r_s(m)$	$r_c(m)$	q_{max}	**Dataset**
100	40	80	10	AoI 1
250				
400				
550				
700				
850				

TABLE VIII: Scenario 2.2 parameters.

n	$r_s(m)$	$r_c(m)$	q_{max}	**Dataset**
400	40	80	10	AoI 1
		95		
		110		
		125		
		140		
		155		

The input to all algorithms in phase II is the sensor locations produced by SPARTA-CC in phase I. Besides, the base station is located at coordinate $(0, 0, z)$, where z is the height of point $(0, 0)$ in the selected dataset. To evaluate the impact of different parameters and environmental conditions, we design four distinct scenarios, namely:

- **Scenario 2.1.** This scenario investigates the influence of varying the number of targets (n) on the algorithm's performance. We consider a range of target values from 100 to 850 with a stride of 150. A total of six network instances are utilized for evaluation, as presented in Table VII.
- **Scenario 2.2.** In this scenario, we examine the effect of adjusting the sensing range (r_c) on the algorithm's performance. The sensing range is incrementally modified in steps of $15(m)$, ranging from $80(m)$ to $155(m)$. This evaluation encompasses six distinct network instances, as outlined in Table VIII.
- **Scenario 2.3.** In this scenario, our aim is to evaluate the impact of the maximum priority value (q_{max}) on the algorithm's performance. We systematically vary q_{max} in increments of 2, spanning from a minimum value of 2 to a maximum value of 10. To conduct this evaluation, we utilize five distinct network instances, which are comprehensively described in Table IX.
- **Scenario 2.4.** In this scenario, we investigate the impact of the environment on the algorithm's performance. The experiments are conducted using two distinct datasets while keeping other parameters at their default values. This evaluation includes two network instances, which are described in Table X.

For each scenario and network instance, we calculate the average number of relay nodes and runtime as performance

TABLE IX: Scenario 2.3 parameters.

n	$r_s(m)$	$r_c(m)$	q_{max}	**Dataset**
400	40	80	2	AoI 1
			4	
			6	
			8	
			10	

TABLE X: Scenario 2.4 parameters.

n	$r_s(m)$	$r_c(m)$	q_{max}	**Dataset**
400	40	80	10	AoI 1
				AoI 2

metrics. These metrics are obtained by running the algorithms 20 times and averaging the results.

3) Combined two phases: In this experimental scenario, with the aim of assessing the performance of our comprehensive two-phase algorithm, SPARTA-GEMSTONE, alongside two of its variants is evaluated. We conduct a comparative evaluation with the two-phase approach proposed by Hanh et al. [20], denoted as GLA-CMFA. All experiments are conducted using the default parameter values mentioned in Section IV-B, ensuring consistency and fairness in the evaluation process.

To obtain statistically significant results, each algorithm is executed 20 times, and the resulting metrics are averaged for further analysis. The performance metrics considered for comparison encompass the average number of nodes and average runtime in each phase, as well as the total number of nodes and total runtime.

D. Experiment results and discussions

1) Q-Coverage results:

- **Scenario 1.1.**
  Figure 8 and Table XI depict the outcomes of Scenario 1.1, which focuses on the evaluation of the number of sensors and computational time. In terms of sensor count, SPARTA-CC outperforms all other algorithms and achieves the same result as vanilla SPARTA (as proved in the proposal), while GN-DSD exhibits the least favorable performance. GLA emerges as the second-best algorithm, surpassing SPARTA-CP by a marginal improvement. Specifically, SPARTA-CC is 1.04 times better than GLA and 1.39 times better than GN-DSD when $n = 850$. The superiority of SPARTA-CC over SPARTA-CP can be attributed to its ability to effectively exploit the overlapping regions among target spheres. Conversely, GN-DSD suffers from two major drawbacks that contribute to its inferior performance: its reliance on cliques, which prevents it from capitalizing on the overlap area between cliques, and its inefficient strategy for eliminating redundant sensors due to sequential erasing without comprehensive consideration of the erasing order's impact.
  Regarding runtime, GLA exhibits the slowest performance among the four algorithms due to its time-consuming mixed integer programming approach. SPARTA has nearly the same running time as GLA. Besides, SPARTA-CC proves to be the fastest when the number of targets (n) is small ($n \leq 400$), while SPARTA-CP showcases the fastest performance in all other cases. Particularly, SPARTA-CC can reduce the running time of vanilla SPARTA by 4256 times in the case $n = 400$. When n is small, the disparities in cluster number and cardinality between SPARTA-CC and SPARTA-CP are negligible. Moreover, the time required for finding connected components is shorter than the time needed for identifying the maximal clique partition. Consequently, SPARTA-CC demonstrates slightly superior runtime. However, as n increases, the cluster size in SPARTA-CC escalates rapidly, causing SPARTA-CC to become slower than SPARTA-CP.

