International Journal of Networked and Distributed Computing
Vol. **9**(**1**); January (2021), *pp.* 33–51
DOI: https://doi.org/10.2991/ijndc.k.201218.001; ISSN 2211-7938; eISSN 2211-7946
https://www.atlantis-press.com/journals/ijndc

Research Article

CHEAP: An Efficient Localized Area Coverage Maintenance Protocol for Wireless Sensor Networks

Gokou Hervé Fabrice Diédié[1,*], Boko Aka[2], Michel Babri[3]

[1]*Laboratoire de Mathématiques et d'Informatique, Université Peleforo Gon Coulibaly, Korhogo BP 1328, Côte d'Ivoire*
[2]*Laboratoire de Mathématiques et d'Informatique, Université Nangui Abrogoua, Abidjan 02 BP 801, Côte d'Ivoire*
[3]*Laboratoire de Recherche en Informatique et en Télécommunication, INPHB, Yamoussoukro BP 1093, Côte d'Ivoire*

ARTICLE INFO

Article History

Received 04 September 2020
Accepted 18 November 2020

Keywords

Coverage hole
detection
recovery
location-allocation
tabu search
wireless sensor network

ABSTRACT

Over the course of operation, a wireless sensor network can experience failures that are detrimental to the underlying application's objectives. In this paper, we address the problem of restoring coverage ratio of a damaged area (*hole*) using only the neighboring nodes. Most existing solutions fail to simultaneously prevent new holes formation, collisions, oscillations, cascaded movements, and overlapped areas. To do this, we propose an intersection points-based strategy to properly locate and characterize any type of coverage hole. Then, we allow nodes, whether or not redundant, to coordinate their movements and ranges in order to effectively eliminate the detected hole. We suggest for that purpose, a tabu search based optimization scheme along with a location-allocation model through a mixed integer linear program. Simulation results show that our protocol significantly increases the network's resilience.

1. INTRODUCTION

Wireless Sensor Networks (WSNs) are composed of a large number of tiny devices that are able to measure various physical quantities in their immediate environment. During the last two decades, WSNs have been utilized in many event-monitoring applications that are related to domains such as security, ecology, agriculture, transportation, or industry [1–3]. Unfortunately besides their limited capabilities, sensor nodes are inherently prone to failures [4], especially when they are randomly deployed in hostile environments. These failures can lead to the occurrence of uncovered areas also called *holes* that can be detrimental to network effectiveness [5]. Hence, coverage maintenance is of paramount importance to have any hope of prolonging network lifetime [6]. This process comprises two phases namely, detection and elimination.

When the underlying application requires a complete area coverage, hole detection use techniques originating from geometry, algebraic topology and graph theory [7–13]; whereas hole elimination solutions can exploit nodes' redundancy, mobility or motility [8,14–18]. Detection strategies must be fast and accurate whereas recovery schemes must eliminate the best the coverage hole while minimizing energy consumption and overlapped areas. This process must avoid creating new holes and be applied to both closed and open holes. It is also desirable that the coverage recovery strategy is localized and considers scenarios involving nodes with different ranges. To address all these requirements some authors opt for hybrid solutions which are essentially based on both nodes' sensing range adjustment (motility) and controlled movements (mobility) [19]. However, schemes commonly proposed usually struggle to simultaneously prevent undesirable effects such as new hole formation, collisions, and oscillations especially when nodes movements are constrained by obstacles.

In this paper, we propose to seamlessly combine mobility, redundancy control, and motility based strategies in order to further increase the network's resilience. The main contributions of this work are as follows:

- a localized intersection points based coverage hole detection scheme that helps to discover in linear time any type of hole (closed, semi-open, open), to minimize the use of geographical information, and to reduce message overhead;
- a location-allocation [20,21] based model and a mixed integer linear programming formulation for the area coverage recovery problem. A fully distributed tabu-search based scheme is applied to make hole elimination decisions. Unlike most existing solutions, our strategy guarantees energy-efficiency and high coverage ratio by simultaneously preventing new holes formation, cascaded movements, collisions, oscillations, especially in the presence of obstacles;
- a novel metric (the *coverage resilience index*) to help better estimate coverage hole elimination protocols' actual fault tolerance ability;
- extensive simulations using various scenarios are performed to validate the proposed algorithms. Results show that our scheme outperforms several recently proposed solutions with respect to

*Corresponding author. Email: *herve.diedie@upgc.edu.ci*
Data availability statement: The data that support the findings of this study are available from the corresponding author [GHFD] upon reasonable request.

the number displaced nodes, coverage ratio, total distance travelled, and energy consumption.

The rest of the paper is organized as follows: Section 2 surveys recent and significant related contributions; then, the proposed solution is detailed in Section 3; the performance evaluation process, the results, and discussions are presented in Section 4 followed by conclusion in Section 5.

2. RELATED WORK

Area coverage maintenance is a two-phase process encompassing hole detection and its elimination. Hole detection phase is aimed at providing maximum information (position, surface, shape, perimeter...) about the damaged areas (holes) resulting from a topology change. Once identified, holes must be healed using the least amount of resources.

2.1. Hole Detection

Area coverage hole detection is part of network boundary detection problem [22]. Methods commonly used for that purpose can be categorized into geometric, algebraic topology, graph theory, and analytic methods [23]. Accuracy and precision are the most important challenges to face, irrespective of the type of holes (close, open, semi-open). Techniques commonly used (virtual grid, intersection points, perimeter coverage, Voronoi tessellation, Delaunay triangulation...) mainly originate from computational geometry. They generally require nodes to know their exact positions. An et al. [24] proposed a combination of cells and triangles in order to reduce the computational complexity. However, this scheme is only limited to closed holes and homogeneous networks. Trong et al. [7] proposed a solution for dynamic holes. Besides detecting holes, this strategy is aimed at predicting the enlargement of their boundaries but has the drawback of increasing message overhead. Amgoth and Jana [8] suggest also using classical square grids. However, this strategy targets only closed holes.

Kang et al. [25] suggest a coordinate-free strategy based on the concept of *critical boundary points* i.e. intersection points which are not covered by any other node. Regrettably, finding such points is time consuming. Sahoo et al. [9] use a similar strategy except that it requires nodes to know their exact positions.

Huang and Tseng [26] used a perimeter coverage based strategy that regrettably requires also sensors to know their exact locations. In order to cope with this shortcoming, Bejenaro [27] suggests a concept called cyclic segment sequence which involves using nodes' local polar coordinates. A hole is detected if every selected arc (*segment*) does not overlap with exactly two other segments. However, this method has a high computational complexity $\mathcal{O}(n^3)$.

Qui et al. [28] proposed a k-coverage Delaunay triangulation oriented strategy. Any hole is now detected if a voronoi cell is covered by less than k nodes. Although innovative, this method has a $\mathcal{O}(n^2 \ln n)$ time complexity. Dai et al. [10] used also a voronoi diagram-based strategy. Unfortunately, the proposed solution is not fully-distributed (the latter diagram is constructed by the sink) and is thus not scalable. Li and Wu [29] proposed to merge isolated empty-circles in order to properly estimate hole's size. If the length of the common side of two Delaunay triangles is greater than their diameters then the two resulting isolated empty circles are considered as a hole. The two circles are merged if their centers are located at the same part of the common side. This strategy is able to detect both close and open holes, but cannot be applied to heterogeneous networks.

Senouci et al. [30] suggest using a collaborative scheme triggered by duly identified stuck nodes. Hole discovery process is based on the classical message forwarding TENT rule [31]. However, this strategy is dedicated to only closed holes. Chu and Ssu [32] used a location-free strategy that explicitly considers obstacles. However, this scheme requires each node to previously collect three-hops neighborhood information. In order to quickly detect holes, Patra and Sau [12] proposed to find a *base cycle* i.e. a cycle in the subgraph induced by each node's neighborhood. Regrettably, this technique is devoted to only closed holes.

To avoid using nodes exact locations, many authors suggest using techniques inspired by homotopy or homology to infer a simplicial complex from the network topology. However, Yan et al. [33] showed that it is impossible to detect with a Rips complex, certain types of holes including *triangular* holes, i.e. damaged areas enclosed by three nodes (two-simplexes). In a recent study, Šorbel et al. [11] used a spanning tree-based strategy for homological coverage verification. Small network segments are gradually merged into larger ones, until a Rips complex is obtained. This merging strategy is helpful to locally compute the first Betti number but hardly scalable since the spanning tree construction scheme is centralized.

2.2. Hole Elimination

Area coverage recovery is also a well studied topic in the literature. Solutions commonly proposed can be classified into redundancy, motility, and mobility based approaches.

Diongue and Thiare [34] proposed to maintain active some nodes (*sentinels*) so that they watch over their sleeping neighbors. This strategy combines node-based and link-based adaptation schemes. The link adaptation technique helps a *sentinel* to dynamically adjust its communication range according to link quality. On the other hand, node adaptation process consists of waking up redundant nodes to replace the failed *sentinels*. Unfortunately, this strategy fails to cope with large scale damages. Sharma and Sharma [18] propose a similar scheme to minimize movements by grouping nodes according to their overlapping coverage ratio. Each cluster is controlled by a node referred to as the *Zone Monitor* (ZM). When a cluster member fails, the ZM orders the sleeping redundant ones to be active in order to eliminate the resulting hole. Nodes' mobility is only used if no redundant node is found. Despite a low message overhead, no solution is proposed to cope with ZM failures.

As for mobility-based solutions, they mainly rely on nodes' locomotion abilities to replace failed neighbors especially after a large scale disaster. The challenges to face include the minimization of the number of candidates, overlapped areas, cascaded movements, the average distance travelled while avoiding collisions, oscillations and new holes formation. Strategies commonly found, involve using (attractive or repulsive) *virtual forces* inspired from quantum physics [35–37]. Li et al. [15] investigated using *Tchebychev point*

instead of targeting traditional points such as centroid or *Voronoi vertex*. Nodes must seek such a point in their *k*-order *Voronoi cell*. This scheme is very useful in 3D environments and helps to minimize oscillations but does not consider obstacles. Habibi et al. [38] use geometric optimization formulation. Although, this strategy provides a high coverage ratio, it cannot prevent the formation of new holes. To cope with this shortcoming, Sahoo and Liao [39] suggest to replace nodes according to their degrees while limiting their movements. They proposed a nonlinear programming formulation combined to a triangulation based scheme. The latter is aimed at minimizing energy consumption due to mobility and coverage overlapping. Regrettably, candidates selection scheme does not consider nodes' residual energy. Qiu and Shen [40] proposed a coordinate-free scheme that first constructs Delaunay triangles (i.e. triangles that have no other node inside) and then conducts nodes movements to meet conditions that suffice to ensure each triangle full coverage. New holes creation are avoided by ensuring that each node finds a s*afe area* capable of preserving its coverage. However, this technique considers only closed holes and requires nodes to synchronize their movements. Saha and Das [16] addressed this problem for heterogeneous networks, but the proposed scheme cannot be applied to open holes. Khelil and Beghdad [41] proposed a self-deployment scheme to move nodes periodically toward the centroid of the polygon induced by the hole. However, such a strategy also requires nodes' movements to be synchronized. Rout and Roy [42] apply a strategy that use obstacles and deployment boundaries as sources of repulsive forces exerting on nodes. This solution enhances coverage ratio and minimizes distance travelled but cannot prevent the formation of new holes. Zhao et al. [43] proposed a novel paradigm referred to as *fruit fly optimization*. The deployment zone is discretized with a virtual grid. All the uncovered areas exert on nodes (*fruit flies*), smells that are able to attract them to suitable areas. This strategy considers the presence of obstacles and quickly converges but can only be applied to homogeneous and dense networks. Ray and De [44] suggest instead, a *glowworm* based heuristic that minimizes the number of overlapped areas but fails to prevent new holes formation.

Solutions aiming at controlling specifically nodes' sensing ranges are more recent [45,46]. Qu and Georgakopoulos [47] used a Voronoi diagram based scheme that allows each node to adjust its range in order to entirely cover its Voronoi cell and check its redundancy. However, this solution is dedicated to only closed holes. Amgoth and Jana [8] proposed a virtual grid based scheme where after neighbor discovery, each node must identify cells that it can cover respectively with its current range and maximum range. These information help nodes to find cells that are not covered by any neighbor and eventually detect holes. In that case, a detection message is sent to alert these neighbors. If such a message is not acknowledged, node must increase its range in order to cover those empty cells. Although precise, this strategy is memory and energy expensive since large messages are required.

So far, only a few solutions have considered hybrid strategies to eliminate area coverage holes. Guvesan and Yavuz [19] proposed to combine motility and mobility based approaches. Regrettably, this protocol applies only to networks composed of nodes equipped with directional antennas. Abolhasan et al. [48] suggest a potential game theory based strategy that allows nodes to move or adjust their sensing ranges more efficiently. However, this solution is devoted to only closed holes and does not prevent the formation of new holes. Joshita et al. [49] proposed to vary nodes' sensing ranges along with a random mobility. This scheme helps to minimize the number of candidates and the number of overlapped areas, but also cannot prevent new holes to appear. Khedr et al. [50] suggest to only move redundant nodes and to add a pro-active scheme where nodes with low residual energy must alert neighbors in order to prevent hole formation. However, this strategy leads to collisions since some neighbors can move to the position of the alert's sender before its actual death.

3. PROPOSED SOLUTION

In this section we first discuss the motivations and objectives of this paper. Then we describe the key assumptions before detailing our solution.

3.1. Motivations and Objectives

Most existing solutions essentially aim to detect and eliminate only closed holes. It is useful to design a solution that can efficiently deal with the main three types of holes (closed, semi-open, and open) regardless of the failure scale.

Few hybrid hole elimination schemes, have been proposed in the literature. They often combine sensing range adjustment (motility) or redundancy control to mobility strategies in order to further reduce nodes' energy consumption. However, they fail to simultaneously face challenges such as overlapped areas, new holes formation, collisions, oscillations, and cascaded movements. This shortcoming need to be addressed especially in the presence of obstacles.

