Localized Ant Colony of Robots for Redeployment in Wireless Sensor Networks

YUAN WANG¹, AHMED BARNAWI², RODRIGO FERNANDES DE MELLO³ AND IVAN STOJMENOVIC¹,²

¹EECS, University of Ottawa, Canada
E-mail: ywang269@uottawa.ca, stojmenovic@gmail.com
²King Abdulaziz University, Jeddah, Saudi Arabia
E-mail: ambarnawi@kau.edu.sa
³ICMC, University of São Paulo, Sao Carlos, Brazil
E-mail: mello@icmc.usp.br

Received: October 6, 2013. Accepted: February 1, 2014.

Sensor failures or oversupply in wireless sensor networks (WSNs), especially initial random deployment, create spare sensors (whose area is fully covered by other sensors) and sensing holes. We envision a team of robots to relocate sensors and improve their area coverage. Existing algorithms, including centralized ones and the only localized G-R3S2 [9], move only spare sensors and have limited improvement because non-spare sensors, with area coverage mostly overlapped by neighbour sensors, are not moved, and additional sensors are deployed to fill existing small holes. We propose a localized algorithm, called Localized Ant-based Sensor Relocation Algorithm with Greedy Walk (LASR-G), where each robot may carry at most one sensor and makes decision that depends only on locally detected information. In LASR-G, each robot calculates corresponding pickup or dropping probability, and relocates sensor with currently low coverage contribution to another location where sensing hole would be significantly reduced. The basic algorithm optimizes only area coverage, while modified algorithm includes also the cost of robot movement. We compare LASR-G with G-R3S2, and examine both single robot and multi robots scenarios. The simulation results show the advantages of LASR-G over G-R3S2.

Keywords: Wireless sensor network; sensor relocation; ant-based algorithm; robot-assisted algorithm

1 INTRODUCTION

A wireless sensor network (WSN) is a collection of sensing devices to monitor real-time environmental conditions in a region of interest (ROI). WSNs
gained attention after the development of Micro-Electro-Mechanical Systems technology (MEMS), which contributed to the high-tech design of smart sensors [22]. Sensor nodes communicate with each other using wireless links, to facilitate distributed sensing tasks. Sensors are expected to cover the whole ROI, thus providing a quality-guaranteed monitoring service. Initial deterministic deployment [10, 13] may achieve optimal area coverage, but could be costly for actuators deploying them, and holes may be generated over time from sensor failures. Initial random sensor node dropping is often preferred, but creates immediately spare sensors and sensing holes.

To improve the area coverage for WSN, methods for relocation of sensor nodes were introduced recently. Relocation can be beneficial for both improving the coverage by recovering sensing holes within ROI after initial random deployment [9], and replacing failed sensor nodes [5,11,21]. Sensors can be static or mobile. Mobile sensors are able to self deploy [11], while one or more robots must be placed to carry static sensors from original locations to more “comfortable” places [2, 8]. Mobile sensors are costly and energy consuming. In this paper, we focus on using mobile robots and static sensors for coverage enhancement with our algorithm, which is inspired by swarm intelligence methods.

Existing (centralized and localized) robot-assisted sensor relocation algorithms replace only fully redundant sensors. This paper is the first to relocate both spare sensors and some active sensors for coverage improvement. Further, we achieve it in a localized way.

Sensor whose coverage area is fully overlapped by neighbouring sensors normally labels itself as spare sensor and turns to “sleep” mode to save energy [23]. In some existing robot-assisted sensor deployment schemes [5, 6, 9], robots detect spare sensors and sensing holes and move spare sensors to cover sensing holes. If robots do not detect spare sensors in their vicinity, they will move to the base station to collect spare sensors from there. There is a tradeoff between number of sensors deployed in the fields, their overall energy consumption, and overall area covered. Except for storing extra spare sensors, a base station is also used to grasp global topological information within ROI in centralized algorithms [5, 6], which results in heavy computational cost to gather information towards base station.

