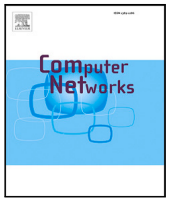
Computer Networks 250 (2024) 110538

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet



Swarm intelligence for hole detection and healing in wireless sensor networks

Giada Simionato [a,b,*], Mario G.C.A. Cimino [a]

[a] Department of Information Engineering, University of Pisa, Italy
[b] Department of Information Engineering, University of Florence, Italy

ARTICLE INFO

Keywords:
Hole detection
Hole healing
Wireless sensor networks
Swarm intelligence
Artificial platelets

ABSTRACT

The increasing demand for wireless sensor networks to monitor specific regions has prompted extensive research on sustaining coverage over time. The main threat to this goal arises from coverage holes caused by random node deployment or failures. This study proposes a swarm intelligence-based algorithm to detect and heal coverage holes. The swarm of agents relies on local and relative information, activating in response to detected holes and navigating a potential field toward the closest hole. The agents quantize their perceptions to disperse efficiently, approaching holes from different directions to accelerate healing. Based on geometric criteria, the swarm deploys at locally optimal positions along hole borders while preventing redundant deployments. Agents deployment update the potential field, guiding the rest of the swarm toward unhealed areas and ensuring dynamic detection and tracking of new holes, even near the region frontier. Experimental studies demonstrate superior coverage restoration compared to state-of-the-art solutions, showing good scalability and flexibility to different hole sizes, shapes, and multiplicity. Moreover, it exhibits high robustness to the corruption of agents' perceptions and to their failure, while efficiently managing the battery level.

1. Introduction

In recent years, there has been a growing demand for systems capable of continuously monitoring and sensing environmental factors across wide areas. This need for continuous and complete coverage of a Region of Interest (ROI) extends to diverse sectors, including sustainable agriculture [1,2], wildlife monitoring [3,4], disaster recovery [5,6], surveillance [7,8], and military applications [9,10]. Typically, to achieve this, a Wireless Sensor Network (WSN) is deployed across the ROI, ensuring that each network node can cover a specific portion of it. The widespread adoption of WSNs can be attributed to the customization, scalability, and interconnectivity of their nodes [11].

However, several factors can interfere with sustaining such coverage requirements: the random positioning of network nodes might result in some areas of the ROI being uncovered [12]. Furthermore, nodes are susceptible to unforeseen failures, whether due to, e.g., hardware damage, hacking, or energy depletion [13,14]. All these situations result in coverage *holes*, which can be classified as *open* or *bounded*, depending on whether the border forms an open or closed curve [15]. Due to the importance of maintaining complete coverage, several studies proposed solutions to detect and restore the sensing in bounded holes [16–18] (i.e., *healing* the holes). However, these approaches require *(i)* mobile network nodes to enable relocation as needed; *(ii)* highly dense networks, in which more nodes are deployed than those required. These assumptions are not usually met, especially in case of massive, unmovable nodes, or when the network functioning leverages its stationarity. Moreover, it is not always feasible to have redundant nodes deployed beforehand, especially when the devices are expensive. Another drawback of these solutions is that they do not consider that the nodes involved in restoring the coverage could fail as well, hindering the healing process.

New methods based on swarm intelligence were developed to address these limitations [6,14]. These approaches use autonomous and mobile external *agents* with limited functionalities, to temporarily restore the coverage until the full-fledged nodes are reinstated. The methods are applicable in stationary networks and do not require any deployment beforehand, eliminating the need for dense networks. They also leverage the inherent robustness of swarm intelligence techniques to complete the healing process even if agents fail. In this work, we extend the solution in [6], which introduces a novel bio-inspired approach for detecting and healing holes using a swarm of resource-constrained agents with limited sensing capabilities. These agents are released in the ROI and rely on local information to detect hole borders and deploy themselves in locally optimal positions for temporary

* Corresponding author at: Department of Information Engineering, University of Florence, Italy.
E-mail addresses: giada.simionato@phd.unipi.it (G. Simionato), mario.cimino@unipi.it (M.G.C.A. Cimino).

https://doi.org/10.1016/j.comnet.2024.110538
Received 6 February 2024; Received in revised form 27 April 2024; Accepted 24 May 2024
Available online 27 May 2024

restoration. The swarm behavior is governed by three key rules inspired by the phenomena regulating blood coagulation: *activation*, wherein agents initiate the healing process upon detecting holes; *adhesion*, where agents adhere to hole borders and emit attractants to enhance collaboration; and *cohesion*, enabling agents to cohere until complete hole healing is achieved.

Despite displaying high robustness, the method in [6] can only effectively handle holes located away from the ROI frontier. Moreover, the agents, following a similar logic and departing from close locations, tend to cluster excessively. This not only hampers the exploitation of the swarm's parallelism during navigation but also generates frequent concurrent deployments in very close positions, wasting resources. In this work, we tackled these issues while also incorporating an analysis of the effectiveness of our method from an energy perspective. Particularly, our main contributions include:

- enabling holes next to ROI frontier, allowing their detection and their healing.
- exploiting the findings in [14] and the benefits brought by the quantization of perceptions [19,20] to speed up the healing process without increasing the number of agents deployed.
- integrating a *commitment* strategy before agents positioning to prevent clustered deployments, thus minimizing the number of resources necessary to restore the coverage.

Moreover, we equip the agents with a detailed model of the battery level, as to showcase the suitability of our method to real-world scenarios. Using energy information, we devise proactive policies to optimize the replacement process in case the agents deplete their energy, extending the duration of WSNs lifetime.

We carry out a thorough validation and evaluation of our solution. We optimize the parameters using the Differential Evolution (DE) algorithm and investigate the performance of our method in response to different internal and external factors. We also assess the robustness to data corruption and agent failure and we compare its performance against several state-of-the-art algorithms, namely [6,10,16,17,21], showing that it outperforms all of them.

The rest of this paper is organized as follows. In Section 2, the literature on the topic is reviewed and, in Section 3, the context is detailed. The algorithm is explained in Section 4, while Section 5 reports and discusses the results of performance evaluation. Finally, in Section 6, conclusions are drawn.

2. Related work

Several studies suggest methods for detecting [22,23] or healing [21,24] coverage holes in WSNs. Most of these approaches rely on geometric properties and topological insights, such as the Voronoi Diagram or its dual, the Delaunay Triangulation. For example, two algorithms were presented to reduce the energy consumption of mobile nodes, aimed at delaying holes formation [13,18]. They employ a distributed Voronoi-based cooperation scheme [18] and insights from intersection points among nodes [13]. However, both methods can only be applied to dense networks, a condition usually not met. While [7] employs the Delaunay triangulation of the network alongside a virtual edge-based technique, [25] utilizes an additively weighted Voronoi diagram for hole identification and patching position calculation. In contrast to our decentralized approach, both solutions are centralized and consequently lack robustness. The same drawback is found in [8,26]. In [8] the holes are detected through network cellularization and geometric criteria and prioritized to enable their healing. [26], instead, uses a sleep/wake schedule for the nodes to reinstate the coverage. Both these solutions rely on a Base Station (BS), introducing a single point of failure. In [10], a BS is also used to gather information from nodes about network density and to displace mobile agents as needed. The distributed approach in [27] leverages nodes position to geometrically compute the centroid of the holes, that is where new agents must deploy. However, knowing all nodes position is not always possible. Another decentralized method is [5], in which a Voronoi diagram is constructed from the network and used to estimate holes position and size. Similarly, [21] uses the network's Voronoi diagram to identify the positions of additional nodes, then filtered by Linear programming and probabilistic sensor model techniques. These approaches only work with homogeneous networks (i.e., having nodes with equal sensing capabilities), as opposed to our method, which can serve also heterogeneous networks. This limitation is shared by the tree-based and chord-based algorithms in [3,23], and by the approach in [16] that uses geometric criteria to decide which, where, and with which priority some nodes must move to restore sensing. The solution proposed by [28] employs trigonometric rules to compute the updated node positions. However, as for [16], the process of choosing which node to move generates a high message overhead, debilitating nodes with limited energy.

A different kind of approaches to hole detection and healing is represented by [15], where a complex messaging protocol is used to detect the hole. The healing proceeds through virtual forces driving the nodes toward holes center while repelling other nodes to increase coverage. Hybrid solutions, as introduced in [1,29,30], merge the advantages of geometry-based and virtual forces-based approaches but require GPS availability.

Another class of solutions uses Deep Learning (DL) and Reinforcement Learning (RL) techniques to detect the holes and compute potential patching positions [2,31,32]. [31] introduces a pure topology-based solution that uses a Convolutional Neural Network trained with transfer learning to detect the polygonal border of the holes from the network layout. [2] detects and estimates the size of the holes through Hybrid Deep RL, while [32] employs a game theory-based RL algorithm to learn the most energy-efficient sleep/wake schedule for network nodes. However, the amount and precision of data needed by these algorithms are nearly unattainable, especially in unknown environments.

Swarm Intelligence (SI) algorithms have proven to be particularly suitable in managing WSNs [33]. Their approaches to hole detection and healing can be divided in two categories: optimization-based (SI-optimization) and behavioral rules-based (SI-behavioral rules). In the former category, the problem is modeled as an optimization task. The result of this optimization process is the new positions that certain nodes in the network must take to restore the coverage. To find these positions, the algorithms virtually move in an n-dimensional solutions space to find those maximizing a fitness function of global coverage. Examples of this category are [12,34], that employ Particle Swarm Optimization (PSO) or an improved version of PSO to update the positions of patching nodes. In [35], the network is cellularized and fed to an active contour model to detect coverage holes. To restore service, PSO is used to find the locations for additional nodes that maximize coverage, while minimizing their traveled distance. Similarly, in [17], network cellularization provides coverage information to evaluate the fitness function in an enhanced version of the Artificial Fish Swarm Algorithm [36]. In contrast to our approach, which depends solely on local information, these methods require global knowledge of the environment and network. Assuming global knowledge relies on having specific systems in place to collect and manage information, which might not always be feasible if the environment is unknown or if the network changes post-deployment. Therefore, these methods follow *offline planning* and cannot adjust to environmental changes, without having to re-collect and re-process global information. Conversely, local knowledge can be acquired in real-time without the need for such specialized infrastructure. In fact, our algorithm exploits *online planning* to be able to react to dynamic scenarios. The latter category encompasses behavioral rules-based approaches in which the behavior of agents, such as robots, is driven by a set of predefined local rules. [6] introduces a hole detection and healing algorithm that uses resource-constrained agents to restore the service. Rules inspired by blood coagulation are followed by the agents to place inside the holes as to

Table 1
Comparison of key features for hole detection and healing algorithms against our solution. ✓̸ represents partial achievement. Methods involved in the quantitative comparison are reported in bold font.

Reference	*Technique used*	*Distributed*	*GPS-free*	*Low density*	*Dynamic scenarios*	*Agent failure*
Sahoo et al. [22]	Geometric	✓			✓	
Li et al. [23]	Geometric	✓	✓			
Aliouane et al. [24]	Geometric	✓			✓	✓̸
Khedr et al. [13]	Geometric	✓			✓̸	
Papi et al. [8]	Geometric			✓		
Narayan et al. [26]	Geometric				✓	
Rath et al. [27]	Geometric					
Singh et al. [3]	Geometric			✓		
Khalifa et al. [16]	**Geometric**	✓				
Hallafi et al. [10]	**Geometric**	✗		✓	✗	✓
Khelil et al. [21]	**Topological**	✗		✓	✗	✗
Qiu et al. [18]	Topological	✓				
Soundarya et al. [7]	Topological		✓	✓		
Davoodi et al. [25]	Topological					✓̸
Duan et al. [5]	Topological					✓̸
Wu et al. [28]	Topological	✓				
Senouci et al. [15]	Virtual forces	✓				✓̸
Kukunuru et al. [1]	Virtual forces	✓̸	✓			✓̸
Kadu et al. [29]	Virtual forces	✓				
So-In et al. [30]	Virtual forces	✓̸				
Lai et al. [31]	DL/RL-based			✓	✓̸	
Chowdhuri et al. [2]	DL/RL-based			✓	✓̸	
Chauhan et al. [32]	DL/RL-based	✓			✓	✓̸
Mehta et al. [34]	SI-optimization					
Wang et al. [12]	SI-optimization					
Yu et al. [35]	SI-optimization			✓		
Yan et al. [17]	**SI-optimization**	✗				
Ours	**SI-behavioral rules**	✓	✓	✓	✓	✓

restore the coverage. In [14], a version of this algorithm is presented, able to detect and heal coverage holes under sparse and coarse perception. These solutions, however, cannot handle holes forming next to the ROI frontier. Moreover, despite showing prompt reaction to holes formation, they cannot provide swift healing while minimizing the number of agents employed. In this work, we leverage the findings in [6,14] to propose a new version of the algorithm able to overcome these issues, providing prompt recovery with fewer resources.