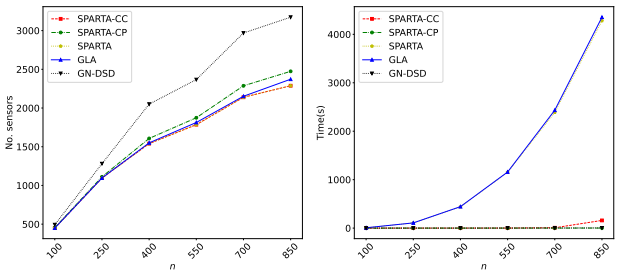


Fig. 8: Performance of the algorithms in Scenario 1.1.

TABLE XI: Performance of the algorithms in Scenario 1.1.

	n	*100*	*250*	*400*	*550*	*700*	*850*
No. sensors	GN-DSD	493	1281	2050	2367	2969	3173
	GLA	449	1096	1550	1813	2152	2372
	SPARTA-CP	455	1111	1607	1874	2287	2474
	SPATAR-CC	**449**	**1096**	**1540**	**1785**	**2139**	**2288**
	SPARTA	**449**	**1096**	**1540**	**1785**	**2139**	**2288**
Time(s)	GN-DSD	0.02254	0.19131	0.62371	1.25556	2.55208	4.21693
	GLA	6.73569	107.25488	442.89113	1157.84351	2434.01801	4353.35800
	SPARTA-CP	0.00561	0.04163	0.12429	**0.26488**	**0.47871**	**0.74173**
	SPATAR-CC	**0.00239**	**0.01186**	**0.10326**	0.68261	6.92089	158.63480
	SPARTA	6.60351	106.06493	439.47009	1153.20148	2389.50916	4280.96800

- **Scenario 1.2.**
  Figure 9 and Table XII provide the results of Scenario 1.2, where the impact of increasing the sensing range (r_s) is examined. Regarding the number of sensors, SPARTA-CC achieves superior performance in all the cases. GLA remains the second-best algorithm, surpassing SPARTA-CP by a small margin. GN-DSD exhibits the least favorable performance among the four algorithms. In a particular case $r_s = 65(m)$, SPARTA-CC is 1.06 times better than GLA and 1.44 times better than GN-DSD. In all cases, the number of sensors decreases as r_s increases due to the increased overlap between target spheres, resulting in more optimal regions for sensor placement.
  Figure 9 also reveals that the runtime of most algorithms experiences only a slight increase as r_s increases, except for SPARTA-CC. Nevertheless, it is still 1.37 times faster than GLA and 1.20 times faster than vanilla SPARTA when $r_s = 95(m)$. This can be attributed to the rapid growth of the size of connected components in SPARTA-CC when r_s increases, whereas cliques in SPARTA-CP face greater difficulty in expanding their size. GLA remains the slowest algorithm among the four, while SPARTA-CP exhibits the fastest runtime when r_s is large (i.e., more overlapping). When $r_s = 95(m)$, it is 2305 times faster than GLA and 2.29 times faster than GN-DSD. The same reasons mentioned in Scenario 1.1 contribute to this outcome.
- **Scenario 1.3.**
  The outcomes of Scenario 1.3 are presented in Figure 10 and Table XIII. The results of all algorithms exhibit nearly linear scaling in relation to the variable q_{max},

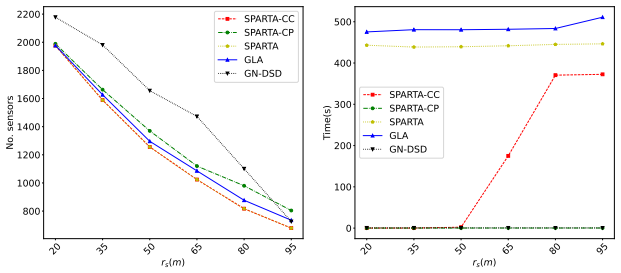


Fig. 9: Performance of the algorithms in Scenario 1.2.

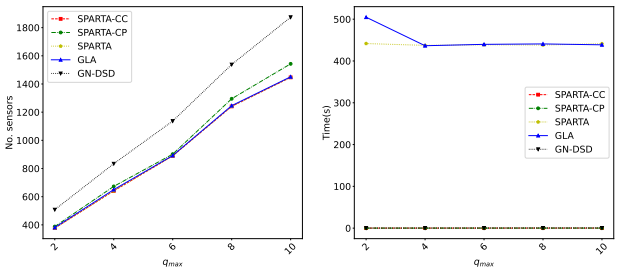


Fig. 10: Performance of the algorithms in Scenario 1.3.

TABLE XII: Performance of the algorithms in Scenario 1.2.

