Wireless Sensor Network Connectivity and Redundancy Repairing with Mobile Robots*

Alberto de San Bernabé Clemente, José Ramiro Martínez-De Dios, Carolina Regoli, and Aníbal Ollero

Robotics, Vision and Control Research Group, University of Seville, Spain
Camino de los Descubrimientos, 41092 Sevilla, Spain
{adesanbernabe,jdedios,aollero}@us.es,
carolina.regoli@gmail.com

Abstract. This chapter presents a method for repairing the connectivity and redundancy of a WSN using mobile robots. It comprises three mechanisms: diagnosis, connectivity repairing and redundancy repairing. During the diagnosis stage the robots survey the scenario learning all required information of the problem. The connectivity repairing mechanism, which takes place after the diagnosis stage, finds the best deployment locations to ensure that the WSN has 1-connectivity. The redundancy repairing mechanism finds the locations where the deployment of new nodes best improves fault tolerance to node failures. The proposed scheme does not use any parameters or assumptions since it acquires all the required information during the diagnosis stage. Besides, its solution is very close to the optimal solution –in all the experiments performed it differed in one node maximum– but requires only a fraction of the time required by the optimal method. The proposed method has been evaluated in simulations and has been validated in experiments carried out in the *CONET Robot-WSN Integrated Testbed*.

1 Introduction

Cooperation among mobile robots and Wireless Sensor Networks (WSN) enhances their individual performance, providing wide variety of complementarities. Robots mobility and their capability of carrying sensors and equipment are useful to enlarge the sensing range and accuracy of static WSN nodes and enlarge the communication ranges of static WSN nodes. Mobile robots have been proposed for WSN localization [3], WSN deployment [21] and WSN data retrieving [19], among others.

Recently, WSN repairing using robots has attracted significant interest. Some of these methods, such as [2,26], detect faults in the network such as coverage holes but do not propose a way to mitigate these faults. Many others focus on solving connectivity failures, see [22,27,10], and most of them propose mobile nodes as tools to repair network connectivity. However, the great majority have been tested only in simulations

* This work was supported by PLANET (European Commission FP7-257649-ICT-2009-5), CLEAR (DPI2011-28937-C02-01) and EC-SAFEMOBILE (European Commission ICT-2011-288082).

A. Koubâa and A. Khelil (eds.), *Cooperative Robots and Sensor Networks 2014*,
Studies in Computational Intelligence 554,
DOI: 10.1007/978-3-642-55029-4_9,

and only few, such as [6,15], were used in real experiments. Existing methods aim at repairing only connectivity disregarding the fact that WSN nodes are rather fragile and a connected network can be suddenly disconnected even if only one node fails.

This chapter presents mechanisms for integral WSN repairing. Their objective is not only to repair connectivity but also to ensure a certain level of redundancy. The method can be divided in three mechanisms: diagnosis, connectivity repairing and redundancy repairing. In the first one the mobile robots explore the scenario in order to identify the WSN status, topology and to estimate the location of the deployed nodes. In connectivity repairing, using this information, the method first computes the locations where new WSN nodes should be deployed in order to ensure that the connectivity of the WSN is repaired. Then, mobile robots are commanded to deploy WSN nodes at the computed locations. These new nodes are integrated in the pre-existing WSN. If the connectivity of the resulting WSN is confirmed as repaired, in the next stage redundancy repairing is applied. Redundancy repairing deploys new WSN nodes at specific locations until a certain level of redundancy is achieved.

The proposed scheme does not use parameters nor requires any assumptions on the WSN nor on the WSN nodes being used. It acquires all the required information during WSN diagnosis stage and only needs as input the desired level of WSN redundancy. The proposed method has been implemented in simulations and has also been experimentally demonstrated in the *CONET Robot-WSN Integrated Testbed* [12].

This chapter is organized as follows. Related work is discussed in Section 2. Section 3 provides a general description of the proposed method. The diagnosis, connectivity repairing and redundancy repairing stages are described in Sections 4-6, respectively. Validation experiments are presented in Section 7. Section 8 closes this chapter with the conclusions.

2 Related Work

Fragility of individual nodes has motivated the development of a good number of methods for WSN diagnosis and repairing. Some focus on the identification of faults and misbehaviors in the networks [2,8], whereas others aim to develop a model of the faults that can occur in WSNs [26]. Others focus on detecting specific types of failures in the network, mainly coverage holes, such as in [23] and [28].

There are a good number of works that propose methods to solve connectivity failures. Some of them study the placement of relay nodes to repair the connectivity, see e.g. [22], while others propose using mobile nodes than can be relocated in order to substitute a failing node or to join two disconnected parts of the network, see [27,10], among others. For example, DARPA LANDROIDS program [5] uses mobile radio relay platforms to improve communications connectivity in non-line-of-sight communications environments such as urban settings. Work [4] uses predictive reasoning to compute the relocation of the mobile nodes. In this work sensor nodes can predict the performance of the WSN in terms of coverage when a node moves in a given direction. Others compute locations where WSN nodes should be deployed in order to repair the connectivity while optimizing utilities such as coverage or energy consumption, see [16,9], among others.

Much research has been carried out based on theoretical analyses. Most of them have been tested only in simulations and their results in many cases show to be unrealistic

when experimented in real settings. Only a few works include experimental validation. In [6] a flying robot is employed to deploy nodes on the ground. Once on the ground, the nodes compute their connectivity map in a distributed way. If the network is disconnected, an algorithm determines waypoints for the helicopter to drop additional nodes. In one of the experiments performed in [15] a mobile node was employed to re-establish the link between static nodes.

However, most works focus on connectivity repairing and disregard redundancy, which is critical for WSN robustness in real applications. Several concepts for redundancy in WSN have been defined. Work [1] presented the concept of N-redundancy of a node and proposed: a method to compute N-redundancy, a technique to estimate the repair time and a repairing algorithm that minimizes the repair time. However, like the aforementioned works, its objective was only to repair the connectivity of isolated parts of the network and did not consider redundancy repairing. The concept of redundancy has also been used in works that study the density of deployments, such as [18,17], but these works did not address network repairing.

This chapter proposes a method for integral WSN repairing which involves connectivity and also redundancy repairing. The proposed method makes no assumptions except the approximate region where nodes are deployed. To the best of our knowledge, it is the first work that validates experimentally the use of robots for ensuring a certain level of redundancy.

3 General Description

Consider a scenario where a number of static WSN nodes have been deployed, but their exact position or the WSN topology are not known. This case is realistic even when the nodes have been manually deployed if we consider the inaccuracies in the GPS receiver. For instance, in an environmental monitoring application WSN nodes could have been thrown randomly from an Unmanned Aerial Vehicle (UAV) over the area of interest. We know the approximate area where nodes were deployed but not their exact location. The objective of the WSN is to collect data from each node in a Base Station (BS). However, the nodes will be located at unknown locations and eventually will be disconnected from the base. We want to devise a method that ensures a certain level of connectivity and redundancy among the deployed nodes. WSN repairing can be triggered after nodes deployment or on demand when the BS detects WSN malfunctioning. The approach selected is to deploy new nodes instead of re-locating existing nodes.

