An Efficient Partial Coverage Algorithm for Wireless Sensor Networks

Habib Mostafaei[1], Antonio Montieri[2], Valerio Persico[3], Antonio Pescapé[2,3]
[1]Università degli Studi Roma Tre (Italy), [2]NM2 srl (Italy), [3]Università di Napoli Federico II (Italy)
habib.mostafaei@uniroma3.it, montieri@nm-2.com, {valerio.persico, pescape}@unina.it

***Abstract*—Wireless sensor networks (WSNs) are currently adopted in a vast variety of domains. Due to practical energy constraints, in this field minimizing sensor energy consumption is a critical challenge. Sleep scheduling approaches give the opportunity of turning off a subset of the nodes of a network—without suspending the monitoring activities performed by the WSN—in order to save energy and increase the lifetime of the sensing system. Our study focuses on *partial coverage*, targeting scenarios in which the continuous monitoring of a limited portion of the area of interest is enough. In this paper, we present PCLA, an efficient algorithm based on Learning Automata that aims at minimizing the number of sensors to activate, such that a given portion of the area of interest is covered and connectivity among sensors is preserved. Simulation results show how PCLA can select sensors in an efficient way to satisfy the imposed constraints, thus guaranteeing better performance in terms of both working-node ratio and WSN lifetime. Also, we show how PCLA outperforms state-of-the-art partial-coverage algorithms.**

***Keywords*-Partial Coverage, Sensor Scheduling, Learning Automata (LA), Wireless Sensor Networks (WSNs)**

I. Introduction

Wireless sensor networks (WSNs) have gained the attention of the research community in the last years and can currently be adopted in a vast variety of domains such as surveillance, health care, and environmental monitoring [1]. Indeed, they have revealed to be a pillar for the Internet of Things and the variety of smart applications stemming out from it [2]–[4].

Wireless network performance [5], [6] and sensing system lifetime are critical concerns in many typical applications, even though WSNs are made up of nodes of low energy. The placement of nodes in improper places and difficulties in changing batteries further exacerbate the lifetime issue. Therefore, strategies for the optimal energy consumption are essential, especially considering that WSNs cannot properly work after a fraction of nodes has run out of energy. Node activity scheduling, i.e. the ability of temporarily turning off just a part of deployed nodes without suspending the monitoring activities performed by the WSN, represents a way to save energy under given constraints (e.g., area coverage, redundancy requirements, etc.) [7].

While *full coverage* applications of WSNs require 100% of the area of interest to be monitored, monitoring only a limited percentage of it is enough for some other applications. This approach is commonly known as *partial coverage* [8]. For instance, the requirements of a WSN aimed at monitoring the environmental temperature or the humidity can be satisfied when just 90% of the zone of interest is covered [9].

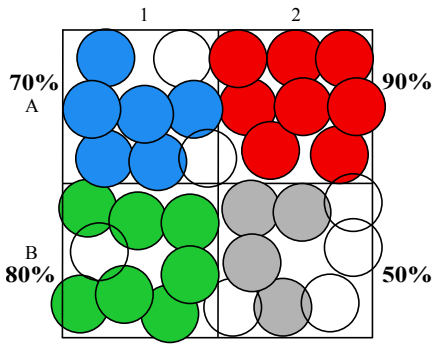


Fig. 1: Partial Coverage with four sub-regions.

Being subjected to more relaxed constraints, partial coverage scheduling is able to guarantee a longer lifetime to a WSN sacrificing some aspects in return. Figure 1 reports a generic example for the partial coverage approach. The figure shows a zone of interest divided into four portions requiring different levels of coverage. For instance, on the one hand, A2 has 90% coverage requirement being a critical area. On the other hand, monitoring 50% of B2 is enough. If having control on sensors placement, more sensors in critical areas could be scattered. For instance, more nodes could be deployed in A2 and less in B2, such that the equipment cost can be reduced and network lifetime can be prolonged under the same hardware cost. Unfortunately, this situation is uncommon: once sensors have been randomly scattered (e.g, from an airplane), a proper solution has to be found *ex post*.

Considering that each sensor is able to cover a certain area, according to its sensing range, partial coverage approaches aim at identifying which sensors have to be activated, such that the overall covered area for each sub-regions respects existing constraints. Moreover, given that wireless sensors have a limited communication range, often applications have connectivity requirements, i.e. each active sensing node has to be placed in the communication range of at least another active node.

In this paper, we investigate the problem of partial coverage in WSNs and propose PCLA (Partial Coverage with Learning Automata), a novel and efficient algorithm to face it. In this study, we assume that all sub-regions have equal coverage requirements and each node has the same sensing and communication capabilities. The proposed solution takes advantage of Learning Automata (LA) to properly schedule sensors into active and inactive state in order to extend the network lifetime. In more details, PCLA by using a number of nodes

first creates a backbone. Then, these nodes use their neighbors to meet the network coverage requirements and connectivity. Simulation results show how PCLA can select sensors in efficient way to satisfy partial coverage requirements, thus guaranteeing better performance in terms of number of active nodes and improving WSN efficiency.

The reminder of this paper is organized as follows. In Section II we survey the related literature; Section III reports the formal definition of *partial coverage* problem and its related concepts; Section IV introduces LA; Section V presents the *PCLA* algorithm to solve the *Partial Coverage* problem. Simulation results are illustrated in Section VI. Finally Section VII reports the concluding remarks.

II. Related Work

This section aims at providing an overall picture of the literature related to the problem of the partial coverage (known also as *p-percent coverage*) in WSNs. Coverage problems in WSNs have been widely studied during recent years. Wang [1] recently surveyed them. Area coverage problems can be divided into *full coverage* and *partial coverage*. Some recent works about this topic can be found in [7], [10]. Although several works about full coverage in WSNs exist, to the best of our knowledge a limited number of solutions has been proposed in the area of *partial coverage*. Gao *et al.* [11] devised two algorithms for partial coverage of WSNs to extend the network lifetime. Their first algorithm is a centralized approach to prolong the network lifetime, while the second solves the problem in a distributed fashion. Both their algorithms can maintain the network connectivity while monitoring the network area. Li *et al.* in [12] devised two methods to preserve partial coverage in WSNs. Their algorithms can guarantee both coverage and connectivity requirements but failed to achieve low time complexity. The concept of Connected Dominating Set (CDS) has widely used. A CDS-based algorithm can be found in [13]. Authors used CDS concept to create a virtual backbone in a network. Their approaches are not used in partial coverage. Wu *et al.* [14] also presented two algorithms for addressing this problem. The first algorithm is named pPCA and is a greedy based. The second one is called CpPCA-CDS and implements a distributed approach. CpPCA-CDS approach is based on CDS to address connected partial coverage problem in WSNs. The main drawback of this work is that its performance depends on DFS search. Therefore, time complexity of their algorithm increases with applying DFS search to find the solutions. In some works authors have also used neighbors information to preserve partial coverage and connectivity in WSNs. Yardibi and Karasan [8] developed a Distributed Adaptive Sleep Scheduling Algorithm (DASSA) for WSNs with partial coverage. In their devised approach each node uses the remaining energy levels and a feedback from the sink node to schedule the activity of its neighbor nodes. However, if a node could not obtain this information from the sink node it is unable to schedule its neighbor nodes. Probabilistic way is another approach to study partial coverage in WSNs. The approach proposed by Gupta *et al.* in [15] is fully distributed and each sensor node does not need any geographical information to find redundant nodes and put them to the sleep state. However the algorithm does not guarantee the connectivity of sensor nodes. Identification of redundant sensors based on a geometric approach is considered in [16]. Hafeeda and Ahmadi [17] studied the coverage problem under both disk sensing and probabilistic sensing models and devised Probabilistic Coverage Protocol (PCP). The PCP computes the maximum possible distance between sensors to ensure that there are no holes in coverage.

In this paper, we focus on the *partial coverage* problem in WSNs. We use LA to find a proper subset of sensor nodes to assure *partial coverage*. The main objective of *PCLA* is to use the smallest number of sensors at any given time to monitor the network area with the desired percentage of coverage. PCLA is able to preserve both coverage and connectivity. In more detail, PCLA uses the coverage graph of the network to select backbone nodes. The selected backbone nodes rely on their neighbors to obtain and preserve partial coverage.

III. Preliminaries and definitions

In this section, we introduce the main concepts and supply the basic definitions for *partial coverage* problem.

A *WSN* is modeled by an undirected connected graph, namely *Coverage Graph* $CG = (V, E)$, where $V = \{S_0, S_1, \ldots, S_N\}$ includes all the nodes randomly deployed in the network including the sink S_0. Each node can sense every event that occurs within its sensing range R_s, and can communicate with other nodes within its communication range R_c. Sensing and communication ranges are defined as the disks with radius R_s and R_c, respectively. E represents the set of the communication links between these nodes. For any node u and v, the edge $(u, v) \in E$ if and only if u and v are within the communication range of each other.

Given a region of interest ϑ whose area is equal to A_ϑ, and a WSN made up of randomly scattered nodes, each having a sensing range R_s—and thus able to cover an area πR_s^2—the *partial coverage problem* consists in identifying a convenient subset of nodes to be activated such that the active nodes are able to cover a given portion P_s of the area of interest.

Some useful metrics in this framework are: (i) the *Average Region Coverage Degree* [14], i.e. $D_\vartheta = \frac{N\pi R_s^2}{A_\vartheta}$, where N is the number of sensors deployed in region ϑ, each having a sensing range R_s; (ii) the *Working-node Ratio* [14], i.e. the fraction $\frac{|\Psi|}{N}$, where Ψ is the set of the active nodes able to cover the fraction P_s of the area of interest ϑ. The former is an index of the resources (i.e. sensing nodes) scattered on the region of interest and takes into account also their sensing capabilities (i.e. the sensing range); the latter is an index of the efficiency of the coverage algorithm.

Formal definition for Partial Coverage problem. Given a two-dimensional region of interest ϑ and a WSN made up of N sensors, the WSN partial coverage problem can be defined as "finding a connected set of nodes $\Psi \subseteq V$ such to minimize $\phi = \frac{|\Psi|}{N}$ and guarantee the coverage of the desired portion $P_s A_\vartheta$ of the region of interest".

Objective: Minimize number of nodes in ϕ, subjected to:

- Ψ is a connected set of nodes;

- Ψ covers at least the area $P_s A_\vartheta$ of ϑ.

Symbols and definitions are summarized in Table I.

TABLE I: Symbols and definitions.

ϑ	Region of interest
A_ϑ	Total area of the region of interest
P_s	Portion to cover of the region of interest
N	Number of sensing nodes
R_s	Nodes' sensing range
R_c	Nodes' communication range
Ψ	Connected set of nodes that guarantees partial coverage

IV. Learning automaton

An automaton is a machine designed to automatically follow a predetermined sequence of operations or respond to encoded instructions. Learning Automata (LA) do not follow predetermined rules, but adapt to changes in the Random Environment (RE). This adaptation is the result of the learning process.

LA are designed to select optimal actions among the set of allowable actions. In more details, a learning automaton has a finite number of actions that can operate. A probability is associated to each of them. Once an action is applied to the environment, the latter generates a reinforcement signal. The reply generated by the environment is used by the automaton to update its action probability vector. By running this procedure, the automaton learns to optimally choose actions among its action-set. The environment is described as a triple $E = \{\alpha, \beta, c\}$ where $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_N\}$ indicates the finite input set (i.e. the actions), $\beta = \{\beta_1, \beta_2, ..., \beta_N\}$ indicates the output set (i.e. the reinforcement signals), and $c = \{c_1, c_2, \ldots, c_N\}$ indicates a set of penalty probabilities, where each element c_i corresponds to one input of action α_i. The probability of action α_i is $p_i(n)$, and the corresponding vector $p(n)$ defines the action probability vector.

For our solution, we consider variable-structure automata [18] and P-model environment (i.e. we assume that β_i can be either 1 or 0).

A learning algorithm T can be defined as in Equation 1:

$$p(n+1) = T[p(n), \alpha(n), \beta(n)] \tag{1}$$

where $p(n)$ and $p(n+1)$ are the action probability vector at the n^{th} and $(n+1)^{th}$ cycle, respectively. The automaton operates as follows. Based on the action probability vector $p(n)$, the automaton randomly selects an action $\alpha_i(n)$, and performs it on the environment. After receiving the environment's reinforcement signal, automaton updates its action probability vector based on Equation 2, and Equation 3:

$$\begin{aligned} p_i(n+1) &= p_i(n) + a(1 - p_i(n)) \\ p_j(n+1) &= (1-a)p_j(n) \quad \forall j, j \neq i \end{aligned} \tag{2}$$

$$\begin{aligned} p_i(n+1) &= (1-b)p_i(n) \\ p_j(n+1) &= \frac{b}{r-1}(1-b)p_j(n) \quad \forall j, j \neq i \end{aligned} \tag{3}$$

where $p_i(n)$ and $p_j(n)$ are the probabilities of action α_i and α_j, respectively, and r is the number of actions. In these two equations, a and b are the reward and the penalty parameter, respectively.

V. PCLA Algorithm

In this section, we describe PCLA algorithm to address partial coverage in WSNs. The main idea behind PCLA is to first select a set of nodes as *backbone nodes*. Then, if partial coverage is not satisfied, additional nodes are selected and activated. Accordingly, our approach consists of two main phases: (i) *learning phase* and (ii) *partial coverage phase*. We provide more details of them in the following.

A. Learning phase

The aim of this phase is selecting the best *backbone nodes* set. A set with the minimum number of nodes is the best set in PCLA. Let Ψ denote the cover set PCLA plans to build. The main goal is to find redundant sensors in the network area A_ϑ, i.e. sensors having a covered area that can be covered with other sensors in their neighborhood. To this aim, at every time the sensor with maximum *coverage increment* C_i—defined as the increment of coverage when node i becomes a working node—is added to Ψ.

Initialization. In the initialization phase, PCLA on each node first gets a snapshot from CG in order to know node's neighbors. This is a key step in PCLA, because it uses CG of network to find suitable nodes to meet *partial coverage* requirements.

Initially, all the nodes are in the active state. For each node, each action α_i means that the neighbor node i is selected to be a working node (i.e. to remain in the active state). The action probability vector $p(n)$ is initialized as follows:

$$p_i(n) = \frac{1}{r} \qquad \forall i \tag{4}$$

where r indicates the action-set count, which is equal to the number of neighbour nodes at this initialization step. For example, if node i has five neighbor nodes, the action probability vector for this node is initially set to $\{0.2, 0.2, 0.2, 0.2, 0.2\}$. This means that node i has five equiprobable actions.

Backbone nodes selection. Each node in the network is equipped with a LA that helps to find the most appropriate backbone nodes set, i.e. those nodes responsible for maintaining connectivity among nodes. To this aim, a node is selected and added to Ψ. LA of this node chooses—accordingly to its $p(n)$—an action among its action-set, i.e. one of its neighbors is selected as a working node. The selected neighbor is added to Ψ, while other unselected neighbors are added to another set Γ. Then, the selected node iterates the procedure by selecting one of its neighbors not already contained in Γ. This process continues until $|\Psi \cup \Gamma| = |V|$. Note that, after this step, each node in the CG belongs to either Ψ or Γ.

At this point, the learning algorithm inside PCLA has to decide on suitability of Ψ set. At each cycle n, the number of nodes in Ψ is compared with a threshold value (T_n). T_n can be initially set to the total number $|V|$ of deployed nodes.

If $|\Psi| < T_n$ all selected actions α_i in Ψ are rewarded from the environment ($\beta_i(n) = 0$). Otherwise, these actions get a penalty from the environment ($\beta_i(n) = 1$). Note that, being PCLA a learning algorithm, it needs some cycles to converge to a stable set. Therefore, this process continues until Ψ set remains fixed in some consecutive cycles. At the end of this phase, we have a set of backbone nodes in Ψ, which are able to preserve connectivity of selected nodes in PCLA algorithm.

The pseudo code for PCLA algorithm is reported in Algorithm 1.

Algorithm 1 PCLA Algorithm

Input:
CG ▷ Snapshot of the network
P_s ▷ Desired partial coverage
Output:
Ψ ▷ Set of selected nodes that guarantees the partial coverage
Parameters:
a ▷ Reward parameter for the update of the action probability vector, where $0 < a < 1$
T_n ▷ Threshold value
$|V|$ ▷ Total number of deployed nodes
Γ ▷ Set of unselected nodes
Initialization:
$p_i(0) = \frac{1}{r}\ \forall\ i$ ▷ r is the number of neighbors
repeat
A node is randomly selected and activated
Its automaton is denoted as S_i
repeat
while S_i has no possible actions **do**
Activated automata are traced back to find an automaton with available actions
end while
Ψ=$\Psi \cup S_i$
Automaton S_i selects one of its actions (a neighbor node S_j) accordingly to its $p(n)$
Each automaton prunes its action-set to avoid the loop
Automaton S_j is activated
Γ=$\Gamma \cup$ Unselected neighbors of S_i
$S_i = S_j$
until $|\Psi \cup \Gamma| < |V|$
if $|\Psi| < T_n$ **then**
$\beta_i(n)$ = 0
$T_n = |\Psi|$
else
$\beta_i(n) = 1$
end if
Enable all the disabled actions
until Ψ remains fixed in some consecutive cycles.
FormPartialCoverage()

B. Partial coverage phase

At the end of the learning phase, PCLA checks whether partial coverage is met. If partial coverage is not satisfied, `FormPartialCoverage()` routine is called. This function uses nodes in Γ to meet partial coverage requirement. At the end of this phase, nodes whose state is active will remain active to monitor the network, while other nodes will switch to idle state in order to save energy. These nodes have possibility to be active in the next round of algorithm, according to LA results.

The pseudo code of `FormPartialCoverage()` is shown in Algorithm 2.

C. PCLA Basic Property

Before presenting the experimental results of the proposed approach, we prove how PCLA preserves both *partial coverage* and connectivity.

Theorem 1: The obtained set Ψ from PCLA can preserve both *partial coverage* and connectivity.

Algorithm 2 `FormPartialCoverage()`

Parameters:
S_j ▷ Available node in Γ
repeat
for all S_j in Γ **do**
if Neighbors of S_j cannot cover S_j area **then**
Activate S_j
Ψ=$\Psi \bigcup S_j$
else
Deactivate S_j
end if
end for
until The desired partial coverage reaches

Proof: To prove this theorem, we firstly construct backbone nodes set based on LA. Then, LA of each node selects one of the actions among its action-set, accordingly to $p(n)$. As we described in the first paragraph of this section, action-set of each LA is formed based on node's neighbors. Therefore, selecting an action by LA of each node preserves node's connectivity, because it lies on the node's neighbors list. Algorithm 2 selects nodes among Γ. As we described in PCLA algorithm, it is obvious that each node in Γ has at least one neighbor in Ψ. Hence, obtained set Ψ from PCLA can preserve both *partial coverage* and connectivity. ■

VI. Performance evaluation

In this section, we provide a comprehensive performance evaluation of the proposed solution. The performance of PCLA is compared to other existing algorithms under varying conditions. Simulation results show that PCLA performs better than state-of-the-art partial coverage solutions in terms of working-node ratio, also for larger network sizes, and in terms of network lifetime it is able to guarantee.

TABLE II: Simulation parameters for the first set of experiments.

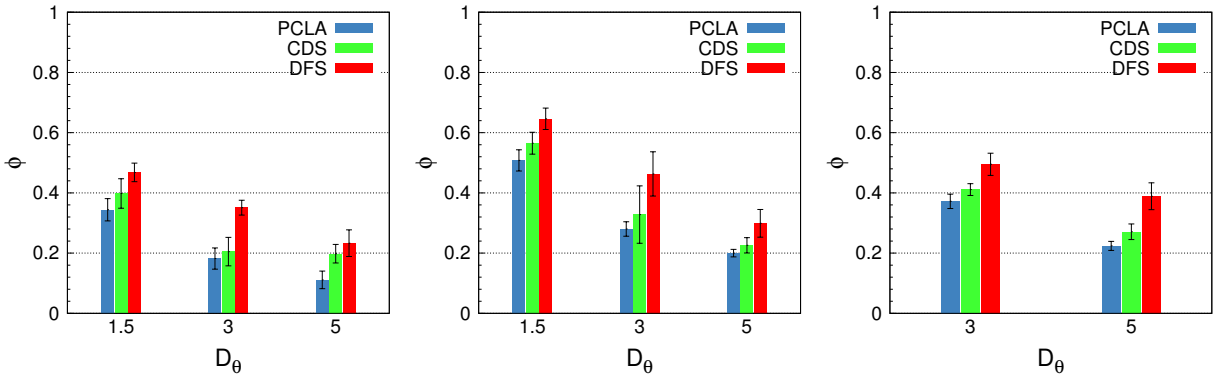
(a) Resource constraints.

Parameter	Values		
A_ϑ (m^2)	400×400		
R_s (m)	50		
R_c (m)	100		
T_n	100		
α	0.1		
N	31	63	105
D_ϑ	1.5	3.0	5.0

(b) Coverage requirements.

Parameter	Values		
P_s	0.6	0.8	1.0

Evaluation Setup. Performance evaluation has been performed through simulation using the WSN simulator [19]. PCLA has been compared to the CDS method proposed in [13] and the CP-PCA-DFS algorithm introduced in [14] (hereafter simply CDS and DFS, respectively). These works have been chosen since the approaches they propose model the network similarly to PCLA and use coverage graph to find a solution. The three algorithms have been compared considering different conditions, in terms of (i) network resources and (ii) coverage requirements. In more details, we have considered as inputs for our simulations: (i) the overall number N of randomly scattered nodes that make up the WSN; (ii) the overall area A_ϑ of the region of interest; (iii) the sensing range R_s and (iv) the communication range R_c of each node; (v)



(a) $P_s = 0.6$ (b) $P_s = 0.8$ (c) $P_s = 1.0$

Fig. 2: Impact of Average Region Coverage Degree (D_ϑ) on Working-Node Ratio for different values of P_s. ϕ exposes a decreasing trend on average for increasing values of D_ϑ. PCLA outperforms the other two algorithms in all the circumstances taken into account.

the coverage requirement P_s demanded to the algorithm. Note that the random placement of the nodes reflects the practical inability to place WSN elements in a controlled manner which derives from common practices (e.g., sensor deployment from an aircraft [20]). All nodes' sensing and communication ranges are assumed to be equal. To compute the network lifetime we used the approach adopted in [20]. All the obtained results have been averaged over 10 simulation runs.

Working-node ratio evaluation. In the first set of experiments, we aim at evaluating the effect of the *Average Region Coverage Degree* (D_ϑ) on the performance of PCLA. Note that—according to Section III—D_ϑ may also be (indirectly) considered an input for our simulations, as being function of N, A, and R_s. By varying the number of nodes, we have defined three different configurations, corresponding to different resource constraints: $D_\vartheta = 5$, $D_\vartheta = 3$, and $D_\vartheta = 1.5$ (see Table IIa). Moreover, we have tested the three solutions under three different coverage-requirement levels: $P_s = 0.6$, $P_s = 0.8$, and $P_s = 1.0$ i.e. full coverage (see Table IIb).

Figure 2 shows how the working-node ratio (ϕ) varies with D_ϑ for the different coverage requirements considered ($P_s = 0.6$, $P_s = 0.8$, and $P_s = 1.0$, respectively) and for the three algorithms. As expected, for all the three algorithms the simulation has reported that ϕ exposes a decreasing trend on average, for increasing values of D_ϑ. Note that results for $D_\vartheta = 1.5$ and $P_s = 1$ are missing because none of the algorithms satisfied connectivity requirements under this configuration. As shown in the figures, PCLA outperforms the other two algorithms, proving to be always the one exposing the lowest values of ϕ in all the circumstances taken into account. PCLA shows also the best relative decrement of ϕ when passing from $D_\vartheta = 1.5$ to $D_\vartheta = 5$. Indeed, the value of ϕ decreases by 67.7% (60.6%) when $P_s = 0.6$ ($P_s = 0.8$); this value corresponds to an improvement of -0.233 (-0.308) in absolute terms. When varying the coverage requirements, we note that increasing P_s has a detrimental impact on the performance of all the algorithms, as ϕ also increases. However, this impact is mitigated when D_ϑ is higher, i.e. by deploying a larger number of nodes. In more details when $D_\vartheta = 5$, PCLA has the lowest performance decrease when passing from $P_s = 0.8$ to $P_s = 1.0$.

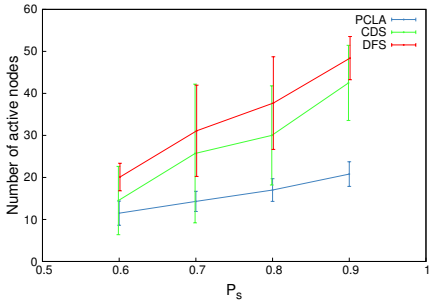
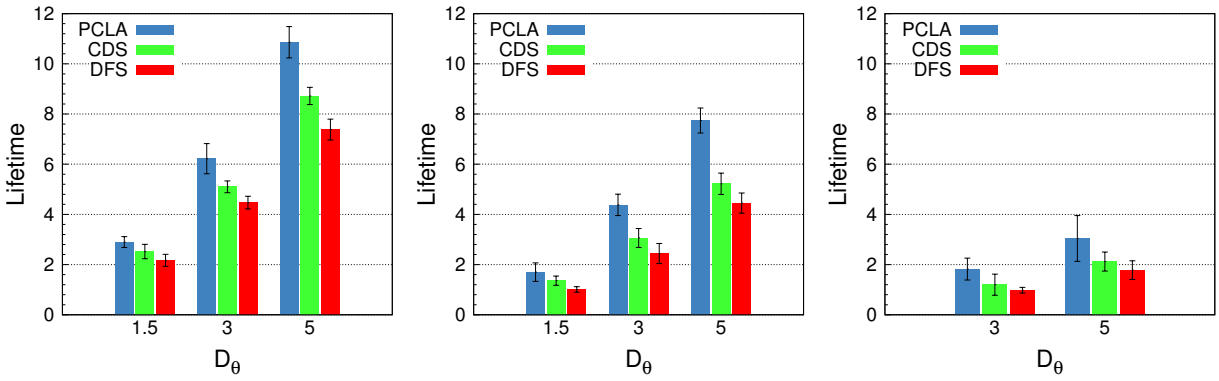


Fig. 3: Number of active nodes for different coverage requirements for a large network. ($N = 200$, $R_s = 50m$). As expected performance decreases for higher P_s. PCLA exhibits a better trend and outperforms other algorithms.

In the following we report the results of further investigations about the impact of the network size on the performance of the three algorithms, considering networks with a larger number of nodes. Figure 3 reports the number of active nodes in Ψ with a high number of nodes ($N = 200$) and coverage requirements varying from $P_s = 0.6$ to $P_s = 0.9$. As expected, for all the algorithms, the number of active sensors is higher when the parameter P_s increases, as a larger portion of the network area has to be monitored. On the other hand, this result shows how the performance of PCLA in terms of active nodes proved to decrease slower than the one of CDS and DFS when increasing coverage constraints. The improvement obtained by using PCLA respect to CDS (DFS) increases when increasing the value for P_s: it amounts in terms of active nodes—on average—to -3 nodes (-8.6) for $P_s = 0.6$, and reaches -21.7 nodes (-27.6) for $P_s = 0.9$.

Lifetime evaluation. As network lifetime—i.e. the time span from the networks initial deployment to the first loss of the required coverage—is a critical performance index for WSNs, our evaluation also took it into consideration. Figure 4 compares the lifetime obtained with the three approaches considered for different values of D_ϑ (taking values in $\{1.5,\ 3,\ 5\}$) and P_s (assuming values in $\{0.6,\ 0.8,\ 1.0\}$). As reported by



(a) $P_s = 0.6$ (b) $P_s = 0.8$ (c) $P_s = 1.0$

Fig. 4: Impact of D_ϑ on network lifetime for different values of P_s.

the analysis, we note that a larger Average Region Coverage Degree (D_ϑ) leads to a longer lifetime for all the approaches considered. On the contrary, the lifetime is shortened by an increase in coverage constraints. PCLA proved to perform better than both CDS and DFS in all the circumstances considered. In more details, the lifetime enhancement that PCLA is able to carry when adopted in place of CDS (DFS) ranges from +15% (+34%) to +52% (+86%).

VII. Conclusion

In this paper, we have investigated the partial coverage problem in WSNs and have devised an approach based on Learning Automata. We have proposed PCLA to find the minimum number of sensors to activate, in order to cover a given portion of the region of interest. Experimental results show that PCLA outperforms state-of-the-art algorithms by exposing the best performance in terms of working-node ratio and network lifetime in all the circumstances taken into account. PCLA achieves also the best relative performance enhancement when increasing the Average Region Coverage Degree. Furthermore, the trend of the performance decrease of PCLA observed when increasing coverage constraints, has proved to be slower than the ones obtained with the other evaluated solutions. Accordingly, the benefit obtained by using PCLA instead of the other algorithms increases when increasing the value for P_s. As a future work we aim at checking the performance of PCLA in the case of nodes with heterogeneous sensing ranges and adopting a probabilistic sensing model.

References

[1] B. Wang, "Coverage problems in sensor networks: A survey," *ACM Comput. Surv.*, vol. 43, no. 4, pp. 1–53, 2011.

[2] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[3] A. Botta, W. de Donato, V. Persico, and A. Pescapé, "On the integration of cloud computing and internet of things," in *Future Internet of Things and Cloud (FiCloud), 2014 International Conference on*. IEEE, 2014, pp. 23–30.

[4] A. Botta, W. de Donato, V. Persico, and A. Pescapé, "Integration of cloud computing and internet of things: A survey," *Future Generation Computer Systems*, vol. 56, pp. 684 – 700, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X15003015

[5] R. Karrer, I. Matyasovszki, A. Botta, and A. Pescapè, "Experimental evaluation and characterization of the *magnets* wireless backbone," in *Proceedings of the First ACM Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization, WINTECH 2006, Los Angeles, California, USA, September 29, 2006*, 2006, pp. 26–33.

[6] R. P. Karrer, I. Matyasovszki, A. Botta, and A. Pescapé, "Magnets-experiences from deploying a joint research-operational next-generation wireless access network testbed," in *Testbeds and Research Infrastructure for the Development of Networks and Communities, 2007. TridentCom 2007. 3rd International Conference on*. IEEE, 2007, pp. 1–10.

[7] H. Mostafaei, M. R. Meybodi, and M. Esnaashari, "A learning automata based area coverage algorithm for wireless sensor networks," *Journal of electronic science and technology*, vol. 8, no. 3, pp. 200–205, 2010.

[8] T. Yardibi and E. Karasan, "A distributed activity scheduling algorithm for wireless sensor networks with partial coverage," *Wirel. Netw.*, vol. 16, no. 1, pp. 213–225, 2010.

[9] M. Demirbas, K. Chow, and C. Wan, "Insight: Internet-sensor integration for habitat monitoring," in *World of Wireless, Mobile and Multimedia Networks, 2006. WoWMoM 2006. International Symposium on a*, 2006, pp. 6 pp.–558.

[10] Y. Qianqian, H. Shibo, L. Junkun, C. Jiming, and S. Youxian, "Energy-efficient probabilistic area coverage in wireless sensor networks," *Vehicular Technology, IEEE Transactions on*, vol. 64, no. 1, pp. 367–377, 2015.

[11] S. Gao, X. Wang, and Y. Li, "p-percent coverage schedule in wireless sensor networks," in *2008 Proceedings of 17th International Conference on Computer Communications and Networks*, Aug 2008, pp. 1–6.

[12] Y. Li, C. Ai, Z. Cai, and R. Beyah, "Sensor scheduling for p-percent coverage in wireless sensor networks," *Cluster Computing*, vol. 14, no. 1, pp. 27–40, 2011. [Online]. Available: http://dx.doi.org/10.1007/s10586-009-0088-9

[13] K. Donghyun, W. Yiwei, L. Yingshu, Z. Feng, and D. Ding-Zhu, "Constructing minimum connected dominating sets with bounded diameters in wireless networks," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 20, no. 2, pp. 147–157, 2009.

[14] Y. Wu, C. Ai, S. Gao, and Y. Li, *p-Percent Coverage in Wireless Sensor Networks*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2008, vol. 5258, book section 21, pp. 200–211. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-88582-5_21

[15] H. P. Gupta, S. V. Rao, and T. Venkatesh, "Sleep scheduling for partial coverage in heterogeneous wireless sensor networks," in *2013 Fifth International Conference on Communication Systems and Networks (COMSNETS)*, Jan 2013, pp. 1–10.

[16] H. Ammari and S. Das, "Centralized and clustered k-coverage protocols for wireless sensor networks," *Computers, IEEE Transactions on*, vol. 61, no. 1, pp. 118–133, Jan 2012.

[17] M. Hefeeda and H. Ahmadi, "Energy-efficient protocol for deterministic and probabilistic coverage in sensor networks," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 21, no. 5, pp. 579–593, May 2010.

[18] K. S. Narendra and M. A. L. Thathachar, *Learning automata: An introduction*. Prentice Hall, 1989.

[19] "Wireless Sensor Networks Simulator," http://www.djstein.com/Projects/Files/Wireless%20Sensor%20Network%20Simulator%20v1.1.zip, Jan. 2016.

[20] H. Mostafaei, M. Esnaashari, and M. Meybodi, "A coverage monitoring algorithm based on learning automata for wireless sensor networks," *Appl. Math. Inf. Sci.*, vol. 9, no. 3, pp. 1317–1325, 2015.