In [9], coverage is optimized by Randomized Robot-assisted Relocation of Static Sensors algorithm; we elaborate only on the best grid-based variant (G-R3S2). In G-R3S2, sensors check coverage area, label themselves as spare sensors if fully covered by other sensors and inform neighbour proxy sensors. Each sensor identifies uncovered arcs on its sensing range perimeter. Uncovered arcs are part of boundaries of polygon-shaped sensing holes. Coverage is improved by robots relocating spare sensors to cover nearby sensing holes. Robot random movement is restricted by walking on virtual grid and applying
Least Recently Visited policy (LRV) [2], which advises robot to move to a least recently visited adjacent grid point. To the best of our knowledge, this is the only known localized algorithm for sensor redeployment. Its main drawback, that motivates this paper, is that only spare sensors are relocated. There exist sensors with little contribution to the coverage, which could be moved to a better location. We also replace deterministic decision making with probabilistic one.

In this paper, we present a new robot-assisted approach to address sensor replacement problem, named Localized Ant-based Sensor Relocation with Greedy Walk (LASR-G). LASR-G aims at spreading sensors for coverage improvement. This approach is inspired by ant-based algorithms for clustering unlabelled datasets [3, 20]. The goal of data clustering algorithms is to group analogous items by placing similar items in their original space of attributes in neighbour regions on a two-dimensional output grid. Contrary to clustering goal, our goal in LASR-G is to spread items (sensors) and balance the density of each subarea within ROI. LASR-G is the first localized algorithm that relocates active sensors (in addition to spare ones).

Robots can pick up (at most one) spare or active sensors under certain criteria, which favors spare sensors. They calculate corresponding picking up or dropping probability and make decisions based on local information before taking a locally best step. Unlike centralized algorithms, LASR-G algorithm has low message overhead since no network wide information is needed by sensors or robots. It aims to prolong sensor’s lifetime and increase sensor area coverage. Our simulation compares its performance with G-R3S2 algorithm [9] in single-robot and multi-robot scenarios, and calculates the coverage percentage and average traveling distance for robots.

The paper is organized as follows. Section 2 provides a brief review of the related work about sensor relocation and ant clustering protocols. Then we introduce two versions of localized robot-assisted sensor relocation algorithms and discuss about novelty in Section 3. Our simulation analysis is presented in Section 4 with the comparison to a competing solution algorithm, followed by conclusion in Section 5.

2 RELATED WORK

2.1 Sensor self-deployment algorithms

Several recent biologically-inspired (particle swarm optimization, genetic, ant colony) algorithms [12, 14, 18, 19] assume that sensors are mobile, and able to search and relocate themselves based on information available on area coverage. These algorithms are centralized and cannot be easily converted to a distributed solution. The main issue is that such solutions require global information on the network status, or local information that does not exist in
the field. For example, several ant colony-based algorithms assume that sensors leave pheromone traces in the field in places where no sensor exists, and finds traces from other sensors on some edges between virtual nodes. Swarm intelligence was applied to other problems in sensor networks, such as intrusion detection [7]. In MSRP [11], mobile sensors move towards destination with guidance of information mesh (iMesh) constructed by messages initiated from spare sensors and relayed in north-south and east-west directions. Sensors near holes detect nearby mobile spare sensor with the help of iMesh. This article has references to other decentralized solutions.

2.2 Coverage Repair in WSN by robots

Three robot coordination protocols: centralized manager, fixed distributed area division, and dynamic distributed area division algorithms, are presented in [16], for replacing failed sensors with spare ones. A central manager robot is needed, with rest of the robots as maintainers, in centralized manager algorithm. Sensors report neighbour node failure to central manager. The manager dispatches maintainers to replace failed sensors with spare ones they are carrying. In fixed and dynamic algorithms, ROI is divided into sub-regions, each guarded by a specific robot. All three algorithms assume robots carrying enough spare sensors during recovery process.