Table 1 summarizes the discussed approaches, providing a qualitative comparison regarding the technique used and key characteristics of hole detection and healing algorithms. Comparison is made against *(i)* whether the methods are *distributed*, and thus more robust; *(ii)* if they do not rely on nodes' absolute position (*GPS-free*), therefore being applicable to GPS-denied scenarios; *(iii)* if they support *low density* networks or demand abundant nodes to function; *(iv)* whether they handle *dynamic scenarios*, i.e., changes in network status and *(v)* if they adapt to *agent failure*.

We compared our algorithm against five of the discussed state-of-the-art methods, namely [6,10,16,17,21]. We selected Khalifa et al. [16] and Hallafi et al. [10] among the geometry-based approaches. These recent methods showed superior performance with respect to those of their category. Moreover, they rely on different geometric concepts, being good representatives of the geometric category. For the same reason, we selected Khelil et al. [21] among the topological approaches. The methods relying on virtual forces were proven to be outperformed by geometric and topological solutions. Therefore, we excluded them from our comparison as they offer no additional insights. DL/RL-based approaches necessitate data that cannot be gathered in unknown scenarios, or infrastructures and computational power that might not be available. Since the assumptions they rely on cannot be met in the scenarios we considered, we opted not to select any. Among the SI-optimization methods, we selected Yan et al. [17] since it outperformed other PSO-based approaches. Simionato et al. [6] was included in the comparison to evaluate the improvements introduced in this work.

Next, we will provide more details about the methods chosen as comparisons (reported in bold font in Table 1). In their work, Khalifa et al. [16] present DHDR, a solution designed for a dense network of mobile nodes. In this approach, nodes respond to the absence of a heartbeat message from a neighboring node by moving toward the detected hole. Consequently, only nodes adjacent to the hole actively contribute to the healing process. The displacement calculation is determined through geometric criteria, considering the intersection points between the sensing disk of each node and that of its neighbors and following a heuristic priority scheme to guide the movement of nodes.

The algorithm in Hallafi et al. [10] uses mobile agents to heal coverage holes. A BS cellularizes the network, selecting the highest-energy nodes as cell heads. Each cell head is responsible for its cell coverage computation, using the information from its neighbors and geometric rules. A sleep-wake schedule is applied to minimize energy consumption in case of high cell density. If the covered area is lower than that of the cell, the head alerts the BS. The BS instructs the agents to navigate toward the cells containing holes to heal them. If the agents are less than the detected holes, the BS selects the highest priority holes using a Grasshopper Optimization algorithm [37].

Khelil et al. [21] detects coverage holes using [22]. For each hole, it appoints a node as hole manager responsible for building the Voronoi diagram from the nodes bordering its hole. Diagram vertices are candidate patching positions for external nodes. The list of locations is iteratively filtered using Integer Linear Programming techniques, and the concept of maximal independent set of points with minimum cardinality, to minimize the number of new nodes.

Yan et al. [17] approach the issue as an optimization task, presenting the FSHR algorithm. This adds leap and rebirth behaviors to the AFSA, enhancing it. To heal the holes, FSHR moves some of the inactive, mobile nodes that are dispersed across a network of static nodes. The nearest mobile nodes are guided to the target positions by the WSN sink nodes, which compute the positions and provide control inputs. To do this, FSHR divides the ROI into discrete grids and optimizes a fitness function that is determined by the percentage of grid points that are covered by both the static nodes and the activated mobile nodes.

These approaches show some limitations: they rely on GPS, which may not be available in critical scenarios, or assume mobile redundant nodes in the network, while our method uses external agents independent of network characteristics. Moreover, they compute new deployment positions without considering unexpected failures, unlike our online planning method which dynamically adapts to environmental changes.

3. System model

We assume the region of interest to be a 2-dimensional area free from obstacles. This area is completely covered by a network of stationary nodes, having communication and sensing capabilities. We model nodes as point-like entities, where their position corresponds to the center of mass (CoM) of the device they represent. The network layout is randomly generated and it is designed to ensure that each point of the ROI is sensed by at least one node, while keeping the overall density low. This means that the failure of a node always results in a portion of the ROI left uncovered, generating holes.

To heal the coverage holes, we make use of a swarm $S = \{s_1, \dots, s_n\}$ of mobile agents, e.g., drones or unmanned ground vehicles. We consider agents to have limited computational and energy resources and restricted sensing capabilities. We model the agents as point-like nonholonomic entities following the unicycle dynamics [38]. As for the nodes, their pose coincides with the robot CoM. The control inputs are the *steering* and *driving* speeds. If specific control values are not explicitly needed for executing maneuvers or reaching a destination, the agents move in the environment maintaining a steady steering $\bar{\omega}$ and driving $\bar{v}$ speed. This abstraction offers broader applicability to different robots, with the potential to map more complex dynamics back to the simplified model we used for movement.

Nodes and agents have sensing and communication capabilities, which we represent through the Boolean disk coverage model [39]. In this model, we assume reliable communication or sensing within a disk of radius r_c or r_s centered on each node and agent, and no service outside. As a result, the communication range r_c represents the maximum distance at which packets can be exchanged and the sensing range r_s is the maximum distance covered. Given the limited coverage capability of the agents, we suppose that their sensing radius $r_{s,a}$ is significantly smaller than that of nodes, denoted as $r_{s,n}$ (i.e., $r_{s,a} \ll r_{s,n}$). However, we stress that our algorithm could work without modifications also for $r_{s,a} \geq r_{s,n}$. To ensure connectivity in the restored area, we assume that the communication range is at least double of the sensing radius for both agents and nodes, that is $r_{c,a} \geq 2r_{s,a}$ and $r_{c,n} \geq 2r_{s,n}$ [40].

To perceive the environment, nodes and agents $\boldsymbol{o_i}$ are equipped with a Range and Bearing (RaB) sensor [41] that provides, in their reference frame, the relative distance d_{ij} and angle φ_{ij} to other nodes and agents $\boldsymbol{o_j}$ within the communication range. These values are computed using hardware geometry and packets arrival time at the antennas array [42]. We leverage this concept by sending lightweight packets containing nodes and agents information to estimate the quantities. This sensor enables continuous environmental assessment, allowing the swarm to adapt to dynamic scenarios without relying on a priori global knowledge or GPS information.

To tighten the gap with real-world scenarios, we endowed the agents with a model of the battery level, detailed in Section 4.3.

4. Proposed method

Our algorithm is inspired by the behavior exhibited by biological platelets during blood coagulation. As the platelets, the swarm stays in an *inactive* state if no holes are detected. In this state, the swarm is clustered in random locations, called Release Points (RPs) that are scattered around the ROI. In real-world scenarios, the RPs represent the locations where the swarm could be potentially released by external operators. Network nodes, instead, evaluate periodically their surroundings looking for coverage holes, based on their RaB readings and geometric rules. When a hole is detected, they start spreading an alert message in the network. This behavior emulates the diffusion of collagen, a substance that leaks inside the bloodstream following an injury and is responsible for activating the platelets [43,44]. The concentration of the collagen is inversely proportional to the wound distance. Similarly, the nodes keep track of their distance to the closest hole, leveraging information from their neighbors (i.e., those with which their sensing disks overlap). In this way, they form a simple potential field that points toward the closest hole.

The inactive agents eavesdrop on nodes' communications, waiting for the alert to *activate*. Once activated, they start following the potential field created by nodes until they are close to the hole border. In biology, platelets initiate the plug by adhering to the border of the wound, becoming *bound*. After that, they start releasing adenosine diphosphate (ADP) and fibrin. The former substance attracts other platelets and induces them to cohere to increase the size of the plug. The latter, instead, cements the bonding with the injury walls and with other platelets [43,44]. We adapt these concepts by making the swarm *adhere* to the hole border, deploying in locally optimal positions computed geometrically. Deployment has the effect of switching the agent state to *bound* and connecting it with the rest of the network (as artificial fibrin). Moreover, a bound agent contributes to the forwarding of the alert message and updates the potential field. This will naturally attract other agents and foster further deployments (as artificial ADP).

Deployment must not occur outside the hole, where the network works properly. In the biological context, to discourage adhesion to the intact part of the vessel, a repulsive substance named prostacyclin is secreted. Similarly, the agents are instructed to discard potential positions that are already covered by other nodes or bound agents.

Our algorithm results in the border of the holes changing and shrinking as the swarm deploys. When the healing is completed, holes are no longer detected and the alert message ceases to be forwarded. The remaining agents may return to the RPs, or to a safe area, waiting to be reactivated. Similarly, platelets return to flowing in the bloodstream when the injury is healed.

Since we consider a swarm with limited capabilities, we stress that our algorithm offers a prompt but temporary solution. For this reason, it can be thought of as a primary hemostasis for the network.

In the next paragraphs, we will detail how hole detection is performed and how the potential field is generated (Section 4.1). We will explain the healing process (Section 4.2), and define the battery model, as well as the techniques employed for energy management (Section 4.3).

4.1. Hole detection

Holes in the coverage are detected by nodes and bound agents through the computation of a coefficient: the *Angular Coverage Ratio* (ACR). This coefficient $c \in [0, 1]$ represents how much of the border of their sensing disk is covered by the sensing disk of other nodes or bound agents. If a node (or bound agent) faces one or more holes, part of its border will not be covered. Particularly, if the ACR of a node or bound agent $\boldsymbol{o_i}$, namely c_i, is below a threshold Th_c, it means that $\boldsymbol{o_i}$ belongs to the set $\mathcal{B}$ of nodes or bound agents bordering a hole. ACR computation relies on the readings from the RaB sensor and involves only the neighbors. For each node and bound agent $\boldsymbol{o_j}$ in the set $\mathcal{N}_i$ of neighbors of $\boldsymbol{o_i}$, $\boldsymbol{o_i}$ follows Eq. (1a) to compute the angle θ_j in the triangle $\boldsymbol{o_j}\hat{\boldsymbol{o_i}}\boldsymbol{q_j}$. Here, $\boldsymbol{q_j}$ is one of the two points of intersection between sensing circumferences, r_{s,o_i} represents the sensing range of $\boldsymbol{o_i}$ (and r_{s,o_j} of $\boldsymbol{o_j}$), and d_{ij} its distance to $\boldsymbol{o_j}$, provided by the sensor. These quantities are used as margins for intervals centered in the direction φ_{ij} in which the neighbors are perceived. The intervals are merged, as in Eq. (1b), to obtain the angular span covered by the neighbors. c_i is then computed according to Eq. (1c) (where $|\Theta_i|$ returns the length of the interval Θ_i) and compared with Th_c to determine the presence of a

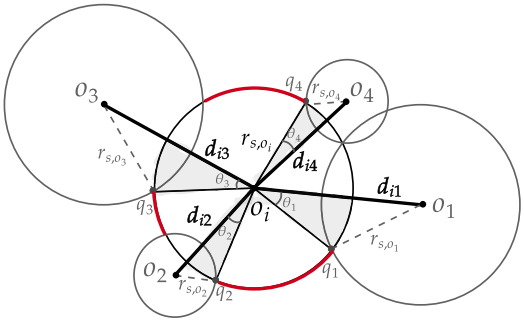


Fig. 1. Example of ACR computation for o_i. The red border shows where it faces holes. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

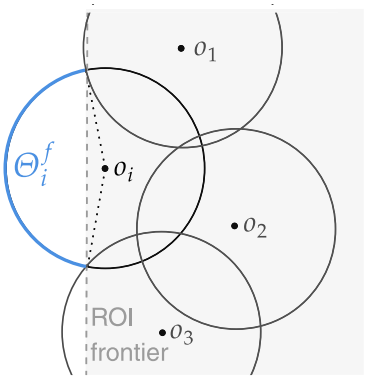


Fig. 2. Geometric interpretation of Θ_i^f (blue border) for the frontier node o_i. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

hole. Th_c represents, therefore, a tolerance accounting for RaB noises and imperfect placement. We assess the impact of different values for Th_c in Section 5.2. ACR computation occurs periodically, enabling prompt detection of holes and tracking of their recovery, even in case of agent failure. Fig. 1 illustrates an example of ACR computation.

$$\theta_j = \arccos\left(\frac{r_{s,o_i}^2 + d_{ij}^2 - r_{s,o_j}^2}{2 r_{s,o_i} d_{ij}}\right) \tag{1a}$$

$$\Theta_i = \bigcup_{o_j \in \mathcal{N}_{o_i}} [\varphi_{ij} - \theta_j, \varphi_{ij} + \theta_j] \tag{1b}$$

$$c_i = |\Theta_i|/2\pi \tag{1c}$$

In this work, we extend the ACR computation to support scenarios in which *frontier nodes*, i.e., those located at the ROI frontier, face the holes. To identify frontier nodes, right after deployment, each node performs Eq. (1) when the network is still intact. All nodes o_i having $c_i < Th_c$, store $\Theta_i^f = [0, 2\pi] \backslash \Theta_i$, a set of intervals representing the portion of the sensing border facing the outside of the ROI (see Fig. 2). Due to the stationarity of the network, the angular span identified by Θ_i^f can always be considered as covered. In this way, only the part of the sensing border left uncovered by a hole can be responsible for a lower ACR. To account for this, we modify Eq. (1c) as follows:

$$c_i = |(\Theta_i \cup \Theta_i^f)|/2\pi. \tag{2}$$

When a node or bound agent o_i detects a hole, it spreads the alert message in the network using a *gossip protocol*, such as [45]. In addition, it stores the *level* ℓ_i representing the one-hop distance to the closest hole, computed as follows:

$$\ell_i = \begin{cases} 0, & \text{if } o_i \in \mathcal{B} \\ \min\limits_{o_j \in \mathcal{N}_i} \ell_j + 1, & \text{otherwise.} \end{cases} \tag{3}$$

The network levels generate a potential field of increasing integer values, whose global minimum is 0 at the holes border. When agents change the border by deploying, the levels are recalculated, updating the potential field and directing the swarm to the remaining unhealed area.

