A Comprehensive Review of Sensor Relocation

Wei Shi
Faculty of Business and Information Technology
University of Ontario Institute Technology
Oshawa, Canada Email: wei.shi@uoit.ca

Jean-Pierre Corriveau
School of Computer Science
Carleton University
Ottawa, Canada Email: jeanpier@scs.carleton.ca

***Abstract*—In this paper, recent literature pertaining to the sensor relocation problem is reviewed. Different proposed solutions are categorized, summarized and compared.**

***Keywords*-Wireless Sensor Network; Actuator; Robot; Relocation; Algorithms.**

I. Introduction

A Wireless Sensor Network (WSN) consists of a collection of sensors equipped with a wireless communication device (e.g., radio transceiver), and an energy source (e.g.,battery). Some sensors, called *stationary* or *static* wireless sensors, are immobile. Others, called *mobile* wireless sensors can move. Wireless sensor and actor networks (WSANs) refer to a group of sensors and actuators (i.e., actors) linked by wireless medium to perform distributed sensing and actuation tasks. Actors are usually resource-rich devices equipped with better processing capabilities, stronger transmission abilities and longer battery life [1] than mobile sensors. A Wireless Sensor and Robot Network (WSRN) combines WSNs with Multi-Robot systems. It consists of sensor and robot nodes that communicate via wireless links to perform distributed sensing and actuation tasks [12]. In WSRNs or WSANs, a set of sensors are often deployed in an unknown environment, such as forests, battlefields, mountains, highways, tunnels and mines, in order to monitor or control physical or environmental conditions. An adequate sensors coverage level is crucial for obtaining a proper phenomenon of interest and the successful completion of the corresponding sensing tasks. But, given an unknown environment, it is very difficult to guarantee a perfect deployment. Moreover, even after sensors are deployed into ideal locations, they often randomly fail at run time for various reasons such as power depletion, hardware defects, and damaging events. This creates unmonitored locations in the given environment. Such locations are often referred as *sensing holes*. In order to fill these sensing holes, spare sensors or previously deployed sensors need to be moved to overcome these failures, or to respond to an event that requires that a sensor be moved to a particular location. We call this procedure *sensor relocation* [14].

Some target environments can be dangerous if not inaccessible to humans. In such environments, given stationary sensors are not able to self-relocate, mobile sensors or robots or actuators are used for relocation in order to track the phenomena of interest or achieve the coverage despite stochastic or unpredictable node failure. Mobile sensors need to have motors, motion control, and GPS modules. Adding mobility to a large number of sensor nodes is expensive. Also, most of the mobile sensors on the market are equipped with a low energy source, such as a battery, in order to reduce cost. So, mobile sensor self-relocation is not suitable for all situations. For example, in a large environment, using powerful robots or actuators (often referred as movement-assisted sensor relocation) may be necessary. In such case, in contrast to mobile sensors, we require robots or actuators that can carry spare sensors and replace the failure sensor when it is necessary. When such robots or actuators cannot carry or do not have at hand any spare sensors, they can pick up *passive* (e.g. sleeping or not active) sensors previously deployed in the target environment and replace the failure sensors with them.

Contrary to robots/actuators- assisted sensor relocation, mobile sensor self-relocation requires sensors to have locomotion and involves two tasks: replacement discovery (i.e., finding a redundant sensor as the replacement of a failed sensor node) and replacement migration (i.e., migrating the replacement to the failed sensor's position [10]). In this paper, we review recent sensor relocation algorithms organizing them into two general categories: mobile sensor self-relocation and robot/actuator assisted sensor relocation. We first present a list of features used to distinguish each solution from the others. Then we briefly summarize the relevant algorithms. Also, we use a comparative matrix to summarize this collection of solutions for the sensor relocation problem.

II. Feature Analysis

From studying the environment setup, assumptions, and basic mechanisms of the reviewed algorithms (see Table I), we identify five useful features for the comparison of these algorithms.