Furthermore, in most studies the performance evaluation process is commonly based on metrics which alone are not sufficient to actually capture the protocol's resilience ability. Hence, a more precise metric should be proposed to better estimate network's fault tolerance.

This paper is specifically aimed at minimizing:

- coverage hole detection and elimination time;
- travel distance, overlapped areas, number of candidates, and movements required to eliminate a hole regardless its type and location;
- risk of new holes formation especially at the network's boundaries.

3.2. Assumptions

We make the following assumptions:

- nodes respectively use the classical Unit Disk Graph and Boolean disk coverage models to communicate and sense events;
- each node u's communication and sensing ranges, respectively denoted by $r_c(u)$ and $r_s(u)$, are such that $r_c(u) \geq 2 \times r_s(u)$;
- each node u knows its position coordinates (x_u, y_u) in the deployment zone A using the underlying localization protocol;
- each node u can estimate the transmission delay $\tau(u, v)$ and the round trip time $rtt(u)$ with each neighbor v;

- two nodes u and v are neighbors (with two intersection points) if $d(u, v) < (r_s(u) + r_s(v))$ and $d(u, v) > (r_s(u) - r_s(v))$. $d(u, v)$ denotes the euclidean distance between u and v. $(r_s(u) + r_s(v)) - d(u, v) >= \lambda$; where λ is a parameter set by the underlying application.
- nodes are capable of adjusting their communication and sensing ranges (motility);
- nodes are able to control their mobility;
- the process is supposed to take place in a two-dimensional euclidian space.

3.3. Description

In this section, we detail our solution referred to as Coverage Hole Elimination Adaptive Protocol (CHEAP). This localized message passing protocol consists of three phases namely: hole detection, hole characterization, and hole elimination (coverage recovery).

We need first to give some definitions that can help to gain a better understanding of our strategy.

Definition 1 (Coverage of a node) *Let i be a point of the deployment zone denoted by A, the coverage of node u denoted by C(u) is such as* $C(u) = \{i \in A: d(u, i) \le r_s(u)\}$; *where* $d(u, i)$ *is the euclidean distance between i and u while* $r_s(u)$ *denotes the sensing range of node u.*

Definition 2 (Perimeter of a node) *Let i be a point in C(u) the coverage of node u while the perimeter of u denoted by P(u) such as* $P(u) = \{i \in C(u): d(u, i) = r_s(u)\}$.

Definition 3 (Parents of an intersection point) *the parents (father and mother) of a point are the nodes of which sensing disks intersection has created this point. Formally, let i be a point of the deployment zone denoted by A. Let u and v two nodes,* $(u, v \in \chi^{(i)}) \Leftrightarrow (i \in P(u)) \wedge (i \in P(v))$. *Where* $\chi^{(i)}$, *P(u), and P(v) respectively denote the set of point i's parents, the perimeter of u, and the perimeter of v.*

Definition 4 (Uncovered arc of a node) *Let P(u) be the perimeter of node u and N(u) the set of its neighbors. A part of P(u) denoted by* $\widehat{a(u)}$ *is an uncovered arc of node u, if none of its neighbors covers this arc. Formally,* $\widehat{a(u)} = \{p \in P(u) : \nexists v \in N(u), r_s(v) \ge d(v, p)\}$.

In order to simplify notations and discussions, each uncovered arc $\widehat{a(u)}$ *will be reduced to* $\widehat{ij}$ *such as* $i, j \in \widehat{a(u)}$ *and* $\exists v, w \in N(u) : (C(u) \cap C(v) = \{i\}) \wedge (C(u) \cap C(w) = \{j\})$. *i and j will be referred to as the border points of* $\widehat{a(u)}$.

Definition 5 (Adjacent uncovered arcs) *Let i, j, k, and l be four points whose coordinates are respectively denoted by* (x_i, y_i),(x_j, y_j), (x_k, y_k) *and* (x_l, y_l). *The uncovered arcs* $\widehat{ij}$ *and* $\widehat{kl}$ *are adjacent if* $((x_j = x_k) \wedge (y_j = y_k)) \vee (x_i = x_l) \wedge (y_i = y_l))$; *in other words, if* $(j = k) \vee (i = l)$.

Definition 6 (Maneuver of a node) *A maneuver is an action (displacement or range change) that a node u can perform in order to be usefully involved in a hole elimination process. Formally, a maneuver m is a vector such as* $m = (x_o, y_o, x_d, y_d, r_o, r_d)$ *and* $\exists i \in H : (x_d - x_i)^2 + (y_i - y_d)^2 < r_d^2$ *where H is a coverage hole, i a point and* $(x_i; y_i)$ *its coordinates.*

$(x_o, y_o, x_d, y_d, r_o, r_d)$ *respectively denote the current position's abscissa, the current position's ordinate, the destination's abscissa, the destination's ordinate, the current range, and the destination's range of node u. Therefore:*

if $(x_o \ne x_d) \vee (y_o \ne y_d)$, *maneuver m implies node u's displacement (mobility);*

if $(r_o \ne r_d)$, *maneuver m implies node u's range change (motility).*

Definition 7 (Elbow room of a node) *node u's elbow room denoted by* $\Phi(u)$ *is the set of its maneuvers. Formally, let* x_u *and* y_u *respectively denote the abscissa and the ordinate of node u. Let I be a set of maneuvers, and F(u) a family such as* $F(u) = \{\eta(u): \eta(u) \subseteq N(u)\}$

$$\Phi(u) = \{(m, \eta(u)) \in I \times F(u) : (x_o = x_u) \wedge (y_o = y_u)\}.$$

$\eta(u)$ *is referred to as node u's immobilized neighborhood.*

Definition 8 (Redundant node) *A node u is redundant if each point within its range is also covered by at least one of its immobilized neighbors. Formally, u is redundant iff* $\forall i \in C(u), \exists v \in \eta(u) : d(v, i) \le r_s(v)$.

Definition 9 (Maximum Redundancy Zone) *The Maximum Redundancy Zone (MRZ) of node u is the region delimited by the convex hull deriving from the cloud of the intersection points between neighbors that belong to node u's immobilized neighborhood. Points located on this hull will be referred to as the Border Points; whereas the others will be referred to as the Interior Points. Let* $\mathrm{IP}(u)$, $\mathrm{IN}(u)$ *and* $\mathrm{MRZ}(u)$ *respectively denote the set of intersection points between node u and its neighbors, the set of intersection points between node u's neighbors, and the set of border points on the convex hull of node u's MRZ; formally,*

$$IP(u) = \{i \in P(u) : \exists v \in \eta(u), i \in P(v)\}$$

$$IN(u) = \{i \in A : \forall v, w \in \eta(u), [(i \in P(v)) \wedge (i \in P(w))]\}$$

$$MRZ(u) = \{i \in (IP(u) \cup IN(u)) : (\forall v \in \eta(u), v \notin \chi^{(i)}) \Leftrightarrow (i \notin C(v))\}$$

Definition 10 (Incompatible maneuvers) *Two maneuvers* $m = (x_o, y_o, x_d, y_d, r_o, r_d)$ *and* $\tilde{m} = (\tilde{x}_o, \tilde{y}_o, \tilde{x}_d, \tilde{y}_d, \tilde{r}_o, \tilde{r}_d)$ *are incompatible in the following cases:*

- $(x_o = \tilde{x}_o) \wedge (y_o = \tilde{y}_o)$ *i.e. maneuvers m and* $\tilde{m}$ *imply an ubiquitous displacement; in other words, they involve the same node;*
- $(x_d = \tilde{x}_d) \wedge (y_d = \tilde{y}_d)$ *i.e. maneuvers m and* $\tilde{m}$ *imply a collision;*
- *maneuvers m and* $\tilde{m}$ *respectively involve two nodes u and v such as* $(x_o \ne \tilde{x}_o) \wedge (y_o \ne \tilde{y}_o)$, *u is redundant and* $v \in \eta(u)$ *or v is redundant and* $u \in \eta(v)$;
- *maneuvers m and* $\tilde{m}$ *respectively involve two nodes u and v such as* $(x_o \ne \tilde{x}_o) \wedge (y_o \ne y_o)$, *u and v are not redundant and* $(v \in N(u)) \wedge (u \in N(v))$.

Definition 11 (State of a node) *Node u can enter into the following states:*

- *Sleep (**SLP**),*

$$(state(u) = SLP) \Leftrightarrow [(r_s(u) = 0) \wedge (\widehat{N}(u) \ne \varnothing)]$$

- *Active (**ACT**),*

$$(state(u) = ACT) \Leftrightarrow (r_s(u) > 0)$$

- *Tentative (**TNT**),*

$$(state(u) = TNT) \Leftrightarrow [(r_s(u) > 0) \wedge ((IP(u) \cap MRZ(u)) \ne \varnothing)]$$

- *Alert (**ALRT**),*

$$(state(u) = ALRT) \Leftrightarrow [(r_s(u) > 0) \wedge (\widehat{kl} \neq \varnothing)]$$

- *Initiator (**INIT**),*

$$(state(u) = INIT) \Leftrightarrow [(r_s(u) > 0) \wedge (init(u) = id_u)]$$

- *Coordinator (**CRD**),*

$$(state(u) = CRD) \Leftrightarrow [(r_s(u) > 0) \wedge (crd(u) = id_u)]$$

- *Candidate (**CAND**),*

$$(state(u) = CAND) \Leftrightarrow [(r_s(u) > 0) \wedge (\Phi(u) \neq \varnothing)]$$

- *Move (**MOV**),*

$$(state(u) = MOV) \Leftrightarrow [(r_s(u) > 0) \wedge (dest(u) \neq \varnothing)]$$

$\widehat{kl}$, id_u, $init(u)$, $crd(u)$, $dest(u)$ respectively denote the adjacent uncovered arc, node u's identifier, the process initiator's identifier, the process coordinator's identifier, the next destination to move to.

3.3.1. Hole detection phase

Each active node must periodically (re)discover its neighbors and check for the presence of a potential coverage hole, following this process:

- step 1: get all Type 1 intersection points (i.e. intersection points with each neighbor);
- step 2: get Type 2 intersection points (i.e. intersection points between neighbors);
- step 3: create a cloud from the two types of intersection points;
- step 4: construct a convex hull from the point cloud created;
- step 5: get from that hull the set of border points (i.e. points that are covered only by their parents); the latter set is referred to as the convex hull of the MRZ;
- step 6: check if the MRZ's convex hull contains Type 1 intersections points; if so, a coverage hole exists; stop.

Let us see two examples. In Figure 1a node u constructs a convex hull (in red) from the cloud of intersection points, i.e. the intersection points with each neighbor (in white) and the ones between neighbors (in gray). Node u tries to derive from this first convex hull the one of the MRZ. To do this, node u focuses on points that are covered only by their parents, namely points bp_1, bp_2, bp_3, and bp_4, respectively covered by parents $\{v3, v4\}$, $\{v2, v4\}$, $\{v1, v2\}$, and $\{v2, v3\}$. There is no white point among them, therefore, there is no coverage hole. Note that, in this example the first convex hull and the one of the MRZ coincide; this is not always the case. Figure 1b shows another example where node u constructs a convex hull from a cloud of intersection points (white and gray). Then node u derives its MRZ's convex hull composed of bp_1, bp_2, bp_3, and bp_4. However, two of these points (bp_1 and bp_2) are white. Node u concludes that arc $\widehat{bp_1bp_2}$ is uncovered. In other words, node u has found a new coverage hole in its neighborhood.

Note that, each node must store the parents' identifiers (f_i, m_i) of all the intersection points i located on the convex hull of its MRZ. Then derive from these information a vector $\{(x_i; y_i; f_i; m_i), (x_j; y_j; f_j; m_j), \tau_{ij}\}$ for each uncovered arc $\widehat{ij}$ found; where τ_{ij} denotes the number of times an alert message (i.e. HOLE message) about $\widehat{ij}$ has been broadcasted so far.

When a non-isolated node u notices the presence of an uncovered arc, it waits for an amount of time $t_{tent}(u)$ randomly and uniformly chosen in a interval defined as a parameter. Upon timer $t_{tent}(u)$ expiration, node u broadcasts a HOLE message in its two-hop neighborhood; if it has not already received such a message; then, node u starts a backoff timer $t_{init}(u)$ and waits for a reply. $t_{init}(u)$ is initiated using its maximum round-trip time and updated using Equation (1) upon receiving a ARC message from any node v. Node u will be considered by its nodes around the hole as the *initiator* of the recovery process to come.

$$t_{init}^{(t+1)}(u) = t_{init}^{(t)}(u) + \sum_{i=1}^{|\pi(u,v)|} \tau_{(i,i+1)} + rtt(v) \quad (1)$$

$\tau_{(i,i+1)}$ denotes the transmission delay between two consecutive nodes on the path between nodes u and v. $rtt(v)$ is the maximum round trip time between node v and its neighbors. $\pi(u, v)$ is the set of intermediary nodes on the path between u and v.

$$t_{alert}(v) = \sum_{w \in \pi(u,v)} rtt(w) \quad (2)$$

Upon receiving a HOLE message, a node v becomes active (if needed) and enters into *Alert* state for a duration denoted by $t_{alert}(v)$. The latter is calculated using Equation (2). Then, if v is a parent of at least one of the border points contained in the HOLE message it must search its perimeter for any uncovered arc adjacent to those sent by node u. Neighbor v must substitute the adjacent arc contained in the HOLE message by the arc it has found before forwarding this message, in turn, to its neighbors. Finally, node v must

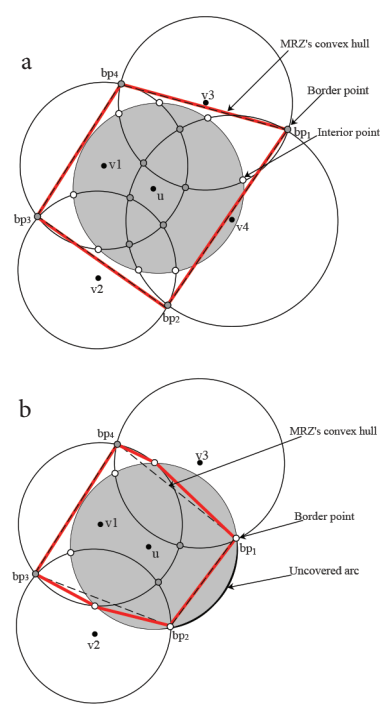

Figure 1 Construction of the Maximum Redundancy Zone (MRZ)'s convex hull (dotted) in order to detect a coverage hole by node u: (a) no coverage hole, no intersection point with neighbors found on this hull (b) coverage hole, two intersection points with neighbors found on this hull.

send to *initiator u* a ARC message via the neighbor who forwarded *u*'s HOLE message. This message contains information about the adjacent uncovered arc node *v* has just found on its perimeter.