4.2. Hole healing

When the inactive agents overhear the alert message, they switch their state to *active* and begin the healing process. In [6], the swarm uses the RaB readings to follow the potential field by moving in the direction of the closest node (or bound agent) with minimum level. Instead in [14], the bearing readings are quantized into K angular sectors. The agents move when facing the sector with the lowest average level of nodes and bound agents within that sector. As proved in [14], its navigation strategy introduces more dispersion in the swarm due to the higher coarseness of the readings. This results in the swarm tackling holes from different directions, parallelizing the healing and thus speeding up the process, compared to [6]. However, the deployment logic in [14] requires more agents. In this work, we theorize that combining the navigation strategy of [14] with the placement logic of [6], here described, could lead to quicker healing without the waste of resources. The effectiveness of this combination is proved in Section 5.2.

Both navigation strategies drive the swarm to the closest hole border. Whenever an agent perceives at least two nodes or bound agents with level $\ell = 0$ (on the border), it computes potential deployment positions. Each position is based on the coordinates of two nodes or bound agents with overlapping sensing disks and level $\ell = 0$, referred to as *parents*. The positions $\mathbf{t_{1,2}}$ are calculated using Eqs. (4) and (6) when both parents share the same type (either both nodes or both agents), or Eqs. (5) and (6) otherwise. Here, o_i and o_j are the coordinates of the parents in the agent reference frame, provided by the RaB sensor. r_{s,o_j} is the sensing range of o_j and d is the distance between the parents computed from RaB data. $\mathbf{k}_\perp$ is the unit vector orthogonal to the unit vector $\mathbf{k}$. We recall that $r_{s,a}$ and $r_{s,n}$ are the sensing ranges of the agents and of the nodes, respectively. Figs. 3a and 3b depict the two symmetrical solutions obtained in the two cases.

$$\mathbf{k} = \frac{\mathbf{o_i} - \mathbf{o_j}}{\|\mathbf{o_i} - \mathbf{o_j}\|} = \frac{\mathbf{o_i} - \mathbf{o_j}}{d} \qquad d_b = d/2 \qquad d_h = \sqrt{r_{s,o_j}^2 - \frac{d^2}{4}} + r_{s,a} \tag{4}$$

$$\mathbf{k} = \frac{\mathbf{o_i} - \mathbf{o_j}}{\|\mathbf{o_i} - \mathbf{o_j}\|} = \frac{\mathbf{o_i} - \mathbf{o_j}}{d} \qquad d_2 = 2\sqrt{r_{s,a}^2 - \left(\frac{d^2 - r_{s,n}^2 - r_{s,a}^2}{2 r_{s,n}}\right)^2} \qquad p = \frac{2d + d_2}{2} \qquad A = \sqrt{p(p-d)^2(p-d_2)} \qquad d_h = 2A/d \qquad d_b = \sqrt{d_2^2 - d_h^2} \tag{5}$$

$$\mathbf{t_{1,2}} = \mathbf{o_j} + d_b \mathbf{k} \pm d_h \mathbf{k}_\perp \tag{6}$$

Since the RaB sensor provides a local and relative perception of the environment, the solutions computed by the agents are only locally optimal. To increase the coverage, we also considered *additional* positions on the midpoint of the segment joining two nonadjacent parents, having distance $d \leq r_{s,o_i} + r_{s,o_j} + 2r_{s,a}$. These solutions t are computed according to Eq. (7) and illustrated in Fig. 3c.

$$\mathbf{k} = \frac{\mathbf{o_i} - \mathbf{o_j}}{\|\mathbf{o_i} - \mathbf{o_j}\|} = \frac{\mathbf{o_i} - \mathbf{o_j}}{d} \qquad \mathbf{k}' = \frac{\mathbf{o_j} - \mathbf{o_i}}{\|\mathbf{o_i} - \mathbf{o_j}\|} = \frac{\mathbf{o_j} - \mathbf{o_i}}{d} \qquad \mathbf{p_1} = \mathbf{o_j} + r_{s,o_j}\mathbf{k} \qquad \mathbf{p_2} = \mathbf{o_i} + r_{s,o_i}\mathbf{k}' \qquad \mathbf{t} = (\mathbf{p_1} + \mathbf{p_2})/2 \tag{7}$$

To prevent deployment onto intact parts of the network, the agent discards from the list of potential positions those already covered by the sensing disk of other nodes or bound agents. To do so, it checks if the

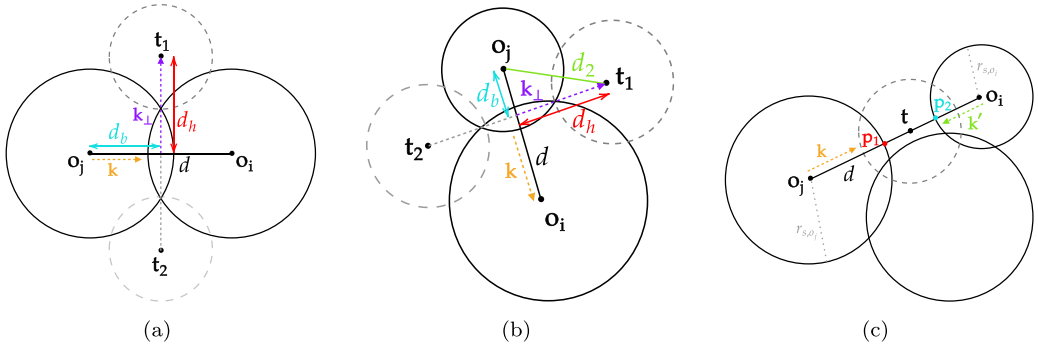


Fig. 3. Geometric representation of locally optimal solutions for parents with same (a) or different sensing radius (b). Additional solution (c).

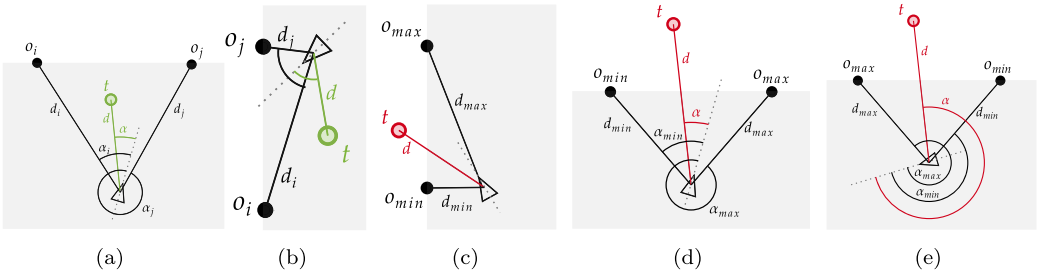


Fig. 4. Geometric representation of different situations in the filtering process of solutions outside the ROI: importance of satisfying both conditions (a) and (b); importance of minimum distance (c); geometric quantities involved in the filtering for agents headed within the range (d) and outside of it (e).

distance from the virtual solution t to the perceived nodes and bound agents o_j is greater than the corresponding sensing range (i.e., $d_{jt} > r_{s,o_j}\ \forall o_j$). If it is not the case, the agent must discard the solution.

Computing potential deployment positions using Eq. (4) or Eq. (5) may yield solutions beyond the ROI when at least one parent is a frontier node. These positions are uncovered since no node or agent is deployed outside the ROI, and thus not automatically discarded. To address this, we introduce the following filtering rule:

Theorem 1. *A potential position t, generated by the parents o_i and o_j according to Eq. (4), Eq. (5) and (6), must be discarded if each of these conditions holds:*

1. *Either o_i or o_j is a frontier node.*
2. *The angle α at which t is observed must fall within the range defined by the angles at which the two parents are perceived.*
3. *The distance d to the position t must be greater than the distance at which the closest parent is perceived.*

Proof. If neither one of the two parents are frontier nodes, t cannot lie farther than them, so it must lie inside the ROI. Conditions 2 and 3 must be both satisfied. If only the condition on the angle is met, based on the geometry underlying how solutions are computed and the assumption that the RPs are scattered inside the ROI, without the condition on distances we could exclude viable positions. In fact, when the condition on the angle is satisfied, viable positions are closer than any of the parents (see Fig. 4a). Similarly, meeting the distance condition ensures that a viable solution will not be found within the cone formed by the agent and its parents (see Fig. 4b). Considering the distance to be greater than the minimum between parents' distances allows the agent to correctly exclude positions, as shown in Fig. 4c. □

Algorithm 1 reports the steps of the filtering process. Lines 1–9 remove the positions already covered, while lines 10–29 implement the filtering rule of Theorem 1. The range in condition 2 must be computed in two ways, depending on the agent heading. If its orientation lies within the range, as in Fig. 4d, the condition is implemented according to lines 18–22. Otherwise, if the agent does not face the range (Fig. 4e), it must follow lines 23–27. We stress that this theorem can be applied only because of the geometry behind the solutions computation and because the RPs lie inside the ROI.

The remaining solutions are then sorted according to a specific *policy*. We consider policies based on the distance to the solutions and the type of their parents. Particularly, we devise the policies *DNAM, NAMD, AMND, closest*, and *random*. The names of the first three strategies reflect the descending priority assigned to the virtual solutions based on their parents type: both Nodes, both Agents, Mixed type, and Additional solutions of Fig. 3c. When positions belong to the same type, precedence is given to the closest one. The *closest* strategy always opts for the nearest computed solution, whereas the *random* strategy selects the closest solution among those associated with a randomly chosen type of parents. The impact of the diverse policies is assessed in Section 5.2. Each agent pursues its highest-ranked solution until either the selected position becomes occupied or its distance is below a threshold Th_b. In the former case, the agent selects a new position, while in the latter, it halts and switches its state to *commit*. In this state, it collects the IDs of other committed agents that are within its sensing range. If it has the lowest ID [46], it deploys, switches to *bound* state and starts providing coverage acting as in Section 4.1. Otherwise, the agent resumes the navigation. We added this commitment strategy to prevent overly clustered deployment and thus waste of resources.

Frontier and loops avoidance. The navigation logic in [14], using quantized information, might drive the agents outside the ROI if the RPs are too close to its frontier, since sectors without readings have lower averages. To prevent this, the agents are instructed to move away from the frontier nodes by keeping them in the rear sectors. This logic might also create loops in agents behavior, with short-term loops caused by two consecutive perceptions demanding opposite actions. To break these loops, agents are subjected to a random impulse, imposing uniformly random control speeds. Long-term loops, instead, occur when agents navigate along arbitrary lengths, resulting in identical perceptions. To avoid these loops, a wiggling behavior is programmed, initially involving n_w^i consecutive instants of random impulses. The wiggle is

Algorithm 1: Solutions Filtering

Data: L the list of solutions t;
R the RaB readings of nodes and bound agents o_i.
Result: Filtered L.

```
foreach t in L do
    foreach o_i in R do
        d_it ← compute distance d(o_i, t);
        if d_it ≤ r_{s,o_i} then
            remove t from L;
            break;
        end
    end
end
ε ← arbitrary small tolerance;
foreach t in L do
    p_1, p_2 ← getParents(t);
    if p_1 is a frontier node or p_2 is a frontier node then
        d, α ← getRaBInfo(t);
        d_1, α_1 ← getRaBInfo(p_1);
        d_2, α_2 ← getRaBInfo(p_2);
        α_min = min(α_1, α_2); α_max = max(α_1, α_2);
        if |α_1 − α_2| ≥ π then
            if [α ≤ (α_min + ε) or α ≥ (α_max − ε)] and
            d > min(d_1, d_2) then
                remove t from L;
            end
        else
            if α ≥ (α_min − ε) and α ≤ (α_max + ε) and
            d > min(d_1, d_2) then
                remove t from L;
            end
        end
    end
end
return L;
```

Table 2
Parameters for battery model.

Parameter	*Value*
Consumption for forward motion	1 J/m
Consumption for rotation motion	0.5 J/2π
Consumption for sensing	0.2 J/s
E_{elec}	50 nJ/bit
ϵ_{fs}	10 pJ/bit/m^2
ϵ_{mp}	13e-17 J/bit/m^4

triggered after t_w^i instants and repeated with a period of T_w instants, increasing each time its duration by one instant. In Section 5.2, the effects of the wiggling parameters will be evaluated.