- Migration techniques (Directed or Shifted migration): Sensor migration occurs when a mobile sensor self-relocates from its current position to its target position. Migration is achieved when a mobile sensor moves all the way to the target location directly. Alternatively, a mobile sensor can achieve sensor replacement by building a multi-hop migration path [10] from its current position to the target location (the position of a failure sensor). Using this *shifting* migration method, energy consumption is distributed across multiple sensor nodes thus prolonging network lifetime. To further optimize the overall energy consumption and response time, a replacement node should always be a redundant sensor geographically closest to the failed sensor node. Regardless of the actual details of a migration technique, a shorter path of migration is always desired.

- Use of proxy: In order to reduce message overhead, several algorithms choose to use proxies to advertise information for sensor nodes. Algorithms in [7], [13] use the closest static node to a sensing hole as a proxy for the latter; whereas [4], [8] uses the closest node to a redundant mobile sensor as proxy.
- Geographic arrangement/Area Partition Methods:
    1) Voronoi diagram based: [11], [13];
    2) Grid-based: the network field is partitioned into a $2 - D$ grid;
    3) Zone-based: a variant of the quorum-based location service [7];
    4) Mesh-based: in a grid sensor network, A-nodes are placed exactly at the intersection points of a grid structure. Any A-node can find out its own role in the grid structure just by counting its number of A-node neighbors. If one considers only the rows and columns of the proxy nodes in the network, these intersect each other and form a mesh structure [8];
    5) Hierarchical Hamilton Cycle: the whole network is partitioned into grids. Each 4 neighboring grids form a Hamilton cycle [1] (i.e., a level-1 cycle in the hierarchical structure). One of them will be selected as the eye (i.e., monitor) to collect all the coverage and connectivity information along such a cycle. And then, each 4 selected eyes form a higher level (i.e., level-2) Hamilton cycle. This process will continue until all the grids are connected in a level-k Hamilton cycle [6].
    6) Equilateral Triangle Tessellation: an equilateral triangle tessellation (TT) is a planar graph composed of congruent equilateral triangles. Given an orientation, say north, and edge length l_e, each sensor is able to locally compute a unique TT containing a point $\mathcal{P}$ as vertex [9].
- Algorithm Structure (Centralized vs. Distributed): Distributed (or Localized) algorithms are typically executed concurrently by different sensors and robots/actuators, each of which having limited information about the status of the rest of the environment. A centralized algorithm often requires a central server that receives and analyzes the data, identifies the sensing hole, plans the migration path and assigns a replacement task to the particular mobile sensor(s), robot(s) or actuator(s).
- Sensor mobility (Robot/actuator assisted vs. mobile sensor self-relocation): [3]–[5], [11] use robot/actuator assisted sensor relocation. But [3], [4] use mobile robots reusing local static sensors whereas in [11] robots are required to always carry enough spare sensors and repair sensing holes only with these sensors. Thus, strictly speaking, [11] does not address sensor relocation per se. Other algorithms reviewed in this survey are mobile sensor self-relocation algorithms.

	Migration Technique	Use of Proxy	Area Partition	Net. Arch.	Sensor Mobility
[3]	Direct	No	Complete Graph	Central.	Robot
[4]-R3S2	Direct	Yes	Plane	Distr.	Robot
[4]-B-R3S2	Direct	Yes	Grid	Distr.	Robot
[5]	Direct	No	Plane	Distr.	Actuator
[6]	Direct	Yes	Hier. Ham. Cyc.	Distr.	Mobile Sensor
[7]	Shifted	Yes	Zone	Distr.	Mobile Sensor
[8]	Shifted	Yes	Mesh	Central.	Mobile Sensor
[9]	Shifted	No	T.T.	Distr.	Mobile Sensor
[11] -Centralized	Direct	No	Plane	Central.	Robot
[11] -Fixed	Direct	No	Square or Hexagons	Distr.	Robot
[11] -Dynamic	Direct	No	Voronoi Diagram	Distr.	Robot
[13]	Direct	Yes	Voronoi Diagram	Distr.	Mobile Sensor
[14]	Shifted	Yes	Grid	Central.	Mobile Sensor