	$r_s(m)$	*20*	*35*	*50*	*65*	*80*	*95*
No. sensors	GN-DSD	2178	1981	1656	1473	1101	726
	GLA	1977	1627	1296	1086	877	735
	SPARTA-CP	1989	1663	1370	1120	980	804
	SPATAR-CC	**1976**	**1589**	**1256**	**1024**	**817**	**679**
	SPARTA	**1976**	**1589**	**1256**	**1024**	**817**	**679**
Time(s)	GN-DSD	0.63355	0.63226	0.60424	0.56801	0.53902	0.50817
	GLA	475.35775	480.76933	480.64530	481.75905	483.63470	510.98250
	SPARTA-CP	0.06972	0.11798	**0.14490**	**0.16947**	**0.19943**	**0.22170**
	SPARTA-CC	**0.02225**	**0.05478**	2.29128	175.03751	370.58600	372.59840
	SPARTA	443.12558	438.68054	439.42220	441.77010	445.09630	446.46020

which represents the maximum allowable quantity of sensors. This behavior can be attributed to the fact that the number of sensors deployed during each iteration of the SPARTA-family algorithm scales linearly with respect to q_{max}. Among the algorithms considered, SPARTA-CC and SPARTA consistently yield the best performance, followed by GLA and SPARTA-CP as the second and third best, respectively, with only marginal differences in their results. Conversely, GN-DSD remains the least effective algorithm in this particular scenario. Specifically, GN-DSD is 1.3 times worse than SPARTA-CC when $q_{max} = 10$.

Regarding runtime, SPARTA-CC proves to be the fastest algorithm across all values of q_{max} examined. The subsequent ordering in terms of increasing runtime is SPARTA-CP, GN-DSD, SPARTA and GLA. The runtime of all algorithms exhibits minimal changes as q_{max} increases due to the fact that the number of iterations remains approximately constant, with only the number of sensors placed during each iteration increasing linearly. Given that the values of n and r_s are fixed, SPARTA-CC consistently achieves the best runtime in all cases where the level of overlap is not substantial enough, as elucidated in previous scenarios. It is 4392 times faster than vanilla SPARTA, 4367 times faster than GLA and 5.78 times faster than GN-DSD when $q_{max} = 10$.

- **Scenario 1.4.**
  The findings from Scenario 1.4, investigating the influence

TABLE XIII: Performance of the algorithms in Scenario 1.3.

	q_{max}	*2*	*4*	*6*	*8*	*10*
No. sensors	GN-DSD	508	834	1137	1538	1874
	GLA	381	650	892	1246	1451
	SPARTA-CP	388	673	902	1295	1543
	SPATAR-CC	**377**	**641**	**890**	**1241**	**1447**
	SPARTA	**377**	**641**	**890**	**1241**	**1447**
Time(s)	GN-DSD	0.20780	0.29585	0.38291	0.52870	0.58103
	GLA	504.73460	436.49640	439.79020	440.79210	438.67990
	SPARTA-CP	0.12060	0.12012	0.12269	0.13145	0.13275
	SPARTA-CC	**0.09725**	**0.09876**	**0.10312**	**0.10046**	**0.10046**
	SPARTA	441.68790	437.32890	438.82250	438.17060	441.26360

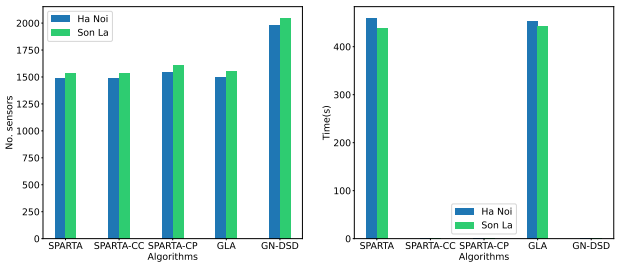


Fig. 11: Performance of the algorithms in Scenario 1.4.

of environment type, are depicted in Figure 11 and Table XIV. The outcomes reveal that the uneven-height area (AoI 1) necessitates a greater number of sensors compared to the even-height area (AoI 2), despite both scenarios having the same sensing range (r_s) and q_{max}. Although the value of n remains constant across the two environments, the high variation in altitude of AoI 1 poses challenges for achieving target sphere overlap. Consequently, the average Euclidean distance between targets becomes longer, leading to a slight increase in the required sensor quantity.

Regarding runtime, the reduced overlap in the flatter environment of AoI 2 results in decreased number of elements in the set I, which leads to less time for sorting, updating the cover set for each point, etc. Moreover, in AoI 2, cluster sizes of both SPARTA-CC and SPARTA-CP also decreases. Additionally, the number of variables in the Mixed Integer Programming (MIP) formulation of GLA also decreases. These modifications contribute to reduced runtime for SPARTA and GLA when operating in a flatter environment. Conversely, in the case of GN-DSD, as it places sensors one by one, its runtime increases with the growing number of required sensors.

2) Q-Connectivity results:

- **Scenario 2.1.**
  Table XV and Figure 12 present the findings from Scenario 2.1, which examines the influence of the target count. Among the various algorithms, GEMSTONE emerges as the most favorable in terms of the number of relay nodes. Specifically, when the number of targets is set to $n = 850$, GEMSTONE achieves a result that is 2.02 times superior to CMFA and 21.81 times better than the FCSA algorithm, which follows a naive approach.

TABLE XIV: Performance of the algorithms in Scenario 1.4.