4.3. Energy management

To account for energy consumption, we extended the battery model in [13]. Three sources of power consumption are considered: movement, transmitting, and receiving packets.

For inactive agents the energy consumption is zero, since it is assumed that they are connected to the power supply. When moving, the energy decreases proportionally to the traveled distance, following the values in Table 2. These values were chosen as to resemble typical power consumption of commercial drones, as further validated by [13]. The fixed decay rate for forward motion is higher than for rotation, to compensate for the energy needed to overcome air resistance. Once deployed, the agents consume constant energy over time to sense the environment.

Transmitting a packet consumes an amount of energy proportional to its size and to the distance of the receiver. In particular, the energy E_{Tx} spent to transmit a l-bit message at distance d can be computed as follows:

$$E_{Tx}(l,d) = \begin{cases} l\ E_{elec} + l\ \epsilon_{fs}\ d^2 & \text{if } d < d_0 \\ l\ E_{elec} + l\ \epsilon_{mp}\ d^4 & \text{if } d \geq d_0 \end{cases} \tag{8}$$

where E_{elec}, ϵ_{fs}, and ϵ_{mp} are coefficients (whose values are reported in Table 2) and $d_0 = \sqrt{\epsilon_{fs}/\epsilon_{mp}}$. The computation considers distance, as signals need amplification for farther destinations, leading to higher energy consumption [13]. Conversely, the energy cost of receiving a packet depends solely on its size, according to Eq. (9). The parameters in Table 2 relative to the transmission and reception of packets are those used in simple models of radio communications and validated by [13,47].

$$E_{Rx}(l) = l\ E_{elec} \tag{9}$$

In our algorithm, packets are exchanged to allow the RaB sensor to compute the distance and direction of other nodes and agents, and to collect essential information. Since perception is not directional, the packets are broadcasted within the communication range. Once activated, the swarm only receives packets from nodes and deployed agents within communication distance, while in the committed state or after deployment, the agents contribute to the exchange of packets. To compute the energy spent on communication using Eqs. (8) and (9), we estimated the size of a packet to $l = 164$ bit, by encoding its content. Each agent starts with 1000 J (400 J in case of limited resources) and it keeps track of its remaining energy. At each time step, according to the state and the environment, the contributions brought by movements and communication are deducted from the current energy level. When an agent's energy drops below zero, it becomes faulty and stops participating in the healing.

In this work, we propose to leverage the information on the remaining energy to prematurely signal agent failure. This allows the agents to modify the potential field, driving the swarm toward them for replacement. Particularly, when the remaining battery level drops below a threshold Th_e, i.e., a fixed percentage of the battery capacity, the agents start acting as faulty, while continuing to provide coverage. This notice enables the swarm to reach their position and deploy the replacements before they actually run out of energy.

We introduce another measure to counteract energy depletion: the incremental release of the swarm. The agents activate upon hole detection, directing all resources toward healing. In real-world scenarios, the uncertainty about hole size may lead to overestimating the required healing capacity, risking wasting resources by releasing unnecessary agents or depleting their energy while patrolling for potential replacements. An incremental release, instead, delays the activation of a fraction of agents, conditioning their involvement in the healing process based on actual need. This approach also allows fully charged agents to quickly replace the failed ones, extending the overall lifespan of the replacements.

We investigate the impact of the incremental release and different values for Th_e in Section 5.5.

5. Performance evaluation

To assess our algorithm performance, we used HDH Simulator [48], a discrete time platform specifically designed to address hole detection and healing problems in WSNs.

We investigated different aspects influencing our method performance. Firstly, we optimized parameters and strategies to identify the most effective setup, examining performance variations with different parameter values (Section 5.2). We assessed the impact on performance of external factors such as hole size, shape, and multiplicity (Section 5.3) and of internal factors, like sensing capabilities (Section 5.4). Additionally, we tested the method robustness, evaluating its

Table 3
Parameters used in the simulations. The default values are highlighted in bold font.

Parameter	*Symbol*	*Value*
Node sensing range	$r_{s,n}$	30 m
Node communication range	$r_{c,n}$	60 m
Agent sensing range	$r_{s,a}$	10 m
Agent communication range	$r_{c,a}$	60 m
Agent cruise driving speed	$\bar{v}$	5 m/s
Agent cruise steering speed	$\bar{\omega}$	0.1 rad/s
Bound threshold	Th_b	0.3 m
Bearing quantization levels	K	4
Time step duration	dt	{0.5, **1**, 2, 4, 10} s
Avg. Perc. RaB distance error	ρ	{0, 1, **1.5**, 5, 20} %
Avg. error on RaB bearing	γ	{0, 2, **5**, 15, 20} °
Avg. Perc. agent failure	P_f	{**0**, 5, 10, 25, 50} %

resilience to potential disruptive elements like noise, data frequency, and both predictable and unpredictable faults (Section 5.5). Finally, we compared our method against several state-of-the-art approaches (Section 5.6).

5.1. Experimental settings

We considered networks composed of 125 randomly deployed nodes. Each network is designed to fully cover the region of interest, allowing small overlapping among nodes sensing area, but without inducing high density. Holes in the coverage are modeled through the failure of a specific number of adjacent nodes that leave part of the region uncovered. Unless expressly stated otherwise, we relied on a default scenario consisting of a single hole caused by the failure of 7 adjacent nodes (medium-sized hole) and the parameters in Table 3. These holes presented an irregular but compact shape. The communication and sensing ranges of nodes and agents were selected to meet Section 3 assumptions and to align with existing technologies, such as [49]. The swarm perceived the environment once per time step using the RaB sensors. The data provided by these sensors was perturbed using two zero-mean Gaussian noises with a constant variance for the bearing information and a variance proportional to the measured distance for the range. Default noise values are reported in Table 3 and are compliant with modern sensors [42,49]. We set the control speeds of the agents to match the values observed in commercial quadrotors, e.g., [50]. By default, the agents were released from a single RP, which represents the most challenging, yet realistic, configuration for our algorithm (as further discussed in Section 5.4). For each experiment, the swarm size was empirically determined and carefully calibrated to avoid both inflated performance resulting from an excess of agents and undue penalization due to their scarcity.

For each experiment, we carried out 70 simulations lasting $T = 1000$ time steps each. The default duration of a time step was set to 1s. In every simulation, we introduced variability to the scenario by randomly altering RPs location, the network topology, and the shape and position of the holes, while ensuring that key characteristics remained comparable.

As a measure of performance, we relied on three metrics: *(i)* average coverage over time; *(ii)* number of deployed agents and *(iii)* average energy spent over time. Specifically, the first metric, ranging from 0 to 1, measures the fraction of the hole area effectively recovered by the deployed agents. We will refer to as *convergence*, the attainment of a stable value by the coverage before the simulation ends. The number of deployed agents, instead, serves as an indicator of agents placement efficiency: when coverage values are comparable, a reduced number signifies more effective resource management by the algorithm. Additionally, at each time step, we collected the average energy consumption of the swarm (according to the battery model detailed in Section 4.3) to quantify the energy demands of the methods.

Fig. 5 exemplifies the healing process in a scenario with multiple holes.

Table 4
Average number of deployed agents and corresponding 95% confidence interval for different navigation strategies and for Simionato et al. [6] in a single 5-failures hole scenario.

	Avg. no. agents	95% CI
Continuous	22.28	7.96
Discrete	22.12	8.13
Simionato et al. [6]	27.78	2.57

5.2. Parameters optimization and strategy comparison

In Section 4.2, we hypothesized that coupling the navigation strategy lead by quantized perception with the placement logic using the continuous RaB information could accelerate the healing process while deploying a similar amount of agents. To prove this hypothesis, we considered two strategies: *Discrete* navigation (as in [14]), where data from RaB sensors is quantized before being fed to the discrete navigation module, and *Continuous* navigation, where RaB readings are directly used as in [6]. Both strategies are coupled with the placement logic described in Section 4.2. Before comparing the performance of these strategies, we optimized their parameters. For the Continuous strategy, we optimized Th_c (threshold for bordering a hole) in the range $[0.75, 1.0]$ and the *policy* for ranking potential deployment positions. To do so, we employed the Differential Evolution algorithm, using Scipy's default parameters and the maximization of the coverage at convergence as fitness function. The exploration of the parameter space was performed on a single-hole scenario with 5 adjacent faulty nodes, using default parameters from Table 3 and a swarm size of 30.

Fig. 6a, shows the fitness landscape. The algorithm's poor performance at low Th_c values is due to nodes and deployed agents tolerating more uncovered border within their sensing disks, hindering agent adhesion by not signaling their presence on the hole border. Conversely, as values approach 1, most nodes and deployed agents signal their presence on the hole border, even when adequately covered. This results in a lack of guidance for the agents, overwhelmed by the number of potential solutions. The plateau in the fitness landscape prompted additional analysis to identify the optimal parameter set. We conducted 50 simulations for each Th_c value in $[0.9, 0.95, 0.98]$ with all policies, introducing randomness in the scenario for a more robust performance assessment. The resulting coverage at convergence, shown in Fig. 6b for all combinations, allowed us to discern the best performing sets of values. However, in time-sensitive contexts, it is also important to analyze the time required to reach convergence, i.e., the transient duration. Fig. 6c illustrates the average coverage over time for the best performing sets, indicating $Th_c = 0.95$ and $closest$ policy as the optimal values.

We repeated the same operation for the Discrete strategy, including the parameters associated with the long-term wiggle: $T_w \in [50, 300]$, $n_w^i \in [1, 10]$, and $t_w^i \in [0, 1000]$. For clarity, Fig. 7 shows only the region of the fitness landscape generated by the DE with values over 0.98. Based on this information, we restricted the parameters space and analyzed the performance over time and at convergence of the best performing sets of parameters. From this analysis emerged that, among those sets, $Th_c = 0.95$ and $closest$ policy confirm as the best values, while there is no statistically significant difference among values for the wiggle parameters, leading us to choose $T_w = 300$, $n_w^i = 7$ and $t_w^i = 750$ as the best set of parameters for the Discrete variant.

Using the same setting and the optimal values, we compared the two strategies by performing 70 simulations on the same set of scenarios. For an early assessment, we included in the comparison the algorithm in [6], to evaluate the improvements introduced in this work. Although [6] employs the Continuous navigation strategy, it lacks the commitment logic and the capability to handle holes adjacent to the ROI frontier. From Fig. 8 and Table 4, we derived that both navigation strategies achieve comparable high values of coverage at convergence

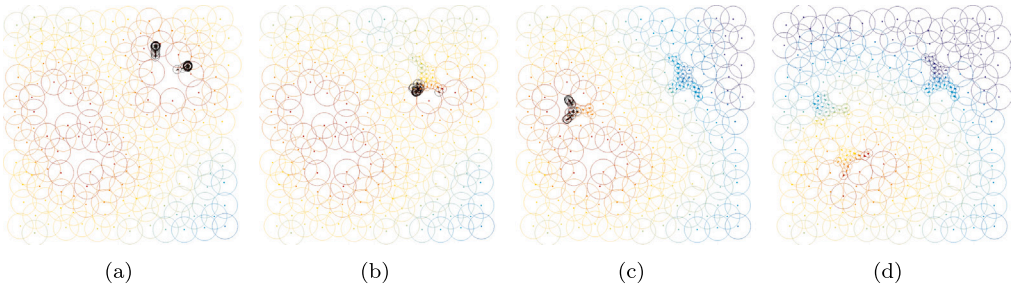


Fig. 5. Snapshots of a multi-hole healing process for subsequent time steps [6].

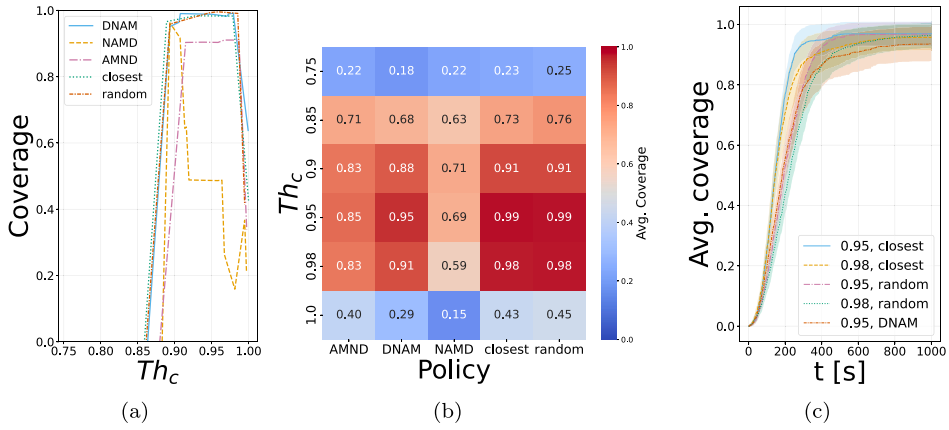


Fig. 6. Parameters optimization for the Continuous strategy: DE landscape (a); plateau analysis using coverage at convergence (b); avg. coverage over time for the best performing sets of parameters. The shaded bands represent the 95% confidence interval.