Table I
CATEGORIZATION OF REVIEWED PAPERS

III. MOBILE SENSOR SELF-RELOCATION

[13] presents a proxy-based sensor deployment protocol that uses a mix of static and mobile sensors. This protocol also solves the sensor relocation problem. Static sensors create a Voronoi diagram [2] and detect the sensing holes. Each mobile sensor selects the closest static sensor as its proxy in order to reduce message overhead. Mobile sensors communicate by exchanging messages with their proxies. A proxy advertises the logical location, physical location and *base price* (i.e., current sensing hole coverage) of its mobile sensor to the network so that all the static sensors can bid to cover the sensing holes. Then the proxy sensors choose the highest bid and send a message to the winning bidder. This bidder becomes a new proxy. All new proxies update the base price of their mobile sensors. Then, proxies

need to check whether hole-exchange is needed. If so, they choose the mobile sensor suitable for exchange and send out an exchange request to the proxy of that mobile sensor. To reduce energy consumption, mobile sensors first move logically and when they find their target location, they move directly toward it to avoid extra or zigzag movement. Once a mobile sensor arrives at a sensing hole, it advertises its base price again and tries to relocate to cover a bigger hole.

[14] proposes a Grid-Quorum based mobile sensor self-relocation protocol that assumes the sensor field is known. That is, the field is divided into equal grids. Each grid has a *head*, which is responsible for collecting information for all the sensors in this grid. A head also determines if there exists redundant sensors and/or sensing holes in its grid. It broadcasts a message containing this information to all other grid heads. To reduce message complexity the grid heads send these messages to a grid quorum rather than to the whole network. The grid heads in a row form a *supply* quorum and the grid heads in a column form a *demand* quorum. When any grid needs more sensors for hole covering, its grid head searches along its demand quorum to discover the closest redundant node. After finding the latter the relocation phase proper starts. Relocation is based on a cascading movement, that is, the sensors relocate in a shifting manner. The difficulty with this phase is to find the best cascading schedules that minimize the overall energy consumption and maximize the remaining energy for each sensor.

[7] presents a distributed algorithm, called ZONER, to solve the mobile sensor relocation problem. This algorithm assumes global coordinates as well as a unique ID for each sensor. In each bounded horizontal request zone, every redundant node registers itself with all the non-redundant nodes inside a vertical registration zone across the entire network. A non-redundant node maintains a one-hop neighborhood map by listening to a periodical HELLO message from the redundant nodes in its residing zone. When a node fails, its specified neighbors query all the non-redundant nodes inside each zone for redundant nodes. Because the request zone intersects with a number of registration zones, the non-redundant nodes in the intersection areas can also provide the requester with redundant node information. Once a satisfactory and available redundant node is identified, it will be relocated in a shifting way to replace the failed node with no change in network topology. In summary, [7] and [14] are both Quorum-based mobile sensor self-relocation protocols. But, ZONER does not require knowledge of the sensor field and thus is applicable to unknown environments.

In [8], a mesh-based mobile sensor relocation protocol (MSRP) is proposed. The sensor nodes are already deployed in an area. Active nodes (A-nodes) construct the network, whereas redundant nodes (R-nodes) are merely scattered in this area. An R-node chooses the closest A-node as its proxy (by sending it a delegation request). Each R-node keeps monitoring the aliveness of its proxy. As soon as an R-node notices its proxy fails, it moves to replace it. The proxies keep the location information of their closest redundant nodes. Proxies construct an information mesh (imesh) and inform all the other A-nodes of their position. Whenever an A-node fails, its westmost, eastmost, northmost and southmost neighbors act as servers, sending a query in each of these four directions and waiting for replies. After receiving replies, the closest proxy node, called the *replacement discoverer*, is selected as a replacement proxy by the server. After finding this proxy, the replacement phase proper starts by constructing a relocation path based on a routing protocol. All the mobile sensors on this path carry out the replacement using shifting. That is, the replacement discoverer moves to the location of the failed node and each intermediate nodes moves to the position of its preceding node on the path. Ultimately, the last proxy node on the path informs the closest redundant node of its movement and moves. Then the redundant node fills its position.