Dataset		AoI 2	AoI 1
No. sensors	GN-DSD	1979	2050
	GLA	1500	1550
	SPARTA-CP	1549	1607
	SPARTA-CC	**1486**	**1540**
	SPATAR	**1486**	**1540**
Time(s)	GN-DSD	0.61159	0.62371
	GLA	452.38630	442.89110
	SPARTA-CP	**0.12793**	0.12429
	SPARTA-CC	0.13573	**0.10326**
	SPARTA	460.41956	439.47009

Our analysis reveals that regarding the number of relay nodes, CMFA performs poorly compared to our proposed algorithm for two main reasons. Firstly, CMFA's simplistic mechanism of repeatedly selecting the shortest remaining edge and evaluating conditions exhibits a greedy nature that hinders its efficiency. Secondly, CMFA relies on a strong assumption that the paths derived from a node in the graph formed by cluster centers must be node-disjoint to fulfill the Q-connectivity constraint. However, this assumption is unnecessary and ultimately leads to inferior outcomes.

As for runtime considerations, GEMSTONE significantly outperforms CMFA, being 129.31 times faster when $n = 850$, while also achieving superior relay node counts. This discrepancy can be attributed to CMFA's utilization of maximum flow algorithms, which are employed multiple times corresponding to the number of edges in the final graph. This iterative process proves time-consuming. In contrast, GEMSTONE utilizes the Prim's algorithm (which is faster than maximum flow algorithms) just one round for every vertex set. Besides, GEMSTONE's runtime remains comparable to that of FCSA, which follows a straightforward and naive approach. The difference in runtime between GEMSTONE and FCSA is negligible.

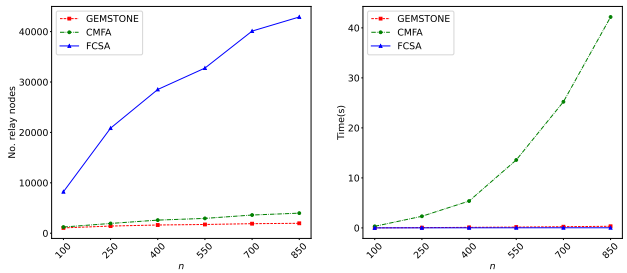


Fig. 12: Performance of the algorithms in Scenario 2.1.

TABLE XV: Performance of the algorithms in Scenario 2.1.

n		*100*	*250*	*400*	*550*	*700*	*850*
No. relay nodes	FCSA	8224	20847	28519	32762	40104	42900
	CMFA	1223	1939	2601	2942	3609	3982
	GEMSTONE	**1058**	**1420**	**1630**	**1734**	**1893**	**1967**
Time(s)	FCSA	**0.00444**	**0.01270**	**0.01886**	**0.02269**	**0.02694**	**0.02578**
	CMFA	0.31839	2.32429	5.36840	13.58329	25.21656	42.17305
	GEMSTONE	0.01064	0.05641	0.11085	0.16450	0.24051	0.32613

- **Scenario 2.2.**
  The outcomes of Scenario 2.2, investigating the influence of the communication range r_c, are depicted in Figure 13 and Table XVI. It is important to note that the value of r_c does not impact the number of links formed by the algorithms but only affects the placement of relay nodes on each link. Consequently, the number of relay nodes is inversely proportional to r_c, resulting in hyperbolic graph shapes.

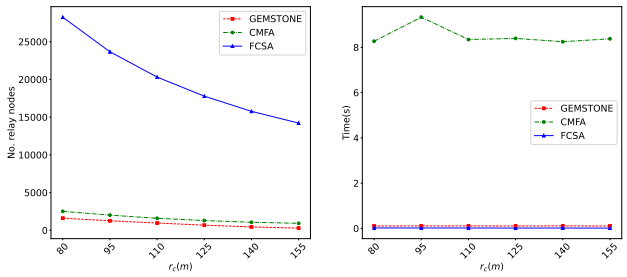


Fig. 13: Performance of the algorithms in Scenario 2.2.

TABLE XVI: Performance of the algorithms in Scenario 2.2.

$r_c(m)$		*80*	*95*	*110*	*125*	*140*	*155*
No. sensors	FCSA	28247	23663	20309	17782	15771	14214
	CMFA	2511	2012	1592	1295	1064	927
	GEMSTONE	**1610**	**1262**	**964**	**682**	**449**	**292**
Time(s)	FCSA	**0.02099**	**0.01928**	**0.01790**	**0.01597**	**0.01647**	**0.01381**
	CMFA	8.27101	9.33300	8.34619	8.39751	8.25033	8.37900
	GEMSTONE	0.10620	0.11004	0.11018	0.10758	0.11217	0.10618

Regarding computational time, the duration required for placing relay nodes is significantly lower compared to that for identifying connections, particularly when the number of targets is substantial. As the connections remain unchanged for different values of r_c, the total runtime of all algorithms remains approximately constant. Furthermore, GEMSTONE proves to be approximately 80 times faster than CMFA and achieves a comparable speed to that of FCSA in this scenario.