Falcon et al. [5] proposed a centralized algorithm 1-TSP-SELPD for replacing reported failed sensors with spare ones in ROI. NP-hard traveling salesman problem is applied to find a global best picking up and dropping route. Robot carrying capacity is a user-defined constant which has an impact on robot picking and dropping decisions.

Li et al. [9] proposed a family of localized robot-assisted sensor relocation algorithms including R3S2, G-R3S2, C-R3S2 and C-G-R3S2. Each robot can carry multiple sensors at a time, with infinite capacity. Robots move within the network to discover sensing holes and redundant (spare) sensors by local communication and relocate the discovered redundant sensors to the encountered sensing hole locations. In R3S2, robots move at random. Spare sensors are detected by neighboring proxy sensors, while sensing holes are reported by sensors which have one or more uncovered sensing perimeter arcs. In G-R3S2, robots random movement is restricted on a virtual grid, and Least Frequently Visited (LFV) [2] policy is adopted, to advice robot to move toward least recently visited adjacent grid points for spare sensor and sensing hole detection. If only spare sensors are locally detected, picking probability is inversely proportional to number of sensors the robot is currently carrying. G-R3S2 leads to a balanced traversing within ROI even without a global view. C-R3S2 and C-G-R3S2 are respectively the variants of R3S2 and G-R3S2 where clustering [1] and virtual force techniques are adopted to merge local sensing holes. However, in all variants, recovering process relies only on
the redeployment of fully redundant (spare) sensors. Therefore more sensors should be involved at initial state for a satisfactory coverage enhancement, especially for small holes. For example, a sensor with area partially covered by neighboring sensors, and adjacent to a small hole, could potentially be relocated just a small distance away to completely eliminate that hole, while not creating new one, or allowing new small hole to be in turn covered by other sensor in a similar move.

2.3 Ant-based Data Clustering Algorithms
Our algorithm is developed from the ant-based data clustering algorithms [3, 20], which are inspired by the “algorithm” of how real ants clean their nests and organize dead bodies in their colonies. In data clustering algorithms, ant-based agents are set into the grid for picking up and dropping data objects according to some clustering principles. The goal is to group similar items in their attributes in neighbour regions of a two-dimensional output grid.

A²CA [20] was proposed with modifications made to the standard ant-clustering algorithm [15]. By applying A²CA, datasets are classified and clusters are formed with similar items grouped together. Ant-based agents pick up or drop data objects based on corresponding probabilities. The probability of picking up an object increases with low density and similarity in neighbourhoods and decreases vice versa. Objects will be dropped if high density of similar objects are detected in its vicinity at a location. Cooling schedule, progressive vision field and pheromone releasing and detecting are applied as modification to converge into a more robust clustering solution. With cooling schedule, threshold constant of picking probability will decrease geometrically after a preset cycle, leading to a slight decline on picking up probability, so that agents will focus on moving objects with larger dissimilarities to neighbours in later stages.

Chen et al. [3] presented an artificial Ant Sleeping Model (ASM) and an ant algorithm for clustering analysis (A⁴C), which are based on behaviors of gregarious ant colonies. Ants, representing data objects, tend to turn to sleeping state when surrounded by similar objects and active state when feeling “unconformable” about neighboring area. By ants self-moving to a more “comfortable” place to sleep, different subgroups are formed and data objects are clustered adaptively at end.

3 ANT-BASED RELOCATION PROTOCOL DESCRIPTION

We design a Localized Ant-based Sensor Relocation Algorithm with Greedy Walk (LASR-G) which can increase sensor coverage in ROI after random deployment even without adding new sensors. We propose two algorithms based on LASR-G, focusing on different optimization criteria in relocation
process. The first algorithm, LASR-G1, is suitable if we only consider the coverage improvement, and allow a robot to travel longer distances to pick up a sensor. LASR-G2 attempts to optimize both coverage improvement and robot traveling distance. The main difference between two proposed algorithms is which of neighbor sensors to select to pickup, after an unloaded robot already decided to target one for pickup.