In [6] a hierarchical Hamilton cycle based algorithm is presented to solve the mobile sensor relocation problem. In this algorithm, the target field is partitioned evenly into r x r grids. In each grid, a head is elected and all the other nodes in this grid are called *spare enabled* nodes. All the nodes have the same sensing range and communication range R. It is also assumed that both network connectivity and full coverage are preserved as long as there exists one node in each grid. This enables each head of a grid to monitor the status of the neighboring heads. Every four neighboring grids form a level-1 Hamilton cycle and one of them is elected as the eye of the Hamilton cycle. The head of the eyes collects the information of redundant sensors and empty grids in these 4 grids. The eyes of four neighboring level-1 Hamilton cycle form a level-2 directed Hamilton cycle and share information. This Hamilton cycle formation is performed recursively until a level-k cycle covering the entire sensor field is built on four level- $(k-1)$ eyes. Whenever a grid head detects an empty grid in its neighboring grid, it starts an intra-level repair process to fix it. In this phase, if there is a redundant sensor in this grid, the grid head sends it to the empty grid, otherwise it moves itself to fill the neighboring grid. Before a grid head moves to fill the empty grid, it sends a notification message to its preceding neighbor head in its residing level-i Hamilton cycle. This process continues until a redundant sensor is found in these four neighboring grids in level-i Hamilton cycle. Otherwise the inter-level repair process will be started. In this phase, the head of the eyes searches the redundant sensor in the upper levels and it continues until a redundant sensor is detected in one of the upper levels.

The two algorithms presented in [9] first construct a network by sensor self-deployment. An equilateral triangle tessellation (TT) layout is required for mobile sensors self-deployment in a $2D$ free field. This new form of sensor self-deployment achieves focused coverage around a Point of Interest (POI). In both algorithms presented in that paper, once the network is formed, namely, when sensors are compactly placed or collide, they may move away from the POI. In order to maintain a connected network of TT layout with hole-free coverage, the extra sensors will relocate in a shifted way to fill any emerging sensing holes. The relocation path is a path linking the failed sensor and a sensor on the outer boundary of the network. This shifted relocation is implemented using proposed greedy advance rules (driving sensors to move inward toward the POI). Because one sensor on the outer boundary moves inside, the other sensors along the boundary may have to rotate a full cycle around the POI according to the rotation rules. The proposed GRG algorithm guarantees optimal hexagonal coverage radius, and near optimal circular coverage radius under the following three assumptions: (a) $r_c \geq \sqrt{3}r_s$; (b) sensors know their own spatial coordinates by GPS devices or any effective localization algorithm; (c) through lower-layer protocols (with possible minor modifications), sensors have the information about their 1-hop neighbors, i.e., location, moving status, and movement destination (if moving).

IV. Robot/Actuator Assisted Sensor Relocation

Mobile sensors used to perform sensor replacement assume global coordinates and location-awareness. This is not cost-effective. In robot and actuator assisted sensor relocation algorithms, the number of robots or actuators is much smaller than the number of sensors. Thus, the cost is expected to be lower than requiring most of sensors to have mobility. Furthermore, in this case, since sensors do not move, the routing algorithm can also be more efficient in delivering sensing data than with mobile sensors. However, except for the very recent papers by Falcon et al. [3] and Fletcher et all [4], how to relocate sensor nodes using robots or actuators remains largely unexplored.

The authors of [11] propose using only a few robots to assist in sensing hole repair/maintain coverage. All robots are mobile and can pick, carry, and unload sensor nodes. When a node fails, a selected robot moves to the location of the failed node and replaces it with a spare sensor it carries. Thus robots do not repair sensing holes using redundant sensors already deployed in the environment. This paper presents three different robot coordination algorithms: a centralized manager algorithm, a fixed distributed manager algorithm, and a dynamic distributed manager algorithm. In all three algorithms, a *manager* is a robot that receives failure reports and determines which robot is to handle a specific failure. In contrast, a *maintainer* is a robot that moves and replaces failed nodes [11]. The centralized algorithm has a central manager: a single robot that a) stays at the center of the sensing area, b) receives failure reports from sensors and c) forwards these reports to all the other robots (the maintainers). This algorithm is efficient in reducing the distance to travel and thus the motion energy of robots because the manager always selects the robot closest to a failure. But, as with all centralized algorithms, the central manager can become a bottleneck, which may considerably hinder overall performance. In the two distributed algorithms of [11], the management responsibility is distributed over the robots, and each robot functions as both a manager and a maintainer. In the fixed algorithm, the sensor area is equally divided into fixed subareas, each of which is handled by a robot independently. Two types of area partition methods are considered: squares or hexagons. The fixed algorithm does avoid the single manager bottleneck problem. However, the controlled flooding messages and possible longer traveling distance (due to each robot having only local knowledge) introduce factors that hinder performance. In the dynamic algorithm, the sensor area is dynamically divided in to Voronoi graphs based on the robots' current locations. The authors compare the three algorithms with respect to motion overhead and messaging overhead. They observe that in the dynamic algorithm, since a sensor node reports a failure to the closest robot, the robot achieves similar traveling distance as in the centralized algorithm without suffering the scalability problem of the fixed algorithm. But this is achieved at the cost of high messaging overhead, as the new location of a moving robot needs to be broadcasted to many sensors. Simulations of the three algorithms were performed and the experimental results show that: a) the centralized and the dynamic algorithm can achieve lower motion (travel distance) overhead and b) both the fixed and the dynamic algorithm are more scalable but also have higher messaging overhead.

In [11], when a sensing hole occurs (due to node failure), mobile robots are assumed to have spare sensors available at all time in order to fill such a holes. In contrast, in [4], robots improve existing network coverage by transferring redundant sensors to the sensing hole positions. Two algorithms that use such a strategy are presented in that paper: algorithm R3S2 and Grid-Based R3S2 (G-R3S2).

In Algorithm R3S2, each robot can be in one of three states: *discovery*, *free* or *busy*. In the discovery state, a robot remains static and periodically transmits a beacon message carrying its current location. Upon receiving this message, the nearby active sensors reply with the location of their adjacent sensing holes and also of any passive (i.e., spare or redundant) sensor they proxy. If there is no sensing hole or passive sensor reported to it, a robot is free and will travel within the region of interest autonomously and asynchronously at random. Otherwise, this robot becomes busy and moves to either pick up a passive sensor or fix a reported sensing hole. A robot always chooses to fill a sensing hole first if it has spare sensors at hand. If there is no hole to fill and it has no spare sensor at hand, it chooses to pick up the closest spare sensor. When there is no hole to fill and its has $S(S \neq 0)$ spare sensors at hand, this robot continues random movement with probability $p = 1 - 1/|S|$. Each time a robot picks up a passive sensor, it informs the corresponding active sensor to remove the passive one from its advertisement list. In the case of two robots competing for the same task, the following competition rules apply: a) if competing for picking up a passive sensor, the robot carrying the smaller number of passive sensors wins; b) if competing for repairing a sensing hole, the robot with the larger number of passive sensors wins; c) if competing for a hole and sensor pair, the robot with the shortest travel distance wins. Note however that none of these rules deal with the situation when two (or more) robots with equal number of passive sensors meet.

G-R3S2 is the implementation of R3S2 on a virtual grid. Contrary to R3S2 in which robots travel at random if free, in G-R3S2 the movement of a free robot is restricted to a grid. During initialization and after performing assigned tasks, robots spontaneously align themselves with the grid by moving to the closest grid point. Beyond this difference, G-R3S2 further restricts a robot's movement by using a *least frequently visited policy*. Namely, each active node keeps track of the number of visits it had. Upon receiving a beacon message from a robot, an active node sends back the number of visits beyond its adjacent sensing hole(s) or passive sensor(s). Each free robot always chooses to perform the tasks reported by the active sensor with the smallest number of visits.

Simulations performed measure three metrics: average number of moves, average moving distance and average number of bits sent. Results show that G-R3S2 improves R3S2 across all these three performance criteria.

Garetto et al. [5] present an even-based relocation protocol using actuators. It does not deal with the fine-grained relocation problem (which is concerned with repairing, for example, a coverage hole created by a node failure) considered in all the other papers of this survey. Instead, that paper focuses on even-based relocation, where the node position and density have to be adapted to properly sample and control a large-scale event.

Last, in [3] the authors address the *carrier-based coverage repair problem* (CBCRP) in a wireless sensor network, which fits into the relocation problem considered in this survey. Contrary to most of the sensor relocation solutions, the algorithm presented in this paper allows "off line" computation, that is, the optimal solution can be computed before the execution of the real algorithm in the base station. This robot-assisted coverage repair problem is modeled as a new variant of the Traveling Salesman Problem or Vehicle Routing Problem, depending on the number of robots used. The algorithm assumes that mobile robots (which are able to carry a limited number of sensors) are located at a base station. The goal of the algorithm is to collect passive sensors all over the network and drop them into the sensing holes with a minimum cost tour robot trajectory so that network coverage can be repaired by replacing all damaged sensors with spare (passive) ones. In algorithm 1-TSP-SELPD, every passive sensor periodically reports its location to the base station, whereas active sensors report the coordinates of any adjacent sensing hole to the base station. Each robot has a fixed carrying capacity and leaves from the base station. In order to address the CBCRP, the authors first introduce a new combinatorial optimization problem, the one-commodity traveling salesman problem with selective pickup and delivery (1-TSP-SELPD). They then explain how they model the CBCRP as a 1-TSP-SELPD with unitary pickup and delivery and solve it using an ant colony optimization (ACO) meta-heuristic. Furthermore, they propose six heuristic functions (that address diverse optimization criteria) to guide the search undertaken by the Max-Min Ant system. Also, a two-step ACO is proposed in order to accelerate convergence in dense networks. Finally, the authors claim that simulations using MATLAB suggest that this later solution outperforms existing similar approaches (though no details are provided with respect to which heuristics were prioritized).

V. Analysis

In dealing with the sensor relocation problem, the following criteria are the ones to optimize:

1) shortest total traveling distance;
2) fast hole healing;
3) minimal message overhead and
4) energy efficiency.

Frequently-used metrics for the evaluation of these criteria include: average number of moves, average total travel distance, and average number of bits sent. But several factors in fact affect these metrics:

- Number of robots/actuators, number of mobile sensors and number of static sensors used: typically, the more mobile sensors, robots and actuators there are, the smaller the total travel distance of an algorithm is, but the larger the energy consumed by mobility becomes.
- Type of communication protocols used: for example, in [11], in order to reduce the message overhead, controlled flooding is used instead of a basic flood protocol.
- Communication radius and Sensing radius: usually the bigger the communication or sensing radius is, the more energy it requires, but the bigger the communication radius is, the fewer number of hops each message needs to be relayed by other sensor nodes.
- Size and the partition of the environment.
- Usage of GPS or other localization equipment and algorithms.
- Restrictions on movements: e.g., algorithms implementing movements based on least visited region of

interest typically end up having better performance than algorithms using random mobile sensor or robot movements.

- Usage of proxy: it is known that it costs a mobile sensor about $30J$ to move one meter, but it only costs the mobile sensor about $0.1J$ to send a message. This means that it costs 300 times more energy to move one meter than to send one message. Simulation results from [13] show that proxy-based algorithms minimize the total travel distance compared to basic bidding protocols. Also, algorithms from [4], [7], [8], [13] show that using of proxies reduce the number of messages sent by mobile sensors or robots. Such observations indicate that proxy-based algorithms have low energy requirements.
- Use of a central manager/server: generally, when compared to distributed algorithms, centralized algorithms have lower message overhead (e.g., see [11]). But the risk of a messaging bottleneck and of a single point of failure is a drawback.

VI. Conclusion

An adequate level of sensor coverage is crucial for obtaining a proper phenomenon of interest and for the successful completion of the corresponding sensing tasks in a WSN. Sensor relocation solutions aim at maintaining the adequate sensor coverage level when the procedure for initial sensors deployment fails or when run time sensor failure occurs. In this paper, we review existing sensor relocation solutions proposed by different researchers. We identify the features of the algorithms put forth, categorize the latter with respect to these features, analyze common evaluation metrics as well as the factors that impact experimental results.

References

[1] I. F. Akyildiz, I. H. Kasimoglu, *Wireless Sensor and Actor Networks: Research Challenges*, $AdHocNetworksJournal(Elsevier)$, Vol. 2, No. 4, pp. 351-367, October 2004.

[2] B. Carbunar, A. Grama, J. Vitek, *Distributed and dynamic voronoi overlay maintenance for coverage detection and distributed hash tables in ad hoc networks*. IEEE International Conference on Parallel and Distributed Systems, pp. 549556, 2004.

[3] R. Falcon, X. Li, A. Nayak, and I. Stojmenovic. *The One-Commodity Traveling Salesman Problem with Selective Pickup and Delivery: an Ant Colony Approach*. IEEE Congress on Evolutionary Computation (CEC), pp. 4326-4333, 2010.

[4] G. Fletcher, X. Li, A. Nayak, and I. Stojmenovic. *Randomized Robot-assisted Relocation of Sensors for Coverage Repair in Wireless Sensor Networks*. The 72^{nd} IEEE Vehicular Technology Conference (VTC 2010-Fall). pp.to appear.

[5] M. Garetto, M. Gribaudo, C.-F. Chiasserini, and E. Leonardi. *A Distributed Sensor Relocation Scheme for Environmental Control*. The 4^{th} IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS), pp.1-10, 2007.

[6] Z. Jiang and J. Wu. *A Hierarchical Structure based Coverage Repair in Wireless Sensor Networks*. The 19^{th} IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), pp.291-296, 2008.

[7] X. Li and N. Santoro. *ZONER: A ZONE-based Sensor Relocation Protocol for Mobile Sensor Networks*. The 6^{th} IEEE International Workshop on Wireless Local Networks (WLN), pp. 923-930, 2006.

[8] X. Li, N. Santoro, and I. Stojmenovic. *Mesh-based Sensor Relo-cation for Coverage Maintenance in Mobile Sensor Networks*. The 4^{th} International Conference on Ubiquitous Intelligence and Computing (UIC), pp. 696708, 2007.

[9] X. Li, H. Frey, N. Santoro, and I. Stojmenovic. *Focused Coverage by Mobile Sensor Networks*. The 6^{th} IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS), pp. 466-475, 2009

[10] X. Li, A. Nayak, D. Simplot-Ryl, and I. Stojmenovic. *Sensor Placement in Sensor and Actuator Networks*. Wireless Sensor and Actuator Networks: Algorithms and Protocols for Scalable Coordination and Data Communication, Wiley, 2010.

[11] Y. Mei, C. Xian, S. Das, Y.C. Hu and Y.-H. Lu. *Sensor Replacement Using Mobile Robots*. $ComputerComm$, 30(13):2615-2626, 2007.

[12] A. Nayak and I. Stojmenovic, *Wireless Sensor and Actuator Networks: Algorithms and Protocols for Scalable Coordination and Data Communication*, John Wiley & Sons, 2010.

[13] G. Wang, G. Cao, and T. LaPorta. *Proxy-Based Sensor Deployment for Mobile Sensor Networks*. The1^{st} IEEE International Conference on MobileAd-hoc and Sensor Systems (MASS), pp. 493502, 2004.

[14] G. Wang, G. Cao, T. LaPorta, and W. Zhang. *Sensor Relocation in Mobile Sensor Networks*. The 24^{th} Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), pp. 23022312, 2005.