- **Scenario 2.3.**
  The results from Scenario 2.3, examining the impact of the variable q_{max}, are presented in Table XVII and Figure 14. In terms of the number of relay nodes, GEMSTONE continues to outperform the other algorithms. Specifically, when $q_{max} = 10$, GEMSTONE exhibits a superiority of 1.55 times over CMFA and 17.10 times over FCSA. Additionally, it can be observed that the number of relay nodes for all algorithms increases linearly in relation to q_{max}.
  In regards to runtime, GEMSTONE is approximately 79 times faster than CMFA when $q_{max} = 10$ while still achieving a comparable speed to FCSA. The Ford-Fulkerson Algorithm employed by CMFA possesses a time complexity of $O(\text{maxflow} \times E)$, where E represents the number of edges in the graph. In our specific problem, the maximum flow cannot exceed q_{max}. Therefore, the computational time of CMFA scales linearly with q_{max}.

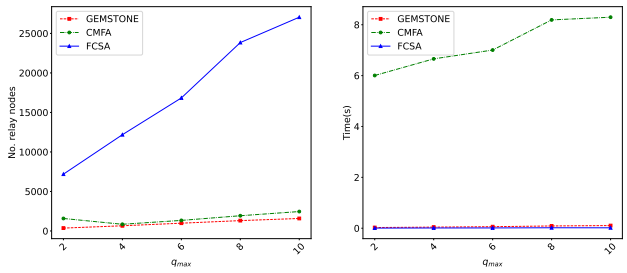


Fig. 14: Performance of the algorithms in Scenario 2.3.

TABLE XVII: Performance of the algorithms in Scenario 2.3.

	q_{max}	2	4	6	8	10
No. sensors	FCSA	7189	12181	16834	23842	27051
	CMFA	1581	848	1338	1928	2458
	GEMSTONE	**363**	**657**	**985**	**1311**	**1582**
Time(s)	FCSA	**0.00521**	**0.00981**	**0.01183**	**0.01854**	**0.01699**
	CMFA	6.00477	6.65820	7.00477	8.18999	8.29570
	GEMSTONE	0.02769	0.04374	0.06098	0.08943	0.10461

- **Scenario 2.4.**
  The findings from Scenario 2.4, investigating the influence of different environment types, are presented in Table XVIII and Figure 15. It is observed that all algorithms yield a slightly higher number of relay nodes when operating in AoI 1, which is a uneven-height area. This can be attributed to the larger distances between targets in such terrain, necessitating the placement of more nodes to establish connections between the targets and the base station. This observation aligns with the explanation provided in Scenario 1.4.

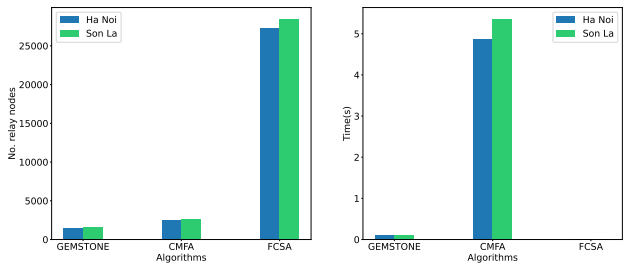


Fig. 15: Performance of the algorithms in Scenario 2.4.

TABLE XVIII: Performance of the algorithms in Scenario 2.4.

Dataset		AoI 2	AoI 1
No. sensors	FCSA	27270	28519
	CMFA	2524	2601
	GEMSTONE	**1527**	**1630**
Time(s)	FCSA	**0.01663**	**0.01886**
	CMFA	4.87683	5.3684
	GEMSTONE	0.10594	0.11085

In terms of computational time, all algorithms demonstrate slightly faster runtimes in the flat area of AoI 2. This disparity arises from the fact that in the uneven-height area of AoI 1, there are more sensors and less overlap between target spheres, which leads to more vertexes in each vertex subset V_i overall. Hence, the MST finding algorithm takes longer to run. Moreover, there is a greater number of relay nodes to be positioned in AoI 1, resulting in a longer time required for node placement compared to AoI 2. Additionally, for CMFA, the sparser distribution of targets in AoI 1 leads to an increase in the number of clusters (i.e., the number of vertices in the graph), which further contributes to longer runtimes.

TABLE XIX: Performance of the algorithms on combined two phases evaluation.