3.1 Assumptions and Definitions
A WSN is randomly deployed in the ROI, with boundary information known to all nodes (including robots and sensors). All the deployed $n$ sensors $S = \{s_0, s_1, \ldots, s_{n-1}\}$ are static. A fixed number of $m$ robots (ants) $A = \{a_0, a_1, \ldots, a_{m-1}\}$ are moving in ROI, for replacing sensors. In each iteration, robot can pick up or drop a sensor. The maximum number of sensors a robot can carry, $N$, is same and limited for each robot. Here we assume $N = 1$, thus once a robot picks up a sensor, it reaches full load and has to drop it before picking up another sensor. We assume that robots always have enough remaining energy to complete relocation process.

We assume physical properties (e.g. sensing radius, battery capacity, computational ability, message storage, etc.) for all the sensors are same. All robots (in multi robots scenarios) are also identical. We define sensing radius for every sensor as $r_s$, sensor’s communication radius $r_c$ and robot’s communication radius $R_c$, with $r_c \leq R_c$. Our algorithms use $r_c$ as parameter, and do not use $R_c$ anywhere, since robot is unable to learn about area coverage from sensors at distance greater than $r_c$, and communication among robots was not explored for improving overall task assignments.

For simplicity, we assume that sensing and communication ranges are a circle-shaped areas, centered at sensor or robot, respectively. Nodes communicate with each other by periodically sending and replying “hello” messages. Each message contains the unique ID and location of the sender. If a message reply was received by a node, it stores sender’s information and regards sender as a neighbour.

By communicating with neighbouring sensors, a sensor can decide whether its sensing range is fully covered by other active sensors, and consequently classify itself as a spare sensor. If so, it turns to “sleeping” mode to save energy. In “sleeping” mode, a sensor no longer monitors surrounding area within sensing range. However it still periodically keeps communicating with neighbour sensors in time synchronized manner. If it detects neighbour sensor failure or removal, it recalculates its sensing area, and turns active if it is no longer fully covered by other sensors.

3.2 Algorithm LASR-G1
Robots travel within ROI to detect sensor density and overlapping of area coverage. Moving routes and coverage improvement for robot $a$ are
Illustrated in Figure 1. Robot \( a \) picks up a nearby sensor \( s_8 \) whose sensing range is mostly overlapped with other sensors, and drops it at a hole nearby located at point \( d \). Before dropping, it makes several steps, each in selected direction, and each step has length (distance from previous location) \( l \). We assume \( l < r_s \).

We describe first our solution that only considers coverage improvement as optimization criterion, without involving travelled distance by robot. Let \( S' \) represent neighbour sensor set, that is, the set of sensors whose distance to robot \( a \) is less than sensor’s communication radius \( r_c \). Let \( \forall s' \in S' \), robot \( a \) calculates coverage ratio of area centered at current location of \( s' \) (denoted as \( d' \)), shown in Equation (1):

\[
f(s', d) = \frac{Q(s', d)}{Q_t}.
\]

\( Q(s', d) \) is the area currently only monitored by sensor \( s' \) at its location \( d \), which also equals to the possible newly created sensing hole area if \( s' \) is picked by robot \( a \). \( Q_t \) is the total sensing area of \( s' \) (by our homogeneity assumption, it is a same for all sensors). This coverage ratio can be calculated by dividing ROI into small grids by rows and columns, and measuring the corresponding area by counting the points within that area. The same equation can be also used to find the coverage ratio for newly covered area,
if we relocate $s'$ and place it at location $d$ (in both cases, we assume that previous and new coverage areas for $s'$ do not overlap).

By calculating and comparing coverage ratio for every $s' \in S'$, robot $a$ is able to find a neighbour sensor $st$ with minimal coverage ratio value. If more than one sensor ties on this value, robot chooses the nearest one.

