Journal of Network and Computer Applications 80 (2017) 67–78

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

A sleep scheduling approach based on learning automata for WSN partial coverage

Habib Mostafaei[a,*], Antonio Montieri[b], Valerio Persico[c], Antonio Pescapé[b,c]

[a] Roma Tre University, Italy
[b] NM2 s.r.l., Italy
[c] University of Napoli Federico II, Italy

ARTICLE INFO

Keywords:
Partial coverage
Sensor scheduling
Learning Automata (LA)
Wireless Sensor Networks (WSNs)
PCLA

ABSTRACT

Wireless sensor networks (WSNs) are currently adopted in a vast variety of domains where sensor energy consumption is a critical challenge because of the existing power constraints. Sleep scheduling approaches have recently attracted the interest of the scientific community, as they give the opportunity of turning off the redundant nodes of a network to save energy and prolong the lifetime of the network without suspending the monitoring activities performed by the WSN.

Our study focuses on the problem of *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, a novel algorithm that relies on Learning Automata to implement sleep scheduling approaches. It aims at minimizing the number of sensors to activate for covering a desired portion of the region of interest preserving the connectivity among sensors. Simulation results show how PCLA can select sensors in an efficient way to satisfy the imposed constraints, thus guaranteeing good performance in terms of time complexity, working-node ratio, scalability, and WSN lifetime. Moreover, compared to the state of the art, PCLA is able to guarantee better performance.

1. 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 (Wang, 2011). Indeed, they have revealed to be a pillar for the Internet of Things and the variety of smart applications stemming out from it (Atzori et al., 2010; Botta et al., 2014, 2016).

Wireless network performance (Karrer et al., 2006, 2007) and sensing system lifetime (Ren et al., 2007; Mao et al., 2007) are critical concerns in many typical applications, since WSNs are made up of nodes with low energy. The placement of nodes in improper places and difficulties in changing batteries further exacerbate lifetime-related issues. 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. Challenges are mainly related to determining whether some portions of the area of interest covered by a sensor are also covered by other sensors, and to determining the order of sensor activation or deactivation (Wang, 2011). 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.) (Akbari Torkestani, 2013).

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 is commonly known as *partial coverage* problem (Yardibi and Karasan, 2010). 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 (Demirbas et al., 2006). Partial coverage scheduling is able to guarantee a longer lifetime to a WSN placing more relaxed constraints on it. Fig. 1 reports a generic example for the partial coverage problem. The figure shows a zone of interest divided into four portions requiring different levels of coverage. For instance, on the one hand, B2 has 90% coverage requirement being a critical area. On the other hand, monitoring 50% of A2 is enough. If we had control on the placement of the sensors, we would scatter more sensors in B2 and less in A2 to either reduce equipment costs or prolong the network lifetime under the same hardware cost. Unfortunately, this

* Corresponding author.
E-mail addresses: mostafae@dia.uniroma3.it (H. Mostafaei), montieri@nm-2.com (A. Montieri), valerio.persico@unina.it (V. Persico), pescape@unina.it (A. Pescapé).

http://dx.doi.org/10.1016/j.jnca.2016.12.022
Received 6 July 2016; Received in revised form 15 November 2016; Accepted 6 December 2016
Available online 13 December 2016

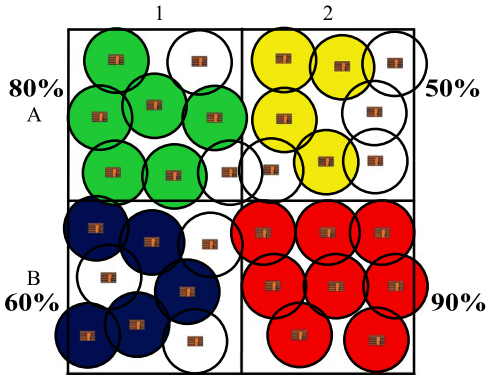


Fig. 1. Partial coverage with four sub-regions having different levels of coverage.

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 to respect the existing constraints. Moreover, since wireless sensors have limited communication ranges, applications often have connectivity requirements, i.e. active sensing nodes have to be placed in the communication range of another active node, at least.

In this paper, we extend our previous work presented in Mostafaei et al. (2016) that addresses the problem of partial coverage in WSNs and propose *PCLA* (Partial Coverage with Learning Automata), a novel and efficient algorithm. The proposed solution takes advantage of Learning Automata (LA) to properly schedule sensors into active or sleep state in order to extend the network lifetime. In more details, each node runs the PCLA algorithm that first creates a backbone by selecting a number of nodes and leveraging the coverage graph of the network. Then, these nodes use their neighbors to meet the network coverage and connectivity requirements. Simulation results show how PCLA can select sensors efficiently to satisfy partial coverage requirements, thus guaranteeing better performance in terms of the number of nodes to activate, with respect to state-of-the-art solutions.

This paper extends our previous work (Mostafaei et al., 2016) in a number of additional contributions: the correctness of the proposed solution has been formally demonstrated; the theoretical framework in which our work lies has been detailed, also providing analytical models; the theoretical complexity of the proposed algorithm has been investigated and compared to the state of the art; additional results related to both scalability of the solutions and lifetime have been reported in the proposed evaluation.

The reminder of this paper is organized as follows. In Section 2 we survey the related literature. Section 3 reports the formal definition of the problem of partial coverage and its related concepts. Section 4 introduces LA. Section 5 presents the PCLA algorithm to solve the partial coverage problem. Simulation results are illustrated in Section 6. Finally Section 7 reports the concluding remarks.

2. Related work

This section provides an overall picture of the literature related to the coverage problems in WSNs, specifically focusing on partial coverage.

Coverage problems have been widely studied in WSNs during recent years (Wang, 2011). Three main classes of problems can be identified: (i) *target coverage*, (ii) *barrier coverage*, and (iii) *area coverage*. The objective of target coverage is to monitor a set of targets with sensor nodes. Recently, novel solutions to this problem have been proposed (Mostafaei and Meybodi, 2013; Mostafaei et al., 2015; Mostafaei and Shojafar, 2015). Differently from partial coverage, target coverage problems require all and only deployed targets to be monitored. On the other hand, barrier coverage problems aim at minimizing the probability of undetected penetration through a sensor barrier (Li and Shen, 2015; Mostafaei, 2015). In the context of barrier coverage, at least k distinct sensors have to detect a penetrating object before it reaches the area of interest.

Area coverage problems can be divided into *full coverage* and *partial coverage* (also known as *p-percent* coverage). While full coverage problems require to continuously monitor the whole of the area of interest (Akbari Torkestani, 2013; Qianqian et al., 2015), partial coverage problems need to monitor a given percentage of the area, usually known in advance. To the best of our knowledge a limited number of works focus on partial coverage, although many works about full coverage in WSNs exist. Some of these works also require that *connectivity* among nodes is preserved. For instance, Shan et al. Shan et al. (2008) devised two algorithms that can maintain the network connectivity while monitoring the network area, guaranteeing a certain percentage of coverage. The first one enforces a centralized approach, while the second one solves the problem in a distributed fashion. Li et al. Li et al. (2011) proposed two methods to obtain partial coverage in WSNs. Their algorithms can guarantee both coverage and connectivity requirements, but fail in achieving low time-complexity. The concept of *Connected Dominating Set* (CDS) has been often leveraged. A CDS is a subset of vertices such that every vertex is either in the subset or adjacent to a vertex in the subset and the subgraph induced by the subset is connected. For instance, the algorithm proposed by Donghyun et al. Donghyun et al. (2009) creates a virtual backbone in the network identifying a CDS. Considering the two solutions presented by Wu et al. Wu et al. (2008) to address connected partial coverage problem in WSNs, pPCA implements a greedy approach, while CpPCA-CDS acts in a distributed way and is based on the construction of a CDS. The main drawback of these solutions is the dependence upon the depth-first search (DFS) that heavily impacts the time complexity (for this reason, hereafter we refer to CpPCA-CDS simply as DFS). *Information about neighbor nodes* is often used to preserve coverage and connectivity in WSNs. Yardibi and Karasan (2010) developed a Distributed Adaptive Sleep Scheduling Algorithm (DASSA) for WSNs. In their approach each node uses the remaining energy levels and a feedback from the sink node to schedule the activity of its neighbor nodes. *Probabilistic approaches* have been also proposed (e.g., to find redundant sensor nodes in a network while preserving partial coverage requirements). Hafeeda and Ahmadi (Hefeeda and Ahmadi, 2010) studied coverage problems under both disk sensing and probabilistic sensing models and propose a Probabilistic Coverage Protocol (PCP) that computes the maximum possible distance between sensors to avoid holes in coverage. Xing et al. Xing et al. (2005) devised a Coverage Configuration Protocol (CCP) to provide different degrees of coverage to the applications. The approach proposed by Gupta et al. Gupta et al. (2013) 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 among sensor nodes. Ammari et al. Ammari and Das (2012) consider the identification of redundant sensors based on a geometric approach. Finally, *WSN lifetime* proved to be an important aspect to consider, indeed. Ren et al. (2007) studied partial coverage considering network lifetime issues. Mao et al. Mao et al. (2007) analyzed the relation between the desired sensing coverage fraction and the minimum number of working sensors. They devised an Energy Aware Partial Coverage Protocol (EAPC) which chooses the minimum number of working sensors based on the residuary energy of the nodes.

This paper addresses the problem of *partial coverage* in WSNs extending the work proposed in Mostafaei et al. (2016). PCLA leverages the probabilistic framework of LA to find a convenient subset of sensor nodes to ensure partial coverage. The main objective of the proposed approach is to use the smallest number of sensors at any given time, thus extending WSN lifetime. PCLA is able to preserve both coverage and connectivity. In more details, it uses the coverage graph of the

Table 1
Comparison of PCLA with state-of-the-art algorithms for partial coverage in WSN.

Algorithm	*PCLA*	*CDS* (Donghyun et al., 2009)	*DFS* (Wu et al., 2008)
Model	Binary sensing model	Binary sensing model	Binary sensing model
Connectivity	Yes	Yes	Yes
Backbone selection method	LA	CDS approach	CDS approach
Desired coverage method	LA	Neighbor nodes	DFS search on neighbors

network to identify the nodes to build a backbone. LA running on backbone nodes rely on the coverage information of their neighbor nodes to construct a connected set that is able to obtain and preserve the partial coverage required.

To the best of our knowledge, recent solutions have largely focused on sensing models that deal with the problem of the full area coverage that exposes different requirements and thus requires different approaches than partial coverage (Wang, 2011). Therefore, in this work we compare PCLA to the algorithms presented in Donghyun et al. (2009) and in Wu et al. (2008) We have chosen these works as they leverage the coverage graph and model the WSN similarly to the proposed PCLA approach. Moreover, they are both two-phase algorithms (as PCLA is), with a first phase addressing the connectivity among nodes, and a second one aimed at obtaining the required coverage. Table 1 summarizes the main characteristics of PCLA and state-of-the-art partial coverage algorithms taken into account. As shown, they all use a binary sensing model and ensure also the connectivity of the nodes. In PCLA however, the selection of the backbone nodes and the fulfilment of the partial coverage requirement are accomplished with the aid of LA. On the other hand, both CDS- and DFS-based approaches construct a connected dominating set which constitutes the backbone assuring the connectivity. In more particulars, the former leverages the neighbor nodes in order to obtain an increment of the coverage until the desired one is met, while the latter performs a depth-first search to reach the required percentage of area to be covered.

3. Preliminaries and definitions

In this section, we define the problem of *partial coverage* also introducing the main related concepts.

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 randomly deployed nodes. Each node can sense every event that occurs within its *sensing range* and can communicate with other nodes within its *communication range*. Sensing and communication ranges are defined as the circles with radius R_s and R_c, respectively. E represents the set of the communication links between nodes. For any pair of nodes u and v, the edge $(u, v) \in E$ if and only if u and v are within the communication range of each other. More formally, the *sensing region* γ_i of the node S_i is defined as the circle with center in S_i and radius R_s. Consequently, the *coverage function* $C_i(x, y)$ of each node S_i can be defined as:

$$C_i(x, y) = \begin{cases} 1, & \text{if } (x, y) \in \gamma_i \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

In other words, the point (x,y) is covered by S_i if it lies in the sensing region of the node.

Given a two-dimensional region of interest ϑ (whose area is equal to A_ϑ) and a WSN made up of N sensors (each able to cover an area of πR_s^2) the WSN partial coverage problem we aim at solving 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 Ψ, subjected to :
$-\Psi$ is a connected set of nodes;
$-\Psi$ covers at least the area $P_s A_\vartheta$ of ϑ.

Some useful metrics in this framework are: (i) the *Average Region Coverage Degree* (Wu et al., 2008), 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* (Wu et al., 2008), 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.

According also to Younis et al. Younis et al. (2008) ϑ is divided into small squared cells that are $\frac{R_s}{\sqrt{5}}$ on a side. This allows a node to completely cover neighbor cells in the four main directions.

We can therefore extend the definition of coverage function for the i^{th} node as follows:

$$C_i(j) = \begin{cases} 1, & \text{if } j \in \gamma_i \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

where $j \in cells$ is one of the cells the area is divided into.

Accordingly, the second constraint can be written as:

$$\frac{|\bigcup_{i,j} C_i(j)|}{|cells|} \geq P_s, \quad \text{where} \quad i|S_i \in \Psi, j \in cells \quad (3)$$

Symbols and definitions are summarized in Table 2.

4. Basics on learning automata

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 interaction between a learning automaton and the random environment is shown in Fig. 2.

The environment is described as a triple $\mathcal{E} = \{\alpha, \beta, c\}$ where $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_N\}$ indicates the finite input set (i.e. the actions), $\beta = \{\beta_1, \beta_2, \ldots, \beta_N\}$ 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 action α_i. The probability of the action α_i is $p_i(n)$, and the corresponding vector $p(n)$ defines the action probability vector.

Table 2
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
γ_i	Sensing region of the node S_i
$C_{(x,y)}(S_i)$	Coverage function of the node S_i
Ψ	Connected set of nodes that guarantees partial coverage

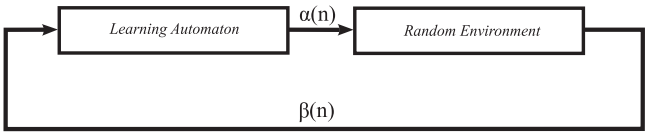


Fig. 2. The relationship between the learning automaton and the random environment. The automaton learns to optimally choose actions $\alpha(n)$ based on the reinforcement signals $\beta(n)$ provided by the environment.

For our solution, we consider variable-structure automata (Narendra and Thathachar, 1989) and a 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 Eq. (4):

$$p(n+1) = T[p(n), \alpha(n), \beta(n)] \tag{4}$$

where $p(n)$ and $p(n+1)$ are the action probability vectors 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 Eqs. (5) and (6):

$$p_j(n+1) = \begin{cases} p_j(n) + a(1 - p_j(n)) & j = i \\ (1-a)p_j(n) & \forall\, j, j \neq i \end{cases} \tag{5}$$

$$p_j(n+1) = \begin{cases} (1-b)p_j(n) & j = i \\ \frac{b}{r-1} + (1-b)p_j(n) & \forall\, j, j \neq i \end{cases} \tag{6}$$

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.

5. PCLA

In this section we describe the PCLA algorithm designed to address the partial coverage problem in WSNs. The main idea behind PCLA is to first identify a set of nodes to build a *backbone* able to guarantee the connectivity among all the nodes. Then, if partial coverage is not satisfied, additional nodes are activated. In the following, we first describe the phases the algorithm is composed of. Then, we discuss its correctness and complexity.

5.1. Algorithm

PCLA consists of two phases: (i) *learning phase* and (ii) *partial coverage phase*.

Learning phase. The aim of the learning phase is selecting a convenient (i.e. limited in number) set of nodes as the *backbone*. This phase consists of an iterative procedure aimed at finding a backbone set that fits with predefined constraints. The learning phase ends when the selected backbone satisfies the constraints and a predefined stop condition is met.

In order to perform the PCLA algorithm in a distributed fashion, each node requires to store the data structure detailed in the following:

- P_s, the desired level of coverage in terms of portion to cover of the region of interest;
- $P_{threshold}$, a threshold value defining the maximum possible value for the product of the probabilities of the actions chosen by LA of each node in Ψ; it is used to enforce the first stop condition of the learning phase;
- T_k, a threshold value defining the maximum number of cycles for the PCLA algorithm; it is used to enforce the second stop condition of the learning phase;
- Ψ, the set of nodes chosen by PCLA and updated in each cycle of the evolution of LA;
- Γ, the set of nodes not chosen by PCLA and made up by the unselected neighbors of the nodes in Ψ;
- E_ψ, the average residual energy of nodes in Ψ; this parameter is useful to select a set of backbone nodes with a high residual energy to keep the connectivity for a longer period of time;
- T_n, a dynamic threshold storing the cardinality of the Ψ set selected at the n^{th} cycle in the learning phase of PCLA; its value is initialized to $|V|$;
- E_n, a dynamic threshold storing the average residual energy of the nodes in the Ψ set selected at the n^{th} cycle in the learning phase of PCLA; its value is initialized to 0;
- n, a counter which keeps the number of cycles of the PCLA algorithm and used to check the stop condition.

This set of global variables is established at the beginning of the *initialization step*. A `HELLO` message containing the values of P_s, $P_{threshold}$, T_k is broadcast within the network. After receiving this message, PCLA on each node starts the process initializing the global variables, and gets a snapshot from the *CG* in order to know node's neighbors. This is a key step of the overall algorithm, because it starts from the *CG* of the network to find suitable nodes with the final goal to meet the partial coverage requirement.

Let Ψ denote the set that PCLA builds and updates iteration by iteration. For each node, each action α_i consists in adding the neighbor node S_i to Ψ. The action probability vector $p(n)$ is initialized as follows:

$$p_i(n) = \frac{1}{r} \quad \forall\, i \tag{7}$$

where r indicates the action-set count, that is equal to the number of neighbor nodes at this initialization step. For example, if node S_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 S_i has five equiprobable actions.

The main goal of the learning phase is to find redundant sensors in the network area A_ϑ. LA running on each node helps to identify the convenient backbone. After the initialization step, a randomly chosen node is added to Ψ.

Each sensor (say S_i), in order to form its action-set, propagates a `DISCOVERY` message to its neighbors (i.e. to the nodes placed within its transmission range). Upon receiving the message, each node replies to the sender S_i that thus forms its action-set based on the received messages by its neighbors. The action-set size of each LA thus depends on the degree of corresponding nodes and consequently of the network density. Its action-set being given, in turn, the LA of a node chooses one of the neighbors to be added to Ψ accordingly to its action probability vector $p(n)$. 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. Upon selecting an action by LA of each node, it updates its data structure (i.e. the values for global variables). In this phase, if no action is available (e.g., if all the neighbors of the selected node are already contained in Ψ or Γ and $\Psi \cup \Gamma \neq V$) then the selection procedure is traced back and restarted from another node, to ensure eventually the successful selection of the backbone nodes. Moreover, after a node has been chosen, LA of each node prunes its action set by disabling the action corresponding to the selected node. In this way, the nodes already selected cannot be chosen anymore and the generation of loops is avoided, increasing also the convergence speed of the PCLA algorithm.

As soon as a candidate backbone has been identified, the suitability of Ψ is evaluated. At each cycle n, the number of nodes in Ψ is compared with a threshold value T_n (that can be initially set to $|V|$). If $|\Psi| < T_n$ and the average residual energy of nodes in Ψ (E_Ψ) is greater than E_n, all selected actions α_i in Ψ are rewarded from the environ-

ment ($\beta_i(n) = 0$). Otherwise, these actions get a penalty from the environment ($\beta_i(n) = 1$). In the former case, each node in Ψ broadcasts a REWARD message among its active neighbors and updates accordingly its action probability vector using Eq. (5). It also updates the values of the global variables. This procedure helps *PCLA* to keep the connectivity of the network for a longer period of time. In the latter case instead, a PENALTY message is broadcast among active nodes in Ψ. It is worth noting that the PCLA algorithm uses the L_{R-I} reinforcement schema for each LA in order to update its action probability vector. Therefore, upon receiving a PENALTY message, the probability of the selected actions remains fixed in Ψ and the disabled actions of activated LA will be enabled again. The update of the action probability vector in this case can be done by setting b parameter to zero in Eq. (6). Updating these values, the learning procedure is implemented.

This process continues until the stop condition is met. In the following this stop condition is exhaustively described. First, the probability of the selected nodes during the last cycle is computed. This probability value can be defined as the product of the probabilities of the chosen actions by LA of each node during the last cycle. It can be computed as follows:

$$\prod_{i=1}^{|\Psi|} \alpha^i_{best} \tag{8}$$

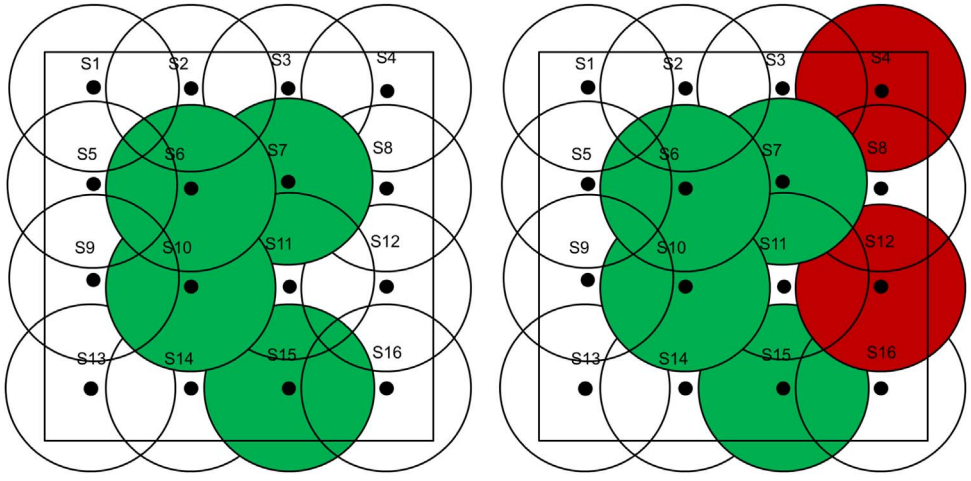
where $|\Psi|$ is the number of nodes in Ψ and α^i_{best} is the best action of LA of node S_i. If the probability value is greater than a threshold defined by $P_{threshold}$, the stop condition is met. Second, the number of cycles reaches a maximum value defined by T_k. Note that PCLA needs some cycles to converge to a stable set. Therefore, this process continues until the nodes in Ψ do not change for ten consecutive cycles. Indeed, in this state, the product of the probabilities of the best action of LA in each node reaches the threshold value. At the end of this phase, the nodes in Ψ are able to preserve the connectivity and identify the desired backbone. In detail, an ACTIVATION message is broadcast among all the nodes in Ψ (i.e. each node sends the message to all its neighbors in Ψ); thus the nodes receiving this message will be active. Other nodes can switch to the idle state in order to save their energy.

Fig. 3 shows a descriptive example of the operating principles of the PCLA algorithm when 16 nodes have been deployed. In detail, in Fig. 3(a) the operations performed during the learning phase are illustrated. At the beginning, the node S_7 is randomly selected. Then, S_7 chooses one of its actions (i.e. one of its neighbors) according to its action probability vector, say node S_6. This process continues until each deployed node has been either selected (i.e. added to Ψ) or not (i.e. added to Γ). Thus, S_{10} and S_{15} are chosen ending the first cycle of the learning phase. At this point, the cardinality $|\Psi|$ (in this example equal to 4) and the average residual energy E_Ψ of the nodes in Ψ are computed. On the one hand, if $|\Psi|$ is less than the threshold value T_n and E_Ψ is greater than the current residual energy E_n, a REWARD message will be sent to the nodes in Ψ. On the other hand, if the previous condition is not satisfied a PENALTY message is sent instead. Upon receiving the REWARD or PENALTY message, each node updates its action probability vector according to Eq. (5) or Eq. (6), respectively. Additionally, in both cases they also update the value of the global variables stored (in this example the value of T_n is set to 4, and E_n to E_Ψ). This process continues until the stop condition is met, preserving the connectivity among the chosen nodes.

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

Algorithm 1. PCLA Algorithm

Input:
$CG = (V, E)$ ▷ Snapshot of the network
P_s ▷ Desired partial coverage
a ▷ Reward parameter for the update of the action probability vector, where $0 < a < 1$
T_n ▷ Cardinality threshold value
E_n ▷ Energy threshold value
Output:
Ψ ▷ Selected nodes
Γ ▷ Unselected nodes
Initialization:
A HELLO message is broadcast within the network
for all nodes in V **do**
$\Psi = \varnothing$
Initialize T_k and $P_{threshold}$
Send a DISCOVERY message
$p_i(0) = \frac{1}{r} \forall\ i$ ▷ r is the number of neighbors
end for
repeat
Randomly select and activate a node S_i
repeat
while S_i has no possible actions **do**
Activated nodes are traced back to find an automaton with available actions
end while
$\Psi = \Psi \cup S_i$
S_i selects one of its neighbors S_j accordingly to its $p(n)$



(a) Learning phase. (b) Partial coverage phase.

Fig. 3. An illustrative example of the selection of nodes in PCLA. The two main phases of our algorithm are highlighted. Nodes selected in the learning phase are colored in green, whereas those selected in the partial coverage phase in red. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Each automaton prunes its action-set to avoid the loop
S_j is activated
Γ=Γ ∪Unselected neighbors of S_i
$S_i = S_j$
until $|\Psi \cup \Gamma| < |V|$
E_Ψ is calculated
if $|\Psi| < T_n$ **and** $E_\Psi > E_n$ **then**
$T_n = |\Psi|$
$E_n = E_\Psi$
$\beta_i(n) = 0 \quad \forall\, i | S_i \in \Psi$ ▷ Reward from the environment
Broadcast a `REWARD` message among all selected nodes in Ψ
else
$\beta_i(n) = 1 \quad \forall\, i \mid S_i \in \Psi$ ▷ Penalty from the environment
Broadcast a `PENALTY` message among all selected nodes in Ψ
end if
Enable all the disabled actions
until the stop condition is met.
An `ACTIVATION` message is broadcast among all the nodes in Ψ
`FormPartialCoverage()`

Partial Coverage Phase. At the end of the learning phase, PCLA checks whether partial coverage is met. If partial coverage is not satisfied, the `FormPartialCoverage()` routine is called. This function uses the nodes in Γ to meet the partial coverage requirement. Specifically, `FormPartialCoverage()` leverages the coverage function and evaluates the number of cells that would be covered by activating each of the nodes in Γ to finally identify convenient nodes to be activated. More in details, a node is activated if it covers cells that cannot be covered by its neighbors. At the end of this phase, nodes in Ψ will remain in the active state to monitor the network, while other nodes will switch to the idle state in order to save energy. To this aim, nodes already in Ψ send an `ACTIVATION` message to the nodes selected during the partial coverage phase.

In Fig. 3(b) we provide an example for the partial coverage phase of PCLA algorithm. Assuming that the set of nodes selected in the learning phase (see Fig. 3(a)) does not satisfy the desired level of coverage, additional nodes can be activated and added to this set. In the proposed example since S_4 and S_{12} can cover an area not yet covered by their neighbors, they are chosen from Γ, activated, and added to Ψ. In this way the partial coverage requirement of the network is met.

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

Algorithm 2. `FormPartialCoverage()`

Input:
$CG = (V, E)$ ▷ Snapshot of the network
P_s ▷ Desired partial coverage
Input/Output:
Ψ ▷ Selected nodes
Γ ▷ Unselected nodes
for all S_j in Γ **do**
if Ψ does not satisfy P_s **then**
if Neighbors of S_j cannot cover S_j area **then**
Ψ=Ψ ∪S_j
Send an `ACTIVATION` message to S_j
else
Deactivate S_j
end if
end if
end for

5.2. Correctness and time complexity

Before presenting the experimental results of the proposed approach, here we show how PCLA preserves both connectivity and partial coverage. Moreover the complexity of the algorithm is evaluated and compared with state-of-the-art solutions.

Correctness of PCLA. *The obtained set Ψ from PCLA can preserve both partial coverage and connectivity.*

Proof. The connectivity among all the working nodes of the WSN is guaranteed by PCLA by construction. Indeed, at the end of the algorithm $\Psi \cup \Gamma = V$, i.e. each node belongs either to Ψ or Γ. Nodes in Ψ are connected due to the iterative procedure adopted: at each iteration i of the learning phase, S_i selects a neighbor within its communication range R_c and add it to the backbone set Ψ. Therefore all the nodes in Ψ are connected. If the coverage constraints require more nodes to be added to Ψ, the `FormPartialCoverage()` routine activates other sensors and adds them to Ψ, until the required percentage of partial coverage is reached. These additional nodes are selected from those in Γ: considering that each node in Γ has at least one neighbor in Ψ by construction, the connectivity is maintained.□.

Time complexity analysis. PCLA algorithm is formed by two nested loops: (i) the inner loop that has a running time proportional to the number of nodes (N) and (ii) the outer loop whose running time depends upon the number of cycles (I). Therefore the complexity of the inner and of the outer loop is equal to $O(N)$ and $O(I)$, respectively. Finally, the running time of `FormPartialCoverage()` routine is also $O(N)$. Due to these contributions, the time complexity of PCLA can be expressed as $O(N \times I) + O(N)$. Therefore, the overall time complexity of PCLA algorithm is $O(N \times I)$.

Experimental results showed how the number of cycles required by the algorithm is markedly lower than the number of nodes in the WSN in most of the cases. Fig. 4 shows the distributions of the number of cycles$I$performed by PCLA for different values of N. For N=100 (N=1000) 17.5 (24.6) cycles are needed, on average. Cases with N=10 represent a notable exception due to the operating mode of PCLA (see Algorithm 1). If we filter out non-realistic values of N, i.e. we consider only larger values for it, we always obtain that $I \ll N$ and therefore we can conclude that the time complexity of PCLA is $O(N)$. In this case, if we reward Ψ for I times, the product of the probabilities of actions in this set reaches the threshold value and the stop condition of *PCLA* is met.

Compared to the state-of-the-art algorithms chosen as term of comparison (see Section 6), PCLA leads to a better time complexity. The time complexity of the CDS-based algorithm is $O(N^{1.6} + Diam)$, where $Diam$ is the diameter of the network (Donghyun et al., 2009). The complexity of the DFS-based algorithm is $O(N + |E|)$ (Wu et al., 2008), as the algorithm can be divided in three main phases: the construction of a CDS ($O(Diam)$), the building of a DFS search tree on the CDS nodes ($O(N + |E|)$), and the addition of nodes to the CDS until the partial coverage requirement is met

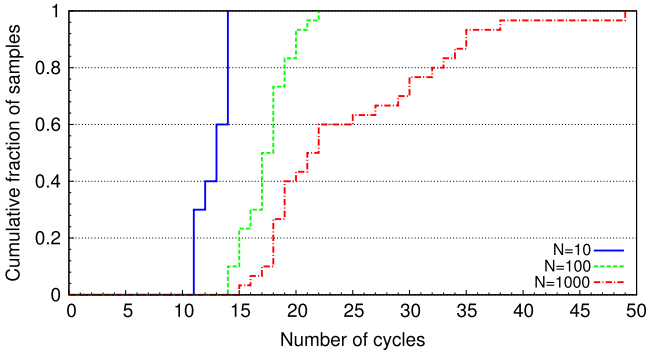


Fig. 4. Distribution of the number of cycles (I) required by PCLA for WSNs having different number of nodes (N). The number of cycles needed by the algorithm is markedly lower than the number of nodes for WSNs having a realistic number of nodes (N = 100, 1000).

($O(N + |E|)$). