Nodes self-deployment for coverage maximization in mobile robot networks using an evolving neural network

Carmelo Costanzo, Valeria Loscri, Enrico Natalizio, Tahiry Razafindralambo

Nodes self-deployment for coverage maximization in mobile robot networks using an evolving neural network

C. Costanzo[a], V. Loscrí[a], E. Natalizio T.[b], Razafindralambo[b]

[a]*DEIS - University of Calabria*
[b]*INRIA Lille - Nord Europe*

Abstract

There are many critical issues arising in Wireless Sensor and Robot Networks (WSRN). Based on the specific application, different objectives can be taken into account such as energy consumption, throughput, delay, coverage, etc. Also many schemes have been proposed in order to optimize a specific Quality of Service (QoS) parameter. With the focus on the self-organizing capabilities of nodes in WSRN, we propose a movement-assisted technique for nodes self-deployment. Specifically, we propose to use a neural network as a controller for nodes mobility and a genetic algorithm for the training of the neural network through reinforcement learning [27]. This kind of scheme is extremely adaptive, since it can be easily modified in order to consider different objective and/or QoS parameters. In fact, it is sufficient to consider a different input for the neural network to aim to a different objective. All things considered, we propose a new method for programming a WSRN and we will show practically how the technique works, when the coverage of the network is the QoS parameter to optimize. Simulation results show the flexibility and effectiveness of this approach even when the application scenario changes (e.g., by introducing physical obstacles).

Keywords: Wireless Sensor and Robot Networks, Neural Network, Genetic Algorithms, Self-Organizing Networks

1. Introduction

Wireless Sensor and Robot Networks (WSRN) are composed by a high number of nodes that need to cooperate to accomplish several tasks (i.e. detect the presence of a malicious intruder in a region of interest, monitor the occurrence of a specific event, etc.). They have been applied in military applications and health care, as well as environmental monitoring and smart agriculture [1]. One of the fundamental issue in such a network is coverage. It is used to determine how well an interest area is monitored and a service can be provided. Nodes deployment is a critical issue for coverage, because it affects costs and detection capability of a WSRN. In literature, existing nodes deployment algorithms can be classified into the following categories: 1) Stationary sensors [2], [3] 2) Mobile Sensors [4] [5] and 3) Mobile Robot [6] [7]. Usually stationary sensors are random deployed in the area of interest. Random deployment is very simple, but the number of nodes to deploy has to be much larger than what is actually required for the full coverage. Furthermore, a random deployment of static nodes could be inefficient since some areas could be densely deployed in respect of others and some coverage holes or network partitions can occur. In some work [4] [5], a large number of stationary nodes and a few mobile sensors are considered. Unfortunately, when the initial deployment of static nodes is random, this kind of approach is not useful to overcome the hole problems. In order to face these issues, it is possible to use only mobile nodes, but in this case it is necessary to implement an efficient and effective technique to allow nodes to move towards a certain position, in order to ensure the best coverage with the smallest amount of movement. In fact, energy consumption and probability to lose connectivity increase when nodes movement increases. These schemes are also known as movement-assisted techniques. An alternative is to use robots for deploying static nodes in a given region. Robot deployment can achieve full coverage with fewer nodes and can guarantee connectivity too. However, the presence of obstacles can challenge robot deployment and have a great impact on the deployment efficiency. To reduce the impact of obstacles in [7] authors propose four traveling orders for the robot movement. Unfortunately, this approach cannot guarantee full coverage and may introduce several sensing redundancies when the robot encounters obstacles. The approaches for coverage can be categorized into three groups: 1) force based

[29], 2) grid-based [8] and 3) computational geometry based [9]. All the three categories can be considered as a sub-class of the movement-assisted category. With the focus on the self-organizing capabilities of nodes in WSRN, we propose a movement-assisted technique for nodes self-deployment. Specifically, we propose to use a neural network as a controller for nodes mobility and a genetic algorithm for the training of the neural network through reinforcement learning. This kind of scheme is extremely adaptive, since it can be easily modified in order to consider different objective and/or QoS parameters. In fact, it is sufficient to consider a different input for the neural network to aim to a different objective. In this work we will show how the scheme works by considering the neural network as a controller for nodes mobility, when the objective is to maximize the coverage area and minimize the number of time steps to achieve the objective. To the best of our knowledge for deployment of wireless nodes able to self-organize, this is the first work based on neural networks and genetic algorithms. This technique has the clear advantage to make nodes able to learn from the environment and adapt their behavior. Based only on local information, nodes ensure a high coverage even when obstacles of irregular shapes are present in the area of interest. Another evident advantage of our algorithm is its simplicity: by the simple introduction of a new input, a different QoS parameter or objective can be taken into account. In respect of other approaches, our algorithm can be considered as a synthesis. In fact, our technique is grid based since it focuses on an interest area divided into cells. It can also be considered as a kind of virtual-force scheme based, because nodes are attracted or rejected from certain positions depending on the presence of other nodes. Finally, our technique belongs to the computational geometry based approach because the only local information needed at the nodes is the relative distance with other nodes and obstacles, and this leads to spatial distributions of the nodes in the area of interest that follow geometric rules (e.g., uniformly spaced position).

The contributions of this work are the following:

- the approach makes behavior emerge from the interaction instead of being pre-programmed in the nodes;
- the methodology used in this exploits both the flexibility of neural networks and the capability to learn of genetic algorithms, which makes it suitable to implement all the self-* properties;
- the proposed solution is simple, effective, distributed and feasible, even in presence of obstacles. It does not require any specific hardware/software that is not already included in mass products and needs only local information to work.

The rest of the paper is organized as follows: in Section we provide a state of the art, Section introduces methodology and algorithm. In Section we show the results of our algorithm and finally Section concludes the work.

2. Related Works

The problem of coverage in WSN and WSRN is closely related to deployment, because a good deployment can improve all the functionalities of the network. Indeed, Meguerdichian in [10] argues that coverage is the primary metric that provides indication about quality of service.