	No. sensors	*Runtime phase I*	*No. relay nodes*	*Runtime phase II*	*No. total nodes*	*Total runtime*
GLA-CMFA	1550	442.89113	2099	64.59364	3649	507.48477
SPARTA-GEMSTONE	**1540**	439.47009	**1630**	**0.11085**	**3170**	439.58094
SPARTA-CC-GEMSTONE	**1540**	**0.10326**	**1630**	**0.11085**	**3170**	**0.21411**
SPARTA-CP-GEMSTONE	1607	0.12429	1593	0.13772	3200	0.26201

3) Combined two phases: Table XIX, Figure 16, and Figure 17 depict the outcomes obtained from the two-phase algorithms. Notably, SPARTA-CC-GEMSTONE exhibits superior performance across all evaluated metrics, significantly surpassing GLA-CMFA, particularly in terms of runtime. Precisely, SPARTA-CC-GEMSTONE achieves a speedup of 4289 times during phase I and 583 times during phase II when compared to GLA-CMFA. Overall, SPARTA-CC-GEMSTONE demonstrates a total speedup of 2370 times over GLA-CMFA. Additionally, concerning the total number of nodes, both SPARTA-CC-GEMSTONE and SPARTA-CP-GEMSTONE outperform GLA-CMFA by 1.15 and 1.14 times, respectively.

A noteworthy observation is that SPARTA-CC-GEMSTONE employs fewer sensors than SPARTA-CP-GEMSTONE but relies on a higher number of relay nodes. This phenomenon indicates a trade-off between the number of sensors and relay nodes. In general, an abundance of sensors results in shorter average distances between them, leading to a reduced need for relay nodes.

V. Conclusion

This research introduces the formulation of the Q-Coverage and Q-Connectivity problems and extends them to the three-dimensional space, aiming to minimize the number of nodes required. Additionally, a two-phase approach is presented to solve these problems. The first phase employs the SPARTA algorithm with a Q-Coverage constraint, while the second phase utilizes the GEMSTONE algorithm with a Q-Connectivity constraint. Comparative analysis against the state-of-the-art in both phases demonstrates that the proposed algorithms significantly outperform existing methods. In-depth analysis is provided to explain the superior performance of SPARTA and GEMSTONE. These algorithms have practical applications in constructing high-quality fault-tolerant WSNs.

In future research, it is possible to integrate additional practical constraints into the algorithms, such as incorporating mobile sensors, considering energy consumption, and accounting for obstacles in the environment. Furthermore, there is potential for optimizing the algorithms in terms of running time and node usage, aiming to provide even better results. These advancements would contribute to the enhanced applicability and efficiency of the proposed algorithms for addressing the given problem.

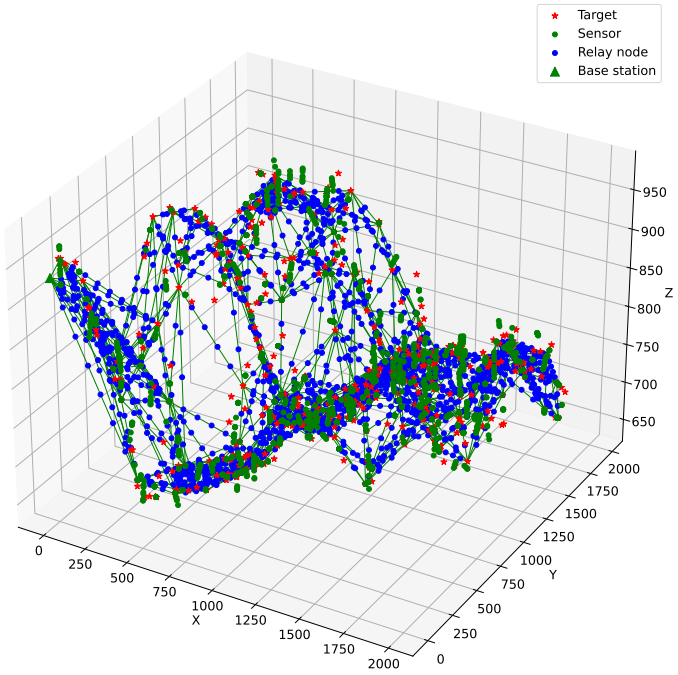


Fig. 16: Result of GLA-CMFA with the default dataset and experiment parameters.

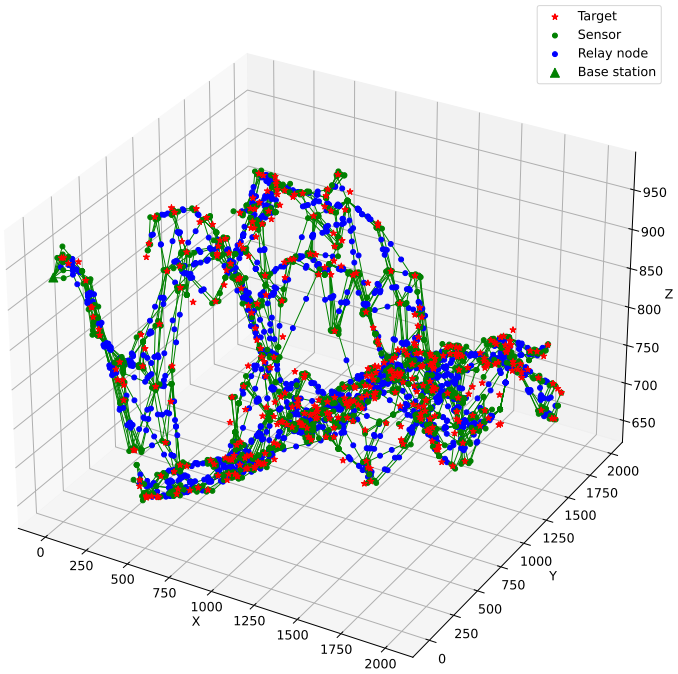


Fig. 17: Result of SPARTA-CC-GEMSTONE with the default dataset and experiment parameters.