Priority is given to HOLE messages which have the lowest τ_{ij}. The length of each arc $\widehat{ij}$ denoted by $\mathcal{L}(\widehat{ij})$ is used to break ties.

3.3.2. Hole characterization phase

From information contained in ARC messages, the *initiator* builds a graph describing the relationships existing between all the uncovered arcs found by its neighbors (see Definition 5).

Upon timer $t_{\text{init}}(u)$ expiration, *initiator u* that has received at least one ARC message must start the characterization process of the newly discovered hole (*closed* or *open*) by checking if the uncovered arcs graph is respectively cyclic or not. Several schemes exist for that purpose in the literature. However, we propose a method based on Theorem 1.

Theorem 1. *Let Ga = (Va, Ea) be a graph that specifically results from a hole detection process; where Va and Ea respectively denote the set of uncovered arcs* $\widehat{ij}$ *and the set of the symmetric links existing between them. Formally,* $Ea = \{(\widehat{ij}; \widehat{kl}) \in Va \times Va : (j = k) \vee (i = l)\}$. *Ga is cyclic if* $|Va| = |Ea|$.

Proof. Any node *w* sending a ARC message has inevitably received at least one HOLE message via a neighbor *u*. Therefore, *w* could have necessarily verified that one of its uncovered arcs denoted by $\widehat{kl}$ is adjacent to the one contained in the HOLE message and denoted by $\widehat{ij}$. Since any HOLE message is sent by only one node *u* referred to as the *initiator*, there is a relationship between an uncovered arc denoted by $\widehat{ab}$ found by the *initiator* and $\widehat{kl}$. In other words, $((\widehat{ab}; \widehat{cd}), \ldots, (\widehat{ij}; \widehat{kl}))$ is a simple path in graph *Ga*. Moreover, this path is unique in *Ga* since only nodes that have received a HOLE message and have found at least one adjacent uncovered arc, have sent a ARC message as a reply. By definition, a cycle is a simple path wherein the number of nodes equals the number of links. $|Va|$ is the length of path $((\widehat{ab}; \widehat{cd}), \ldots, (\widehat{ij}; \widehat{kl}))$; therefore $(|Va| = |Ea|) \Leftrightarrow G_a$ is cyclic.

Corollary 1 *If Ga = (Va, Ea) is not cyclic and if* $|Va| \geq 3$ *then* $\exists \widehat{ij}, \widehat{uv} \in Va : (\widehat{ij}; \widehat{uv}) \notin Ea$ *and we have* $(d(i, v) \neq 0) \wedge d(j, u) \neq 0)$.

Corollary 2 *Topology induced by Ga = (Va, Ea) is the hole's concave border; if* $\widehat{ij} \in Va$, *points i and j are located on its convex hull.*

From Theorem 1 and Corollaries 1 and 2 we give a formal definition of a coverage hole.

Definition 12 (Coverage hole) *Let H be a set of points in the area of interest A and Ga = (Va; Ea) a graph of uncovered arcs discovered in its neighborhood; if H ⊂ Va then H is a hole. Moreover,*

$$H \text{ is}: \begin{cases} \text{Closed} & \text{if } ((|Va| \geq 3) \wedge (|Va| = |Ea|) \\ \text{Semi-open} & \text{if } ((|Va| \geq 3) \wedge (|Va| \neq |Ea|) \\ \wedge (\exists \widehat{ij}, \widehat{uv} \in Va : \Omega > dth)) & \\ \text{Open} & \text{otherwise} \end{cases}$$

with $\Omega = \max(d(j; u), d(i; v))$.

d(j; u) and d(i; v) respectively denote the euclidean distances between point j and point u then between point i and point v. Ω is referred to as the "closure distance" and dth denotes a threshold.

Therefore, in order to characterize the detected hole the *initiator* must just ensure that conditions specified in Definition 12 are met. Furthermore, the *initiator* must identify all the boundary nodes and choose among them the one who can best lead and supervise the rest of this operation. The latter node will be referred to as the *coordinator*. Indeed, in order to be a good *coordinator*, a node must be as close as possible to the centroid of the newly detected hole. This issue is trivial for closed holes but crucial for energy efficiency, in the case of open or semi-open holes. Figure 2a–2c show some examples of graphs that an *initiator* could build during a hole characterization process.

The *coordinator* will be chosen by the *initiator* using the *graph of parents* denoted by *Gp* = (*Vp*, *Ep*) built from the border points (see Definitions 3 and 4) contained in ARC messages.

The *graph of parents* induces a ring (cycle), a tree or a chain (path) when the coverage hole is respectively closed or open. *Coordinator* election process complies with the following rules:

- if this graph is cyclic (see Theorem 1) the *initiator* becomes the *coordinator*;
- otherwise, the *initiator* is the root of a tree containing two sets of nodes or *branches* denoted by b_1 and b_2 such as $|b_2| \geq |b_1|$ with $|b_2| > 0$ and $|b_1| \geq 0$; then the node located $\hat{h}$ hops from the *initiator* on branch b_2 is chosen as the *coordinator*. $\hat{h}$ is calculated using Equation (3).

$$\hat{h} = \lfloor \frac{|b_1| + |b_2| - 1}{2} \rfloor - (|b_1| - 1) \tag{3}$$

After choosing another node as a *coordinator*, the *initiator* must send a COORD message containing information needed to conduct the coverage recovery process such as the uncovered arcs, the type of topology induced by the detected coverage hole, and

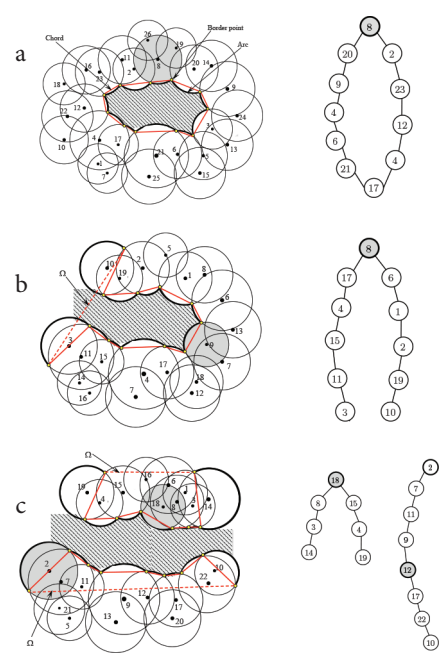

Figure 2 Different types of holes and their corresponding *graphs of parents*: (a)–(c) on left gray node is the *initiator*, on right gray node is the *coordinator*.

information that help to update the timer t_{alert} of all the nodes that have entered into *Alert* state.

The coverage recovery process is started by the *coordinator* just after its election upon receiving a COORD message or directly, if it was the former *initiator*.

3.3.3. Hole elimination phase

The entire coverage recovery process is supervised by the *coordinator*. This process consists of three phases, namely: *relocation sites definition*, *candidates selection*, and *candidates migration*.

- ***Relocation sites definition***

This phase starts with the location of hole H's centroid denoted by G. To do this, *coordinator* must find the set of border points starting by the ones located on its own perimeter. These points will be referred to as p_i in the remainder of this description; i denotes the index granted by the *coordinator*. The latter gets each border point p_i's coordinates respectively denoted by x_i and y_i from information contained in the uncovered arcs.

When the hole is closed, point G corresponds to the barycenter of the irregular polygon induced by its convex hull. G's coordinates respectively denoted by x_G and y_G, are calculated using Equations (4)–(6).

$$x_G = \frac{1}{6\times\mathcal{A}(H)}\sum_{i=1}^{n}(x_i + x_{i+1})(x_i y_{i+1} - x_{i+1} y_i) \tag{4}$$

$$y_G = \frac{1}{6\times\mathcal{A}(H)}\sum_{i=1}^{n}(y_i + y_{i+1})(x_i y_{i+1} - x_{i+1} y_i) \tag{5}$$

$$\mathcal{A}(H) = \frac{1}{2}\sum_{i=1}^{n}(x_i y_{i+1} - x_{i+1} y_i) \tag{6}$$

However, if hole H is open, G is located such as $d(u, G) = \Delta k \times \hat{r}_s$; where $d(u, G)$ denotes the distance between *coordinator* u and G on the direction vertical to the chord existing between border points of its uncovered arc (see Figure 3b). Δk (i.e. the range offset) is defined as a parameter and $\hat{r}_s$ denotes nodes' maximum range.

After locating the centroid G, coordinator u must discretize hole H linking each border point p_i to G. Figure 3a and 3b depict results respectively obtained with a closed hole and an open one.

Each resulting triangle will be referred to as a *zone* in the remainder of this description. coordinator u must also determine inside hole H all the relocation sites. To do this, each segment $\overline{p_i G}$ is split (see Figure 4a and 4b) into $\lfloor d(p_i; G)/(2\hat{r}_s - \lambda)\rfloor$ parts (i.e. with $\lfloor d(p_i; G)/(2\hat{r}_s - \lambda)\rfloor - 1$ points) if $d(p_i; G) > (2\hat{r}_s - \lambda)$; or into two parts (by putting just one point) otherwise. $d(p_i; G)$ denotes the length of $\overline{p_i G}$ and λ is a constant (see Section 3.2).

After hole discretization process, coordinator u broadcasts in its two-hop neighborhood a SITE message containing information about the border points and those located inside hole H. Then, it starts a timer whose duration $t_{site}(u)$ is initiated using the maximum round trip time experienced among its neighbors. This timer is updated using Equation (7) upon receiving any response (CANDIDATE message) from a neighbor v.

$$t_{site}^{(t+1)}(u) = t_{site}^{(t)}(u) + \sum_{i=1}^{|\pi(u,v)|}\tau_{(i,i+1)} + rtt(v) \tag{7}$$

$\tau_{(i,i+1)}$ denotes the transmission delay between two consecutive nodes on the path between nodes u and v. $rtt(v)$ is the maximum round trip time between node v and its neighbors. $\pi(u, v)$ is the set of intermediary nodes on the path between u and v.

Upon receiving a SITE message, any node v in *Alert* state must define its elbow room $\Phi(v)$ (see Definition 7). Indeed, in coordinator u's two-hop neighborhood each node v must find with respect to each of its own neighbors, the farthest external position denoted by $\hat{\sigma}_v$ which helps to take part in the hole elimination process without compromising its coverage. For example, in Figure 4c, on receiving a SITE message from the coordinator, node 2 starts looking with its maximum range (first dotted circle) for all the possible actions to take in order to be involved in the hole elimination process. Thus, node 2 notices that it can reach the hole from its current position (i.e. its range is strictly greater than the distance to at least one border point). Then, since node 2 is before neighbor 1 (i.e. closer to hole's centroid G than node 1), node 2 will try to find with respect to neighbor 1 on the direction vertical to the chord ij, the farthest position it may move to, in order to reach the hole without compromising the coverage of the area of interest (second dotted circle). Formally, if w, v, $r_s(w)$, $\hat{r}_s(v)$, and $\hat{\sigma}_v$ respectively denote node 1's position, node 2's current position, node 1's maximum range, node 2's maximum range, and node 2's farthest position with respect to neighbor 1, then the coordinates of $\hat{\sigma}_v$ will be determined using Equations (8)–(12) λ is a parameter set by the underlying application as discussed in Section 3.2.

$$x_{\hat{\sigma}_v} = x_w + \frac{d(w,\hat{\sigma}_v)\times(x_v - x_w)}{d(w,v)} \tag{8}$$

$$y_{\hat{\sigma}_v} = y_w + \frac{d(w,\hat{\sigma}_v)\times(y_v - y_w)}{d(w,v)} \tag{9}$$

$$d(w, \hat{\sigma}_v) = (r_s(w) + \hat{r}_s(v)) - \lambda \tag{10}$$

$$d(v, \hat{\sigma}_v) = d(w, \hat{\sigma}_v) - d(w, v) \tag{11}$$

$$d(w, v) = \sqrt{(x_w - x_v)^2 + (y_w - y_v)^2} \tag{12}$$

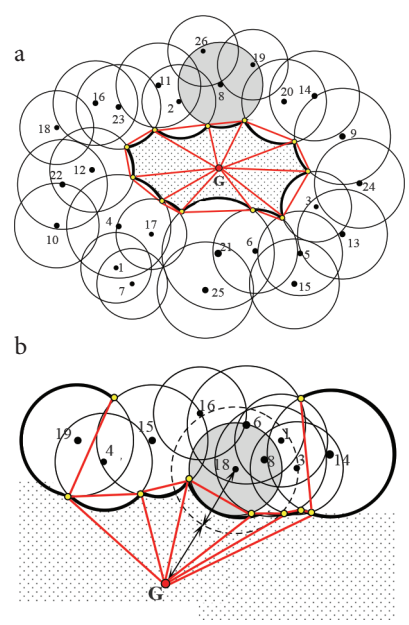

Figure 3 | Hole discretization by a coordinator (in gray): (a) hole is closed (b) hole is open.