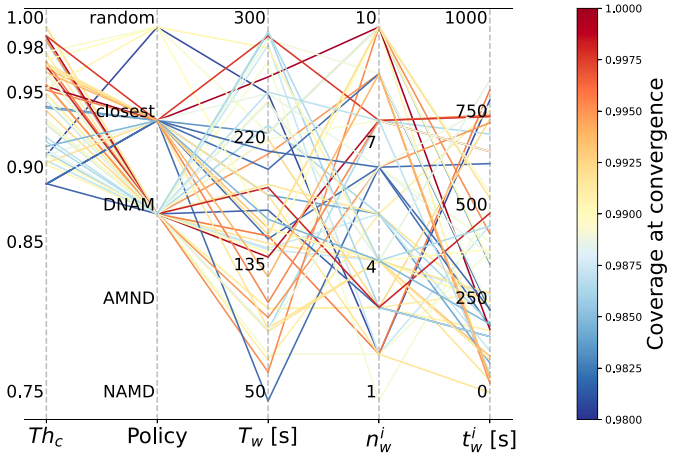


Fig. 7. DE landscape from Discrete parameters optimization. Only instances with coverage at convergence greater than 0.98 are reported for clarity.

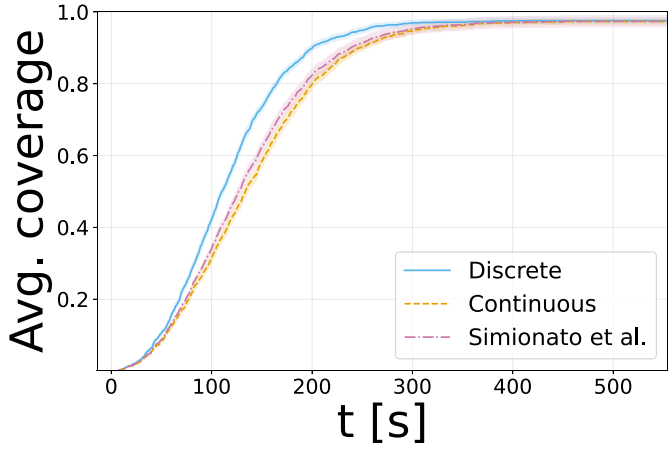


Fig. 8. Average coverage over time for the two navigation strategies and for Simionato et al. [6] in a single 5-failures hole scenario. The shaded bands represent the 95% confidence interval.

while deploying a similar number of agents. Using the same navigation strategy of Continuous, the shape of the coverage over time of [6] is similar to the Continuous curve, except for the slightly quicker transient caused by the simultaneous placements of agents in close positions. The values in Table 4 show how the commitment strategy, introduced in this work, allows obtaining similar coverage using less agents. Overall, the Discrete strategy presents a much quicker transient that the other two curves, thus confirming the initial hypothesis and becoming the default navigation strategy for the rest of the evaluation. Using this logic brings also the advantage of reducing the impact that proportional error has on the long distances usually encountered during navigation: the perturbation is absorbed by the quantization levels.

5.3. External factors analysis

To study the adaptability of our algorithm to different scenarios, we analyzed the impact that external factors have on performance. We consider as external factors those not under our control, such as the size, shape, position, and multiplicity of holes.

To adjust the hole size, we varied the parameter m, representing the number of faulty adjacent nodes. We investigated various hole dimensions, including minimum ($m = 1$), small-medium ($m = 5$), medium ($m = 10$), large ($m = 20$), and extremely large ($m = 40$) configurations. These latter values were included to test the limits of our method. For each dimension, the swarm size was selected according

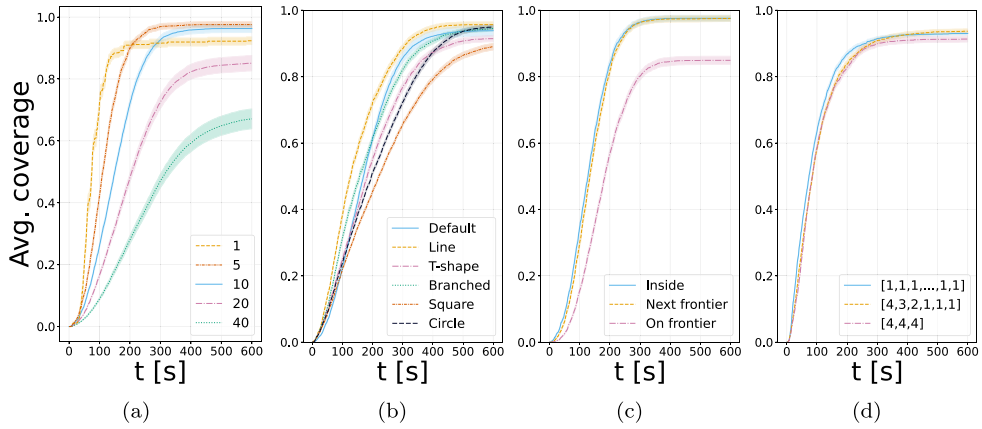


Fig. 9. Avg. coverage over time for different hole sizes (a); shapes (b); locations (c) and multiplicities (d). The shaded bands represent the 95% confidence interval.

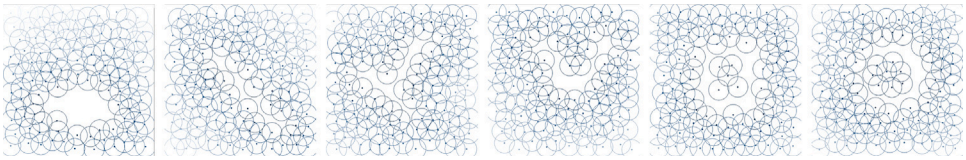


Fig. 10. Examples of scenarios with the shapes under analysis.

Table 5
Values of the geometric parameters regulating the primitives shape.

Primitive	*Parameter*	*Value*
Line	Length	100 m
T-shape	Longitudinal length	200 m
	Cross length	200 m
Branched	Main length	80 m
	Max. branch length	10 m
	No. branches	3
Square	Side length	120 m
Circle	Radius length	100 m

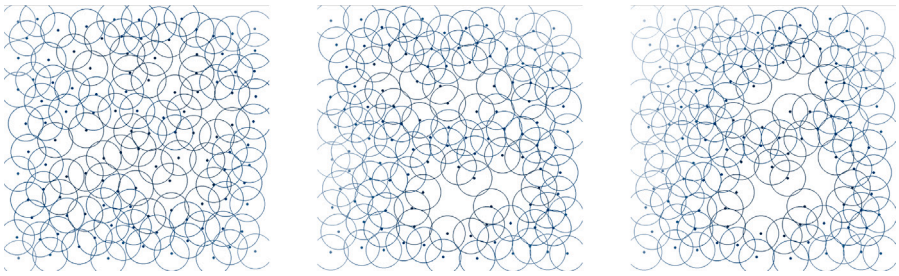
Table 6
Average number of deployed agents and corresponding 95% confidence interval for different hole locations in a single 7-failures hole scenario.

	Avg. no. agents	95% CI
Inside	33.60	11.65
Next frontier	32.47	13.23
On frontier	43.51	12.20

to the criteria in Section 5.1. Fig. 9a shows the average coverage over time for different values of m. As expected, the duration of the transient increases with the size of the hole: the larger the area, the longer it takes to traverse and cover it. Our method attains very high coverage across most dimensions. Yet, for the case $m = 1$, we note a decrease of the coverage at convergence, mainly caused by the limited pool of potential deployment positions. However, we recall that the coverage metric is calculated based on the hole area. Hence, a small uncovered area has a proportionally greater impact on smaller holes: deploying an agent to cover it would waste resources. In this analysis, we included massive holes (up to 1/3 of the network) to study the behavior of the algorithm in extreme scenarios. Despite the large hole, our method covers more than 69% of the area. The reduction in coverage is primarily linked to the necessity for the swarm to divide in response to the attraction from various directions. However, few agents might become trapped by small uncovered areas within one branch, failing to reconcile with the rest of the swarm and providing the support required for larger uncovered areas. In summary, our method quickly achieves high coverage, especially for small to medium holes (the most frequent in real-world scenarios), maintaining good performance even in the presence of massive failures.

We then investigated whether the shape of the hole could affect the performance. We considered different geometric primitives: line, t-shape, branched, square, and circle. To guarantee a fair analysis, the parameters regulating the shapes (see Table 5) were chosen to obtain similar areas. The resulting hole size is comparable to the failure of 15 adjacent nodes. The inclusion of the square and circle also aimed at studying how the algorithm responded to holes containing isolated but intact nodes. Fig. 10 exemplifies the scenarios generated by the different shapes. We involved 90 agents in the healing process. The average coverage over time reported in Fig. 9b indicates that the algorithm reaches high coverage independently from the shape of the hole. The longer transient produced by squared and circular holes can be attributed to the swarm not deploying as it moves through the intact parts of the hole, slowing down the healing process. For all shapes, the method deploys a statistically indistinguishable number of agents.

In this work, we modified the generation of the potential field and the logic for filtering potential positions, to allow restoring holes next to the ROI frontier. To confirm that healing such holes yields similar performance to those within the ROI, we performed 70 simulations considering a single hole with $m = 7$, the default parameters, and releasing 45 agents. Fig. 9c shows coverage at convergence statistically indistinguishable, but with a transient duration slightly longer in case of holes next to the frontier. This occurs because statistically, holes at the border are more likely to be farther from the release point, requiring more time for the swarm to reach them. Once reached, the healing process proceeds at a similar rate, as confirmed by the slope of the curves. In both cases, the number of deployed agents is very similar (see Table 6). Despite we assumed to work with bounded holes, we tested our algorithm also in presence of open holes, obtained by the failure of frontier nodes (Fig. 9c, pink line). As expected, the method still provides high coverage but converges on lower values. As the ROI frontier cannot be defined, agents cannot ascertain whether



(a) $m = [1, 1, 1, \cdots, 1, 1]$ (b) $m = [4, 3, 2, 1, 1, 1]$ (c) $m = [4, 4, 4]$

Fig. 11. Examples of the multi-hole scenarios under analysis.

Table 7
Average number of deployed agents and corresponding 95% confidence interval for different numbers of release points in a single 15-failures hole scenario.

	No. of RPs					
	1	2	4	8	16	40
Avg. no. agents	69.97	76.10	76.36	76.48	76.76	76.94
95% CI	39.85	11.64	9.58	9.12	7.92	7.93

they will deploy inside or outside the ROI. The metric computation only considers the area and agents deployed within the ROI, meaning that only a fraction of the deployed agents actually contributes to the coverage. However, these results suggest that by inserting virtual or physical landmarks acting as ROI delimiters, our algorithm could be used, without modifications, also in case of open holes.

Finally, we investigated how performance varies in presence of multiple holes. We considered three categories: *(i)* small isolated holes (12 single failures); *(ii)* a mixture of small and medium holes (6 holes with $m = [4, 3, 2, 1, 1, 1]$); *(iii)* three larger holes ($m = 4$ each). Despite the variability, the overall uncovered area is comparable and similar to the failure of 12 adjacent nodes. We used the default parameters in Table 3, with 5 RPs of 16 agents each. Fig. 11 exemplifies the multi-hole scenarios considered, while Fig. 9d shows the average coverage over time. As it is possible to see, the curves behave similarly, achieving high level of coverage at convergence, while using comparable number of agents. This indicates that the method naturally adapts to different hole layouts.

5.4. Internal factors analysis

In the evaluation of our algorithm, we assessed the impact that factors under our control have on performance. In particular, we carried out simulations varying the number of RPs in $\{1, 2, 4, 8, 16, 40\}$. We included the last case to simulate an idle swarm randomly scattered in the ROI. For each value, we released 80 agents, equally distributed among the RPs. We used the default parameters in scenarios with a single hole of 15 failures. The results in Fig. 12a for a single RP confirm the phenomenon already discussed in Section 5.3 for massive holes. Multiple RPs enable the swarm to approach the hole from various directions. This not only accelerates the healing process through parallelization, as indicated by the increasingly shorter transients, but also allows the swarm to heal different parts of the hole simultaneously, preventing entrapment, as indicated by the higher coverage at convergence. However, when the number of RPs exceeds 8, the difference in the transient duration drastically decreases: how the swarm approaches the hole does not differ significantly. This enables releasing the swarm from fewer points (a more realistic setting), without compromising on the responsiveness of the solution or the number of deployed agents (see Table 7).