An important categorization regarding deployment for wireless networks as summarized in [11], classify algorithms as belongin to random deployment, incremental deployment and movement-assisted deployment. Random deployment is the fastest and most practical way to deploy a network, even if it does not ensure a uniform distribution. For this reason it is often used, as in our case, only in the initial phase of the movement-assisted deployment. Incremental deployment is a centralized approach, which places nodes one at time. The computation of optimal location for each node is based on information gathered by nodes already deployed. Hence, computation and time costs explodes when nodes number increases. In the last years, the most used approach to deploy a network is the movement-assisted, because it can achieve a uniform coverage with reasonable time and costs.

In [12] and [13], the approaches for coverage are categorized in three groups: force based [29] [14], grid-based [8], [15], [16] and computational geometry based [17], [18] and [9]. All these approaches can be considered as a sub-group of the movement-assisted approach.

In particular for the first group, in [14] authors propose a virtual force algorithm (VFA) to improve the coverage after an initial random placement over a region with obstacles. Even if their method works in presence of obstacles, it seems massively dependent on the mobility and energy of the sensors. Also authors of [19] consider obstacles. They develop an efficient technique to place sensors by means of a robot. Algorithm in [9] first deploys sensors

along the boundaries and the region based on the sensing radius of the sensors. Successively, they apply delaunay triangulation and add some sensors in order to reach full coverage. Both in [19] and [9], the methods developed seem suitable for simple regular regions and obstacles, like areas consisting of long straight lines. When obstacles and regions become sufficiently and arbitrarily irregular, these methods become inefficient. To consider obstacle in [20], authors introduce the concept of virtual force. Using this kind of Coulomb force proportional to the distance with other nodes or obstacles each nodes is able through attraction and repulsion to maximize the covered area. This is a cluster-based approach, so it shows all the drawbacks related to a centralized scheme. In [16] the whole region is divided into single-row and multi-row regions, in order to guarantee both coverage and connectivity. Specifically, holes are covered through sensors deployed along the boundaries, and connectivity is preserved by maintaining a constant distance computed from sensing and communication radii. Also this method seems suitable only for regular regions and obstacles.

Concerning the Grid quorum based movement [4], typically the sensors field is partitioned into many cells and the number of sensors in a cell is the load of the cell. Coverage and connectivity depend on the size of the cell. This deployment strategy cannot guarantee connectivity and cannot provide Points of Interest (PoI) coverage.

Regarding the computational geometry based approaches, In [5] authors propose an algorithm based on Voronoi regions. Each node, after the computation of its own Voronoi region, compares it with its sensing range in order to determine the presence of holes. In this case, the node moves to an improved location and recalculate the new Voronoi region until convergence is reached. This approach provides an uniform deployment in a distributed way, but is not suitable to use in scenarios with obstacles. Another distributed algorithm, known as pull & push [21], produce an hexagonal tiling, by attracting the nodes in low density regions and repulsing nodes from high density regions. This algorithm does not require manual tuning of variables related to the particular working scenario, even if it works only in absence of obstacles. Coulumb's law is also the base of [22], in this work authors propose a distributed algorithm inspired by equilibrium of molecules to obtain an uniform coverage. Each node calculates its lowest energy point and the distance from other nodes in a distribute manner, all computation requires exact location information and does not take in account the presence of obstacles.

The technique we propose in this work can be considered as a movement-assisted approach, in which the region of interest is sub-divided in specific cells (grid-quorum approach) and by the usage only of node-node and node-obstacle distance (computational geometry approach) attracts or repulses other nodes (virtual force approach) towards the best position for improving the coverage. A desirable property of our technique is its evolutionary nature, namely, its behavior varies in a "natural" fashion depending on the the evolution process of the network. In previous schemes, such as Voronoi and VF schemes, the behavior is imposed in a pre-programmed fashion. Instead, our approach allows self-organization of nodes and it does not need any prior knowledge of the environment, even in presence of obstacles. The novelty is that the deployment is not based on some geometric or physic rule pre-implemented in the nodes but the way to behave is learned directly from the environment during the training phase. The rules of movement learned are very general and not strictly related to any particular scenario.

3. A Genetic Algorithm for Nodes Self-Deployment

In order to solve the coverage problem through self-deployment of nodes in a wireless sensor and robot network, we propose to use neural networks and genetic algorithms in a combined approach. The neural network is used to model the behavior of the single node, whereas the objective of the genetic algorithm is to select the most performing population of nodes as a whole. In Section 4, we will show the effectiveness of the combined algorithm. Instead, in this Section we will illustrate the simplicity in the usage of the two main elements of the combined approach and we will introduce background and assumptions used in this work. The flowchart in Figure 1 describes the most relevant steps of the nodes self-deployment algorithm. The first three steps illustrate the initialization of simulation setup, neural network and genetic algorithm, while the following four contain the core algorithm.

In the following subsections we will refer to the flowchart as a general scheme of the proposed algorithm and we will provide details on each step.

3.1. Background and assumptions

The model proposed in this work is discrete in both time and space. We decided to use discretization for sake of simplicity, even though the method could be easily extended to a continuous model, where we would expect even

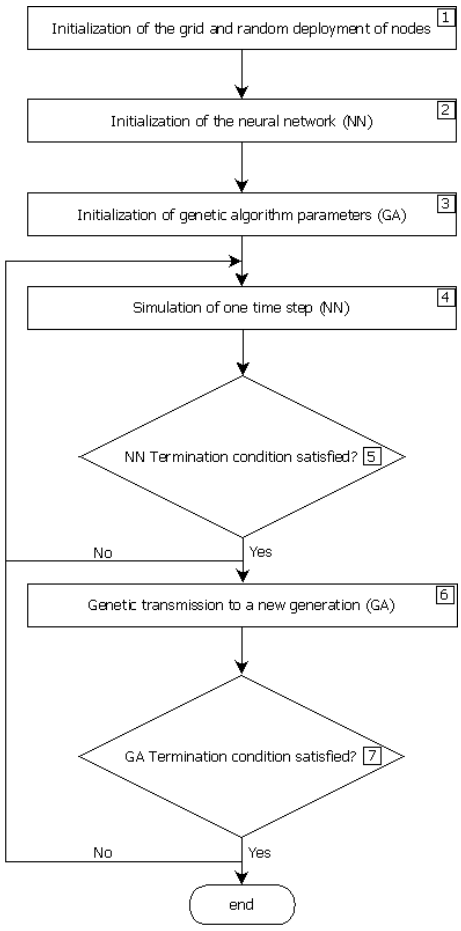


Figure 1: Description of the neural network training phase.

better results at the cost of some complication in the computation required from the nodes. Hence, the field to cover is a square grid constituted of square cells, where also physical obstacles can be present. We consider one cell and one time step as discrete units of space and time, respectively. Also the sensing radius r of the nodes becomes a positive integer, and it expresses the number of cells that nodes are able to cover in each of the four main direction (north, south, east and west). Therefore, the coverage area of a node is a square area composed of $(2r + 1)^2$ cells. In Figure 2, the area covered by a node is coloured in light green. Cells coloured with a higher intensity of green have a higher degree of coverage (also referred to as k-coverage).

In the described scenario, we use the following assumptions regarding the behavior of nodes:

- In one time step nodes can move of one cell in one of the four main direction (north, south, east and west).
- Nodes are able to estimate the distance (in number of horizontal and vertical cells) between themselves and other nodes, and between themselves and obstacles.

The first assumption is necessary to express a maximum amount of distance that a node can travel in a single time step. We decided, again for sake of simplicity, to exclude four secondary directions (north-east, north-west, south-east and south-west). However, it would be very easy to extend the set of possible movements to include also the missing directions, and the algorithm would improve its performance. The second assumption allows nodes to avoid other nodes and obstacles, and to calculate the right position to move to in order not to create unwanted overlapping coverage areas with their neighbors. The boundaries of the field are considered from the nodes as obstacles. For nodes to measure the relative distances with other nodes or obstacles is possible through the measurement of the round trip

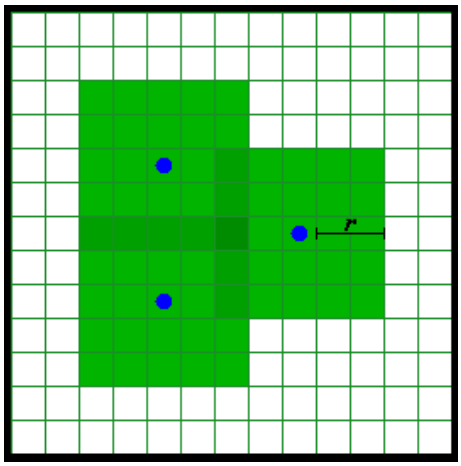


Figure 2: Representation of sensing radius and cell overlapping.

time or by infrared [23], ultrasonic [24] or visual techniques [25]. In order to guarantee connectivity, we could also assume that the communication range of the nodes is at least twice their sensing range, as in [26], nonetheless this assumption is not strictly necessary for the coverage algorithm. When the mentioned assumptions are satisfied, no additional hardware or software is needed onboard the nodes, and also no localization systems are required. Basically, nodes are able to place themselves and maximize the coverage only with the knowledge of the local environment around themselves.

3.2. Neural Network (NN)

The behavior of each node is controlled by a fully connected, recurrent and time-discrete artificial neural network. The neural network is composed by input, output and hidden neurons, as classified also in Figure 3. The number of hidden neurons depends on the complexity of the problem, and in our case 2 hidden neurons are sufficient for our objective.

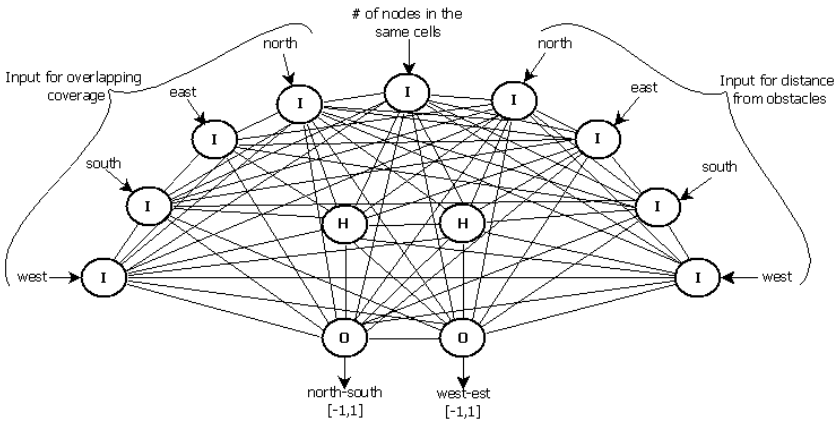


Figure 3: Neural network architecture of one node for the self-deployment problem.

The input of the neural network is detected from the environment and is related to the goal of interest. In our case, the input for each node is constituted by:

1. the number of cells already covered by the neighbors, one input for each direction;
2. the distance from obstacles, one input for each direction;

3. the number of nodes placed in the same cell.

The first set of input has to objective to make node spread and maximize the coverage by avoiding overlapping. The second aims to allow nodes to learn how to avoid obstacles that constitute an impediment to movement, sensing and communication. The last set of input is specifically useful when many nodes are initially deployed in the same cell. Regarding the input, it is important to make the following remarks:

- all the mentioned input can be computed at each node from the measurement of the distance with other nodes (input 1 and 3) and between node and obstacles (input 2);
- the measurement is limited to the coverage range of each node.

Therefore, as mentioned in the previous subsection, only local information about relative distances is enough to feed the neural network and make the algorithm work.

In order to map the n-dimensional input into the m-dimensional output, each neuron uses a real-valued activation function and a time-varying real-valued connection with every other neuron of the network.

$$out_j(k) = F\left(\sum_{i \in N} w_{ij} \cdot out_i(k-1) + b_j\right) \tag{1}$$

The output of neuron j, at the time step k is the same for all the connections originating in j, therefore we indicate with out_j the output of neuron j towards all other neurons of the network. The output of neuron j is computed as in (1), where N is the set of all the neurons of the neural network, w_{ij} is the weight of the incoming connection from neuron i to neuron j and b_j is the bias of neuron j. Weights can have either an excitatory or inhibitory effect. In our case, the activation function F is a simple linear threshold function as expressed in equation 2:

$$F(x) = \begin{cases} -1.0 & if \quad x \leq -1.0 \\ x & if \quad -1.0 < x < 1.0 \\ 1.0 & if \quad x \geq 1.0 \end{cases} \tag{2}$$

The output of the two output neurons is the output of the neural network for each node. It is composed of two real numbers that can vary in the interval $[-1 \div 1]$, as it is clear from (2). Depending on these two real values, the node decides the action to make. The set of possible actions is limited to move of one cell in one of the four admitted directions or stay still in the current cell.

The whole process described above is executed in one time step, for each node of the WSRN, as shown in Step 4 of the flowchart in Figure 1. This phase of the training of the neural network consists in the execution of the described process until a termination condition is satisfied, as in Step 5 of the flowchart. In our case, the termination condition is either the execution of a certain fixed number of time steps or the complete coverage of the field.

3.3. Evolutionary genetic algorithm (GA) for the neural network training phase

In order to train the neural network, in a self-organizing perspective, reinforcement learning is used. In fact, the performance of the neural network is evaluated by a fitness function.

The global optimization method used for training the neural network is a genetic algorithm. The genetic algorithm is encoded in the neural network through the genes, which are the values of the weights of the connection between each couple of neurons and bias of each neuron. A chromosome (also referred as a member of a population in the following) is constituted by an evolving set of genes, and a population is constituted by a fixed number of chromosomes. The goal of genetic algorithms is to improve the fitness function through the transmission of genes of a part of the current population from one generation to the next, as in Step 6 of the flowchart in Figure 1. The transmission of genes to newer generations continues until a termination condition for the genetic algorithm occurs (Step 7). In our work, the condition to terminate the genetic algorithm is the simulation of a fixed number of generations.

The transmission of genes to the new generation is based on the selection, mutation and crossover of the members of the old generation that achieved the highest value of fitness function. Thus, the fitness function has a very important role in the evolution process, since it is used as a feedback for successive generations. The objective of our work is to improve the coverage of the field. Hence, we have to relate the fitness function to the coverage achieved by each

member of a population. The coverage achieved is computed by counting the distinct cells covered by all the nodes of each member of the population, upon the termination condition of Step 5 is satisfied. Thus, the fitness function could be simply represented by the number of covered cells. However, we want also to take into account the "speed" of a chromosome in reaching the maximum coverage. Therefore, we define the fitness function as follows:

$$fitness\_function = \#covered\_cells + (\#time\_steps\_max - \#time\_steps\_actual) \quad (3)$$

In (3), *#time_steps_max* is the maximum number of time steps fixed as a termination condition of Step 5 and *#time_steps_actual* is the actual number of time steps used in that phase of the training by the member of the population. When the difference between these two terms is not zero, it means that the neural network covered the whole field in a number of time steps lower than the maximum allowed. Therefore, an increase of the fitness function by one unit in respect of the value achieved by the previous generation can result as an effect of one the two following causes:

- one more cell has been covered in the same number of time steps;
- the same number of cells has been covered in one time step less.

3.4. Communication complexity

The proposed algorithm is based on local communication only. Indeed, a node only needs to know the position of its neighboring nodes and the surrounding obstacles to take a movement decision. In the sequel, the term "broadcast" stands for message propagation in a node's neighborhood and the term "flooding" refers to network-wide message propagation.

After a node's movement, an update on node's position is broadcasted. The information contained in the broadcasted message consists of the node Id and the node position (x, y). Here it is worth noting that, the message size is constant, therefore the message size is in $O(1)$.

Nodes will update their positions and broadcast their new information at each time steps. Since the number of time step is a constant value given as a parameter in our algorithm, this leads to a message sending complexity of $O(n)$ where n is the number of nodes. In order to select the best population to generate the next generation of our genetic algorithm, each node needs to flood the parameters of its genetic algorithm (constant number of parameters) and the value of its fitness function (number wiht a constant size), the message size for the flooding is thus in $O(1)$. A flooding algorithm from the literature can be used to disseminate the information. For example, MPR [28] provides a complexity in $O(\Delta^2)$ where Δ is the maximum node degree, which is the maximum number of one-hop neighbors of a node.

4. Performance evaluation

In this section we will first introduce the simulation setup and then we will show the qualitative and quantitative results of our approach, when used to solve the problem of coverage. We are mainly interested in the coverage achieved by our scheme in respect of the achievable coverage. Nonetheless, we will investigate the mechanisms of genetic transmission and learning that lie behind the usage of the fitness function. The neural network approach does not share the basic assumption of the usual approaches, because while in the usual approach the behavior of nodes is pre-programmed (e.g., the attraction-repulsion forces in the force-based approach), neural network makes the behavior emerge from learning and interaction of the nodes. Our first objective is to focus on this emerging behavior, its effectiveness in its simplicity. Thus, a performance comparison of our scheme with a scheme using another approach would not be fair and it would be out of the scope of this work.

4.1. Simulation setup

The proposed scheme is evaluated by simulations using FREVO [1], which is an open source framework for evolutionary design. In Table 1, we reported all the simulation parameters used in this work. We consider a 40 cells x 40 cells field, where a variable number of nodes (n) is deployed according to a random uniform distribution and a

[1] http://www.frevotool.tk

variable number of obstacles is present (o). Also obstacles are placed according to a random uniform distribution in the field, but they can not create areas inaccessible for the nodes. Cells containing obstacles are subtracted from the achievable coverage. In our scheme, obstacles are considered impenetrable, thus limiting both nodes movement and coverage. In fact, all the cells in the coverable area of a node that are shadowed by an obstacle are not considered covered. The shadow area depends on the sensing range of the node. From some simple calculations, we can derive an overestimation of the average number of cells shadowed (cs) when an obstacle is present in the coverage area of a node:

$$cs = \frac{\sum_{l=0}^{r-1}(l+1)m(r-l)^2}{4r(r+1)} \tag{4}$$

where m is the number of cells in the first frame of cells around the node (8 when cells are square). If we assume $r = 2$, from (4) we can see that the presence of an obstacle inside the coverage area of a node creates a shadow effect that involves, on average, 2 cells (one for the physical presence of the obstacle and one because of the shadow). This is actually an overestimated value that does not take into account the possible overlapping shadow areas of obstacle, in fact the exact average value is slightly lower (1.94 cells for each obstacle). In any case, this means that even when the number of nodes is exact to cover the whole field, there will be areas uncovered due to the shadow effect of obstacles.

Table 1: Simulation parameters

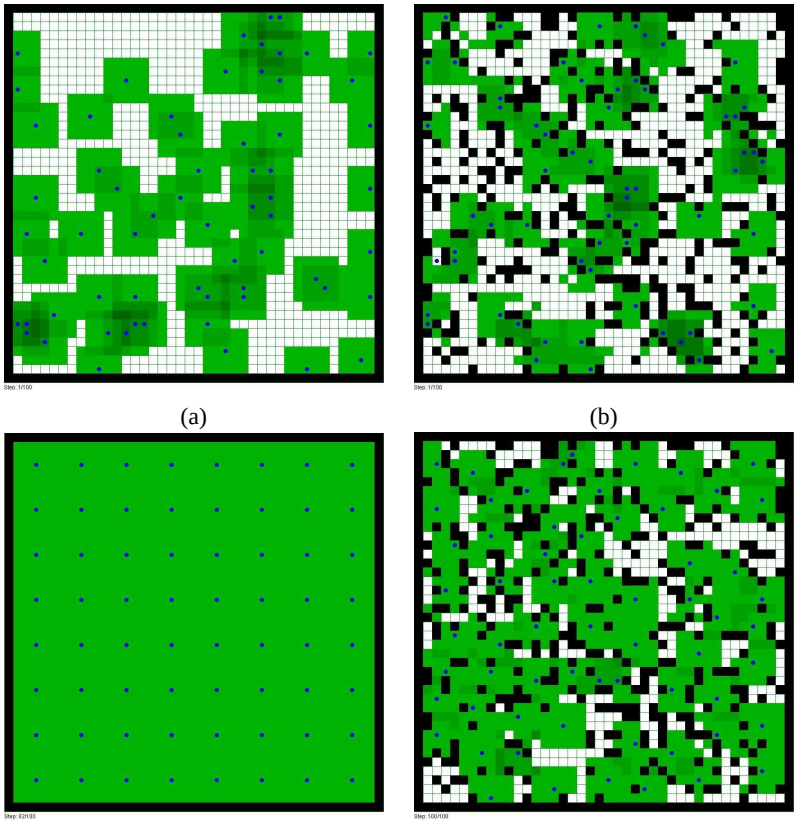
Scenario parameters	
Grid height (h)	40 cells
Grid width (w)	40 cells
Number of nodes (n)	32÷96
Sensing radius (r)	2 cells
Percentage of obstacles (o)	0÷20 %
Maximum number of time steps (#time_steps_max)	100
Number of runs (runs)	10
Neural network parameters	
Total number of neurons (N)	13
Number of input neurons (I)	9
Number of hidden neurons (H)	2
Number of output neurons (O)	2
Genetic algorithm parameters	
Population size (P)	300 chromosomes
Number of generation (g)	100 generations
Percentage of elite selection (e)	15%
Percentage of mutation (mu)	45%
Percentage of crossover (c)	30%
Percentage of randomly created offsprings (off_c)	5%
Percentage of randomly selecting an offspring from previous generation (off_s)	5%

For our quantitative analysis, we will take into account only the physical presence (and not the shadow effect) of the obstacles by using the term *coverable area*, which indicates the total number of cells where no obstacles are present. Before starting the real simulation, we ran some test simulations in order to determine the right number of time steps and generations needed to reach a stable coverage. These results are illustrated in the next subsection, but the values used for the simulations are reported in Table 1, along with the parameters related to the neural network setup and the genetic algorithm setup. All the results have been averaged over 10 different runs in order to respect a confidence interval of 95%.

4.2. Results

A representation of the initial random deployment of the nodes is reported in Figure 4a and 4b, the figures show the scenario with 64 nodes, no obstacles (a) and 10% of obstacles (b), respectively. The figures are also useful to understand how the discretization has been realized, how the coverage area of a node and the overlapping areas are intended, and finally how obstacles impact on movement, coverage and communication. In Figure **??**c and **??**d we show the snapshot of the same scenarios after a training phase of 100 generations. It this possible to appreciate how

nodes have been able to learn the correct placement, in order to cover the whole field when no obstacle are present (c) and as much as possible when obstacles are present (d).



(a) (b) (c) (d)

Figure 4: Initial random deployment of 64 nodes ($r = 2$ cells) in a $40x40$ cells grid with no obstacles (a) and with 10% of obstacles (b). WSRN self-deployment using the evolved neural network with no obstacles (c), and with 10% of obstacles (d).

Before discussing the results about the coverage, we want to justify our choice to use 100 time steps and 100 generation in the neural network simulation and the genetic algorithm, respectively.

Figure 5a shows the achieved coverage (in % in respect of the achievable coverage) after 100 time steps for variable number of nodes (32, 64 and 96) and percentage of obstacles (0%, 10% and 20%). As we can see, after a few number of time steps (less than 30), nodes have learnt enough to reach a stable placement and they are not be able to improve furtherly their coverage. Only the case with 64 nodes and no obstacles shows that nodes continue learning till the 80th time step. In any case, 100 steps is a good value for all the simulated scenarios. In the same way, Figure 5b shows the value of the fitness function (that includes coverage and "speed" in covering the area) in the same aforementioned scenario. In all the simulated scenarios, the neural network is able to learn how to maximize the fitness function after about 25 generation, therefore 100 generations seems a good termination condition for the training phase.

From Figure 5b we can also appreciate other valuable information, which is the "quantity" of information that generation by generation is learnt from the nodes, and also their "speed" in learning new information. In order to give a quantitative idea of these two information, we defined the index IL, which stands for incremental learning, as follows:

$$IL = \frac{fitness(g) - fitness(1)}{fitness(100)} \tag{5}$$

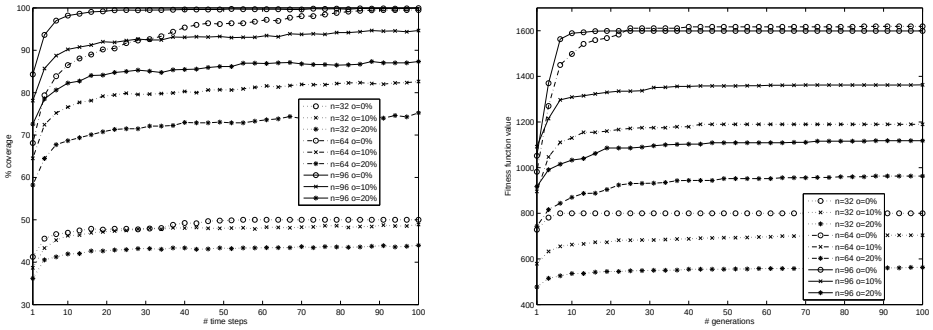


Figure 5: Progress of the achieved coverage over the time by using the neural network (a). Evolution of the neural network over generations in terms of achieved fitness value (b).

where $fitness(x)$ is the value of the fitness function at the generation x. In Table 2, we reported the values of IL for all the simulated scenario, when the generation increases. We can observe that when the number of nodes is enough to cover the whole field (64 and 96 nodes), an increase in the percentage of obstacles affects the capacity to learn. This is not the case when the number of nodes is low (32 nodes) and they have more space to move without interfering with each other. When no obstacles are present, the configuration with 64 nodes is the "optimal" to cover the whole simulated field with no overlapping areas. In the Table we can see that in the configuration with 64 nodes, nodes are able to learn more and quicker than the corresponding configurations with 32 and 96 nodes. This means that an "optimal" number of nodes turns into an optimal number of learners. Improvements in the learning process would not come from an increase in the number of nodes beyond the optimal value but by an evolutionary step, such as a new input for the neural network on the nodes.

Table 2: Incremental learning generation-by-generation for different scenarios

g	n=32 o=0	n=32 o=10	n=32 o=20	n=64 o=0	n=64 o=10	n=64 o=20	n=96 o=0	n=96 o=10	n=96 o=20
5	0.06625	0.08665	0.07993	0.24768	0.15798	0.09034	0.29268	0.11959	0.07245
10	0.09000	0.11932	0.10480	0.31872	0.19664	0.13188	0.33583	0.15921	0.10376
20	0.09000	0.14631	0.12078	0.36195	0.22185	0.17342	0.34209	0.17461	0.15116
30	0.09000	0.14773	0.12966	0.38851	0.23445	0.19626	0.34209	0.19002	0.16011
40	0.09000	0.15483	0.13854	0.39222	0.23782	0.20872	0.34209	0.19369	0.16637
50	0.09000	0.16193	0.13854	0.39222	0.24706	0.21703	0.34209	0.19516	0.17174
60	0.09000	0.16761	0.14387	0.39222	0.24706	0.21703	0.34209	0.19736	0.17174
70	0.09000	0.17188	0.14387	0.39222	0.24706	0.22118	0.34209	0.19809	0.17352
80	0.09000	0.17188	0.14565	0.39222	0.24706	0.22534	0.34209	0.19809	0.17800
90	0.09000	0.17614	0.14920	0.39345	0.24706	0.22845	0.34209	0.19809	0.17979
100	0.09000	0.17756	0.15275	0.39345	0.24706	0.22845	0.34209	0.19883	0.17979

Figure 6a shows the percentage of coverage achieved calculated in respect of the coverable area, as ratio of the number of covered cells over the number of total cells in the grid, except the obstacles. The neural network approach is compared with the maximum achievable coverage, the coverage achieved by a random deployment, which we consider as the best and the worst case, respectively and an approach based on the concept of virtual forces among the nodes, the VFA [29]. The figure is drawn for 10% of obstacles when the number of nodes increases. From Figure 6a, we can see that for very low and very high number of nodes the curve of the neural network approach and the maximum achievable coverage are very close to each other. For intermediate number of nodes the two curves are more distant, in particular for 64 nodes we have both the best improvement in respect of the random deployment and the highest distance from the maximum achievable. VFA follows the same shape of our approach, but in the same conditions is not able to reach the same coverage. This behavior is even more evident in Figure 6b, where we plotted the coverage efficiency of a single node, in terms of average number of cells covered by a node. Also in this figure the neural network approach, the random deployment and the maximum achievable, are plotted when the node number

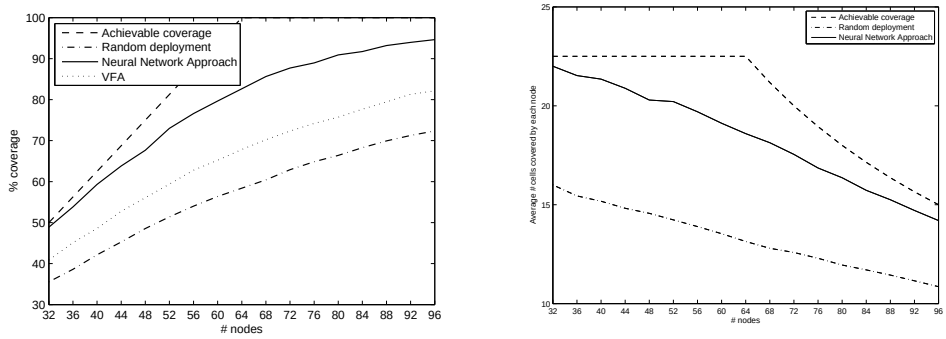


Figure 6: Comparison of achieved coverage percentage (a) and nodes coverage efficiency (b) between the neural network approach, the random deployment, the maximum achievable coverage and the VFA, when the number of nodes increases and 10% of obstacles are present.

increases. Note that the average value of maximum achievable coverage per node is constantly 22.5 cells (and not 25, because 10% of obstacles is uniformly ditributed on the field) for number of nodes lower than 64, and then it decreases more than linearly, whereas the coverage efficiency for the neural network approach decreases linearly in the interval considered.

For all the aforementioned considerations about the configuration with 64 nodes, and because it is the exact number of nodes to cover the whole considered area without overlapping, we decided to analyse its performance more in detail. In Figure 7 we plot the achieved coverage in respect of the percentage of obstacles present on the field for the neural network approach, the random deployment and the VFA. We have not plotted the achievable coverage, because in the scenarios with 64 nodes the whole area (100%) is coverable. Obviously, in the presence of obstacle, our neural network has the possibility to exploit the learning capabilities and outperform the random deployment approach. The interesting consideration is that for an increasing percentage of obstacles, the neural network approach decreases its performance more than linearly. As we already mentioned, the presence of a one-cell obstacle counts almost as two cells lost in the coverage, because of the physical impediment and the shadow effect, therefore a high percentage of obstacles strongly limits the evolutionary capabilities of the nodes. The only solution to change the slope of the curve in Figure 7 is to allow nodes to make an evolutionary step by including more input related to the presence of obstacles. Concerning the VFA, we considered the obstacles in an explicit fashion and we give to the obstacles a repulsion force. As we can observe from the 7, VFA follows the same shape of our approach, but the total coverage that is able to reach is always smaller than our scheme.

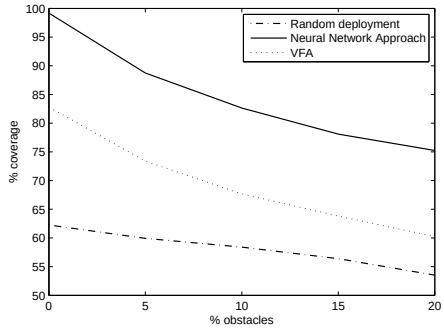


Figure 7: Comparison of achieved coverage percentage between the neural network approach, the random deployment and the VFA, when the number of nodes is fixed to 64 and the percentage of obstacles varies.

Finally, we want to understand the effect of the time in the learning process of the nodes. We designed the fitness

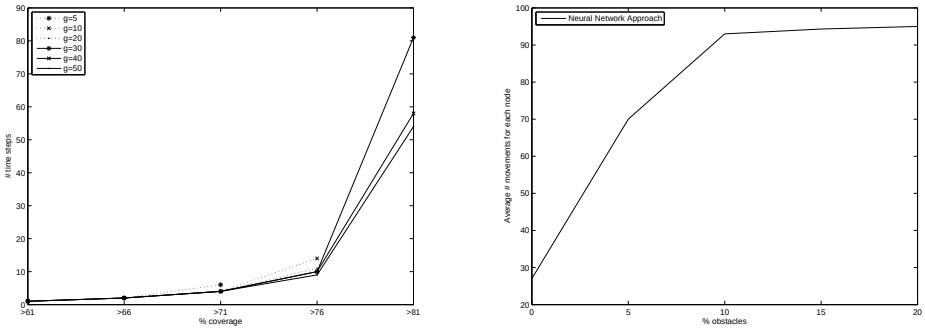


Figure 8: Time step in which a given coverage percentuage threshold is reached for the scenario with 64 nodes and 10% of obstacles (a), Average number of movements needed by each node to reach a stable coverage, when the number of nodes is fixed to 64 and the percentage of obstacles varies (b).

function in a way that would award quick populations to transmit their genes to successive generations. Figure 8a shows the time step when a given coverage threshold is achieved by the population of different generations. The scenario is still 64 nodes and 10% of obstacles. We can see that, when the generation increases, higher percentage of field are covered by the nodes, and, above all, that the same percentage of coverage is achieved in a shorter time. This is exactly the behavior we tried to impress into population genes by including the time into the fitness function. Since we claim that the proposed neural network approach is feasible and appliable to a real case, we also investigated the number of nodes movements needed to reach the expected coverage. In Figure 8b, we plotted the movements needed in the scenario with 64 nodes and an increasing percentage of obstacle. The number of movements gives a measure of the energy needed for the deployment. In our scheme only one movement is admitted for each time step. Thus, the energy consumption is implicitly reduced when the the number of time steps needed to reach a certain coverage is reduced, by exploiting the learning capabilities of the nodes.

5. Conclusion and future direction

In this work an algorithm for nodes self-deployment aimed to maximize coverage in WSRN has been proposed. The algorithm is based on neural networks and genetic algorithms. Results show how the evolving neural network approach is suitable to solve the given problem even in presence of obstacle. The approach used represents a synthesis of the most usual approaches, because it uses concepts belonging to all the three most important categories of moviment-assisted deployment algorithms. But, while in the usual approaches, the behavior of the nodes is pre-programmed and can lead to unwanted situations, in the neural network approach the behavior emerges from the learning process and the interactions with the surrounding environment. The proposed algorithm achieves an high coverage of the field while minimizing the time steps needed, and consequently the number of movements and the energy consumption. The most interesting observation is that the same approach can be used, by introducing few modifications, to solve different problems and pursue different objectives. Future works can include: the definition of new input for the neural network in order to allow an evolutionary step and improve the coverage in presence of a massive quantity of obstacle: the determination of a new neural network for the implementation of another self-* property for autonomous WSRNs, and the design of new algorithms for several simultaneous objectives.

[1] J. Wu, Handbook on theoretical and algorithmic aspects of sensor ad hoc wireless, and peer-to-peer networks, Auerbach Publication, US.
[2] B. Carbunar, A. Grama, J. Vitek, O. Carbunar, Redundancy and coverage detection in sensor networks, ACM Transaction on Sensor Networks 2 (2006) 94–128.
[3] H. Gupta, Z. H. Zhou, S. R. Das, Q. Gu, Connected sensor cover: Self-organization of sensor networks for efficient query execution, IEEE/ACM Transaction on Networks 14 (2006) 55–67.
[4] S. Chellappan, W. Gu, X. Bai, D. Xuan, B. Ma, K. Zhang, Deploying wireless sensor networks under limited mobility constraints, IEEE Transactions on Mobile Computing 6 (10) (2007) 1142–1157.
[5] G. Wang, , G. Cao, T. La Porta, Movement-assisted sensor deployment, IEEE Transactions on Mobile Computing 5 (2006) 640–652.

[6] Y. C. Wang, C. C. Hu, Y. C. Tseng, Efficient deployment algorithms for ensuring coverage and connectivity of wireless sensor networks, Proceedings of the First International Conference on Wireless Internet (2005) 114–121.
[7] M. A. Batalin, G. S. Sukhatma, The design and analysis of an efficient local algorithm for coverage and exploration based on sensor network deployment, IEEE Transaction Robotics 23 (2007) 661–675.
[8] X. Bai, D. Xuan, Z. Yun, T. H. Lai, W. Jia, Complete optimal deployment patterns for full-coverage and k-connectivity ($k \leq 6$) wireless sensor networks, in: Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing, MobiHoc '08, 2008, pp. 401–410.
[9] C. Wu, Y. Lee, K.C.and Chung, A delaunay triangulation based method for wireless sensor network deployment, Computer Communications 30 (2007).
[10] S. Meguerdichian, F. Koushanfar, M. Potkonjak, M. B. Srivastava, Coverage problems in wireless ad-hoc sensor networks, in: INFOCOM 2001. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies, 2001, pp. 1380–1387.
[11] J. Chen, E. Shen, Y. Sun, The deployment algorithms in wireless sensor net works: A survey, Information Technology Journal 8 (3) (2009) 293–301.
[12] B. Wang, H. B. Lim, D. Ma, A survey of movement strategies for improving network coverage in wireless sensor networks, Computer Communications 32 (13-14) (2009) 1427– 436.
[13] N. A. A. Aziz, K. A. Aziz, W. Z. W. Ismail, Coverage strategies for wireless sensor networks, World Academy of Science, Engineering and Technology 50 2009.
[14] Y. Zou, K. Chakrabarty, Sensor deployment and target localization based on virtual forces, in: In IEEE INFOCOM'03, Vol. 2, 2003, pp. 1293–1303.
[15] X. Shen, J. Chen, Z. Wang, Y. Sun, Grid scan: A simple and eective approach for coverage issue in wireless sensor networks, in: IEEE International Communications Conference, Vol. 8, 2006, pp. 3480–3484.
[16] Y. C. Wang, C. C. Hu, Y. C. Tseng, Efficient placement and dispatch of sensors in a wireless sensor network, IEEE Transactions on Mobile Computing 7 (2) (2008) 262–274.
[17] G. Wang, G. Cao, T. Porta, Movement-assisted sensor deployment, in: In IEEE INFOCOM'04, Vol. 4, 2004, pp. 2469–2479.
[18] S. Megerian, F. Koushanfar, M. Potkonjak, M. Srivastava, Worst and best-case coverage in sensor networks, IEEE Transaction on Mobile Computing 4 (2005) 84–92.
[19] C.-Y. Chang, C.-T. Chang, Y.-C. Chen, H.-R. Chang Obstacle-resistant deployment algorithms for wireless sensor networks, IEEE Transaction on Vehicular Technology 58 (6) (2009) 2925–2941.
[20] Y. Zou, K. Chakrabarty, Sensor deployment and target localization in distributed sensor networks, ACM Transactions on Embedded Computing Systems 3 (2004) 61–91.
[21] N. Bartolini, T. Calamoneri, E. G. Fusco, A. Massini, S. Silvestri, Push&pull: autonomous deployment of mobile sensors for a complete coverage, Wireless Networks 16 (2010) 607–625.
[22] N. Heo, P. Varshney, A distributed self spreading algorithm for mobile wireless sensor networks, in: IEEE Wireless Communications and Networking Conference, Vol. 3, 2003, pp. 1597–1602.
[23] F. Arvin, K. Samsudin, A. R. Ramli, A short-range infrared communication for swarm mobile robots, in: Proceedings of the 2009 International Conference on Signal Processing Systems, IEEE Computer Society, 2009, pp. 454–458.
[24] J. Majchrzak, M. Michalski, G. Wiczynski, Distance estimation with a long-range ultrasonic sensor system, IEEE Sensors Journal 9 (2009) 767–773.
[25] H. Zhang, L. Wang, R. Jia, J. Li, A distance measuring method using visual image processing, in: 2nd International Congress on Image and Signal Processing, CISP '09, 2009, pp. 1–5.
[26] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, G. Gill, Integrated coverage and connectivity configuration in wireless sensor networks, in: The 1st International Conference on Embedded Networked Sensor System, 2003, pp. 28–39.
[27] W. Elmenreich, G. Klinger. Genetic evolution of a neural network for the autonomous control of a fourwheeled robot, in: The Sixth Mexican International Conference on Artificial Intelligence (MICAI'07), Aguascalientes, Mexico, November 2007.
[28] A. Laouiti, A. Qayyum, L. Viennot, Multipoint relaying: an efficient technique for flooding in mobile wireless networks, in: 35th Annual Hawaii International Conference on System Sciences HICSS, 2002, pp. 3898–3907.
[29] Howard, A.; Mataric, M. J.; Sukhatme, G. S.: Mobile sensor network deployment using potential field: a distributed scalable solution to the area coverage problem. Proc. of Int. Symp. Distrib. Auton. Robot. Syst. 2002, 299-308.