The proposed method employs the concepts of N-connectivity and N-redundancy. N-connectivity is a metric used to define the minimum number of nodes that need to be removed in order to partition a graph. If a graph has N-connectivity, it remains connected even if any N-1 nodes are removed. If a network has 0-connectivity it means that there are disconnected nodes.

N-redundancy, defined in [1], represents a local measurement of the goodness of the connectivity among the neighbors of each node. A node *i* is N-redundant if we need to remove at least N nodes neighbors of node *i*, besides removing node *i* itself, in order to break the connectivity between two neighbors of node *i*. 0-redundancy nodes are fragile points of the network: if node *i* fails the network becomes disconnected. We consider a

network robust if all its nodes have at least 2-redundancy. A WSN is N-redundant when all its nodes have N-redundancy or higher.

N-connectivity is a metric for the whole network while N-redundancy is a metric for each node. They offer interesting complementarities. N-connectivity naturally allows analyzing if the network is connected. However, being a global metric, it is not very useful to detect the fragile points of the network. For improving fault tolerance we prefer to use N-redundancy, which is a local metric and naturally finds the nodes in the network that are more critical. Knowing which are the fragile nodes is exploited by our method, which drives the search of the repairing locations using heuristics and reducing computer burden.

The proposed scheme deploys new WSN nodes such that the resulting WSN has at least 1-connectivity and N-redundancy, a desired level N of redundancy. The scheme uses three mechanisms:

- **Diagnosis**: One or several robots carrying onboard WSN nodes survey the scenario to learn the status of the WSN, including the locations of the deployed nodes and the WSN topology. It also learns a model of the distribution of Packet Reception Rate (PRR) with distance for the deployed static nodes.
- **Connectivity repairing**: This mechanism deploys WSN nodes to ensure that the resulting WSN has 1-connectivity. If the network contains nodes or clusters of nodes disconnected from the base station, the method computes the best locations where new nodes should be deployed in order to reduce the number of clusters in the network to one.
- **Redundancy repairing**: It takes place when the WSN has 1-connectivity. It iterates computing the locations where new nodes should be deployed in order to best improve the redundancy of the static nodes that have redundancy lower than N. The iterations continue until all the nodes in the network achieve the desired N-redundancy.

4 WSN Diagnosis with Mobile Robots

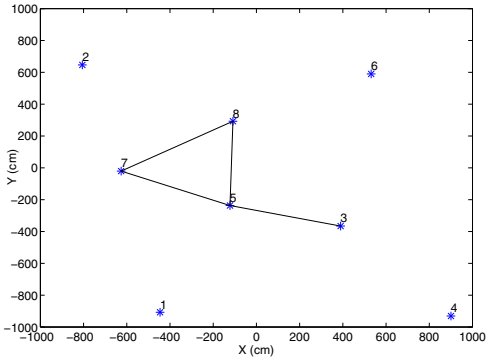
Assume we have available one or more mobile robots that carry one onboard WSN node. Each robot can measure its own location and can communicate with its onboard node using a bidirectional protocol that enables transmission of commands, requests and data. The scenario is discretized into cells and one or more mobile robots are commanded to survey the scenario passing over the centers of each of the cells. During the survey robots make measurements and ask static nodes. As a result, each robot computes the location of the nodes it has discovered, the topology of the static WSN and gathers data to compute a PRR-range model of the static nodes.

During the survey when a robot is close to the center of a cell it commands its onboard node to broadcast a beacon packet. Static nodes that receive the beacon packet respond by transmitting a response packet. When the onboard node receives the packet it measures its Radio Signal Strength Indication (RSSI). The location of each node can be determined from these RSSI measurements. A number of RSSI-based localization methods have been developed including range-based methods, such as multilateration

[24], or range-free methods, such as ROC-RSSI [14], which rely on geometric considerations, or fingerprinting methods, such as [11], which compare measurements with a previously obtained RSSI map. In our case, the location of a node is simply assigned to the cell with the highest RSSI value.

This survey also allows obtaining the topology of the WSN. Every static node dynamically learns which are its neighbor nodes by a simple protocol that periodically broadcasts packets among static nodes. We consider that node *j* is neighbor of node *i*, i.e. *i* and *j* are connected, if node *i* has received more than a certain percentage of responses from node *j* over the last *m* broadcasting periods. Thus, when a static node *i* responses to the robot node, the packet contains the ID of the sender and also of the sender neighbors. Thus, each robot can easily build the local connectivity matrix *LCM* of the static nodes in the area it surveyed.

Each robot transmits its *LCM* to the Base Station, which joins them all. A global connectivity matrix *GCM* of size *n x n* is generated, being *n* the number of nodes discovered. If entry *GCM(i,j)="1"* it means node *j* and *i* are neighbors. Figure 1 shows a simple example and its global connectivity matrix. Neighbor nodes are depicted with a line in the figure.



Global Connectivity Matrix
GCM

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

Fig. 1. Random deployment and its global connectivity matrix *GCM* in a simple simulation

The Base node connected to the BS is also one of the static nodes discovered. *GCM* contains the topology of the network and allows us to know if all the nodes are connected, and the network has at least 1-connectivity, or instead there are isolated parts of the network and therefore is divided in two or more clusters. Algorithm 1 shows a simple method that computes a cluster matrix *C* that represents the groups of connected nodes in the WSN topology. Along this paper we will use the term cluster when referring to isolated groups of connected WSN nodes.

C is an *n x n* matrix which entries can be either "1" or "0". If *C(i,j)*="1" it means that node *j* belongs to cluster *i*. Thus, only rows in *C* with at least one entry with value "1" represent a cluster. Rows in *C* with all entries "0" do not represent a cluster. *C* allows expressing cases in which all nodes are disconnected from each other (*n* clusters) and also cases in which all nodes are connected (1 cluster).

Algorithm 1 performs similarly as depth first search to create *C*. First, *C* is initialized with a zero *n x n* matrix. The algorithm searches the first entry in *GCM* with value "1" and declares a new cluster. Then, it finds the nodes belonging to that cluster iteratively analyzing the connected nodes either directly or through other nodes of the same cluster (line 6 of the algorithm). Entries in *GCM* with values "1" already assigned to a previous cluster are not analyzed. The algorithm ends after analyzing all the entries in *GCM*.