To study the versatility of our method, we assessed performance changes with different agent sensing capabilities. To model diverse sensing hardware, we varied the sensing range in $r_{s,a} \in \{5, 10, 15, 30\}m$,

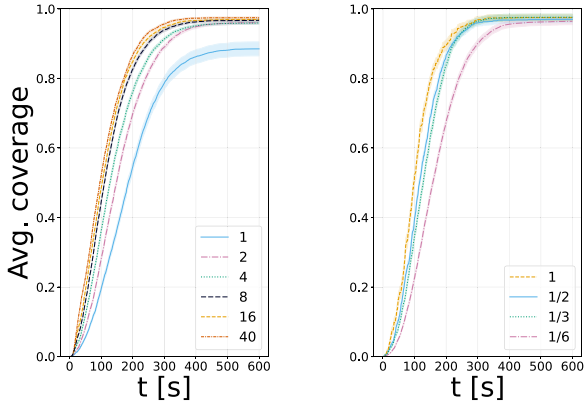


Fig. 12. Avg. coverage over time for different numbers of release points (a) and values of the ratio between sensing ranges (b). The shaded bands represent the 95% confidence interval.

representing $\{1/6, 1/3, 1/2, 1\}$ of the node sensing range, respectively. To accommodate the different coverage power, we adjusted the size of the swarm accordingly, i.e., using 140, 45, 20, and 8 agents, to heal a single 7-failures hole. We used default parameters and a single RP. According to Fig. 12b, there is no statistically significant difference among the values of the coverage at convergence. This indicates the adaptability of our algorithm to diverse agent sensor systems. The duration of transients, on the other hand, is influenced by the coverage of individual agents. As discussed in Section 5.3 for the variation in the hole size, if the sensing disk is smaller, it will take the swarm a longer time to cover the hole.

5.5. Robustness analysis

Employing swarm intelligence techniques brings the advantage of inherent robustness against destructive factors. To support this claim, we investigated how the algorithm performance reacts to two types of disruption: corrupted data and agents failure.

To account for varying precision levels among RaB sensors of different quality and the errors introduced by environmental factors, we varied the noise affecting the RaB readings. More precisely, the percentage error on the distance measurements was set to $\{0, 1, 1.5, 5, 20\}\%$, while the bearing noise to $\{0, 2, 5, 15, 20\}°$, respectively. In this analysis, we included also the case of ideal $(0\%, 0°)$ and strongly perturbed $(20\%, 20°)$ perception, associated with extreme conditions. For all noise levels, 45 agents from a single RP were tasked with healing a medium hole caused by 7 adjacent failures. The average coverage over time, shown in Fig. 13a, demonstrates that the higher the noise, the longer the transient. Strong errors compromise the computation and evaluation of potential deployment positions, generating conflicting decisions

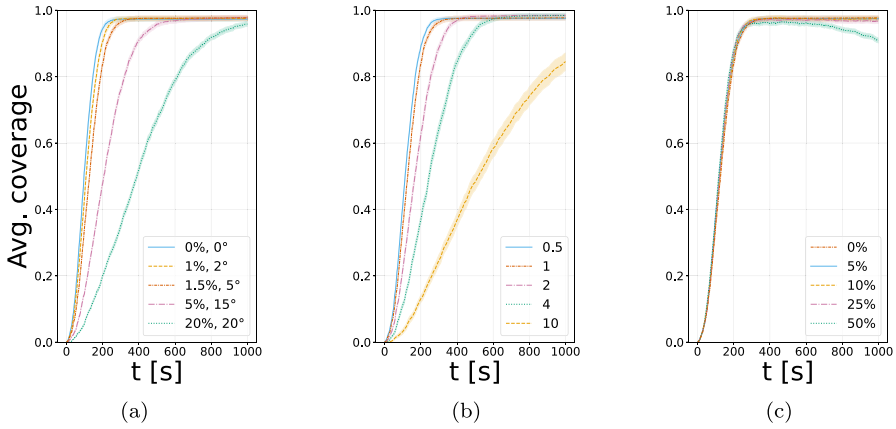


Fig. 13. Avg. coverage over time for different levels of noise affecting RaB readings (a); perception frequencies (b); percentages P_f of agent failure (c). The shaded bands represent the 95% confidence interval.

Table 8
Average number of deployed agents and corresponding 95% confidence interval for different noise levels affecting RaB readings in a single 7-failures hole scenario.

	(ρ, γ) [%,°]				
	(0,0)	(1,2)	(1.5,5)	(5,15)	(20,20)
Avg. no. agents	33.22	33.38	33.61	33.39	31.99
95% CI	11.71	11.89	11.65	11.16	10.88

Table 9
Average number of deployed agents and corresponding 95% confidence interval for different percentages of agent failure (P_f) in a single 7-failures hole scenario.

	P_f [%]				
	0	5	10	25	50
Avg. no. agents	33.61	33.87	34.16	34.59	31.06
95% CI	11.65	11.68	11.85	11.29	10.37

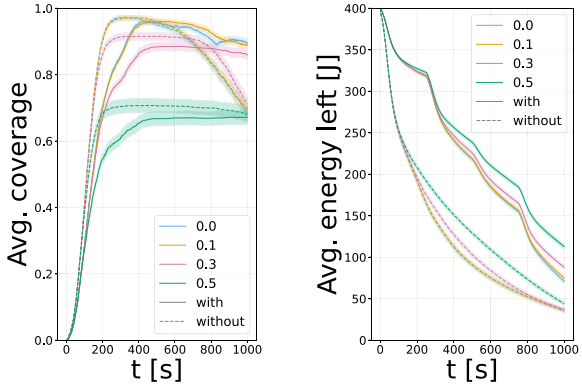


Fig. 14. Analysis of the effect of incremental release and proactive placement. Avg. coverage over time for different values of Th_e and with or without incremental release (a) and avg. energy left over time (b). The shaded bands represent the 95% confidence interval.

that delay the swarm. However, the swarm eventually achieves very high coverage, employing on average the same number of agents (see Table 8) regardless of the noise level.

Data corruption can also be related to a lower update frequency of RaB readings. Varying this frequency allowed us to model packet loss, with consequent retransmission, and environmental factors that could hinder the perception. In the same setting of the previous experiment, we fed the agents with one reading every $\{0.5, 1, 2, 4, 10\}$ seconds. By keeping the default cruise driving speed of 5 m/s, this corresponds to a *cognitive speed* (i.e., the ratio of perception frequency to motion speed) of $\{0.4, 0.2, 0.1, 0.05, 0.02\} m^{-1}$, respectively. Fig. 13b highlights how the duration of the transient is affected by the perception frequency. A more frequent evaluation of the environment enables the agents to quickly respond to changes, such as others taking the chosen position or altering their path to follow an updated potential field. Conversely, a sparser update forces the swarm to backtrack its last actions, thus slowing the healing. The agents' interchangeability allows achieving high coverage values with a similar number of deployed agents, even when access to the readings is extremely limited.

The interchangeability of the swarm was further evaluated by analyzing performance variations to unpredictable and predictable agent failure. The first type of fault is aimed at modeling the abrupt and unexpected failure of part of the swarm, e.g., due to malfunctioning hardware or hacking. To simulate agent failure, we conducted simulations where a fraction P_f of agents fail independently. We accomplished this by making each agent fail with probability $p = 1 - (1 - P_f)^{1/T}$ at every step of the simulation. We recall that T is the total number of time steps in the simulation. To objectively assess the algorithm recovery capability from agent failure, we increased the swarm size by precisely the number of agents expected to fail. This approach ensures that any fluctuations in performance are not attributed to a shortage of agents. We used the same scenario as the previous experiment and the default parameters in Table 3. In Fig. 13c we show the coverage trends for $P_f = \{0, 5, 10, 25, 50\}\%$. Except for $P_f = 50\%$, there is no statistically significant difference among the different trends of the average coverage over time or the number of deployed agents (reported in Table 9). This indicates that our algorithm is highly robust to agent failure. In the last case, instead, the reduction in the coverage toward the simulation end does not indicate poor robustness. Instead, as agents replace failed peers, they have access to a more restricted set of solutions, resulting in sub-optimal placement compared to the initial hole. This implies that we might require more agents than those deployed when we anticipate massive substitutions.

An example of predictable fault is energy depletion. In Section 4.3, we outlined the agent battery model. We proposed a preventive measure where agents below a fixed energy threshold act as faulty and we suggested releasing the swarm incrementally. To analyze the effectiveness of these countermeasures, we set the threshold $Th_e = \{0, 0.1, 0.3, 0.5\}$. For each value, we compared the performance with and without incremental release. In particular, we considered releasing the swarm in 4 stages, with 40%, 30%, 10%, and 20% being released every 250 time steps. The scenario is the same as the other robustness experiments but with 60 agents involved, to account for their partial failure. Fig. 14a reports the average coverage over time for all combinations. Irrespective of the threshold, solutions without incremental

release suffer a decrease in the coverage, especially toward the end of the simulations. This occurs because moving requires more energy than sensing the environment after deployment. As a result, the agents patrolling a healed hole waiting to replace their peers will run out of energy sooner than those deployed. When these latter will require a replacement, the majority of agents may no longer be available. This phenomenon is corroborated by the curves of the average energy left over time, reported in Fig. 14b. Without incremental release, the swarm energy decreases monotonically, while with it, the decline is slowed by the introduction of fresh agents (reflected in the peaks). Fig. 14a suggests that choosing excessively high or low values for Th_e could have a detrimental effect. When agents signal their failure prematurely ($Th_e = 0.5$), a significant portion of the area is redundantly covered by both the agents and their replacements. Given the limited swarm size, this may leave more distant areas uncovered. In contrast, if failure is signaled near actual energy depletion, the availability of agents for replacement will be limited (especially for large holes). From this experiment emerged that an incremental release of the swarm is beneficial, not only to prevent agent failure but also for a more efficient resource management when the size of the hole is unknown. Furthermore, the Th_e value represents a trade-off between achieving a higher coverage peak (lower values) and maintaining a more consistent coverage (higher values). This is particularly relevant when dealing with agents limited in both energy resources and number, allowing the selection of a behavior that better aligns with the scenario.

5.6. State-of-the-art comparison

We compared the performance of our algorithm against several state-of-the-art solutions: *(i)* Khalifa et al. [16], *(ii)* Yan et al. [17], *(iii)* Hallafi et al. [10], *(iv)* Khelil et al. [21], and *(v)* Simionato et al. [6].

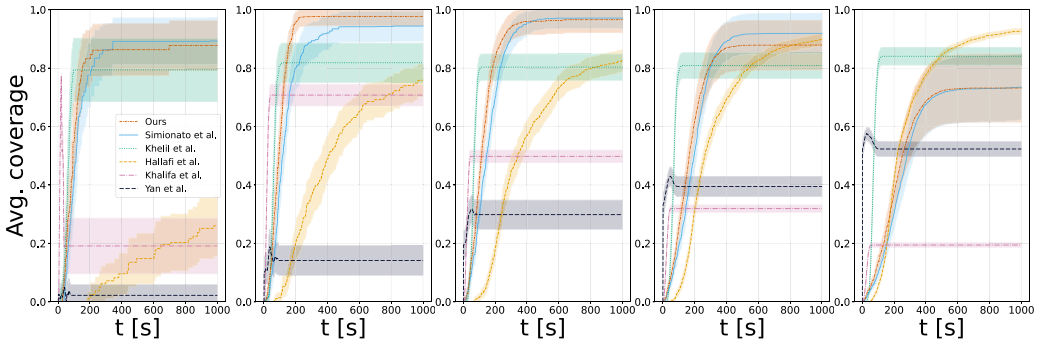
To ensure fairness in the comparison in terms of timing of movement, we endowed the methods with the same dynamics and control speeds of our approach, without affecting their logic. For all experiments of 70 simulations each, we considered squared regions of interest (as requested by [10,17]). The methods were evaluated on the same set of randomly generated scenarios, using the parameters in Table 3 for our algorithm and those reported in the corresponding studies for the other methods. To allow meaningful confrontation, we provided the same healing capacity (i.e., number of agents) for all methods. To accommodate the requirements of [16,21], the agent sensing range was set equal to that of the nodes. The comparison methods necessitate GPS signal. Since our algorithm is affected by noise, we perturbed the GPS signal using a zero mean Gaussian model with standard deviation of 4.9 m [51]. The algorithm of Hallafi et al. [10] explicitly outlines the sub-regions within the ROI that mobile agents need to reach. However, the method for identifying the location of the hole within that specific area, and where the agents deploy, is not clearly defined. To address this issue, we endowed the agents with a basic navigation strategy, i.e., a random walk constrained within the associated region, and a greedy deployment policy. We recall that, in all methods, coverage is calculated as the ratio of the recovered area to the initial hole area, in contrast to [16,17], where the reference area is the entire ROI. This approach ensures that the performance is not skewed by the massive contribution from the intact network.