References

[1] A. Tripathi, H. P. Gupta, T. Dutta, R. Mishra, K. K. Shukla, and S. Jit, "Coverage and connectivity in wsns: A survey, research issues and challenges," *IEEE Access*, vol. 6, pp. 26 971–26 992, 2018.

[2] M. Farsi, M. A. Elhosseini, M. Badawy, H. Arafat Ali, and H. Zain Eldin, "Deployment techniques in wireless sensor networks, coverage and connectivity: A survey," *IEEE Access*, vol. 7, pp. 28 940–28 954, 2019.

[3] S. Pundir, M. Wazid, D. P. Singh, A. K. Das, J. J. P. C. Rodrigues, and Y. Park, "Intrusion detection protocols in wireless sensor networks integrated to internet of things deployment: Survey and future challenges," *IEEE Access*, vol. 8, pp. 3343–3363, 2020.

[4] D. Kandris, C. Nakas, D. Vomvas, and G. Koulouras, "Applications of wireless sensor networks: An up-to-date survey," *Applied System Innovation*, vol. 3, no. 1, 2020, ISSN: 2571-5577.

[5] J. Amutha, S. Sharma, and J. Nagar, "Wsn strategies based on sensors, deployment, sensing models, coverage and energy efficiency: Review, approaches and open issues," *Wireless Personal Communications*, vol. 111, no. 2, pp. 1089–1115, Mar. 2020, ISSN: 1572-834X.

[6] B. Wang, *Coverage Control in Sensor Networks*. Springer London, 2010.

[7] S. Harizan and P. Kuila, "Nature-inspired algorithms for k-coverage and m-connectivity problems in wireless sensor networks," in *Design Frameworks for Wireless Networks*, S. K. Das, S. Samanta, N. Dey, and R. Kumar, Eds. Singapore: Springer Singapore, 2020, pp. 281–301, ISBN: 978-981-13-9574-1.

[8] A. Boukerche and P. Sun, "Connectivity and coverage based protocols for wireless sensor networks," *Ad Hoc Networks*, vol. 80, pp. 54–69, 2018, ISSN: 1570-8705.

[9] M. Karatas, N. Razi, and H. Tozan, "A comparison of p-median and maximal coverage location models with q–coverage requirement," *Procedia Engineering*, vol. 149, pp. 169–176, 2016, International Conference on Manufacturing Engineering and Materials, ICMEM 2016, 6-10 June 2016, Nový Smokovec, Slovakia, ISSN: 1877-7058.

[10] Manju, P. Bhambu, and S. Kumar, "Target k-coverage problem in wireless sensor networks," *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 23, no. 2, pp. 651–659, 2020.

[11] P. Szczytowski, A. Khelil, and N. Suri, "Dkm: Distributed k-connectivity maintenance in wireless sensor networks," in *2012 9th Annual Conference on Wireless On-Demand Network Systems and Services (WONS)*, 2012, pp. 83–90.

[12] X. Bai, D. Xuan, Z. Yun, T. H. Lai, and W. Jia, "Complete optimal deployment patterns for full-coverage and k-connectivity (k≤6) wireless sensor networks," in *Proceedings of the 9th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, ser. MobiHoc '08, Hong Kong, Hong Kong, China: Association

for Computing Machinery, 2008, pp. 401–410, ISBN: 9781605580739.

[13] J. Pu, Z. Xiong, and X. Lu, "Fault-tolerant deployment with k-connectivity and partial k-connectivity in sensor networks," *Wireless Communications and Mobile Computing*, vol. 9, pp. 909–919, Jul. 2009.

[14] Z. Yun, X. Bai, D. Xuan, T. H. Lai, and W. Jia, "Optimal deployment patterns for full coverage and k-connectivity $(k \leq 6)$ wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 18, no. 3, pp. 934–947, 2010.

[15] R. Yarinezhad and S. N. Hashemi, "A sensor deployment approach for target coverage problem in wireless sensor networks," *Journal of Ambient Intelligence and Humanized Computing*, Jun. 2020, ISSN: 1868-5145.

[16] S. Peng and Y. Xiong, "A lifetime-enhancing method for directional sensor networks with a new hybrid energy-consumption pattern in q-coverage scenarios," *Energies*, vol. 13, no. 4, 2020, ISSN: 1996-1073.

[17] R. Ozdag, "Optimization of target q-coverage problem for qos requirement in wireless sensor networks," *Journal of Computers*, pp. 480–489, Jan. 2018.