At the end of the algorithm every row of *C* that contains at least one "1" represents a cluster of the network. For every row, the columns filled with "1" correspond to nodes that are in that cluster. Figure 2 shows matrix *C* corresponding to the example in Fig. 1. The resulting five clusters are $C_1 = \{1\}$, $C_2 = \{2\}$, $C_3 = \{3,5,7,8\}$, $C_4 = \{4\}$ and $C_5 = \{6\}$: it requires connectivity repairing.

Algorithm 1. Computation of *C* from *GCM*

1. Cluster Matrix *C* is initialized as n x n matrix of zeros
2. j=1, $C(j,1) = 1$
3. **for** k=1:n **do**
4. **if** $\sum_i C(i,k) = 0$ **then**
5. $C(j,k) = 1$
6. $C(j,m) = 1 \; \forall$ node *m* that is neighbor of node *k* in *GCM*
7. j++
8. **end if**
9. **end for**

Cluster Matrix
C

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Fig. 2. Cluster matrix *C* computed for the example in Fig. 1

In the diagnosis stage also a PRR-range model for the deployed nodes is computed. Each robot is assumed to know its location during the survey. Also, the location of the static nodes has been estimated as described above. Thus, it is possible to build a PRR-range model using the response packets transmitted by the static nodes and the distances between the robot and the static nodes when the beacon packets were transmitted. In this model range is divided in intervals and the mean percentage of messages received over all the messages broadcasted is taken as the PRR in this interval. Figure 3 shows the PRR-range model obtained in network diagnosis experiments performed in the *CONET Robot-WSN Integrated Testbed*.

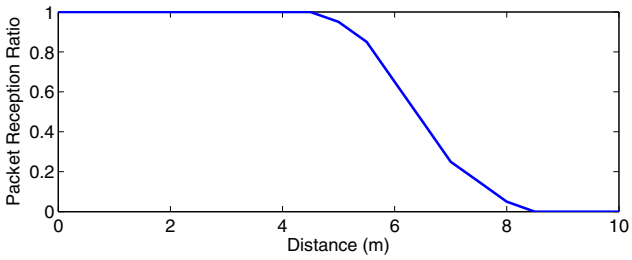


Fig. 3. PRR-range model obtained in the *CONET Robot-WSN Integrated Testbed*

Finally, it is decided if there is need or not to activate the connectivity repairing mechanism. If network is composed of more than one cluster, the connectivity repairing stage is triggered. If the network is composed by only one cluster of nodes, there is no need to activate connectivity repairing.

5 WSN Connectivity Repairing

This mechanism calculates the locations where new nodes should be deployed in order to optimally improve WSN connectivity by joining disconnected clusters. At the end of the previous stage the locations of the nodes and the topology of the clusters are known. The connectivity repairing mechanism is divided in two steps, see Algorithm 2. In the first step new nodes are deployed to join nearby clusters in larger clusters –we call them superclusters. In the second step all superclusters are joined in one cluster covering the entire network, achieving 1-connectivity.

To group clusters in superclusters first we need to know which clusters can be joined into the same supercluster. Using the PRR-range model obtained in the diagnosis stage, R is computed as the range that ensures a desired PRR level, $PRR(d) \geq X, \forall d \leq R$. X is taken as 0.85 in the experiments. If the closest nodes of two different clusters are separated by a distance lower than $2R$, then these clusters can be joined by deploying a new node. These clusters are considered neighbors and can be grouped into the same supercluster.

Consider a WSN with N disconnected clusters. We can compute the cluster connectivity matrix *CCM*. If entry *CCM(i,j)="1"* it means that clusters i and j are neighbors. Otherwise, *CCM(i,j)="0"*. The size of *CCM(i,j)* is *NxN*. *CCM* is also a connectivity matrix. We can also use Algorithm 1 to obtain superclusters of neighbor clusters. The supercluster matrix *SC*, of size *NxN*, is obtained after applying Algorithm 1 to *CCM*. If *SC(i,j)="1"* it means that cluster j belongs to supercluster i. In short, a recursive approach is followed to define how WSN nodes are organized at different levels.

Figure 4 shows an example with 8 nodes, which after the diagnosis stage, are organized in five clusters: $C_1 = \{1\}$, $C_2 = \{2,3,4\}$, $C_3 = \{5,6\}$, $C_4 = \{7\}$ and $C_5 = \{8\}$. C_1 and C_2 are at a distance lower than $2R$, thus they are neighbor clusters. The same applies to C_1 and C_3, C_2 and C_3 and C_4 and C_5. Thus, these clusters are organized in two superclusters: $SC_1 = \{C_1, C_2, C_3\}$ and $SC_2 = \{C_4, C_5\}$. Dashed lines in Fig. 4 shows the

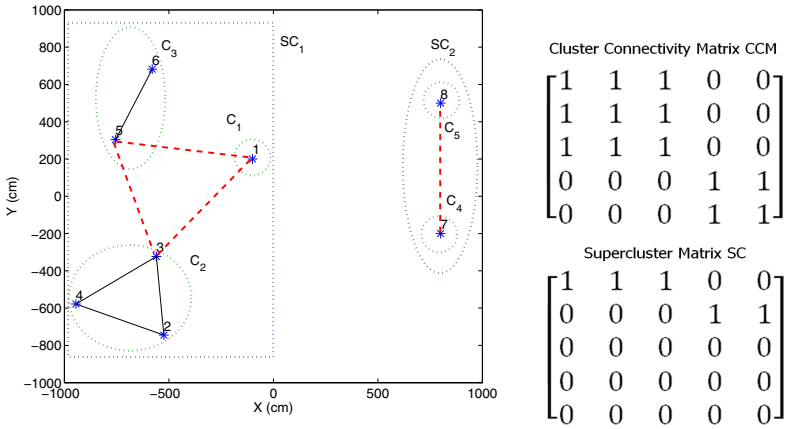


Fig. 4. Superclusters in an example. The links between clusters are represented by dashed lines. *CCM* and *SC* are also shown.

imaginary lines of length $R < d < 2R$ that would join the closest nodes of two different neighbor clusters.

The next step is to compute the locations where to deploy the minimum number of nodes that ensure 1-connectivity within each supercluster. We interpret the graph resulting from considering the WSN topology (in black in Fig. 4) and the imaginary lines between clusters (dashed in Fig. 4). As shown in Fig. 4 two different cases can arise. In SC_1 imaginary lines between clusters, for brevity from now on *ilbcs*, form a polygon (a triangle). In SC_2, the *ilbcs* do not form a polygon.

When the *ilbc* in a supercluster do not form a polygon the repairing location is selected among the cells which distance to both vertices of the *ilbc* are simultaneously lower than R. These locations ensure connectivity between both clusters. Then, the repairing location is selected analysing the effect of deploying a node at each of the selected cells. The location is selected as that in which deploying the node maximizes the repairing impact of the deployment, i.e. creates the highest number of new connections. If case of tie, the quality of the new links is analyzed: the repairing location is selected as the cell with the highest average PRR of the new connections originated by deploying the node in that cell.