To measure the scalability, we analyzed how performance compares when varying the size of a single hole. As in Section 5.3, we set $m = \{1, 5, 10, 20, 40\}$ with a corresponding swarm size of $\{2, 6, 11, 21, 41\}$. Fig. 15 shows the resulting coverage over time. The trends for various m values indicate a decline in Khalifa et al.'s algorithm performance with increasing hole size. This is because this method involves in the healing process only nodes at the border of the hole, therefore drawing on an insufficient amount of resources for tackling larger holes. The peak for $m = 1$ results from the excessive movements of healing nodes, that leave new parts uncovered. Conversely, the method in Yan et al. seems to benefit from larger holes. Since its fitness function is based on the coverage of specific points (see Section 2), the larger the hole, the more points it will include, leading to more informative values for the objective function and thus to a more effective placement of mobile agents. For a tiny hole, the probability that it contains these points tends to zero, lacking on cues on where to place the agents. The initial step displayed in Fig. 15 indicates the presence of mobile agents resting inside the hole. Once activated, they instantly provide coverage. Hallafi et al.'s algorithm performs better, especially for larger holes, showcasing the power of swarm-inspired behaviors. However, the lack of additional guidance inside the sub-regions translates into a much longer transient. With only random movements to guide the agents, the time required to find uncovered areas lengthens in the presence of smaller holes. Khelil et al.'s method achieves consistent and high coverage across various hole sizes. However, as the hole size increases, there is a proportional rise in the computational power required, following a quadratic trend (see Section 5.6.1). The short transient period is a result of the algorithm running only once to determine the final positions that mobile agents can approach directly. Nevertheless, this makes it less responsive to changes. As briefly shown in Section 5.2, Simionato et al.'s approach exhibits similar behavior to our algorithm but with longer transients and employing more agents. In general, our algorithm outperforms all the methods, except for massive holes. As discussed in Section 5.3, our algorithm seems to be particularly scalable. However, for extremely large holes, the swarm might get trapped in some parts of the hole, failing to complete the healing. In this case, the freedom of movement of [10,21] allows overcoming this issue and reaching higher coverage.

As a measure of flexibility, we then considered the same multi-hole scenarios of Section 5.3 and 15 agents. The results in Fig. 16 show both Khalifa et al. and Yan et al.'s methods performing better with fewer holes. In the former, the nodes facing multiple holes cannot move, and hence cannot participate in the healing process. For the latter, instead, there is a slightly higher probability that fewer but bigger nodes will contain more points useful for evaluating deployment configurations. Hallafi et al.'s and our algorithms are not significantly influenced by hole multiplicity. The algorithm in Khelil et al. achieves high coverage for all multiplicities, performing slightly better in case of less, bigger holes. For small holes, the vertices of the Voronoi diagram might not be well-centered or not placed within the holes, thus penalizing the overall coverage. As before, Simionato et al.'s method attains high coverage at convergence, but at the cost of longer transients and more agents required. In all cases, our algorithm outperforms the others.

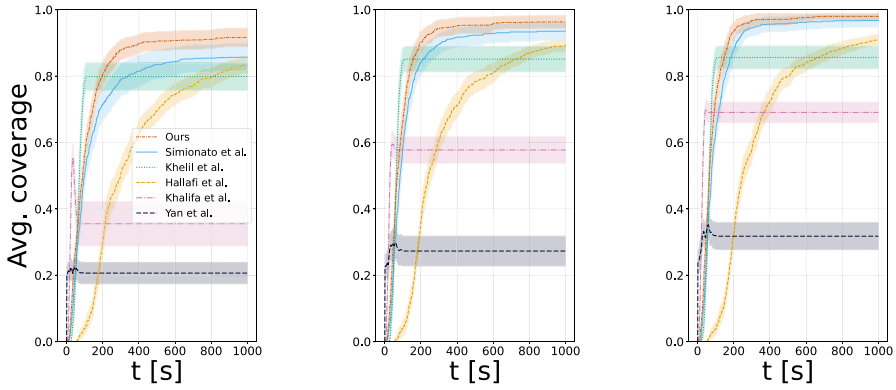
We compared the robustness to agent failure by forcing the failure of a fraction $P_f = \{0, 5, 10, 25, 50\}\%$ of agents. In this case, we adapted the formula in Section 5.5 considering the appropriate duration of activity of the swarm and not just the simulation time. We selected a single hole caused by the failure of 7 adjacent nodes and employed $\{8, 9, 9, 10, 12\}$ agents, respectively. The robustness of Yan et al.'s algorithm, inferred by the trends in Fig. 17, is only apparent: the low coverage achieved implies that the failure affected mostly mobile agents not actively contributing to the healing. Khalifa et al.'s method, lacking a countermeasure, confirms to be strongly affected by agent failure. Even in Khelil et al.'s algorithm, the absence of a countermeasure for agent failure results in a reduction of coverage. However, the decrease is slower compared to Khalifa et al.'s method. This is because Khelil et al.'s deployment approach allows for strong agent overlap, providing redundancy in coverage when failures occur. Conversely, swarm-based solutions exploit the interchangeability of their agents, showing high robustness to agent failure.

Besides confronting the average coverage over time, we endowed each method with the battery model in Section 4.3 and the battery parameters of Table 2. For all methods, we employed the shortest encoding possible for messages. This allowed comparing the average energy spent by the different solutions. Table 10 collects the average decay rate of agents' energy. On average, Khalifa et al. and Yan et al.'s



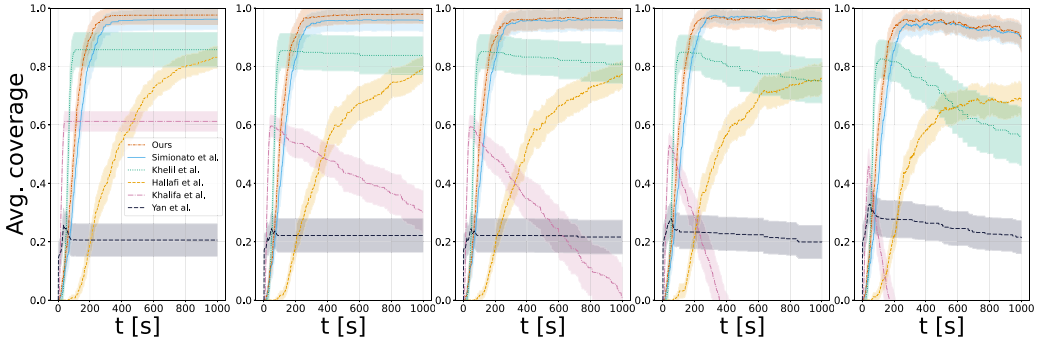
(a) $m = 1$ (b) $m = 5$ (c) $m = 10$ (d) $m = 20$ (e) $m = 40$

Fig. 15. Comparison among the methods on the avg. coverage over time for different hole sizes m. The shaded bands represent the 95% confidence interval.



(a) $m = [1, 1, 1, \cdots, 1, 1]$ (b) $m = [4, 3, 2, 1, 1, 1]$ (c) $m = [4, 4, 4]$

Fig. 16. Comparison among the methods on the avg. coverage over time for different multi-hole scenarios. The shaded bands represent the 95% confidence interval.



(a) $P_f = 0\%$ (b) $P_f = 5\%$ (c) $P_f = 10\%$ (d) $P_f = 25\%$ (e) $P_f = 50\%$

Fig. 17. Comparison among the methods on the avg. coverage over time for different agent failure percentages P_f. The shaded bands represent the 95% confidence interval.

Table 10
Average decay rate of battery level in [J/s] and corresponding 95% confidence interval for the methods under comparison.

Method	Avg. decay rate	95% CI
Ours	0.82	0.02
Khalifa et al. [16]	8.16	0.23
Yan et al. [17]	9.67	0.27
Hallafi et al. [10]	1.53	0.04
Khelil et al. [21]	**0.24**	**0.09**
Simionato et al. [6]	0.81	0.03

algorithms require similar, and higher, energy per second to accommodate the complex messaging protocol and the exchange of information necessary for the optimization, as well as the movement of the nodes. In Hallafi et al. and Khelil et al. the computations are delegated to the base station or to the hole managers, respectively. Hence, agents spend their energy solely on movement, with Hallafi et al.'s method requiring more energy on average due to its agents extended mobility. Our algorithm exhibits a decay rate comparable to that of Simionato et al. indicating that the enhancements introduced in this work do not cause higher energy consumption. Overall, our algorithm encompasses informed movements and a low amount of data exchanged, resulting in an efficient energy management.

5.6.1. Complexity analysis

Consider a network of N nodes and k mobile agents. In our algorithm, nodes and agents operate on their RaB readings, that collect information about the m nodes and agents within their communication

Table 11
Complexity analysis for the methods under comparison. Hallafi et al.'s values are of the complexity of the computations performed by the base station and cell agents, respectively. N is the number of network nodes; n that of the one-hop neighbors; b that of the nodes bordering the hole; m that of the nodes and agents within the communication radius; k the number of the agents; j that of the nodes in the same cell and l the pixels in the sensing disk.

Method	Hole detection	Hole healing
Ours	$\mathcal{O}(m)$	$\mathcal{O}(m^3)$
Khalifa et al. [16]	$\mathcal{O}(bn^2)$	$\mathcal{O}(1)$
Yan et al. [17]	$\mathcal{O}(Nk^2)$	$\mathcal{O}(1)$
Hallafi et al. [10]	$\mathcal{O}(N)$, $\mathcal{O}(Njl)$	$\mathcal{O}(Nk)$, $\mathcal{O}(1)$
Khelil et al. [21]	$\mathcal{O}(n^2)$	$\mathcal{O}((b+k)^2)$
Simionato et al. [6]	$\mathcal{O}(m)$	$\mathcal{O}(m^3)$

range (i.e., $m \ll N$ and $m \leq k$). To detect a hole, they filter out active agents' data from the readings and, for each of the remaining nodes and bound agents, they perform mathematical computations to obtain the ACR and the level. In the worst case scenario, these operations are $\mathcal{O}(m)$. To heal the holes, the agents filter the readings for the data of nodes or bound agents of level zero ($\mathcal{O}(m)$). To find pairs of parents, $\binom{m}{2}$ operations are needed. For each pair, mathematical computations are performed to find one or two solutions. Each solution is then checked against every node and bound agent perceived. These operations have an overall complexity of $\mathcal{O}(m^3)$. Ranking the solutions is $\mathcal{O}(m \log m)$, while moving toward the best one is $\mathcal{O}(1)$. In case no solution is found, moving in the direction of the closest location with minimum level is $\mathcal{O}(m)$. The commitment strategy requires $\mathcal{O}(m)$ computations. Overall, the healing process has $\mathcal{O}(m^3)$ complexity.

Table 11 reports the complexity analysis for our method and those under comparison. Each approach operates on a different set of nodes and depends on particular parameter choices. Therefore, we express complexity in relation to these factors and establish relations among them to facilitate comparison. We denote by b the number of nodes that border a hole and by n the number of one-hop neighbors of a node. j represents the number of nodes in the same cell, when the network is cellularized. This number depends on the chosen grid resolution. l is the number of pixels contained in the sensing disk area and grows with higher sensing capabilities. These quantities are linked according to $n \leq b \leq m < j \ll N < l$ and $k < N$. From Table 11, we draw that our method has the lowest complexity in detecting holes. For the hole healing stage, it exhibits a higher complexity than the methods where the agents are just instructed to drive toward specific points ($\mathcal{O}(1)$). The higher computational complexity in this stage arises from the online planning performed by the agents, which involves continuously re-evaluating environmental conditions but offers adaptability to changes and failures. Nevertheless, our method's complexity is lower than some other approaches. We stress that our algorithm has the same complexity of Simionato et al.'s, meaning that our improvements did not impose additional computational requirements.

6. Conclusion

In this work, we proposed a swarm intelligence-based algorithm for hole detection and healing in WSNs. The swarm navigates the network toward the closest hole and deploys in it, starting from the borders, until the restoration is complete. This approach focuses on minimizing the number of agents deployed and enables the healing of holes regardless of their position in the region. After conducting a thorough evaluation, we verified that the integration introduced in this work, combining the deployment logic with a navigation strategy based on highly quantized bearing information, enhances performance. This merge accelerates the healing process and introduces robustness to sensor readings affected by noise, all while deploying fewer agents. The algorithm exhibited good scalability and flexibility, achieving high coverage across various hole sizes, diverse shapes, and in the presence of multiple holes. It also showed its applicability to different agent sensing capabilities, and extreme robustness to data corruption, both in terms of noise affecting the readings and of lower frequency in their updates. We demonstrated how the algorithm not only can sustain massive agent failures (expected or unexpected), but it exploits the energy information, provided by the newly added battery model, to proactively prevent further loss of coverage, thus extending the WSN lifetime. These results highlight the applicability of our method even in extreme scenarios. Our algorithm demonstrated superior performance compared to several state-of-the-art methods in nearly all cases. However, in the event of significant node failures, we noticed a decline in coverage. This occurs as the swarm tends to branch out, resulting in some agents focusing on covering partially healed holes rather than supporting the rest of the swarm in restoring coverage across larger areas. To overcome this limitation, future research should focus on developing strategies to redirect the swarm if no deployment occurs within a specific time frame. Moreover, working toward the inclusion of obstacles in the environment could further reduce the gap with real-world applications.