[18] S. Mini, S. K. Udgata, and S. L. Sabat, "Sensor deployment and scheduling for target coverage problem in wireless sensor networks," *IEEE Sensors Journal*, vol. 14, no. 3, pp. 636–644, 2014.

[19] D. Arivudainambi, S. Balaji, R. Pavithra, and R. Shakthivel, "Energy efficient sensor scheduling for q-coverage problem," in *2017 IEEE 22nd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, IEEE, 2017, pp. 1–6.

[20] N. T. Hanh, H. T. T. Binh, V. Q. Truong, N. P. Tan, and H. C. Phap, "Node placement optimization under q-coverage and q-connectivity constraints in wireless sensor networks," *Journal of Network and Computer Applications*, p. 103 578, 2023.

[21] D. Arivudainambi and R. Pavithra, "Coverage and connectivity-based 3d wireless sensor deployment optimization," *Wirel. Pers. Commun.*, vol. 112, no. 2, pp. 1185–1204, May 2020, ISSN: 0929-6212.

[22] S. Balaji, Priyanka, and Anitha, "Optimal deployement of sensors in 3d - terrain with q-coverage constraints," in *2018 IEEE SENSORS*, 2018, pp. 1–4.

[23] S. Mini, S. K. Udgata, and S. L. Sabat, "Sensor deployment in 3-d terrain using artificial bee colony algorithm," in *International Conference on Swarm, Evolutionary, and Memetic Computing*, 2010.

[24] W. Chen, P. Yang, W. Zhao, and L. Wei, "Improved ant lion optimizer for coverage optimization in wireless sensor networks," *Wireless Communications and Mobile Computing*, vol. 2022, 2022.

[25] R. Song, Z. Xu, and Y. Liu, "Wireless sensor network coverage optimization based on fruit fly algorithm.," *International Journal of Online Engineering*, vol. 14, no. 6, 2018.

[26] A. N. Njoya, C. Thron, J. Barry, *et al.*, "Efficient scalable sensor node placement algorithm for fixed target coverage applications of wireless sensor networks," *IET Wireless Sensor Systems*, vol. 7, no. 2, pp. 44–54, 2017.

[27] Y.-h. Kim, C.-M. Kim, D.-S. Yang, Y.-j. Oh, and Y.-H. Han, "Regular sensor deployment patterns for p-coverage and q-connectivity in wireless sensor networks," in *The International Conference on Information Network 2012*, 2012, pp. 290–295.

[28] W. Wang, H. Huang, F. He, F. Xiao, X. Jiang, and C. Sha, "An enhanced virtual force algorithm for diverse k-coverage deployment of 3d underwater wireless sensor networks," *Sensors*, vol. 19, no. 16, p. 3496, 2019.

[29] R. Özdağ and M. Canayaz, "Optimization of sensor deployment for k-coverage in wireless sensor networks," 2018.

[30] S. K. Gupta, P. Kuila, and P. K. Jana, "Genetic algorithm approach for k-coverage and m-connected node placement in target based wireless sensor networks," *Computers & Electrical Engineering*, vol. 56, pp. 544–556, 2016, ISSN: 0045-7906.

[31] N. T. Hanh, H. T. T. Binh, N. Q. Huy, N. Van Thanh, and B. T. Q. Mai, "Node placement optimization under k-coverage and k-connectivity constraints in wireless sensor networks," in *2022 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2022, pp. 01–08.

[32] S. Afizudeen and R. Pavithra, *Grundy number based optimal sensor placement in 3d wireless sensor network*, 2023.

[33] A. Singh, A. Rossi, and M. Sevaux, "Matheuristic approaches for q-coverage problem versions in wireless sensor networks," *Engineering Optimization*, vol. 45, no. 5, pp. 609–626, 2013.

[34] S. K. Gupta, P. Kuila, and P. K. Jana, "Genetic algorithm for k-connected relay node placement in wireless sensor networks," in *Proceedings of the Second International Conference on Computer and Communication Technologies: IC3T 2015, Volume 1*, Springer, 2016, pp. 721–729.

[35] M. S., S. Udgata, and S. Sabat, "M-connected coverage problem in wireless sensor networks," *ISRN Sensor Networks*, vol. 2012, Mar. 2012.

[36] A. K. Srivastava and S. K. Gupta, "Eerp: Energy-efficient relay node placement for k-connected wireless sensor networks using genetic algorithm," in *Ambient Communications and Computer Systems*, Y.-C. Hu, S. Tiwari, K. K. Mishra, and M. C. Trivedi, Eds., Singapore: Springer Singapore, 2019, pp. 3–10, ISBN: 978-981-13-5934-7.

[37] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2022.

[38] E. Doukhnitch, M. Salamah, and E. Ozen, "An efficient approach for trilateration in 3d positioning," *Computer Communications*, vol. 31, no. 17, pp. 4124–4129, 2008, ISSN: 0140-3664.

[39] J. Bhasker and T. Samad, "The clique-partitioning problem," *Computers & Mathematics with Applications*, vol. 22, no. 6, pp. 1–11, 1991, ISSN: 0898-1221.