In case *ilbcs* form a polygon, the method identifies the locations within the polygon where the minimum number of nodes should de deployed to establish 1-connectivity in the supercluster. First, the method tries to join all the clusters using only one node. If there is at least one cell within the polygon which distance to all the clusters is lower than R, then the supercluster can be connected using only one node. If there are more than one possible cell, the best is selected measuring the repairing impact as described above. Otherwise, the algorithm tries to connect all the clusters using two new nodes. A combination of two cells within the polygon that enable connection among all clusters is selected. If several solutions exist, the best combination using the above criteria is selected. In case no combination of two cells exist, the procedure is iteratively repeated trying solutions with one more node until a solution is found. Notice that a solution is

guaranteed to exist with a number of nodes equal to the number of *ilbcs* that form the polygon. Also the number of *ilbcs* will be lower or equal to the number of clusters in the supercluster.

Figure 5 shows with asterisks the set of deployment locations selected to join the clusters into superclusters SC_1 and SC_2. The connections originated from the new nodes are depicted with dashed lines. In this case the new node 9 joins C_1, C_2 and C_3 into supercluster SC_1 and node 10 joins C_4 and C_5 into SC_2.

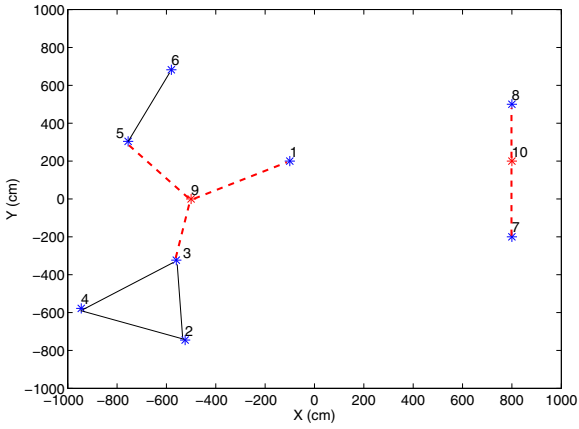


Fig. 5. Solutions selected to join C_1, C_2 and C_3 into supercluster SC_1 and C_4 and C_5 into SC_2. The selected deployment positions for the new nodes are represented with asterisks.

The next step is to join all superclusters in one. The first step is to identify the shortest imaginary lines that join two different superclusters. The length of these lines are higher than $2R$: it is not possible to join two superclusters with only one node. Thus, the approach is to deploy *chains* of new nodes along these imaginary lines. The minimum number of lines needed to join S superclusters is *S-1*. However, there are usually more than one combination of *S-1* lines between superclusters that join all the network. The method selects the combination of lines that join all superclusters with the lowest total length.

For those imaginary lines the optimal locations in which to place nodes is sought between cells which distance to these lines is lower than a certain distance d. The number of nodes Nn that should be deployed depends of the length of the line, $Nn \geq L/2R$, being L the length of the line and R the range that ensures a certain level of PRR according to the PRR-range model computed in the diagnosis stage. The locations of the nodes along the lines are selected using their repairing impact: solutions that join the superclusters with the highest number of new connections are selected; and in case of tie, the solution which new connections have the highest average PRR is selected.

Figure 6 shows the solution selected to join the superclusters in the example in Fig. 5. The shorter lines joining superclusters SC_1 and SC_2 is that between nodes 1 and 10. Two nodes are necessary to join both superclusters. The proposed repairing locations

Algorithm 2. Connectivity repairing algorithm

```
1.  Step1: Join clusters into superclusters
2.  Compute CCM
3.  Obtain SC applying Algorithm 1 to CCM
4.  Determine all imaginary lines between clusters ilbcs
5.  for Each supercluster in SC do
6.     if ilbcs do not form a polygon then
7.        Select the cell close to ilbcs that produces the best connectivity impact
8.     end if
9.     if ilbcs form a polygon then
10.       Determine minimum number of cells that join the clusters
11.       Select the combination of cells that produces the best connectivity impact
12.    end if
13. end for
14.   Step2: Join superclusters
15. Determine the shortest set of lines that joins all superclusters
16. for Every line do
17.    Determine the minimum number of nodes to join the superclusters along each line
18.    Select the combination of cells that produces the best connectivity
19. end for
```

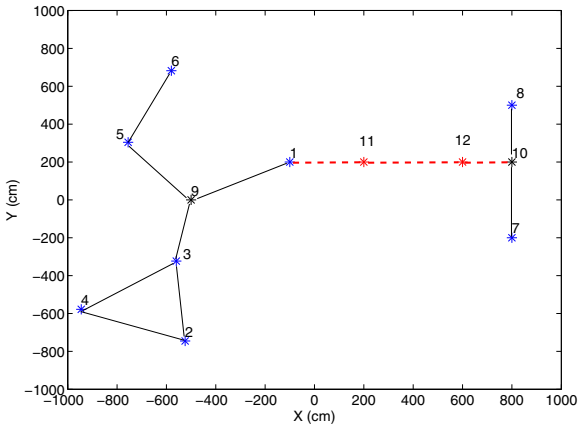


Fig. 6. Solutions adopted to join the superclusters SC_1 and SC_2 in the example in Fig. 4.

selected are marked with red asterisks as nodes 11 and 12 and the new connections are represented as dashed lines.

Finally, robots are commanded to deploy nodes at the repairing positions. New deployed nodes integrate with the existing WSN. If the BS node confirms that it receives packets from all nodes: the connectivity repairing is confirmed and the connectivity matrix *GCM* is updated. If the *BS* does not receive packets from all the nodes a new WSN diagnosis is needed.

6 WSN Redundancy Repairing

The redundancy repairing mechanism is activated if there are nodes with a lower N-redundancy than the desired level. This stage assumes that the network has 1-connectivity, i.e. it is applied after the connectivity repairing stage. This mechanism iteratively calculates the position where the deployment of a new node best improves the N-redundancy of the entire network. The iterations keep on until every node has at least the desired N-redundancy level.

The first step is to compute the redundancy of every node in the network. To do this we follow the concept of N-redundancy as stated in [1] and make use of the *GCM* obtained in the diagnosis mechanism and updated in the connectivity repairing mechanism. A node *i* has N-redundancy when if removed, at least N more nodes need to be removed to break the connectivity between two neighbors of *i*. Nodes with 0-redundancy are critical. A WSN is said to be robust if all nodes have a least 2-redundancy.

Algorithm 3 shows the method to compute the N-redundancy of node *i*. The algorithm can determine if node *i* has redundancy higher or equal to 2. If node *i* has redundancy lower than N, the method computes its redundancy. The algorithm operates with *GCM'*, a copy of *GCM* where the connections of the node *i* have been removed. If a pair of neighbors of *i* are disconnected in *GCM'*, redundancy of node *i* is set to 0. Otherwise, we analyze the impact of removing the connections of neighbors of node *i*. If removing the connections corresponding to any other neighbor of node *i* disconnects two neighbors of node *i* then redundancy of node *i* is 1. Otherwise, its redundancy is 2 or higher.

Algorithm 3. Computation of N-redundancy of node *i*

```
1.  Copy GCM in GCM'
2.  Initial Redundancy 2
3.  Remove all links with node i in GCM'
4.  Compute cluster matrix C' corresponding to GCM'
5.  Redundancy 0 if node i has only one neighbor
6.  for any possible pair of nodes j and k neighbors of i do
7.      if j and k are in different clusters then
8.          Redundancy 0
9.      else
10.         if Removing any other node, nodes j and k are in different clusters then
11.             Redundancy 1
12.         end if
13.     end if
14. end for
```

For example, node 7 in Fig. 7 has 0-redundancy because removing its connections disconnects nodes 3 and 8. On the other hand, node 3 has 1-redundancy: besides node 3 itself we need to remove one node, e.g. node 1, to disconnect two neighbors of node 3, e.g. 2 and 7. The redundancy of every node is shown on top of its identifier.

If the redundancy of a node is lower than the desired N-redundancy level, redundancy repairing mechanism is triggered, see Algorithm 4. Redundancy repairing iteratively computes the location in which to deploy a node that produces the larger increase in redundancy. A brute-force solution that simulates the redundancy increment when deploying a node at each scenario cell would involve high computational cost and result in bad scalability. Many cells are not within the radio coverage of any static node. Also, there are locations that even within the coverage range of a static nodes, are not useful to improve redundancy. Therefore, the proposed mechanism only analyzes the scenario cells that satisfy the following two conditions: a) the cell is within the convex hull of all the deployed nodes and b) the cell is within the radio coverage (distance is lower than R) of at least two static nodes.

The computation of the convex hull of a set of points can be easily performed with algorithms such as [25,20]. Figure 7 shows the convex hull that envelopes the nodes of a cluster. The edges of the polygon can be either connections between static nodes or imaginary lines between disconnected nodes (dashed lines). The N-redundancy of every node is shown on top of the node identifier.

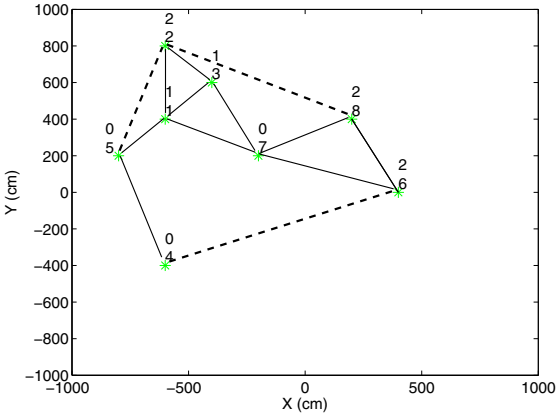


Fig. 7. Example of computation of N-redundancy and convex hull in a node cluster

The cells that do not satisfy both conditions are discarded. The redundancy repairing cells are selected analyzing the impact on redundancy improvement of deploying a node on that cell. The first criterion is the number of nodes that increase their redundancy from 0 to 1 (or higher) as a result of deploying a node at that cell. Nodes with 0-redundancy, such as node 7 in Fig. 7, are critical nodes in the WSN: the proposed method gives priority to solving fragile points. The second criterion is the number of nodes that increase their redundancy from 1 to 2 (or higher). The second criterion is used if there is a tie when using the first criterion. In case the tie keeps, the third criterion is the number of new connections originated by deploying a node at that cell. If necessary, the fourth criterion is to select the cell that obtains the highest average PRR in the connections it originates.

Once a repairing position is selected, *GCM* is updated with the connections produced by the new node. Then, the next iteration of the WSN redundancy repair mechanism starts. The iterations continue until all nodes have at least N-redundancy. Once they have been computed, the robots are commanded to deploy the nodes, provided that the needed number of new nodes are available.

Algorithm 4. N-redundancy repair algorithm

1. Compute the convex hull as in [25,20]
2. Select cells within the convex hull that connect at least two nodes
3. **while** N-redundancy not achieved **do**
4. Analyze the effect of deploying one node in every selected cell
5. Select the cell that produces best impact on redundancy
6. **end while**

Keeping with the example as in Fig. 7, Fig. 8 shows an example that requires two iterations to achieve the desired 2-redundancy in all the nodes. The deployment of the first node, see Fig. 8-left, increases N-redundancy from 0 to 1 or 2 in nodes 7, 4 and 5. N-redundancy is also increased from 1 to 2 in nodes 3 and 1. Hence, deploying only one new node, 1-redundancy is ensured in the whole network. The deployment of the second node, node 10, see Fig. 8-right, increases N-redundancy from 1 to 2 in nodes 9, 5, 7 and 8. Therefore, 2-redundancy is ensured in the whole WSN and the redundancy repairing mechanism finishes.

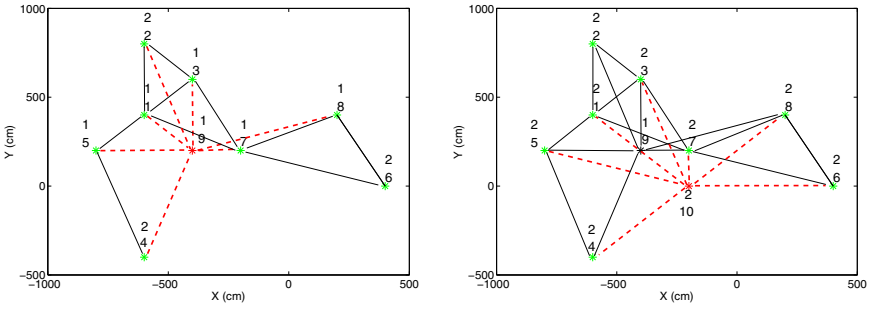


Fig. 8. First and second iterations of the WSN redundancy repairing mechanism in the example of Fig. 7. Left) First iteration. The deployment of node 9 ensures 1-redundancy in the WSN. Right) Second iteration. The deployment of node 10 ensures 2-redundancy in the WSN.

7 Validation

7.1 Simulations

The proposed method is compared with the optimal algorithm in order to assess its performance. The addressed problem is known be NP-complete, [13]. The algorithm

that minimizes the number of nodes consists in trying each possible solution. First, it tries to solve the problem using one node. If no solution is found, all combinations with two nodes are tried. It keeps until a solution is found. Figure 9 shows the computational time and the number of nodes required by both methods to solve the problem with 1-redundancy in 10 different 20x20 m^2 scenarios with 5 nodes. In these simulations both methods were executed in MATLAB. The time is represented in logarithmic scale. In all scenarios the solution found by our method is very near the optimal, it uses the same number of nodes or only one less, and requires times that are only a small fraction of the optimal. In many cases, e.g. scenarios 2, 4, 6, 7 and 8, our method gives the same solution but requiring significantly less time. In scenario 10, our method solves the problem adding one node in 1.88 s, while the optimal method solves it with 4 nodes but requires 221970 s.

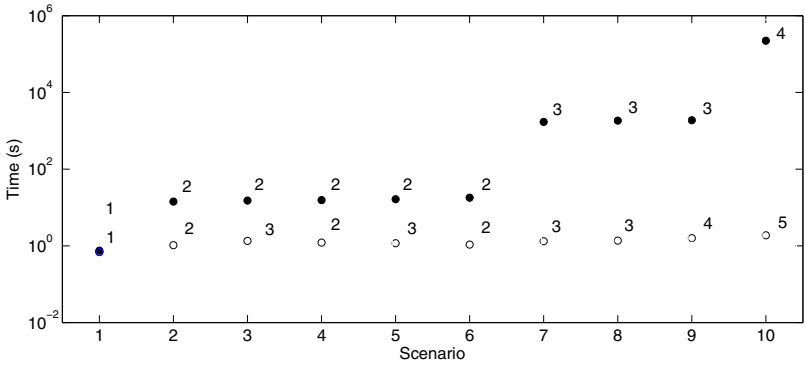


Fig. 9. Computational time and number of nodes required for achieving 1-redundancy in 10 different scenarios. The solutions of the optimal algorithm are represented by full markers while those of the proposed method are represented by empty markers. The number of nodes required to solve each scenario is shown next to every marker.

Figure 10 shows the computational times and number of nodes resulting from solving the problem in the scenario in Figure 4 but scaled in the range [0.5, 3.5]. In scaled scenarios, distances from nodes location to the scenario center were multiplied by a scaling factor. The nodes communication range were kept unaltered. The proposed method scales linearly only with the area of the convex hull that envelopes nodes, which is typically a fraction of the total area.

The proposed method discretizes the scenario into cells. Its dependence with the coarseness of the discretization is also analyzed. Low resolution leads to solutions that use more nodes than needed. High resolution leads to high computing burden. To consider both effects discretization coarseness is selected depending on the nodes range and the size of the convex hull. Notice that the convex hull size is unknown before the redundancy step. The diagnosis and connectivity steps are performed using a fixed resolution, e.g. 1x1 m per cell. Notice that connectivity is very efficient and its burden has low dependence with cell resolution.

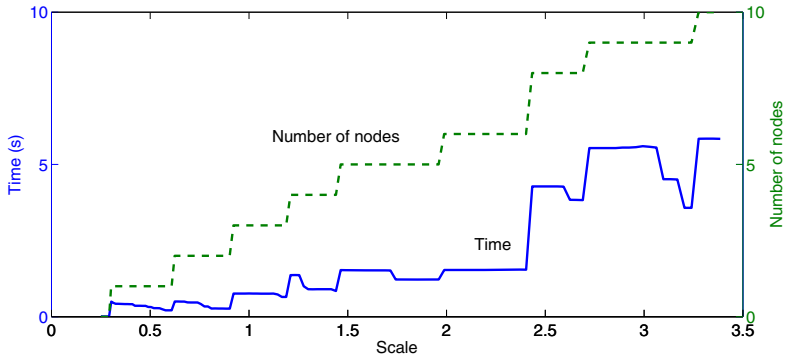


Fig. 10. Computational time and number of additional nodes needed to achieve 1-redundancy in the scenario in Figure 4 at different scales.

Then, once the nodes range and the convex hull size have been computed, the resolution is selected in the range [*R*/2, *R*/5] depending on the convex hull size. The objective is that the convex hull has around 150 cells. Of course, larger scenarios could produce more than 150 cells even if R/2 resolution is selected. In these cases, the method scalability ensures reaching a solution in reasonable times, as analyzed below.

The scenario in Fig. 4 was solved using a discretization in the range [25-625] cells. Figure 11 shows the computational times employed by the proposed algorithm to achieve 1-connectivity, 1-redundancy and 2-redundancy. The computational times in the three cases behave linearly against the number of cells in the discretized scenario. The times to compute 1-connectivity solutions are negligible even in highly discretized scenarios. The times to compute 1-redundancy solutions are lower than 1 second. The times to compute 2-redundancy solutions are larger and they increase linearly with the number of cells.

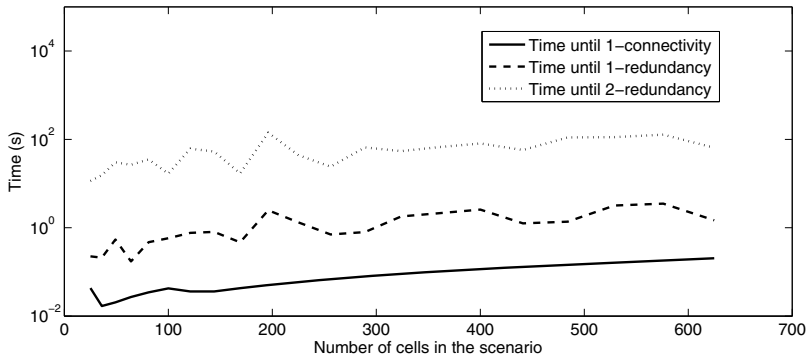


Fig. 11. Computational times to solve the scenario of Figure 4 with a growing number of cells

7.2 Experiments

The proposed method has been also experimented in real settings in the *CONET Robot-WSN Integrated Testbed*. The *CONET Testbed*[1] is a remote open tool to assess and compare multi-robot and WSN methods and algorithms. It is composed by 5 Pioneer 3-AT robots and 140 WSN nodes of different models. Each robot is endowed with a Hokuyo range finder and an Microsoft Kinect camera, GPS and Inertial Measurement Unit, among others. The testbed is installed since 2010 at the basement of the building of the School of Engineering of Seville (Spain). For this experiment one robot was equipped with a robotic arm for node deployment and retrieving, see Fig. 12-left.

In these experiments twenty nodes and a Base node have been deployed in the testbed room. The fleet of Pioneer 3-AT robots cooperatively perform the WSN diagnosis mechanism to compute: locations of the nodes, the connectivity *GCM* and cluster *C* matrices and the PRR-range model. Figure 12-right shows the WSN topology in one experiment after the WSN diagnosis stage. The circles represent the discovered nodes at their estimated locations. The connectivity between nodes is also depicted. In this example, the WSN is fragmented into three clusters: the biggest cluster, $C1$, includes 10 nodes and the Base node, the other two, $C2$ and $C3$, are disconnected from the Base.

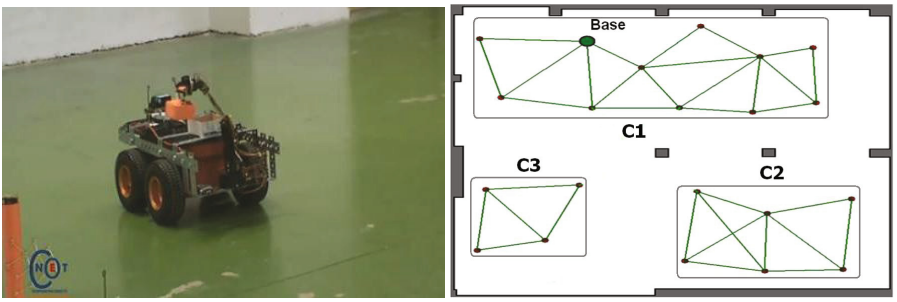


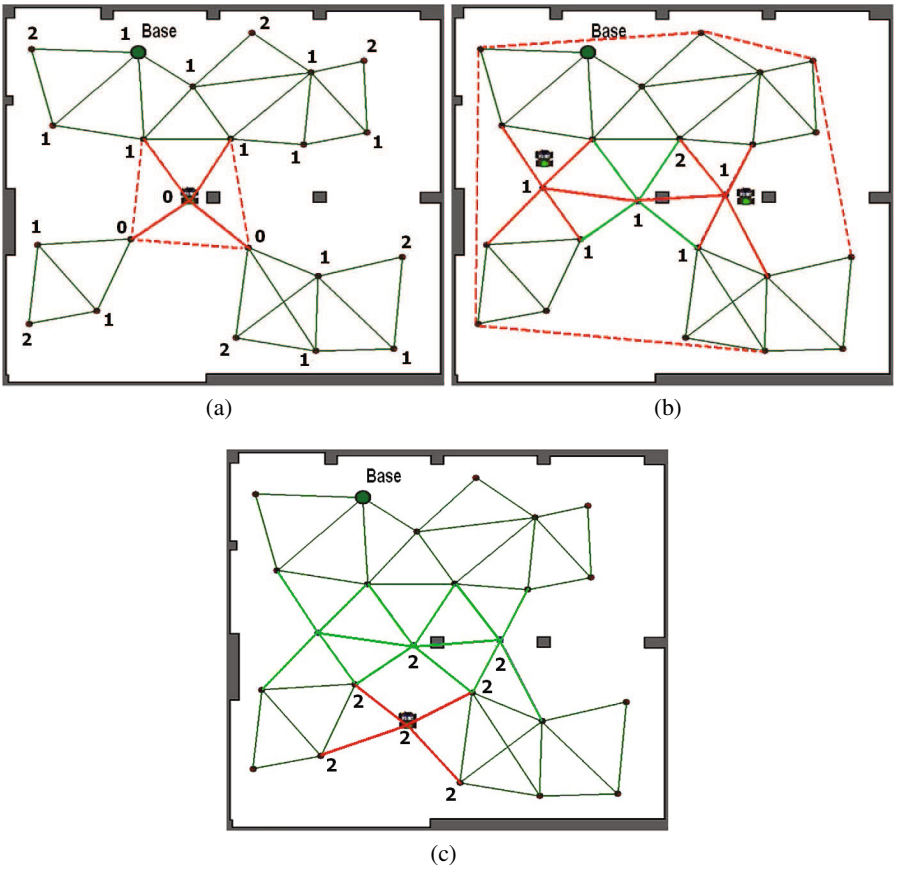
Fig. 12. (Left) Mobile robot deploying a node with its robotic arm. (Right) Connectivity after the WSN diagnosis stage. The WSN is fragmented in three clusters.

Since the network is fragmented the WSN connectivity repairing mechanism is started. It is detected that the distance between the closest nodes is lower than $2R$. Thus, the three clusters can be joined in one supercluster. The method determines that one single node deployed between the three clusters can join them in only one supercluster. Figure 13-a shows with dashed lines the links between the closest nodes of the three clusters. In this case the links between $C1$ and $C2$, between $C2$ and $C3$, between $C3$ and $C1$ form a four sided polygon together with already existing connection between the nodes of $C1$. The solution is sought inside the area delimited by the polygon. The best solution is to deploy one node at the center of the polygon. Then, a robot of the fleet is assigned with the task of deploying a node at the computed position. A simple task allocation based on Market-based auctioning is used, [7]. Once deployed, the new node establishes links with its new

[1] `https://conet.us.es`

neighbors, depicted in red solid lines in Fig. 13-a. The WSN has *1-connectivity* and the connectivity repairing stage is considered finished.

The initial redundancy of every node is shown in Fig. 13-a. There are three nodes with 0-redundancy and thus the WSN redundancy repairing mechanism is triggered. Figure 13-b shows the convex hull that envelopes all the nodes. The edges of the polygon are connections between static nodes or imaginary lines between nodes with no radio connection, shown as dashed red lines in the figure. In this case all the cells inside the convex hull are within the radio coverage of at least two nodes, so in every iteration the WSN redundancy repairing mechanism computes a solution within the convex hull.



(a) (b)

(c)

Fig. 13. Steps in the WSN connectivity and redundancy repairing mechanisms. (a) Network after the WSN connectivity repairing. One node is deployed. The new connections are shown as red lines. (b) Network after achieving 1-redundancy in the WSN redundancy repairing stage, the new connections after deployment of two nodes are shown in red colour. (c) Network state after the deployment of one more node.

After two iterations and the deployment of two more nodes 1-redundancy is achieved in the whole network. After deployment, see Fig. 13-b, ten -10- new connections are created -in red in the figure- and all nodes have at least 1-redundancy. Then, another iteration of the redundancy repairing mechanism was carried out to achieve 2-redundancy. The location selected was computed as described in Section 6. After deployment, four new connections were created and a total of seven WSN nodes increase their N-redundancy from 1 to 2. A summary of this experiment can be seen in this video[2].

8 Conclusions

In this chapter a system for connectivity and redundancy repairing of WSN with mobile robots has been presented. The system consists of three mechanisms: diagnosis, connectivity repairing and redundancy repairing. After the diagnosis stage, the connectivity repairing mechanism finds the optimal deployment locations to ensure that the WSN has 1-connectivity. The redundancy repairing mechanism is executed iteratively. At each iteration, it finds the location where the deployment of a new node best improves N-redundancy of all the nodes in the WSN.

The proposed scheme does not use any parameters. It acquires all the required information during WSN diagnosis stage and only needs as input the desired level of N-redundancy. It has been successfully validated in experiments carried out in the *CONET Robot-WSN Integrated Testbed*[3].

The method uses a iterative approach. It assumes that all necessary nodes are available for deployment. However, when it is not the case, the method uses the nodes available in the best possible way. Each node is deployed optimizing its impact following a step-by-step optimization. The use of globally optimal methods is currently under research. On the other hand, the problems of connectivity and redundancy are coupled but in our method are treated independently. The search for a global solution that simultaneously solves the two problems is under development.

References

1. Atay, N., Bayazit, B.: Mobile wireless sensor network connectivity repair with k-redundancy. In: Chirikjian, G.S., Choset, H., Morales, M., Murphey, T. (eds.) Algorithmic Foundation of Robotics VIII. STAR, vol. 57, pp. 35–49. Springer, Heidelberg (2009)
2. Bourdenas, T., Sloman, M.: Towards self-healing in wireless sensor networks. In: IEEE Sixth Intl. Workshop on Wearable and Implantable Body Sensor Networks, pp. 15–20 (2009)
3. Caballero, F., Merino, L., Gil, P., Maza, I., Ollero, A.: A probabilistic framework for entire WSN localization using a mobile robot. Robotics and Autonomous Systems 56(10), 798–806 (2008)
4. Coles, M., Azzi, D., Haynes, B.: A self-healing mobile wireless sensor network using predictive reasoning. Sensor Review, Emerald Group 28(4), 326–333 (2008)
5. Corbett, D.R., Gage, D.W., Hackett, D.D.: Robotic communications and surveillance-the DARPA LANdroids program. In: Wang, D., Reynolds, M. (eds.) AI 2011. LNCS, vol. 7106, pp. 749–758. Springer, Heidelberg (2011)

[2] `http://www.youtube.com/watch?v=1rKjdT-dCjc`
[3] `https://conet.us.es`

6. Corke, P., Hrabar, S., Peterson, R., Rus, D., Saripalli, S., Sukhatme, G.: Deployment and connectivity repair of a sensor net with a flying robot. In: Experimental Robotics, I. (ed.) Experimental Robotics IX. STAR, vol. 21, pp. 333–343. Springer, Heidelberg (2006)
7. Dias, M.B., Stenz, A.: Opportunistic optimization for market-based multirobot control. In: Proceedings IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems, pp. 2714–2720. Lausanne, Switzerland (2002)
8. Fagiolini, A., Babboni, F., Bicchi, A.: Dynamic distributed intrusion detection for secure multi-robot systems. In: IEEE Intl. Conf. on Robotics and Automation, ICRA 2009, pp. 2723–2728 (2009)
9. Ganeriwal, S., Kansal, A., Srivastava, M.: Self aware actuation for fault repair in sensor networks. In: Proceedings of the 2004 IEEE Intl. Conf. on Robotics and Automation, ICRA 2004, vol. 5, pp. 5244–5249 (2004)
10. Gui, C., Mohapatra, P.: SHORT: self-healing and optimizing routing techniques for mobile ad hoc networks. In: Proceedings of the 4th ACM Intl. Symposium on Mobile ad Hoc Networking & Computing, pp. 279–290. ACM (2003)
11. Honkavirta, V., Perala, T., Ali-Loytty, S., Piché, R.: A comparative survey of wlan location Fingerprinting methods. In: 6th Workshop on Positioning, Navigation and Communication, pp. 243–251 (2009)
12. Jiménez-González, A., Martínez-de Dios, J.R., Ollero, A.: An Integrated Testbed for Cooperative Perception with Heterogeneous Mobile and Static Sensors. Sensors 11, 11516–11543 (2011)
13. Lin, G., Xue, G.: Steiner tree problem with minimum number of Steiner points and bounded edge-length. Information Processing Letters 69(2), 53–57 (1999)
14. Liu, C., Wu, K., He, T.: Sensor localization with ring overlapping based on comparison of received signal strength indicator. In: Intl. Conf. on Mobile Ad-Hoc and Sensor Systems, pp. 516–518 (2004)
15. Luthy, K., Grant, E., Henderson, T.: Leveraging rssi for robotic repair of disconnected wireless sensor networks. In: IEEE International Conference on Robotics and Automation, pp. 3659–3664 (2007)
16. Ma, H., Liu, Y.: On coverage problems of directional sensor networks. In: Jia, X., Wu, J., He, Y. (eds.) MSN 2005. LNCS, vol. 3794, pp. 721–731. Springer, Heidelberg (2005)
17. Machado, R., Tekinay, S.: Bounds on the error in estimating redundancy in randomly deployed wireless sensor networks. In: IEEE Intl. Conf. on Sensor Technologies and Applications, pp. 319–324 (2007)
18. Machado, R., Ansari, N., Wang, G., Tekinay, S.: Adaptive density control in heterogeneous wireless sensor networks with and without power management. IET Communications 4(7), 758–767 (2010)
19. Martinez-deDios, J., Lferd, K., de San Bernabé, A., Núñez, G., Torres-González, A., Ollero, A.: Cooperation between UAS and wireless sensor networks for efficient data collection in large environments. Journal of Intelligent and Robotic Systems, 1–18 (2012)
20. Pateiro-López, B., Rodríguez-Casal, A.: Generalizing the convex hull of a sample: The R package alphahull. Journal of Statistical Software 34(5), 1–28 (2010)
21. Popa, D.O., Lewis, F.L.: Algorithms for Robotic Deployment of WSN in Adaptive Sampling Applications. In: Li, Y., Thai, M.T., Wu, W. (eds.) Signals and Communication Technology, pp. 35–64. Springer US (2008)
22. Sitanayah, L., Brown, K.N., Sreenan, C.J.: Fault-Tolerant relay deployment based on length-constrained connectivity and rerouting centrality in wireless sensor networks. In: Picco, G.P., Heinzelman, W. (eds.) EWSN 2012. LNCS, vol. 7158, pp. 115–130. Springer, Heidelberg (2012)
23. Watfa, M.K., Commuri, S.: Boundary coverage and coverage boundary problems in wireless sensor networks. International Journal of Sensor Networks 2(3-4), 273–283 (2007)

24. Wang, X., Bischoff, O., Laur, R., Paul, S.: Localization in wireless ad-hoc sensor networks using multilateration with rssi for logistic applications. Procedia Chemistry 1, 461–464 (2009)
25. Wu, W., Li, L., Wang, J.: An improved Graham algorithm for determining the convex hull of planar points set. Science of Surveying and Mapping 35(6), 123–125 (2010)
26. Yoo, G., Jung, J., Lee, E.: Fault Management for Self-Healing in Ubiquitous Sensor Network. In: IEEE Second International Conference on Future Generation Communication and Networking Symposia, FGCNS 2008, vol. 5, pp. 21–25 (2008)
27. Younis, M., Lee, S., Guptam, S., Fisher, K.: A localized self-healing algorithm for networks of moveable sensor nodes. In: IEEE Global Telecommunications Conference, pp. 1–5 (2008)
28. Zhang, C., Zhang, Y., Fang, Y.: Localized algorithms for coverage boundary detection in wireless sensor networks. Wireless Networks 15(1) (January 2009)