CRediT authorship contribution statement

Giada Simionato: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Formal analysis, Data curation, Conceptualization. **Mario G.C.A. Cimino:** Writing – review & editing, Supervision, Project administration, Methodology, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

Work partially supported by the University of Pisa, Italy, in the framework of the PRA 2022 101 project "Decision Support Systems for territorial networks for managing ecosystem services". Work partially supported by the Italian Ministry of Education and Research (MIUR, Italy) in the framework of the FoReLab project (Departments of Excellence). Work partially supported by the European Commission under the NextGenerationEU program: (i) Partenariato Esteso PNRR PE1 - "FAIR - Future Artificial Intelligence Research" - Spoke 1 "Human-centered AI", (ii) PNRR - M4 C2, Investment 1.5 "Creating and strengthening of innovation ecosystems", building "territorial R&D leaders", project "THE - Tuscany Health Ecosystem", Spoke 6 "Precision Medicine and Personalized Healthcare", (iii) National Sustainable Mobility Center CN00000023, Italian Ministry of University and Research Decree n. 1033—17/06/2022, Spoke 10.

References

[1] N. Kukunuru, D. RajyaLakshmi, A. Damodaram, Hybrid approach for detecting and healing the coverage-hole in wireless sensor network, in: 2014 International Conference on Signal Propagation and Computer Technology, ICSPCT 2014, IEEE, 2014, pp. 110–115.

[2] R. Chowdhuri, M.K.D. Barma, Node position estimation based on optimal clustering and detection of coverage hole in wireless sensor networks using hybrid deep reinforcement learning, J. Supercomput. (2023) 1–33.

[3] P. Singh, Y.-C. Chen, Sensing coverage hole identification and coverage hole healing methods for wireless sensor networks, Wirel. Netw. 26 (3) (2020) 2223–2239.

[4] S. Zhai, Z. Tang, D. Wang, Z. Li, X. Chen, D. Fang, F. Chen, Coverage hole detection and recovery in wireless sensor networks based on RSSI-based localization, EUC, in: 2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing, vol. 2, IEEE, 2017, pp. 250–257.
[5] C. Duan, Y. Luo, G. Fu, J. Feng, H. Chang, Research on coverage reinforcement strategy in ocean surveillance wireless sensor networks, in: 2022 IEEE 22nd International Conference on Communication Technology, ICCT, IEEE, 2022, pp. 829–835.
[6] G. Simionato, F.A. Galatolo, M.G. Cimino, Swarms of artificial platelets for emergent hole detection and healing in wireless sensor networks, in: Proceedings of the Genetic and Evolutionary Computation Conference, 2023, pp. 75–83.
[7] A. Soundarya, V. Santhi, An efficient algorithm for coverage hole detection and healing in wireless sensor networks, in: 2017 1st International Conference on Electronics, Materials Engineering and Nano-Technology, IEMENTech, IEEE, 2017, pp. 1–5.
[8] F. Papi, H. Barati, Hdrm: A hole detection and recovery method in wireless sensor network, Int. J. Commun. Syst. 35 (8) (2022) e5120.
[9] S. Devi, A. Sangwan, A. Sangwan, M.A. Mohammed, K. Kumar, J. Nedoma, R. Martinek, P. Zmij, The use of computational geometry techniques to resolve the issues of coverage and connectivity in wireless sensor networks, Sensors 22 (18) (2022) 7009.
[10] A. Hallafi, A. Barati, H. Barati, A distributed energy-efficient coverage holes detection and recovery method in wireless sensor networks using the grasshopper optimization algorithm, J. Ambient Intell. Humaniz. Comput. 14 (10) (2023) 13697–13711.
[11] P. Nicopolitidis, M.S. Obaidat, G.I. Papadimitriou, A.S. Pomportsis, Wireless Networks, John Wiley & Sons, 2003.
[12] J. Wang, C. Ju, H.-j. Kim, R.S. Sherratt, S. Lee, A mobile assisted coverage hole patching scheme based on particle swarm optimization for WSNs, Cluster Comput. 22 (1) (2019) 1787–1795.
[13] A.M. Khedr, W. Osamy, A. Salim, Distributed coverage hole detection and recovery scheme for heterogeneous wireless sensor networks, Comput. Commun. 124 (2018) 61–75.
[14] G. Simionato, M. Parola, M.G. Cimino, Impressionist hole detection and healing using swarms of agents with quantized perception, in: 2023 IEEE Symposium Series on Computational Intelligence, SSCI, IEEE, 2023, pp. 1213–1220.
[15] M.R. Senouci, A. Mellouk, K. Assnoune, Localized movement-assisted sensordeployment algorithm for holedetection and healing, IEEE Trans. Parallel Distrib. Syst. 25 (5) (2013) 1267–1277.
[16] B. Khalifa, Z. Al Aghbari, A.M. Khedr, A distributed self-healing coverage hole detection and repair scheme for mobile wireless sensor networks, Sustain. Comput. Inf. Syst. 30 (2021) 100428.
[17] L. Yan, Y. He, Z. Huangfu, A fish swarm inspired holes recovery algorithm for wireless sensor networks, Int. J. Wirel. Inf. Netw. 27 (2020) 89–101.
[18] C. Qiu, H. Shen, K. Chen, An energy-efficient and distributed cooperation mechanism for k-coverage hole detection and healing in WSNs, IEEE Trans. Mob. Comput. 17 (6) (2017) 1247–1259.
[19] N. Hoff, R. Wood, R. Nagpal, Effect of sensor and actuator quality on robot swarm algorithm performance, in: 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2011, pp. 4989–4994.
[20] T. Aust, M.S. Talamali, M. Dorigo, H. Hamann, A. Reina, The hidden benefits of limited communication and slow sensing in collective monitoring of dynamic environments, in: International Conference on Swarm Intelligence, Springer, 2022, pp. 234–247.
[21] A. Khelil, R. Beghdad, A. Khelloufi, 3HA: hybrid hole healing algorithm in a wireless sensor networks, Wirel. Pers. Commun. 112 (1) (2020) 587–605.
[22] P. Kumar Sahoo, M.-J. Chiang, S.-L. Wu, An efficient distributed coverage hole detection protocol for wireless sensor networks, Sensors 16 (3) (2016) 386.
[23] X. Li, D.K. Hunter, K. Yang, Wlc12-1: Distributed coordinate-free hole detection and recovery, in: IEEE Globecom 2006, IEEE, 2006, pp. 1–5.
[24] L. Aliouane, M. Benchaiba, HACH: healing algorithm of coverage hole in a wireless sensor network, in: 2014 Eighth International Conference on Next Generation Mobile Apps, Services and Technologies, IEEE, 2014, pp. 215–220.
[25] M. Davoodi, E. Delfaraz, S. Ghobadi, M. Masoori, Hole detection and healing in hybrid sensor networks, 2021, arXiv preprint arXiv:2106.10659.
[26] V. Narayan, A. Daniel, CHHP: coverage optimization and hole healing protocol using sleep and wake-up concept for wireless sensor network, Int. J. Syst. Assur. Eng. Manag. 13 (Suppl 1) (2022) 546–556.
[27] R.K. Rath, H.K. Kondaveeti, R. Nanda, S. Pattnaik, S.K. Satapathy, C2R: A mechanism for coverage and connectivity, hole recognition and repairment in wireless sensor network, in: 2022 International Conference on Knowledge Engineering and Communication Systems, ICKES, IEEE, 2022, pp. 1–7.
[28] M. Wu, An efficient hole recovery method in wireless sensor networks, in: 2022 24th International Conference on Advanced Communication Technology, ICACT, IEEE, 2022, pp. 1399–1404.
[29] R. Kadu, K. Malpe, Movement-assisted coverage improvement approach for hole healing in wireless sensor networks, in: 2017 Second International Conference on Electrical, Computer and Communication Technologies, ICECCT, IEEE, 2017, pp. 1–4.
[30] C. So-In, T.G. Nguyen, N.G. Nguyen, An efficient coverage hole-healing algorithm for area-coverage improvements in mobile sensor networks, Peer-to-Peer Netw. Appl. 12 (3) (2019) 541–552.
[31] Y.-H. Lai, S.-H. Cheong, H. Zhang, Y.-W. Si, Coverage hole detection in WSN with force-directed algorithm and transfer learning, Appl. Intell. 52 (5) (2022) 5435–5456.
[32] N. Chauhan, P. Rawat, S. Chauhan, Reinforcement learning-based technique to restore coverage holes with minimal coverage overlap in wireless sensor networks, Arab. J. Sci. Eng. 47 (8) (2022) 10847–10863.
[33] M. Ahlawat, M. Bhatia, D. Goyal, P. Gupta, Swarm intelligence based optimization in WSN: A review, Int. J. Comput. Sci. Manage. Stud. 13 (2) (2013).
[34] S. Mehta, A. Malik, A swarm intelligence based coverage hole healing approach for wireless sensor networks, EAI Endorsed Trans. Scalable Inf. Syst. 7 (26) (2020) e8.
[35] X. Yu, M. Xu, L. Cheng, N. Hu, A novel coverage holes detection and holes recovery algorithm in wireless sensor networks, in: The 27th Chinese Control and Decision Conference, 2015 CCDC, IEEE, 2015, pp. 3640–3644.
[36] F. Pourpanah, R. Wang, C.P. Lim, X.-Z. Wang, D. Yazdani, A review of artificial fish swarm algorithms: Recent advances and applications, Artif. Intell. Rev. 56 (3) (2023) 1867–1903.
[37] Y. Meraihi, A.B. Gabis, S. Mirjalili, A. Ramdane-Cherif, Grasshopper optimization algorithm: Theory, variants, and applications, IEEE Access 9 (2021) 50001–50024, http://dx.doi.org/10.1109/ACCESS.2021.3067597.
[38] K.M. Lynch, F.C. Park, Modern Robotics, Cambridge University Press, 2017.
[39] B. Wang, Coverage Control in Sensor Networks, Springer Science & Business Media, 2010.
[40] H. Zhang, J.C. Hou, et al., Maintaining sensing coverage and connectivity in large sensor networks., Ad Hoc Sens. Wirel. Netw. 1 (1–2) (2005) 89–124.
[41] Á. Gutiérrez, A. Campo, M. Dorigo, J. Donate, F. Monasterio-Huelin, L. Magdalena, Open e-puck range & bearing miniaturized board for local communication in swarm robotics, in: 2009 IEEE International Conference on Robotics and Automation, IEEE, 2009, pp. 3111–3116.
[42] U-blox, 2023, URL: https://www.u-blox.com/en/product/xplr-aoa-3-kit.
[43] M.F. Alam, K. Rahman, Platelet substitutes, in: Nanotechnology for Hematology, Blood Transfusion, and Artificial Blood, Elsevier, 2022, pp. 429–449.
[44] K.G. Link, M.G. Sorrells, N.A. Danes, K.B. Neeves, K. Leiderman, A.L. Fogelson, A mathematical model of platelet aggregation in an extravascular injury under flow, Multiscale Model. Simul. 18 (4) (2020) 1489–1524.
[45] A.-M. Kermarrec, M. Van Steen, Gossiping in distributed systems, ACM SIGOPS Oper. Syst. Rev. 41 (5) (2007) 2–7.
[46] K. McGuire, C. De Wagter, K. Tuyls, H. Kappen, G.C. de Croon, Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment, Science Robotics 4 (35) (2019) eaaw9710.
[47] W. Heinzelman, A. Chandrakasan, H. Balakrishnan, Energy-efficient communication protocol for wireless microsensor networks, in: Proceedings of the 33rd Annual Hawaii International Conference on System Sciences, Vol. 2, 2000, p. 10, http://dx.doi.org/10.1109/HICSS.2000.926982.
[48] G. Simionato, F.A. Galatolo, 2023. https://github.com/GiadaSimionato/HDHSim.git.
[49] Terabee, 2021. URL: https://terabee.b-cdn.net/wp-content/uploads/2021/02/Specification-Sheet-Evo-60m.pdf.
[50] DJI, 2023. URL: https://www.dji.com/it/mini-2/specs.
[51] F. Van Diggelen, P. Enge, The world's first GPS MOOC and worldwide laboratory using smartphones, in: Proceedings of the 28th International Technical Meeting of the Satellite Division of the Institute of Navigation, ION GNSS+ 2015, 2015, pp. 361–369.

Giada Simionato is a Ph. D. student at the Department of Information Engineering of the University of Pisa. She received her master's degree in Artificial Intelligence and Robotics in 2021 from the University La Sapienza of Rome. Her research interests include Swarm Robotics and Bio-Inspired Computing in Multi-Agent Systems. She is a member of the Machine Learning and Process Intelligence (MLPI) Team at the University of Pisa. She is (co-)author of several international conference papers on multi-agent systems. She is a teaching assistant for the course of Programming Fundamentals since 2022 at the University of Pisa.

Mario G.C.A. Cimino is an Associate Professor at the Department of Information Engineering of the University of Pisa (Italy). His research lies in the areas of Information Systems and Artificial intelligence. He is (co-)author of more than 100 international scientific publications. He is an Associate Editor of the Journal of Granular Computing (Springer) and the Journal of Ambient Intelligence and Humanized Computing (Springer). He is Vice-Chair of the IEEE CIS Task Force "Intelligent Agents", IEEE Computational Intelligence Society. He was co-chair of the 2021 and 2022 IEEE Symposium on Intelligent Agents (IEEE SSCI).