Based on coverage ratio value in Equation (1), the probabilities of robot $a$ picking up and dropping decisions are given as Equations (2) and (3), respectively:

$$Pp(a, d) = f(st, dt)^{ep},$$

$$Pd(a, d) = f(sl, d)^{ed},$$

where $d$ is the current location of robot $a$, $st$ is the targeted neighbour sensor with minimal coverage ratio, $dt$ is the location of $st$, $sl$ is the currently loaded sensor of $a$. When a loaded robot $a$ arrives at a new location $d$, it calculates dropping probability before taking any action. If robot decides to drop sensor $sl$, it leaves $sl$ directly at its current location.

Parameters $ep$ and $ed$ control picking up and dropping probabilities respectively. They follow a modification schedule: for each cycle (e.g. 100 steps), the value of $ep$ increases and $ed$ decreases by a geometric progression scheme controlled by constants $vp$ and $vd$ in Equations (4) and (5), until it meets the preset stopping criterion.

$$ep \leftarrow ep \times vp,$$

$$ed \leftarrow ed \times vd.$$
action is finished, robot takes a greedy step to a more “comfortable” place to repeat the tasks. However, if picking up or dropping task is not successful, robot keeps same load status and walks to search for other suitable place. Both greedy walk and random walk may be taken, where random walk is selected whenever a preset number of continuous greedy steps $t$ is reached. (that is, whenever robot keeps walking $t$ continuous greedy steps with step length $l$ with same load status).

With or without carrying sensor, robots needs to decide where to move to. The set of neighbouring sensors $|S'|$ dynamically changes. Greedy movements focus on the selection among four directions: north, south, east and west. Based on local information, robot $a$ estimates fitness function values at four points $dn$, $ds$, $de$ and $dw$, respectively, each at distance $l$ from current location $d$. Fitness function is either pickup $Pp(a, dx)$ or dropoff $Pd(a, dx)$ probability, where $dx$ is one of $de$, $dw$, $dn$, and $ds$. As shown in Figure 2, robot at current location $d$ has information of sensor $s_1$ to $s_5$. It is not able to accurately measure fitness function at $de$ because it is not aware of neighboring sensor $s_6$ of $de$. However it estimates that $s_1$ is not a neighbor of $de$, and calculates fitness function based on estimated neighborhood $s_2$ to $s_5$ (sensors in the intersection of two sensor communication ranges, drawn in darker color) of $de$. Robot compares fitness values at four selected candidate points, and chooses one to move to, and repeats this calculation and move selection from the new location.

An example is given in Figure 1 for an overall introduction of LASR-G. Assuming robot $a$ is unloaded at location $d0$, fitness function $g(a, d0)$ and
picking up probability $P_p(a, d0)$ are calculated based on neighbour sensors from $s_1$ to $s_8$. If picking decision is made, $a$ chooses $s_8$ as targeted sensor and walks straight towards $s_8$ to pick it up. Then $a$ chooses a local best step to walk away. Here it compares estimated fitness function value for four directions and chooses east-most direction for the first step. After calculating dropping probability at current location, robot decides to continue taking greedy walk with sensor loaded. When it arrives at location $d$, robot decides to leave loaded sensor at $d$ based on corresponding dropping probability at $d$. Then unloaded robot $a$ will switch to consider candidate pickup probabilities.

Applying greedy method may cause local loops and rotation in robot movements if robot keeps the same load status for several steps. This could lead robot to stuck in a small part of subregion, unable to find an appropriate place to change load status. To expand its “view”, robot will switch to random move with a larger step length $q \times l$ if and only if a preset number of steps $t$ is reached without any change of robot’s load status, as illustrated in Figure 3. During random walk, robot will move in random direction along a straight line without stopping. It will switch back to greedy walk afterwards. Robot keeps switching between greedy walk and random walk until it finds a “comfortable” place to change load status. Then robot starts greedy walk again.

3.3 Algorithm LASR-G2
While LASR-G1 only considers coverage improvement as the optimization criterion, LASR-G2 looks at the trade off between added coverage and robot traveling distance. At each step, robot collects information from all
neighbouring sensors during a time window. Let $S'$ be the set of neighbouring sensors, at distance at most $rc$ from current location $d$, excluding sensor $sl$ that robot $a$ is carrying (if loaded). Contribution on robot decision is calculated for $\forall s \in S'$. Fitness function of robot $a$ at location $d$ is given in Equation (6):

$$g(a, d) = C \cdot \sqrt{\ln|S'| \cdot e^{|S''|} \cdot \sum_{s \in S'} e^{-\frac{|d-D(s)|^2}{2\sigma^2}}}.$$  

Equation (6)

$C$ is a constant to be optimized, $|S'|$ is the number of sensors in set $S'$. $S''$ represents the set of spare sensors within communication range $rc$, $S'' \subset S'$. $|S''|$ is the number of elements in set $S''$. $D(s)$ is the location of sensor $s$. $|d-D(s)|$ represents the distance between robot $a$ and sensor $s$, $\sigma^2$ stands for the variance of distribution of sensors to the robot. Influence of sensors outside communication range on fitness function can be ignored.

In Equation (6), we use Gaussian activation function to express the relation between distribution of neighbour sensors and impact of robot picking up decisions. Sensors closer to robot (the centre of communication range) have more impact on robot picking decisions than those near communication range perimeter.

Based on fitness function value in Equation (6), the probabilities of robot $a$ picking up and dropping a sensor at $d$ are given as Equations (7) and (3), respectively:

$$Pp(a, d) = \left(\frac{g(a, d)}{1 + g(a, d)}\right)^{ep}.$$  

Equation (7)

Thus dropping probability calculations for LASR-G1 and LASR-G2 are the same. The difference between the two algorithms is on how robot derives picking up probability and makes picking decisions. Unlike LASR-G1 which simply chooses to pick up a targeted sensor with minimal coverage ratio, picking up probability in LASR-G2 is obtained by balancing coverage improvement and robot traveling distance. Similar to LASR-G1, parameters $ep$ and $ed$ in LASR-G2 also follow a modification schedule referred in Equation (4) and (5).

In LASR-G2, sensors near communication range have limited contribution to value of fitness function (picking up probability). Since robot $a$ cannot get information outside its current communication range, it will estimate $g(a, de)$ by information in the overlap area of communication ranges for current and east-most location (dark pink area in Figure 2). Therefore, when applying Equation (6) for $g(a, de)$, $S'$ and $S''$ are only obtained from the overlapping
region. After estimating fitness values for four directions $g(a, dn)$, $g(a, ds)$, $g(a, de)$ and $g(a, dw)$, robot will move to a locally best direction.

It remains to show how robot decides which sensor to relocate, if the decision to pick up a sensor is made. Robot $a$ detects neighbouring spare sensors each time it walks to a new area and remembers them. Let $V$ represent spare sensors set. Robot updates set $V$ whenever it learns that some spare sensors were moved, or more spare sensors are detected. $\forall s' \in S', V$, a unique value $Z(s', d, d')$ is obtained. Since we focus on both improving coverage and minimizing robot traveling distance, Equation (8) includes both factors:

$$Z(s', d, d') = \left(1 - \frac{Q(s', d')}{Qt}\right) \cdot e^{-\frac{|d-d'|^2}{2\delta^2}},$$  

(8)

where $d$ is current location of the unloaded robot $a$, $d'$ is location of sensor $s'$. Robot chooses the sensor $s'$ which has maximal value $Z(s', d, d')$ as targeted sensor $st$. If selected $st$ is an element of set $V$, it is deleted from $V$ when picked by robot. Note that other robots may not learn that $st$ has been moved away, after it entered their $V$ set. This may cause their wrong choices, visits to places where spare sensors used to be, and selecting alternative sensors for moving, with additional movements.