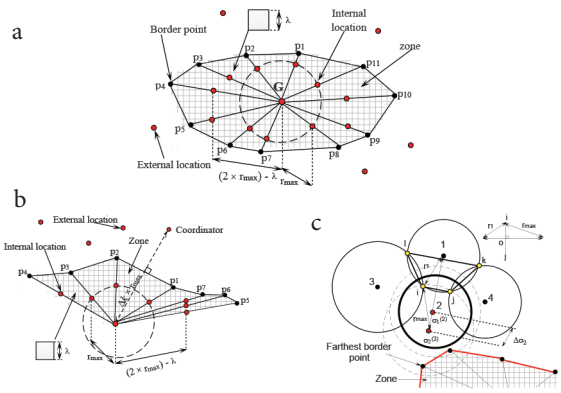

Figure 4 | Finding potential positions (in red): (a) by the coordinator for a closed hole (b) by the coordinator for an open hole (c) by a potential candidate (Virtual grid is only used to estimate the coverage ratio in each zone. See online version for colors).

The farthest position found with respect to neighbor 1 requires node 1, 3, and 4 to be immobilized (i.e. $\eta(\mu) = \{1, 3, 4\}$). Note that node 2 has also found that choosing its current position as a potential relocation site does not require any neighbor to be immobilized (i.e. $\eta(u) = \varnothing$).

Node 2 will try to do the same with each of its other neighbors located further away with respect to G. Therefore, Node 2's elbow room is the set of actions to perform in order to reach the hole without compromising its coverage. Redundant potential candidates generally have the largest rooms for maneuver. Indeed, such nodes can freely move toward several sites (external or not) without compromising deployment zone coverage. To do this, these nodes must select sites (especially the internal ones) by immobilizing their neighborhood; knowing that a node may be redundant with respect to different subsets of neighbors. For example, in Figure 5a node 1 (in gray) is redundant with respect to two subsets of neighbors namely {2, 3, 5, 6} and {2, 3, 4, 5, 6}; therefore, any displacement toward a site will require node 1 to immobilize only the smallest set of neighbors namely, {2, 3, 5, 6} (i.e. $\eta(\mu) = \{2, 3, 5, 6\}$). In Figure 5b node 1 and node 4 are respectively redundant in relation to {2, 3, 5, 6} or {2, 3, 4, 5, 6} and {5, 6, 7, 8, 9}. Therefore, the two subsets of neighbors to be immobilized during the displacements of node 1 and node 4, are respectively {2, 3, 5, 6} and {5, 6, 7, 8, 9}.

Note that like any potential candidate, redundant nodes must make sure that they have enough energy to move to any selected site and reach at least one zone from that position (see Definition 6).

Also note that any area coverage redundancy check protocol found in the literature can be used to detect redundant nodes [14]. Although, we suggest the MRZ-based one we proposed in our previous works [51,52].

If a node v has a sufficient elbow room (i.e. $\Phi(v) \neq \varnothing$) it becomes a formal candidate by sending CANDIDATE message to the *coordinator* and starts a timer for $t_{\text{cand}}(v)$ units of time. This message contains information about its elbow room, and state. $t_{\text{cand}}(v)$ is estimated using Equation (13) in terms of the round trip time on the path $\pi(u, v)$ between v and coordinator u. The latter information is contained in the SITE message sent by the *coordinator*.

$$t_{\text{cand}}(v) = \sum_{w \in \pi(u,v)} rtt(w) \tag{13}$$

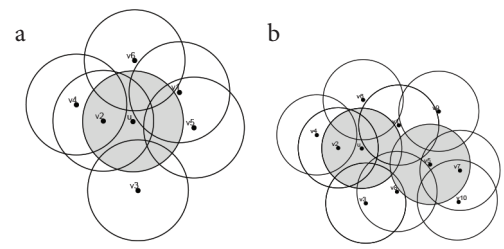

Figure 5 | Neighbors immobilization process by a redundant candidate (in gray): (a) with multiple redundancy neighborhood. (b) with a redundant neighbor.

From mobility point of view, there are three types of formal candidates (with respect to their states) which might be involved in the coverage recovery process namely: nodes that are unable to move, nodes with reduced mobility, and those with full mobility. These types of candidacy respectively correspond to intermediary states MO, MRM, and MM. Note that *coordinator* may be a candidate as well.

- ***Candidates selection***

This last phase allows *coordinator u* to ensure coverage recovery optimization. Indeed, after timer $t_{\text{site}}(u)$ expiration, *coordinator u* must choose the maneuvers which help to best eliminate the hole while minimizing overlapped areas and energy wastes.

We formulate this issue as a *location-allocation* problem [20,21]. The latter has several well-known variants in combinatorics and geometry (*facility location*, *p-median*, *p-center, set covering, Weber problem*...) [53,54].

Solutions commonly proposed in the literature have many applications in domains such as industry, geography, transportation, logistics, marketing etc. We formulate this problem using the following mixed integer linear program:

Let x_{ij} be the area exclusively covered in zone j by the candidate involved in maneuver i. Let

$$y_i = \begin{cases} 1, & \text{if maneuver } i \text{ is allowed} \\ 0, & \text{otherwise} \end{cases}$$

$$z_{ij} = \begin{cases} 1, & \text{if maneuvers } i \text{ and } j \text{ are incompatible} \\ 0, & \text{otherwise} \end{cases}$$

and let ϕ_{ij}, f_i and d_j respectively denote the overlapped area obtained in zone j when allowing maneuver i, the energy ratio dissipated after maneuver i and the total area to be covered (the demand) in zone j. Note that, I and J respectively denote the set of maneuvers to be allowed and the set of zones.

$$\min \sum_{j \in J} d_j - \sum_{i \in I} \sum_{j \in J} x_{ij} + \sum_{i \in I} \sum_{j \in J} \phi_{ij} + \sum_{i \in I} f_i y_i \tag{14}$$

s.t.:

$$\sum_{i \in I} x_{ij} = d_j, \ \forall j \in J \tag{15}$$

$$\sum_{j \in J} x_{ij} + \sum_{j \in J} \phi_{ij} > 0, \ \forall i \in I \tag{16}$$

$$f_i y_i \leq 1, \ \forall i \in I \tag{17}$$

$$y_i + y_j + z_{ij} \leq 2, \ \forall i \in I; \ \forall j \in I \tag{18}$$

$$\sum_{i \in I} y_i \geq 1 \tag{19}$$

$$x_{ij} \in \mathbb{R}_+, \forall i \in I; \forall j \in J \tag{20}$$

$$y_i \in \{0, 1\}, \forall i \in I \tag{21}$$

$$\phi_{ij} \in \mathbb{R}_+, \forall i \in I; \forall j \in J \tag{22}$$

$$z_{ij} \in \{0, 1\}, \forall i \in I; \forall j \in I \tag{23}$$

$$f_i \in [0, 1], \forall i \in I \tag{24}$$

$$d_j \in \mathbb{R}_+^*, \forall j \in J \tag{25}$$

Equation (14) states the objective namely, minimize the uncovered areas, the overlapped areas inside the hole and the total energy consumption. Equation (15) states that each zone j must be entirely covered. Equation (16) suggests that any maneuver i should reach at least one zone j. This constraint ensures that each candidate is active and is able to reach at least one zone. Equation (17) ensures that the residual energy of the candidate concerned by a maneuver is enough if allowed. Equation (18) prevents *coordinator* to allow incompatible maneuvers. This constraint prevents illogic maneuvers (ubiquity), new hole formation, collisions, and oscillations. Equation (19) ensures that any solution allows at least one maneuver. Equations (20)–(22) define the decision variables. Equations (23)–(25) define the constants.

Energy costs are calculated using Equation (26); where εdist(.) and εrange(.) are two functions respectively related to the underlying mobility and motility energy consumption models. Er(v) is the residual energy of the candidate v concerned by maneuver i.

$$f_i = \frac{\mathcal{E}\text{dist}(i) + \mathcal{E}\text{range}(i)}{Er(v)} \tag{26}$$

$$\sum_{i \in I} x_{ij} \leq dj, \forall j \in J \tag{27}$$

Note that in practice, it is difficult to comply with the constraint suggested by Equation (15) since nodes are randomly deployed and their density constantly decreases. Therefore, *coordinator* can relax Equation (15) by Equation (27) if $n < \psi^*$; where n denotes the numbers of available redundant candidates whereas ψ^* denotes the maximum number of candidates required to totally cover hole H; its value is obtained using Equation (28) inspired from Tóth [55]; where $\mathcal{A}(\cdot)$ and $\mathcal{P}(\cdot)$ denote two functions helping to respectively assess area and perimeter of hole H; $\hat{r}_s$ denotes nodes' maximum range.

$$\psi^* = \left(\frac{2}{3\sqrt{3}} \times \frac{\mathcal{A}(H)}{\hat{r}_s^2}\right) + \left(\frac{2}{\pi\sqrt{3}} \times \frac{\mathcal{P}(H)}{\hat{r}_s}\right) + 1 \tag{28}$$

Any method of the literature can be used to estimate the area of hole H. As suggested by Figure 4a and 4b, an intuitive method is to add up the calculated areas (e.g. using Heron's formula) of the zones (triangles) created during the hole discretization process. For closed holes, $\mathcal{A}(H)$ may also be calculated using Equation (6) while $\mathcal{P}(H)$ is trivially obtained by adding the length of chords existing between border points p_i.

According to a method commonly used in the literature d_j, x_{ij} and ϕ_{ij} can be estimated by discretizing each zone j with a virtual grid of patterns such as points, squares etc. [56–58]. Figure 6 illustrates how to estimate these values. Let j (in red) be a zone of a detected hole discretized with a grid of 24 cells (d_j = 24). In this example, we consider that a cell is entirely covered by a node if its four corners are within the node's range (i.e located at a distance inferior or equals to the circle's radius). r, s, t, u, v, w denote six nodes that can reach zone j. Node r exclusively covers six cells (namely cells 3, 4, 5, 9, 10, and 11); node s covers cells 19 and 20; node t cells 15, 21, and 22; node u cells 1, 7, 8, and 13; node v cell 2. Instead, node w does not exclusively cover any cell. Therefore, $x_{rj} = 6$, $x_{sj} = 2$, $x_{tj} = 3$, $x_{uj} = 4$, $x_{vj} = 1$, $x_{wj} = 0$. Nodes r, s, u yield no overlapped areas; instead, node t yields overlapped area in cell 20, node v in cells 1, 7, and 8; node w in cells 13, 19 and 20. Therefore, $\phi_{rj} = 0$, $\phi_{sj} = 0$, $\phi_{tj} = 1$, $\phi_{uj} = 0$, $\phi_{vj} = 3$, $\phi_{wj} = 3$.

Note that, in this example we have discretized zone j with large squares only to simplify our explanations. One could also have used smaller grid cells or even points [57] to increase precision and minimize estimation errors like those yielded by coverage in cells 14 and 16.

Coordinator defines incompatibilities z_{ij} between two maneuvers i and j with respect to their origins (ubiquity), destinations (collision) or immobilizations contained in candidates' elbow rooms (see Definitions 7 and 10)

Regrettably, location-allocation problem has been proven NP-hard [59–61]. Therefore, we propose an approximate solution based on tabu search metaheuristic [62].

A *local reorganization scheme s* (i.e. a simplified version of a feasible solution) will denote a set of n maneuvers, each related to a unique candidate. Formally, $s = \{i \in \text{I}: y_i = 1\}$ with $|s| = n$. To find an initial solution, *coordinator* first looks for the best (i.e. less energy consuming) maneuver toward the centroid; then, if found, allows it and store the concerned candidate's identifier.

For each candidate v (except the one eventually sent to the centroid), *coordinator* will look for all the maneuvers of v that are compatible with the ones already allowed. If v is redundant then *coordinator* chooses the most energy consuming maneuver of v; otherwise, the less energy consuming one. The set of all maneuvers thus allowed will be referred to as the *initial local reorganization scheme*.

During the optimal *local reorganization scheme* search process, neighbor solutions will be obtained by shifting two maneuvers.

During the decision making process, *coordinator* will use candidates' maximum range denoted by $\hat{r}_s$. Whereas in a heterogeneous

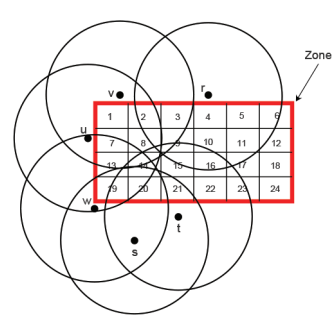

Figure 6 | Exclusive and overlapped area coverage estimations in a virtual grid-based discretized zone.

network, $\hat{r}_s$ will denote the smallest value chosen among nodes' maximum range.

A *local reorganization scheme* $s = \{i_1, i_2, \ldots, i_n\}$ corresponds to decisions made by *coordinator* for each of the n candidates taking account of maneuvers' incompatibilities. For instance, applying a move on i_1 consists in finding for the concerned candidate a new maneuver i_1' which is compatible like i_1, to other maneuvers $\{i_2, i_3, \ldots, i_n\}$ in s. If i_1' exists and if set $\{i_1', i_2, \ldots, i_n\}$ is not inside the tabu list LT then $\{i_1', i_2, \ldots, i_n\}$ is considered as a neighbor solution of s. Another neighbor solution will involve maneuver i_2 and so on. $N(s)$ denotes the set of the neighbor solutions of s. Formally, $N(s) = \{\{i_1', i_2 \ldots i_n\}, \{i_1, i_2' \ldots i_n\} \ldots \{i_1, i_2 \ldots i_n'\}\}$. Note that, the tabu list's size is limited in order to store the last $\rho^* |N(s)|$ solutions; $\rho > 0$ is a parameter.

- ***Candidates migration***

After finding an optimal *local reorganization scheme*, *coordinator u* broadcasts a MOVE message containing its decisions. Upon receiving such a message, candidates start moving to their new relocation sites. MOVE messages are forwarded via paths used by SITE messages.

Nodes will optimize their ranges using any scheme of the literature [63–65] upon reaching destination and timer $t_{move}(u)$ expiration. Redundant nodes will eventually enter into sleep state.

Figure 7 depicts the rationale behind CHEAP. Figure 8 shows the state diagram of the discussed coverage hole detection and elimination schemes.

Table 1 summarizes the variables used in the different algorithms.

Algorithms 1–4 formally describe coverage hole detection, characterization, and elimination processes. Algorithms 5 and 6 specifically describe the optimal *local reorganization scheme* search process.