4 EXPERIMENTAL VALIDATION

We have conducted experiments to test the feasibility of our Localized Ant-based Sensor Relocation Algorithms with Greedy Walk (LASR-G1 and LASR-G2), and compare the proposed algorithm with Grid-based Randomized Robot-assisted Relocation of Static Sensors (G-R3S2) [9]. We analyse the performance for both algorithms based on three metrics. TD metrics measures traveling distance, while CP and MCP measure coverage percentage (in two different ways). In simulation, we terminate G-R3S2 algorithm when no spare sensor exists or its coverage reaches 0.99.

1. Traveling Distance (TD): For single robot, it refers to the total traveling distance until robot movement terminates. For multi robots, it indicates average traveling distance of all robots.

2. Coverage Percentage (CP): The percentage of area which is monitored by at least one sensor over the given ROI, when robot movement terminates. CP is measured when traveling distance (TD) for LASR-1 and LASR-2 reach TD value of G-R3S2 when it stops (achieving the best coverage G-R3S2 can get). CP is used to for compare the coverage among algorithms when each robot walks same distance.
3. Maximal Coverage Percentage (MCP): The maximal percentage of area which is monitored by at least one sensor over given ROI. MCP is measured when increase of coverage value is less than 0.005 within 500 steps. MCP shows the maximal coverage percentage each algorithm can reach regardless of TD value for each robot.

4.1 Simulation Initialization

We generate the simulation of our work in Java and execute on an AMD Athlon 7450 Dual-Core Processor at 2.40GHz with 2.75GB of RAM under Windows XP. Sensors are initially randomly deployed within a rectangular 600 × 600 ROI. The numbers of sensors \( n \) and robots \( m \) are preset by users. Here we set \( rs = 25, rc = 50 \) and \( Rc = 100 \). Robot step length \( l \) is set to 10. Parameter values for this experiment are set as follows: \( \sigma = 12, \delta = 8, C = 0.6, q = 8, t = 5 \). The initial value of \( ep, ed, vp \) and \( vd \) are 1.50, 1.15, 1.005 and 0.99 respectively; these values recurse as shown in Equations (4) and (5) every 200 steps.

To simplify the analysis, we do not consider run-time sensor node failure during relocation process in our simulation. We will compare the performance by modifying the number of sensors \( n \) and robots \( m \), and compare coverage improvement and average traveling distance for each robot during different relocation periods.

4.2 Simulation Analysis

Number of Robots

We will compare LASR-G1, LASR-G2 and G-R3S2 algorithms over the measured aspects, in single and multi robot scenarios. Within the tested ROI, 350 sensors are initially randomly deployed. Figures 4(a), (b) and (c) illustrate the influence of \( m \) (number of robots) on CP, MCP and TD values for LASR-G1, LASR-G2 and G-R3S2. To simplify the simulation, we stop G-R3S2 when no more spare sensor exists within ROI. Stopping criteria for LASR-G1 and LASR-G2 are set differently for Figure 4(a), (b) and (c).

![FIGURE 4](image-url)

Impact of robot number over CP and TD. (a) and (b) are final CP with different stopping criteria; (c) shows robot TD for three algorithms.
Impact of number of robots $m$ on CP is presented in Figure 4(a). We evaluate CP under same average robot traveling distance. Both LASR-G1 and LASR-G2 stop when TD value reaches the one measured when G-R3S2 stops. With the increment of $m$, LASR-G2 rapidly increases on final CP, while value for LASR-G1 and G-R3S2 show slower increment. This result indicates that when taking same TD, LASR-G2 repairs WSN and improves CP faster than other two algorithms.

Figure 4(b) shows MCP values for three algorithms. LASR-G1 and LASR-G2 stop when coverage improvement is less than 0.005 within 500 steps. It shows LASR-G1 can reach maximum CP since it is designed to always pick sensors with most overlap in local area. LASR-G2 has lower value since it needs to balance TD and CP all the time when choosing targeted sensors.

Figure 4(c) compares TD among three algorithms. For fairness, LASR-G1 and LASR-G2 stop when corresponding CP values reach stopping value for G-R3S2. Average TD value for LASR-G2 is always shorter than other two for tested scenarios. With only spare sensors defined as targeted sensors, robots take longer TD to explore and pick up spare sensors in G-R3S2. It confirms that by reducing overlapping among sensing range, it can reach same coverage improvement with less TD for robot detecting and moving spare sensors.

Number of Sensors

We evaluate the impact of number of sensors $n$ on relocation performance for LASR-G1, LASR-G2 and G-R3S2. The simulation analysis is shown in Figure 5. We set the number of robots $m$ to 4. Figure 5(a) shows the relation between CP and number of sensors $n$. We compare CP value when three algorithms reach same TD value for each robot when G-R3S2 stops. LASR-G2 outperforms the other two when $n$ is limited. With the increment of $n$ ($n > 400$), number of spare sensors in tested region increases, leading to a better performance of CP for G-R3S2.
Figure 5(b) illustrates the relation between MCP value and number of sensors $n$ deployed in tested region. Both LASR-G1 and LASR-G2 stop when coverage improvement is less than 0.005 in 500 steps. With the increase of $n$, both the random initial coverage and MCP would rise. A stable trends in the curve for LASR-G1 and LASR-G2 indicate the performance is not highly related to $n$, comparing with G-R3S2 which presents an obvious increase of curve with more sensors involved in. When $n$ is limited (e.g., $n = 300$), the growth rate of either LASR-G1 or LASR-G2 on MCP is larger than of G-R3S2. Because of restricted number of spare sensors generated, performance of G-R3S2 is constrained. When $n$ reaches 400, MCP values for three algorithms are mostly same. Besides, LASR-G1 has higher MCP than LASR-G2, since the former one considers only coverage improvement while latter one chooses targeted sensors based on both CP and TD.

Figure 5(c) explains the relation between average TD for each robot and number of sensors $n$ when reaching same CP as G-R3S2 stops. We can observe that robots take less steps to reach same coverage in LASR-G2 than the other two, because it balances both CP and TD value on traveling within ROI. For scenarios with $n \geq 400$ in tested region, LASR-G1 and LASR-G2 take larger TD since most part is already well deployed after random node dropping and previous robot-assisted relocation, causing robot surrounding area in vicinity similar when applying greedy walk. In that case, it takes longer moving distance for robot to find a “comfortable” place for loaded sensor.

5 CONCLUSIONS

We study sensor relocation for coverage maintenance in WSN after initial random deployment. Inspired from Ant-Clustering methods, we proposed localized robot-assisted algorithms, LASR-G1 and LASR-G2, which focus on redeploying both spare sensor and some active sensors whose sensing area is significantly overlapped with neighbours. Robots make dropping and picking up decisions based on probabilities, which in turn depend on added or lost coverage area by sensor addition or removal. Greedy walk is applied to guide robots move to local best location in its vicinity. Longer random walk is applied to guide robots move to local best location in its vicinity. Longer random walk is applied to escape from local minima. Through extensive simulations, we validate our algorithms and compare them to the only existing localized solution.

In future work, we will consider robot and sensor energy consumption and node failure after certain time period. We will analyse scenarios with robot carrying $N > 1$ sensors. Robot decisions will then also depend on the number of loaded sensors. Further, schedules of modification parameters $v_p$
and $vd$ can be updated based on neighborhood and hole status. We assume robots have same physical properties in this paper, however one can expend our work using heterogeneous robots and sensors.

We assumed a simplified model of circular area coverage. Future work could consider probabilistic area coverage, using a model such as in [4]. We also assumed that robots do not collaborate among them directly. If robots could communicate to each other, they may exchange local information and their tasks, using, for example, auction-like method [17]. This is an interesting area for future extension of our algorithm.

ACKNOWLEDGEMENTS

This work was partially supported by NSERC Discovery grant and NSERC Collaborative Research Development Grants: “Robot-assisted deployment and maintenance of wireless sensor networks for area and event monitoring” (CRDPJ445199-12; 2013-2015).

REFERENCES