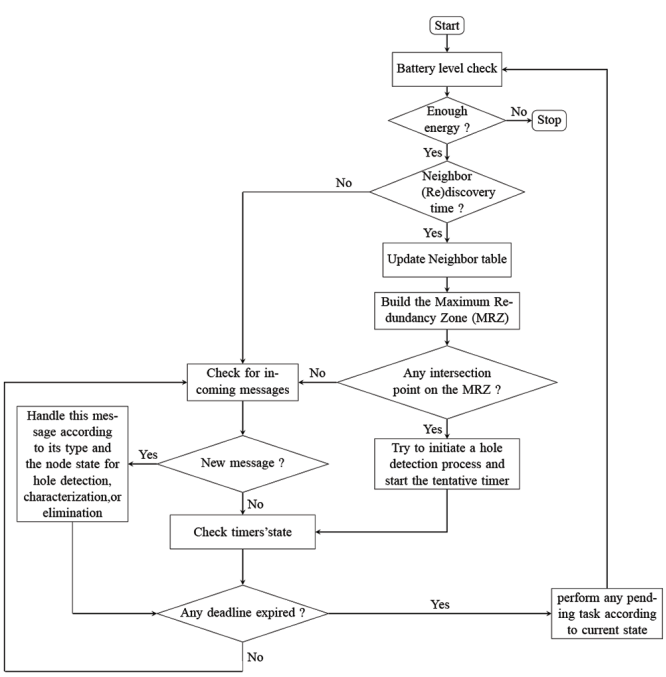

Figure 7 | Flowchart of CHEAP.

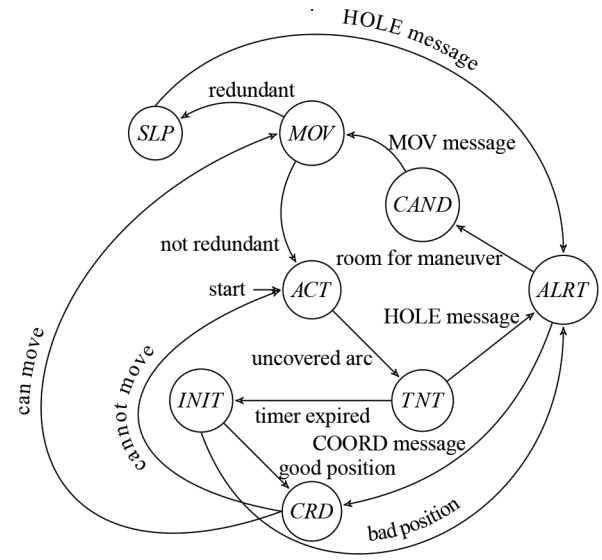

Figure 8 | State diagram of CHEAP.

4. PERFORMANCE EVALUATION

In order to verify and validate our scheme, we chose OMNeT++ 5.5 simulator [66] to evaluate our solution with respect to different metrics. Results are compared to those obtained with three major related protocols proposed in the literature: DECM by Qiu and Shen [40], HORA by Sahoo and Liao [39], and ZBFR by Sharma and Sharma [18].

We used the communication and the sensing energy consumption models respectively proposed by Heinzelman et al. [67] and by Halgamuge et al. [68]. We also used a mobility energy consumption model [see Equations (29)–(33)] inspired from the method proposed by Society of Robots [69].

$$Em = (2 \times Ec) + Ep + \overline{E} + \overline{C} \tag{29}$$

$$Ec = \frac{1}{2} \times m \times v^2 \tag{30}$$

$$Ep = m \times g \times h \tag{31}$$

$$\overline{E} = \hat{\delta} \times d \times Ec \tag{32}$$

$$\overline{C} = Er \times (1 - ea \times eb \times ec \times ed) \tag{33}$$

where Ec denotes the kinetic energy, Ep is the potential energy, $\overline{E}$ represents energy lost due to frictions, $\overline{C}$ corresponds to the amount of energy drained due to conversions. m is the mass of the node, v the velocity, g denotes the gravity, h is the travel height, $\hat{\delta}$ the deceleration rate (number of re-accelerations per unit of distance), d is the distance travelled, Er denotes node's residual energy. ea, eb, ec, ed respectively denote chemical, mechanical, electrical and thermal conversions efficiency.

Since this process is supposed to take place in the plan (see Section 3.2), potential energy is null $((Ep = 0) \Leftrightarrow (h = 0))$.

For each experiment, we deployed several networks; each of them is composed of randomly and uniformly distributed nodes. To randomly vary the average times between two consecutive faults, we used the Weibull distribution combined with the Uniform

Table 1 | Main global variables

Name	Definition
$crd(u)$	coordinator of the hole elimination process involving node u
D	set of relocation sites inside a hole
$deadline$	current deadline of a timer
$dest(u)$	destination node u is allowed to move to
Δk	range offset for open hole's centroid
$Ea(u)$	set of edges between arcs' parents discovered by node u
$Er(u)$	node u's residual energy
Eth	residual energy threshold
G	centroid of a hole
$\Phi(u)$	set of maneuvers that node u can perform (its *elbow room*)
I	set of candidates' maneuvers
id_u	node u's identifier
$init(u)$	initiator of the hole detection process involving node u
$IP(u)$	intersection points between node u and its neighbors
$IN(u)$	intersection points between node u's neighbors
$itmax$	maximum number of iterations
J	set of sub-areas inside a hole (zone)
K	set of nodes' sensing ranges
λ	length of a virtual grid cell
ls	size of the tabu list
$MRZ(u)$	set of points on the convex's hull of node u's MRZ (see Definition 9)
$N(u)$	node u's neighbor table
$NH(u)$	set of candidates for the coverage recovery process led by node u
$\eta(u)$	node u's immobilized neighborhood (see Definition 7)
P	set of points on hole's border
$r_s(u)$	current sensing range of node u
$\hat{r}_s$	maximum sensing range
$rtt(v)$	round trip time with neighbor v
s^*	optimal reorganization scheme
$state(u)$	current state of node u (see Definition 11 for possible values)
$Sr(u)$	set of uncovered arcs so far reported by node u
$t_{alert}(u)$	duration of node u's on alert state
$t_{cand}(u)$	duration of node u's candidate state
t_{discov}	duration between two neighbor (re)discovery periods
$t_{init}(u)$	duration of node u's initiator state
t_{max}	maximum duration of tentative state
$t_{mov}(u)$	duration of node u's on move state
$t_{site}(u)$	duration of node u's coordinator state
$t_{sleep}(u)$	duration of node u's sleep state
$t_{tent}(u)$	duration of node u's tentative state
Υ^{ij}	set of parents of points i and j (see Definition 3)
$Va(u)$	set of arcs' parents
$\chi^{(i)}$	parents of point i (see Definition 3)
x_G	abscissa of the hole's centroid
y_G	ordinate of the hole's centroid

distribution as described in Table 2. Faults were injected using these factors under an unfair distributed daemon.

Fault injection consisted in randomly constructing an irregular polygon so that only intersecting nodes experience a fault (displacement or failure). Note that battery exhaustion is only due to energy losses and self-discharge.

In the following sections, we first detail all the experiments we have conducted. Then we analyze and explain their results. In the course of these experiments we evaluated the impact of the factors described in Table 2 on the selected metrics. Note that each experiment was repeated 100 times. The results were obtained with a 95% confidence interval. The simulation parameters are summarized in Table 3.

Algorithm 1 | Hole detection process by a node u

Require: $Er(u)$, Eth, t_{discov}, t_{max}, K, ls, $deadline$, Δk... ▷ see Table 1
$Er(u) \leftarrow$ get_residual_energy() ▷ check battery level
while ($Er(u)>Eth$) **do** ▷ is residual energy enough ?
  if ((current time() = $delay\_DISC$)) **then**
    if ($state(u) = ACT$) **then**
      $N(u) \leftarrow$ neighbor_discovery()
      $IP(u) \leftarrow$ get_points_with_neighbors ($N(u)$)
      $IN(u) \leftarrow$ get_points_among_neighbors ($N(u)$)
      $MRZ(u) \leftarrow$ get_MRZ_hull ($IN(u)$, $IP(u)$, $N(u)$)
      if (($IP(u) \cap MRZ(u)) \neq \emptyset$) **then** ▷ found uncovered arcs
        $state(u) \leftarrow TNT$
        $t_{tent}(u) \leftarrow$ random_uniform ([0; t_{max}])
        $deadline \leftarrow$ current_time() + $t_{tent}(u)$
      end if
    end if
    $delay\_DISC \leftarrow$ current_time() + t_{discov}
  end if
  receive *message* **from** v ▷ new message from a neighbor v
  switch *message* **do**
    case HOLE
      $\widehat{ij} \leftarrow$ get_uncovered_arc (*message*)
      $\Upsilon^{ij} \leftarrow$ get_parents ($\widehat{ij}$)
      if ($state(u) \in \{ACT, TNT, SLP\}) \wedge (id_u \in \Upsilon^{ij})$ **then**
        $IP(u) \leftarrow$ get_points_with_neighbors ($N(u)$)
        $IN(u) \leftarrow$ get_points–among–neighbors ($N(u)$)
        $MRZ(u) \leftarrow$ get_MRZ_hull ($IN(u)$, $IP(u)$, $N(u)$)
        $\widehat{kl} \leftarrow$ argmax (are_adjacent ($\widehat{ij}$, $\widehat{xy}$))
          $\widehat{xy} \in (IP(u) \cap MRZ(u))$
        $r_s(u) \leftarrow \hat{r}_s$
        $state(u) \leftarrow ALRT$
        $init(u) \leftarrow message.init$ ▷ initiator's id
        if ($\widehat{kl} \neq \emptyset$) **then**
          $t_{alert}(u) \leftarrow$ get_path_delay (*message*)
          $deadline \leftarrow$ current_time() + $t_{alert}(u)$
          send HOLE($init(u)$, $\widehat{kl}$) **to** w, $\forall w \in N(u)\backslash\{v\}$
          send ARC({$\widehat{ij}$; $\widehat{kl}$}, $init(u)$) **to** v
        else
          forward_or_delete (*message*, $N(u)$, v, ttl_{max})
        end if
      else
        forward_or_delete (*message*, $N(u)$, v, ttl_{max})
      end if
    case ARC
      if (($state(u) = INIT) \wedge (message.init = init(u))$) **then**
        {$\widehat{ij}$; $\widehat{kl}$} $\leftarrow$ get_pair_of_arcs (*message*)
        $t_{init}(u) \leftarrow$ get_path_delay (*message*)
        $deadline \leftarrow deadline + t_{init}(u)$
        $Va(u) \leftarrow (Va(u)\backslash\{\widehat{ij}\ ;\ \widehat{kl}\}) \cup \{\widehat{ij}\ ;\ \widehat{kl}\}$
        $Ea(u) \leftarrow Ea(u) \cup \{\widehat{ij}\ ;\ \widehat{kl}\}$
      else
        forward_or_delete (*message*, $N(u)$, v, ttl_{max})
      end if
    otherwise
      elimination (*message*, v, $IP(u)$, ...) ▷ see Algorithm 2
  end switch
  detection_timers ($IP(u)$, $IN(u)$, ...) ▷ see Algorithm 3
  elimination_timers ($IP(u)$, $IN(u)$, ...) ▷ see Algorithm 4
  $Er(u) \leftarrow$ get_residual_energy() ▷ check battery level
end while

Algorithm 2 | Hole elimination process by a node u

```
Require: message, v, IP(u), MRZ(u), deadline, ... ▷ see Table1
switch message do
    case COORD
        if (state(u) = ALRT) ∧ (message.coord = id_u)) then
            P ← find_border_points (Va(u), Ea(u))
            G ← get_centroid (P, Δk) ▷ Equations (4) and (5)
            J ← triangulation (P, G) ▷ Hole discretization
            create_virtual_grid (J)
            D ← set_internal_sites(P, r̂_s, G.x, G.y, λ)
            state(u) ← CRD
            send SITE(D, P) to w, ∀w ∈ N(u)\v
            t_site(u) ← max{rrt(v), ∀v ∈ N(u)}
            deadline ← current_time() + t_site(u)
            Φ(u) ← get_elbow_room (D, P)
            message.ttl ← ttl_max
        end if
    case SITE
        if ((state(u) = ALRT) ∧ (message.init = init(u))) then
            Φ(u) ← get_elbow_room (message.D, message.P)
            if (Φ(u) ≠ ∅) then
                send CANDIDATE (Φ(u), message.coord) to v
                state(u) ← CAND
                t_cand(u) ← get_path_delay (message)
                deadline ← current_time() + t_cand(u)
            end if
        end if
    case CANDIDATE
        if ((state(u) = CRD) ∧ (message.init = init(u))) then
            D ← D ∪ get_external_locations (message)
            I ← I ∪ get_travel_paths (message)
            NH(u) ← NH(u) ∪ {v}
            t_site(u) ← get_path_delay (message)
            deadline ← deadline + t_site(u)
            message.ttl ← ttl_max
        end if
    case MOVE
        if ((state(u) = CAND) ∧ (message.init = init(u))) then
            d ← get_new_location (message.s*)
            state(u) ← MOV
            t_mov(u) ← get_travel_time (d)
            travel to d
            deadline ← current_time() + t_mov(u)
        end if
end switch
forward_or_delete (message, N(u), v, ttl_max)
```

Algorithm 3 | Detection timers check for a node u

```
Require: IP(u), MRZ(u), Va(u), Ea(u), deadline, Δk ... ▷ see Table 1
if (current_time() = deadline) then
    switch state(u) do
        case TNT
            Cr ← get_arcs (IP(u)∩MRZ(u))
            for all xy⌒ ∈ (Cr ∩ Sr(u)) do
                xy⌒.τ ← xy⌒.τ + 1 ▷ increase arc's detection counter
            end for
            Nr ← argmin (xy⌒.τ ≥ 0)
                     xy⌒ ∈ (Cr\Sr(u))
            ij⌒ ← argmax (get_arc_length(xy⌒))
                     xy⌒ ∈Nr
            send HOLE (ij⌒, id_u) to v, ∀v ∈ N(u)
            state(u) ← INIT
            t_init(u) ← max{rrt(v), ∀v ∈ N(u)}
            deadline ← current_time() + t_init(u)
            Sr(u) ← Sr(u) ∪ { ij⌒ }
        case INIT
            if (Va(u) ≠ ∅) then
                Vp ← get_parents (Va(u))
                Ep ← link_between_parents (Ea(u))
                crd(u) ← find_coordinator (Vp, E p)
                if (crd(u) = id_u) then
                    P ← find_border_points (Va(u), Ea(u))
                    G ← get_centroid(P, Δk) ▷ Equations (4) and (5)
                    J ← triangulation(P, G)
                    create_virtual_grid(J)
                    D ← set_internal_sites(P, r̂_s , G.x, G.y, λ)
                    state(u) ← CRD
                    send SITE(D, P) to w , ∀w ∈ N(u)\v
                    t_site(u) ← max{rrt(v), ∀v ∈ N(u)}
                    deadline ← current_time() + t_site(u)
                    Φ(u) ← get_elbow_room(D, P)
                else
                    if ((Va(u) ≠ ∅) ∧ (Ea(u) ≠ ∅)) then
                        send COORD(Va(u), Ea(u)) to crd(u)
                        state(u) ← ALRT
                    end if
                end if
            else
                state(u) ← ACT
                init(u) ← 0
            end if
        case ALRT
            state(u) ← ACT
            init(u) ← 0
    end switch
end if
```

4.1. Average Hole Detection Time

During this experiment, when a fault was injected, simulation daemon recorded the occurrence time, hole localization, and information about boundary nodes. Hole detection time was determined by the simulation program after having received a signal from all the boundary nodes. The resultant detection times were averaged at the end of the simulation (i.e. after all sensors died). Note that detection time of an undetected hole was obtained by subtracting the fault injection time from the simulation duration.

Figure 9a and 9b suggest that for all the evaluated protocols, average hole detection time increases according to network size and fault scale. However, CHEAP provides the lowest detection times. These results are essentially due to the fact that, unlike other protocols, CHEAP does not define waiting times according to nodes' degrees. Indeed, to avoid endless edge flipping during the Delaunay triangulation, DECM sets a timer which depends on nodes' degrees; while the performance of HORA is due to the mobility invitation messages sent. ZBFR yields the worst times because of the heartbeat message used for large agreement regions as well as for failures involving the Zone Monitor.

Algorithm 4 | Elimination timers check for a node u

Require: $I, J, D, NH(u), itmax, ls, deadline, s^*$ .... ▷ see Table1
if (current_time() = *deadline*) **then**
  switch $state(u)$ **do**
    case *CRD*
      $s^* \leftarrow$ reorganization $(I, J, K, ..)$ ▷ see Algorithm 5
      if $(s^* \neq \emptyset)$ **then**
        send MOVE (s^*) **to** v, $\forall v \in NH(u)$
      end if
      if $(\Phi(u) \neq \emptyset)$ **then**
        $d \leftarrow$ get_new_location(s^*)
        $state(u) \leftarrow MOV$
        $t_{mov}(u) \leftarrow$ get_travel_time(d)
        travel to d
        $deadline \leftarrow$ current_time() $+ t_{mov}(u)$
      end if
    case *CAND*
      $state(u) \leftarrow ACT$
    case *MOV*
      $r_s(u) \leftarrow$ get_new_range(s^*)
      if (check_redundancy()) **then**
        $state(u) \leftarrow SLP$
        $t_{sleep}(u) \leftarrow$ get_sleep_time()
        $deadline \leftarrow$ current_time() $+ t_{sleep}(u)$
      else
        $state(u) \leftarrow ACT$ ▷ return to normal state
        optimize_range() ▷ see [63–65]
      end if
    case *SLP*
      $state(u) \leftarrow ACT$ ▷ return to normal state
  end switch
end if

Algorithm 5 | Tabu-based optimal reorganization scheme search

Require: $I, J, D, G, K, NH(u), itmax, ls, s^*$
$s \leftarrow$ get_initial_solution$(I, K, G, NH(u))$ ▷ see Algorithm 6
$s^* \leftarrow s$
$LT \leftarrow \emptyset$ ▷ tabu list initialization
$nitr \leftarrow 0$
while $(nitr < itmax)$ **do**
  $N(s) \leftarrow \{s' \mid \text{are\_neighbors}(s', s) \wedge (s' \notin LT)\} : |N(s)| \leq ls$
  $\delta \leftarrow 1$
  if $(N(s) \neq \emptyset)$ **then**
    $\hat{s} \leftarrow \underset{s' \in N(s)}{\operatorname{argmin}} f(s')$ ▷ best neighbor (Equations (14)–(27))
    if $(f(\hat{s}) < f(s^*))$ **then** ▷ Equations (14)–(27)
      $s^* \leftarrow \hat{s}$
    end if
    update_list $(LT, N(s))$
    $\delta \leftarrow |N(s)|$
  else
    $s \leftarrow \underset{s' \in LT}{\operatorname{argmin}} (f(s'))$ ▷ aspiration
    $LT \leftarrow LT \backslash s$
  end if
  $nitr \leftarrow nitr + \delta$
end while

4.2. Average Hole Elimination Time

In this experiment, hole elimination time was determined by the simulation program after having received a signal from all the

Algorithm 6 | Initial reorganization scheme generation by a coordinator u

Require: $I, K, G, NH(u)$
Ensure: s
$k \leftarrow \max(K)$ ▷ use maximum sensing range
$\Lambda \leftarrow \underset{i \in I}{\operatorname{argmax}}$ (check_destination(i, G))
$C^* \leftarrow \emptyset$
if $(\Lambda \neq \emptyset)$ **then**
  $i^* \leftarrow \underset{i \in \Lambda}{\operatorname{argmin}} (\varepsilon\, dist(i))$
  $\tilde{I} \leftarrow \tilde{I} \cup \{i^*\}$ ▷ update the authorized maneuvers list
  $s \leftarrow s \cup \{(i^*; k)\}$ ▷ update the solution
  $C^* \leftarrow$ get_candidate_id(i) ▷ get the node involved
end if
for all $v \in NH(u) \backslash C^*$ **do**
  $\Upsilon \leftarrow \underset{i \in I}{\operatorname{argmax}}$ (check_origin(i, v) + compatible $(i, \tilde{I})$)
  if $(\Upsilon \neq \emptyset)$ **then**
    if (is_movable(v)) **then** ▷ can v move ?
      $i^* \leftarrow \underset{i \in \Upsilon}{\operatorname{argmax}} (\varepsilon\, dist(i))$ ▷ assign the farthest maneuver
    else
      $i^* \leftarrow \underset{i \in \Upsilon}{\operatorname{argmax}} (\varepsilon\, dist(i))$ ▷ assign the nearest maneuver
    end if
    $\tilde{I} \leftarrow \tilde{I} \cup \{i^*\}$ ▷ update authorized maneuvers list
    $s \leftarrow s \cup \{(i^*; k)\}$ ▷ update the solution
  end if
end for
return s

Table 2 | Fault tolerance factors

Factors	Unit/Description	–	+
Fault scale*	Area of the hole	$\mathcal{U}(1;50)$	$\mathcal{U}(55;100)$
MTBF**	Average time between two faults	$\mathcal{W}(\alpha = \mathcal{U}(2;10);\ \beta = 3)$	$\mathcal{W}(\alpha = \mathcal{U}(100;500);\ \beta = 3)$
Fault effect***	Degree of severity	0	$\mathcal{U}(1;2)$

*Value to square; unit is m^2. **(Mean Time Between Failures) expressed in seconds (simulated time). ***nodes displacement = 0, damaged sensing unit = 1, node failure = 2. $\mathcal{U}$, Uniform distribution; $\mathcal{W}$, Weibull distribution.

candidates when reaching their final destinations. The resultant elimination times were averaged after all sensors died. Note that the elimination time of an uncovered hole was obtained by subtracting the fault injection time from the simulation duration.

Figure 10a and 10b shows that average hole elimination time increases according to network size and fault scale when using all the evaluated protocols. However, CHEAP provides the lowest values, i.e. approximately half the times obtained with the other evaluated protocols. This is due to CHEAP's ability to avoid nodes' cascaded movements which increase the coverage recovery process' latency. This is particulary true with DECM whose hole elimination time suffers from the *cooperative movement* scheme proposed by its authors. Performance of HORA is also due to the delay induced by the nodes' mobile region and the nearest assistant node selection processes. As for ZBFR, its back up nodes

Table 3 | Simulation parameters

Parameter	Value
Deployment area	1000 m × 1000 m
Number of sensors	100 to 1000
Sink's position	(450;200)
Sensors' communication ranges	{15;35;54;70;83;98;117;127} m
Sink's communication range	250 m
Sensors' initial energy (*Ei*)	2.5 J
Self-discharge per second	0.1 μJ
Threshold energy (*Ethr*)	100 μJ
E_{elec}	50 nJ/bit
e_{fs}	10 nJ/bit/m^2
e_{amp}	0.0013 nJ/bit/m^4
d_0	87 m
Message length (*l*)	2000 bits
t_{tlmax}	2
U_{sup}	2.7 V
I_{sens}	25 mA
t_{sens}	0.25 ms
Data size(*b*)	200 bits
Mass (*m*)	0.5 kg
Maximum velocity (*v*)	0.06 m/s
Deceleration rate ($\hat{\delta}$)	$\mathcal{U}(1;3)$
Chemical efficiency (*ea*)	90%
Mechanical efficiency (*eb*)	70%
Electrical efficiency (*ec*)	95%
Thermal efficiency (*ed*)	100%
Virtual grid cell length (λ)	7 m
Range offset (Δk)	3
Maximum number of iterations (*itmax*)	200

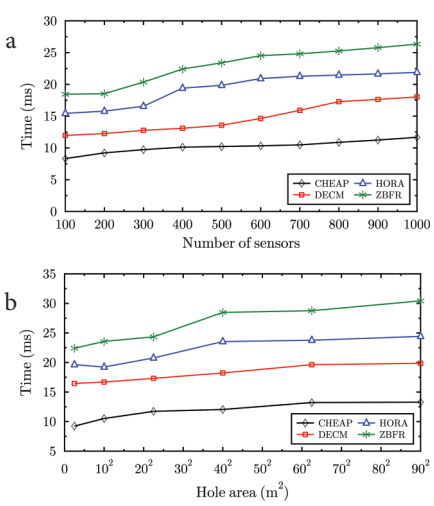

Figure 9 | Average hole detection time: (a) Impact of network size; (b) Impact of fault scale for network size = 1000.

designation process (sleeping nodes activation + redundancy check + sleep rescheduling) conducted by the Zone Monitors contributes to increasing hole elimination delay.

4.3. Average Mobilization Ratio

This experiment was aimed to evaluate each protocol's ability to minimize the number of candidates required to eliminate a hole.

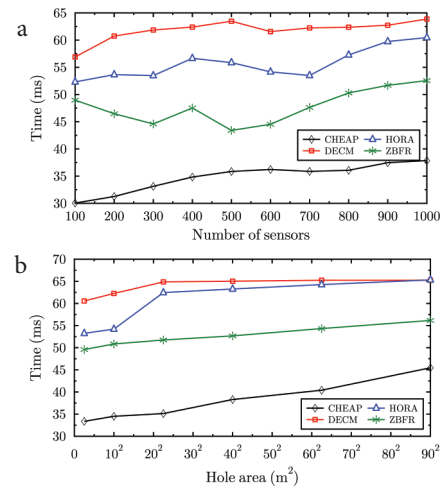

Figure 10 | Average hole elimination time: (a) effect of network size; (b) effect of fault scale for network size = 1000.

To do this, the simulation program recorded the set of potential candidates located two hops from the hole created after fault injection. After hole elimination, the simulation program calculated the ratio between the number of potential candidates and the number of candidates actually used.

After the last sensor died, the experiment was stopped and the resulting ratios were averaged. Figure 11a and 11b respectively suggest that regardless of the protocol evaluated, average mobilization ratio decreases while the network size grows but increases according to fault scale. Intuitively, the higher is the node density the lower is the number of candidates to be relocated. On the other hand, the larger is the coverage hole, the higher will be the number of required candidates. HORA yields the highest ratios (60–55%) especially when the number of sensors is large. Indeed, the mobility invitation-based scheme used in HORA tends to move all the possible candidates, even when the hole is already eliminated. The performances of DECM are more mitigated with an average ratio of 45% when the network size is 1000 and the hole area is 90 × 90 m^2. This is due to the cooperative movement mechanism applied in DECM to prevent generating new holes during node movements. Indeed, this strategy can lead to cascaded movements especially when node density is low. On the contrary, the strategy based on nodes' redundancy and mobility helps ZBFR to better cope with these cascaded movements, the low average ratios we obtained (50–42%). A similar strategy was applied in CHEAP; however, the node motility (i.e. range control) scheme helps to better minimize the number of moving candidates; hence, the lowest ratios (i.e. 40–35%).

4.4. Average Elasticity

We carried out this experiment in order to investigate each protocol's ability to prevent new hole formation during candidates' relocation. After hole elimination, the simulation program searched for a hole in candidates' coverage. This metric denotes the ratio between the number of candidates who did not make any coverage hole and the total number of candidates. The resulting ratios were averaged after all sensors died.

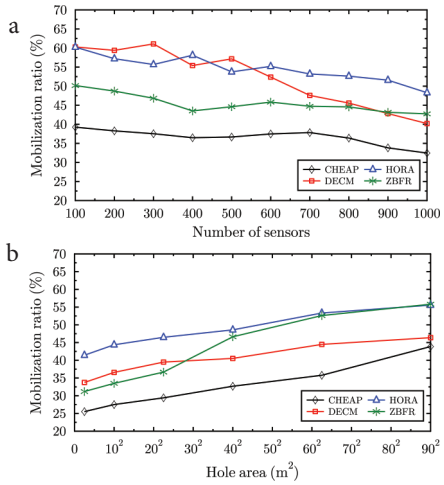

Figure 11 | Average mobilization ratio: (a) effect of network size; (b) effect of fault scale for network size = 1000.

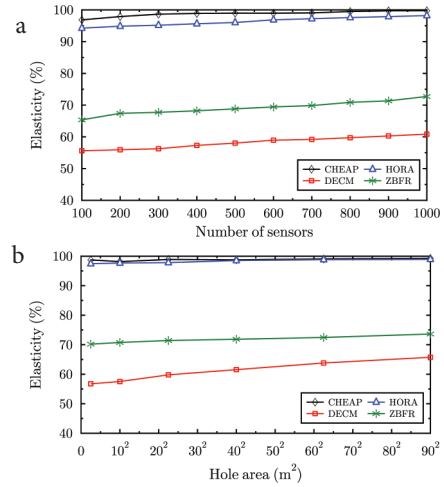

Figure 12 | Average elasticity: (a) effect of network size; (b) effect of fault scale for network size = 1000.

Figure 12a and 12b shows that elasticity increases according to network size and fault scale irrespective of the protocol used. These results are due to the fact that when node density is high, hole creation probability by cascaded movements is low; the same goes for the number of nodes to be relocated. DECM provides the lowest ratios (around 55%). This is because DECM can generate unnecessary node movements which reduce the Delaunay triangulation accuracy. This shortcoming is detrimental to safe area calculation and inevitably leads to new holes while candidates are moved. ZBFR yields better ratios (65–68%) because the strategy applied is mainly based on redundant nodes (back up nodes); but an insufficient number of backup nodes forces ZM to move the nearest neighbor even if a new hole is created. Thus, ZBFR actually eliminates holes through cascaded movements. The latter scheme is inefficient for holes located in the area of interest's periphery. This issue is addressed by the linear non linear programs respectively proposed by CHEAP and HORA who both yield ratios above 90%. However, when using CHEAP, values oscillate between 96% and 99.75%. Indeed, unlike HORA, CHEAP strives to control both nodes' ranges (motility) and mobility. These results prove the relevance of the strategy that allows each node to define a elbow room since it helps to take account of all the meaningless and risky movements.

4.5. Average Migration Ratio

The goal of this experiment was to assess each protocol's ability to minimize the travel distance required to eliminate a hole. For that purpose, after fault injection, the simulation program estimated and recorded the distance of each potential candidate to the hole's centroid. Then, after hole elimination the simulation program calculated for each candidate, the ratio between the distance actually travelled and the distance supposed to be covered.

Note that for undetected holes the ratio is assumed to be 100% for each candidate. This experiment was ended after all sensors died.

The results depicted in Figure 13a and 13b suggest that irrespective of the protocol used, the average migration ratio decreases according to network size but increases with fault scale. These performances are due to the fact that the higher is node density

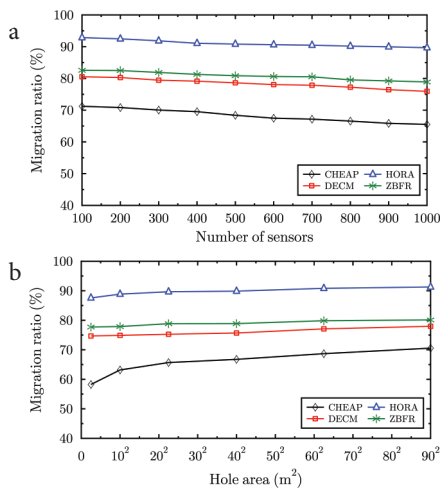

Figure 13 | Average migration ratio: (a) effect of network size; (b) effect of fault scale for network size = 1000.

the shorter is the distance to be travelled by candidates (especially, the boundary nodes) to eliminate any hole; but, for a specific network size, the larger is a hole, the longer is the distance travelled. HORA provides the worst ratios (above 90%). Indeed, in this scheme the only criterion for candidates selection is their overlapping degrees. Ignoring nodes residual energy inevitably leads to high distances travelled. When using ZBFR or DECM, the average migration ratio respectively reaches 82% and 80%. ZBFR also does not consider candidates' residual energy but unlike HORA, the Zone Monitors relocate the nearest backup nodes (redundant nodes). DECM use a similar scheme when helping candidates find the shortest path to their safe area. Regrettably, all these performances are mitigated by the cascaded movements attached to both solutions. CHEAP outperforms the three other protocols by providing a 70% ratio on average. This performance is mainly due to the minimization scheme applied by the *coordinator* while selecting candidates. This strategy helps to avoid using nodes' cascaded movements. Indeed, when using CHEAP only mutually compatible movements are allowed, this strategy prevents new hole formation and minimizes the number

of movements required for coverage restoration. Furthermore, unlike DECM and HORA, CHEAP considers both nodes' residual energy, nodes' motility, redundant nodes' mobility, and obstacles during the candidates selection process.

4.6. Average Elimination Ratio

In order to evaluate each protocol's ability to restore the coverage degree we conducted an experiment where the simulation program had to calculate the area of the newly created hole just after having injected a fault. After nodes' relocation, the resulting coverage ratio was estimated. Note that each hole which was not eliminated was assumed to have 0% coverage ratio. This experiment ended after all sensors died.

Figure 14a and 14b shows that regardless of the protocol used, the average elimination ratio increases according to network size, but decreases with fault scale.

This is because when node density is high, the number of candidates is large enough to restore the coverage. However, intuitively for the same network density, the average elimination ratio decreases when the area to be restored increases. All the evaluated protocols, provide ratios greater than 90%. DECM yields the lowest ratios (between 93% and 95.5%). This is due to the inaccuracy of Delaunay triangulation. Indeed, as discussed in Section 4.4 this scheme can lead to some errors when safe areas are calculated especially with open holes or those located in the outskirts of the area of interest. These errors are detrimental to hole full coverage. HORA and ZBFR provide better results (respectively between 95.2% and 95.6% and between 96.5% and 97%) because these schemes mainly consider redundant nodes when trying to eliminate a coverage hole; but these performances are mitigated by the fact that coverage redundancy check is a NP-complete problem [51,52] which requires approximate solutions that often lead to several false negative cases. The latter prevent efficient candidates mobility and hole full coverage. CHEAP provides the highest ratios varying between 96.5% and 98%. This is due to the fact that during the hole elimination process, when making decisions *coordinator* uses candidates' maximum range while targeting primarily the centroid. The latter position often provides the highest coverage ratio when using node's maximum range.

4.7. Energy Efficiency and Network Lifetime

In order to estimate the amount of energy depleted during hole detection and elimination processes, we used the same experimental setup as described in the previous sections. Since we mainly aimed at investigating each protocol's ability to minimize energy losses, we injected only faults that displaced nodes. We conducted this experiment respectively until a node died [First Node Dies (FND)] and until all nodes died [Last Node Dies (LND)].

Figure 15a and 15b then 16a and 16b show that irrespective of the protocol used and the lifetime definition (FND or LND), the energy oriented network lifetime decreases while network size and fault scale grow. These results are due to the fact that energy losses are mainly due to communication activities. The latter increase according to nodes' degree (number of neighbors). By contrast, large scale disasters (especially when several nodes

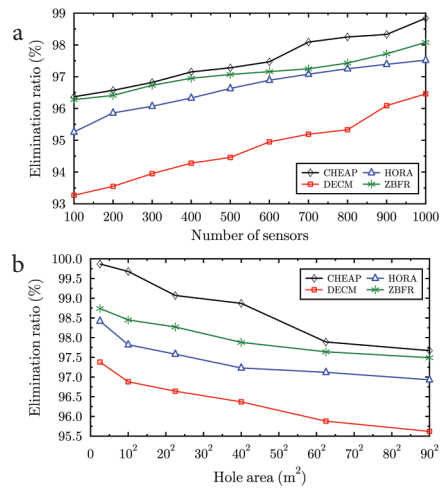

Figure 14 | Average elimination ratio: (a) effect of network size; (b) effect of fault scale for network size = 1000.

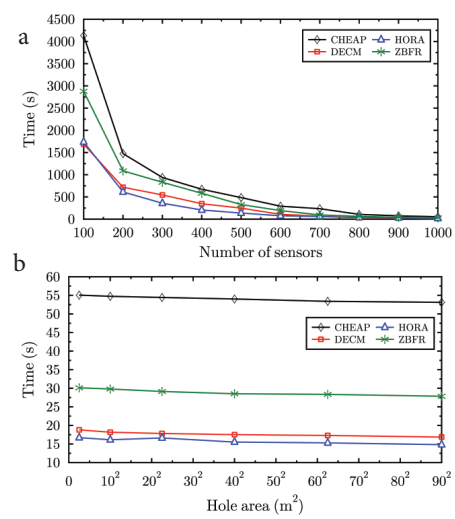

Figure 15 | Network lifetime (until First Node Dies): (a) effect of network size; (b) effect of fault scale for network size = 1000.

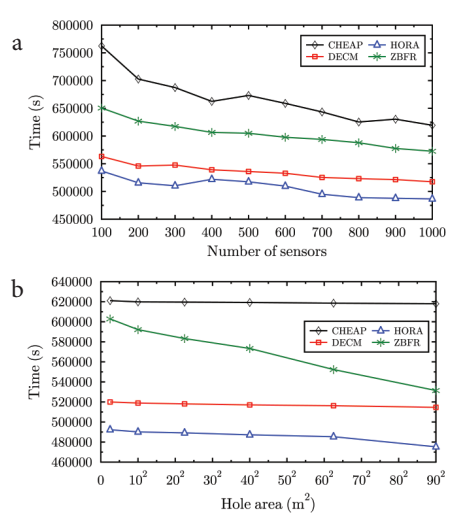

Figure 16 | Network lifetime (until Last Node Dies): (a) effect of network size; (b) effect of fault scale for network size = 1000.

become isolated) tend to decrease nodes' average degree and reduce therefore their energy consumption. However, since faults are essentially local, they have a relatively small impact on the network's lifetime. CHEAP obtains the best results. They are actually due to its performances in terms of message complexity and average migration or mobilization ratio, as discussed in previous sections. By contrast DECM, ZBFR, and HORA use more energy consuming strategies such as cascaded movements. Indeed, ZBFR uses *heart-beat* signalization scheme for hole detection while DECM leads to unnecessary node movements due to Delaunay triangulation inaccuracy (see Section 4.4). HORA does not consider residual energy to select candidates for coverage recovery. The latter strategy is particularly energy inefficient.

4.8. Coverage Resilience Index

In order to evaluate each protocol's ability to maximize network resilience denoted by φ, we propose to aggregate the metrics discussed above using a weighted geometric mean as expressed by Equation (34). Let τ_m, l_s, τ_g, τ_e, and ε respectively be the average mobilization ratio, the average elasticity, the migration ratio, the average elimination ratio, and the average energy consumption ratio.

$$\varphi = (1-\tau_m)^{w_1} \times ls^{w_2} \times (1-\tau_g)^{w_3} \times \tau_e^{w_4} \times (1-\varepsilon)^{w_5}, \sum_{i=1}^{5} w_i = 1 \qquad (34)$$

where $w_1 \ldots w_5$ denote the weighting coefficients of the aggregated metrics. We chose to use an equal weighting scheme (i.e. $\forall i \in \{1, .., 5\}$, $w_i = 0.2$).

By definition, a candidate's energy consumption ratio is the amount of energy lost during the entire coverage maintenance process divided by its residual energy at the time of fault injection.

Figure 17a and 17b shows that regardless the protocol used, resilience increases according network size, but decreases as fault scale grows. These results are due to the fact that high node density intuitively fosters coverage recovery speed and ratio. Since this resilience index is a composite metric, any performance depends on the results discussed in previous section. CHEAP provides a coverage resilience index between 70% and 74.2%. In other words, our contribution enables network to efficiently maintain coverage in 70–74.2% of all cases; while the three other protocols' results oscillate between 45% and 57%. These results prove that the coordination and location-allocation-based strategy combined with nodes' range control is more relevant than any iterative movements-based scheme to eliminate coverage holes.

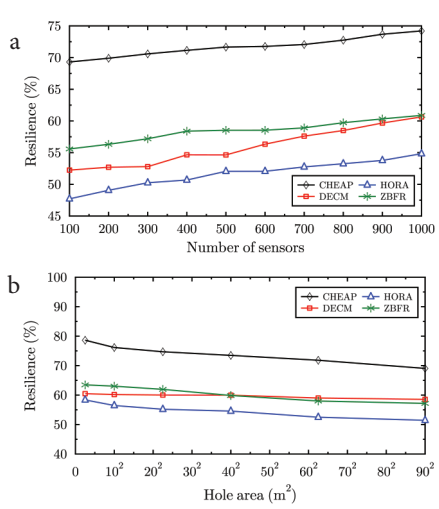

Figure 17 | Resilience: (a) effect of network size; (b) effect of fault scale for network size = 1000.

5. CONCLUSION

In this paper we addressed the area coverage restoration problem in self-organized mobile WSNs with the objective of maximizing the network's resilience. We proposed a localized solution referred to as CHEAP that uses first, a geometric approach based on nodes' sensing disks crossings to effectively detect any type of holes; then a tabu search based heuristic that combines nodes' redundancy, motility and mobility-based strategies to restore the lost coverage. Simulation results have confirmed that regardless of the type of hole, CHEAP outperforms several major previous schemes. The *coverage resilience index* metric we proposed, explicitly revealed that the location-allocation-based scheme we applied helps to effectively minimize risks of new hole formation, average distance travelled, nodes' movements, overlapped areas, and total power consumption; while maximizing coverage ratio.

In future work, we plan to extend this solution to three-dimensional environments.

CONFLICTS OF 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.

REFERENCES

[1] Y. Wang, Y. Zhang, J. Liu, and R. Bhandari, Coverage, connectivity, and deployment in wireless sensor networks, in: S. Patnik, X. Li, Y.M. Yang (Eds.), Recent Development in Wireless Sensor and Ad-hoc Networks. Signals and Communication Technology, Springer, India, 2014, pp. 25–44.

[2] T. Ojha, S. Misra, N.S. Raghuwanshi, Wireless sensor networks for agriculture: the state-of-the-art in practice and future challenges, Comput. Electron. Agric. 118 (2015), 66–84.

[3] L. Muduli, D.P. Mishra, P.K. Jana, Application of wireless sensor network for environmental monitoring in underground coal mines: a systematic review, J. Netw. Comput. Appl. 106 (2018), 48–67.

[4] S. Chouikhi, I.E. Korbi, Y. Ghamri-Doudane, L.A. Saidane, A survey on fault tolerance in small and large scale wireless sensor networks, Comput. Commun. 69 (2015), 22–37.

[5] S. Abdollahzadeh, N.J. Navimipour, Deployment strategies in the wireless sensor network: a comprehensive review, Comput. Commun. 91–92 (2016), 1–16.

[6] H. Yetgin, K.T.K. Cheung, M. El-Hajjar, L.H. Hanzo, A survey of network lifetime maximization techniques in wireless sensor networks, IEEE Commun. Surv. Tutor. 19 (2017), 828–854.

[7] N.D. Trong, N.P. Le, P.V. Hau, N.V. Khanh, A distributed protocol for detecting and updating hole boundary in wireless sensor

networks, Proceedings of the Sixth International Symposium on Information and Communication Technology - SoICT 2015, ACM Press, New York, USA, 2015, pp. 171–178.

[8] T. Amgoth, P.K. Jana, Coverage hole detection and restoration algorithm for wireless sensor networks, Peer-to-Peer Netw. Appl. 10 (2017), 66–78.

[9] P.K. Sahoo, M.J. Chiang, S.L. Wu, An efficient distributed coverage hole detection protocol for wireless sensor networks, Sensors 16 (2016), 386.

[10] G. Dai, H. Lv, L. Chen, B. Zhou, P. Xu, A novel coverage holes discovery algorithm based on voronoi diagram in wireless sensor networks, Int. J. Hybrid Inf. Technol. 9 (2016), 273–282.

[11] D. Šoberl, N.M. Kosta, P. Škraba, Decentralized computation of homology in wireless sensor networks using spanning trees, in: A. Holzinger, P. Kieseberg, A. Tjoa, E. Weippl (Eds.), Machine Learning and Knowledge Extraction. Lecture Notes in Computer Science, Springer International Publishing, Cham, 2017, pp. 25–40.

[12] S. Patra, B. Sau, Detecting hole boundary nodes in WSN under distributed environment, 2016 IEEE 6th International Conference on Advanced Computing (IACC), IEEE, Bhimavaram, India, 2016, pp. 716–721.

[13] A. More, V. Raisinghani, A survey on energy efficient coverage protocols in wireless sensor networks, J. King Saud Univ. Comput. Inform. Sci. 29 (2017), 428–448.

[14] B. Wang, Sensor activity scheduling, in: Coverage Control in Sensor Networks, Computer Communications and Networks, Springer, London, 2010, pp. 121–153.

[15] F. Li, J. Luo, W. Wang, Y. He, Autonomous deployment for load balancing *k*-surface coverage in sensor networks, IEEE Trans. Wireless Commun. 14 (2015), 279–293.

[16] D. Saha, A. Das, Coverage area maximization by heterogeneous sensor nodes with minimum displacement in mobile networks, 2015 IEEE International Conference on Advanced Networks and Telecommuncations Systems (ANTS), IEEE, Kolkata, India, 2015, pp. 1–6.

[17] T.Y. Lin, H.A. Santoso, K.R. Wu, Global sensor deployment and local coverage-aware recovery schemes for smart environments, IEEE Trans. Mobile Comput. 14 (2015), 1382–1396.

[18] K.P. Sharma, T.P. Sharma, ZBFR: zone based failure recovery in WSNs by utilizing mobility and coverage overlapping, Wireless Netw. 23 (2016), 2263–2280.

[19] M. Amac Guvensan, A. Gokhan Yavuz, Hybrid movement strategy in self-orienting directional sensor networks, Ad Hoc Netw. 11 (2013), 1075–1090.

[20] L. Cooper, Location-allocation problems, Oper. Res. 11 (1963), 331–343.

[21] H.A. Eiselt, G. Laporte, Objectives in location problems, in Facility Location: A Survey of Applications and Methods, Springer Nature, New York, 1995, pp. 151–180.

[22] R. Beghdad, A. Lamraoui, Boundary and holes recognition in wireless sensor networks, J. Innov. Dig. Ecosyst. 3 (2016), 1–14.

[23] S. Das, M.K. DebBarma, Hole detection in wireless sensor network: a review, in: P. Sa, S. Bakshi, I. Hatzilygeroudis, M. Sahoo (Eds.), Recent Findings in Intelligent Computing Techniques. Advances in Intelligent Systems and Computing, Springer, Singapore, 2018, pp. 87–96.

[24] W. An, N. Qu, F.M. Shao, X. Xiong, S. Ci, Coverage hole problem under sensing topology in flat wireless sensor networks, Wireless Commun. Mobile Comput. 16 (2014), 578–589.

[25] Z. Kang, H. Yu, Q. Xiong, Detection and recovery of coverage holes in wireless sensor networks, J. Netw. 8 (2013), 822–828.

[26] C.F. Huang, Y.C. Tseng, The coverage problem in a wireless sensor network, Mobile Netw. Appl. 10 (2005), 519–528.

[27] Y. Bejerano, Coverage verification without location information, IEEE Trans. Mobile Comput. 11 (2012), 631–643.

[28] C. Qiu, H. Shen, K. Chen, An energy-efficient and distributed cooperation mechanism for k-coverage hole detection and healing in WSNs, 2015 IEEE 12th International Conference on Mobile Ad Hoc and Sensor Systems, IEEE, Dallas, TX, USA, 2015, pp. 73–81.

[29] W. Li, Y. Wu, Tree-based coverage hole detection and healing method in wireless sensor networks, Comput. Netw. 103 (2016), 33–43.

[30] M.R. Senouci, A. Mellouk, K. Assnoune, Localized movement-assisted sensor deployment algorithm for hole detection and healing, IEEE Trans. Parallel Distrib. Syst. 25 (2014), 1267–1277.

[31] Q. Fang, J. Gao, L.J. Guibas, Locating and bypassing holes in sensor networks, Mobile Netw. Appl. 11 (2006), 187–200.

[32] W.C. Chu, K.F. Ssu, Location-free boundary detection in mobile wireless sensor networks with a distributed approach, Comput. Netw. 70 (2014), 96–112.

[33] F. Yan, A. Vergne, P. Martins, L. Decreusefond, Homology-based distributed coverage hole detection in wireless sensor networks, IEEE/ACM Trans. Netw. 23 (2025), 1705–1718.

[34] D. Diongue, O. Thiare, An energy efficient self-healing mechanism for long life wireless sensor networks, in: T. Sobh, K. Elleithy (Eds.), Innovations and Advances in Computing, Informatics, Systems Sciences, Networking and Engineering, Lecture Notes in Electrical Engineering, Springer, Cham, 2015, pp. 599–605.

[35] G. Wang, G. Cao, T.F. La Porta, Movement-assisted sensor deployment, IEEE Trans. Mobile Comput. 5 (2006), 640–652.

[36] Y. Wang, J. Gao, J.S.B. Mitchell, Boundary recognition in sensor networks by topological methods, Proceedings of the 12th Annual International Conference on Mobile Computing and Networking - MobiCom'06, ACM Press, New York, USA, 2006, pp. 122–133.

[37] Y. Zou, K. Chakrabarty, Sensor deployment and target localization based on virtual forces, IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No.03CH37428), IEEE, San Francisco, CA, USA, 2003, pp. 1293–1303.

[38] J. Habibi, H. Mahboubi, A.G. Aghdam, Distributed coverage control of mobile sensor networks subject to measurement error, IEEE Trans. Autom. Control 61 (2016), 3330–3343.

[39] P.K. Sahoo, W.C. Liao, Hora: a distributed coverage hole repair algorithm for wireless sensor networks, IEEE Trans. Mobile Comput. 14 (2015), 1397–1410.

[40] C. Qiu, H. Shen, A delaunay-based coordinate-free mechanism for full coverage in wireless sensor networks, IEEE Trans. Parallel Distrib. Syst. 25 (2014) 828–839.

[41] A. Khelil, R. Beghdad, ESA: an efficient self-deployment algorithm for coverage in wireless sensor networks, Proced. Comput. Sci. 98 (2016), 40–47.

[42] M. Rout, R. Roy, Dynamic deployment of randomly deployed mobile sensor nodes in the presence of obstacles, Ad Hoc Netw. 46 (2016) 12–22.

[43] H. Zhao, Q. Zhang, L. Zhang, Y. Wang, A novel sensor deployment approach using fruit fly optimization algorithm in wireless sensor networks, 2015 IEEE Trustcom/BigDataSE/ISPA, IEEE, Helsinki, Finland, 2015, pp. 1292–1297.

[44] A. Ray, D. De, An energy efficient sensor movement approach using multi-parameter reverse glowworm swarm optimization algorithm in mobile wireless sensor network, Simul. Modell. Pract. Theory 62 (2016), 117–136.
[45] J. Wang, S. Medidi, M. Medidi, Energy-efficient k-coverage for wireless sensor networks with variable sensing radii, GLOBECOM 2009 - 2009 IEEE Global Telecommunications Conference, IEEE, Honolulu, HI, USA, 2009, pp. 1–6.
[46] C.T. Vu, Y. Li, Delaunay-triangulation based complete coverage in wireless sensor networks, 2009 IEEE International Conference on Pervasive Computing and Communications, IEEE, Galveston, TX, USA, 2009, pp. 1–5.
[47] Y. Qu, S.V. Georgakopoulos, A distributed area coverage algorithm for maintenance of randomly distributed sensors with adjustable sensing range, 2013 IEEE Global Communications Conference (GLOBECOM), IEEE, Atlanta, GA, USA, 2013, pp. 286–291.
[48] M. Abolhasan, Y. Maali, A. Rafiei, W. Ni, Distributed hybrid coverage hole recovery in wireless sensor networks, IEEE Sens. J 16 (2016), 8640–8648.
[49] K. Lakshmi Joshitha, S. Jayashri, A novel redundant hole identification and healing algorithm for a homogeneous distributed wireless sensor network, Wireless Personal Commun. 104 (2018), 1261–1282.
[50] 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.
[51] H.G. Diédié, M. Babri, S. Oumtanaga, Redundancy detection protocol for area coverage control in heterogeneous wireless sensor networks, Int. J. Comput. Sci. Issues 12 (2015), 100–110.
[52] H.G. Diédié, B. Aka, M. Babri, Area k-coverage optimization protocol for heterogeneous dense wireless sensor networks, Int. J. Adv. Comput. Sci. Appl. 8 (2017), 327–336.
[53] J. Krarup, D. Pisinger, F. Plastria, Discrete location problems with push-pull objectives, Discrete Appl. Math. 123 (2002), 363–378.
[54] R.Z. Farahani, M. SteadieSeifi, N. Asgari, Multiple criteria facility location problems: a survey, Appl. Math. Modell. 34 (2010), 1689–1709.
[55] P. Brass, Geometric problems on coverage in sensor networks, in: I. Bárány, K.J. Böröczky, G.F. Tóth, J. Pach (Eds.), Geometry – Intuitive, Discrete, and Convex. Bolyai Society Mathematical Studies, Springer Nature, Berlin, Heidelberg, 2013, pp. 91–108.
[56] S.M. Kwon, J.S. Kim, Coverage ratio in the wireless sensor networks using Monte Carlo simulation, 2008 Fourth International Conference on Networked Computing and Advanced Information Management, IEEE, Gyeongju, South Korea, 2008, pp. 235–238.
[57] Y. Liu, L. Suo, D. Sun, A. Wang, A virtual square grid-based coverage algorithm of redundant node for wireless sensor network, J. Netw. Comput. Appl. 36 (2013), 811–817.
[58] A.K. Idrees, K. Deschinkel, M. Salomon, R. Couturier, Distributed lifetime coverage optimization protocol in wireless sensor networks, J. Supercomput. 71 (2015), 4578–4593.
[59] O. Kariv, S.L. Hakimi, An algorithmic approach to network location problems. II: the p-medians, SIAM J. Appl. Math. 37 (1979), 539–560.
[60] R.J. Fowler, M.S. Paterson, S.L. Tanimoto, Optimal packing and covering in the plane are NP-complete, Inform. Process. Lett. 12 (1981), 133–137.
[61] N. Megiddo, A. Tamir, On the complexity of locating linear facilities in the plane, Oper. Res. Lett. 1 (1982), 194–197.
[62] F. Glover, Future paths for integer programming and links to artificial intelligence, Comput. Oper. Res. 13 (1986), 533–549.
[63] B.S. Panda, D.P. Shetty, Minimum range assignment problem for two connectivity in wireless sensor networks, in: R. Natarajan (Ed.), Distributed Computing and Internet Technology, ICDCIT 2014, Springer, Cham, 2014, pp. 122–133.
[64] G.H.F. Diédié, B. Aka, M. Babri, Energy-efficient ant colony based k-hop clustering and transmission range assignment protocol for connectivity construction in dense wireless sensor networks, J. Comput. Sci. 14 (2018), 376–395.
[65] G.H.F. Diédié, B. Aka, M. Babri, Self-stabilising hybrid connectivity control protocol for WSNs, IET Wireless Sens. Syst. 9 (2019), 6–24.
[66] A. Varga, OMNeT++ Simulator, Available from: http://www.omnetpp.org (accessed on July 2016).
[67] W. Heinzelman, A.P. Chandrakasan, H. Balakrishnan, An application-specific protocol architecture for wireless microsensor networks, IEEE Trans. Wireless Commun. 1 (2002), 660–670.
[68] M.N. Halgamuge, M. Zukerman, K. Ramamohanarao, H.L. Vu, An estimation of sensor energy consumption, Progr. Electromagn. Res. B 12 (2009), 259–295.
[69] Society of Robots (SOR), Robot Energy, Available from: https://www.societyofrobots.com/mechanics_energy.shtml (accessed on July 2020).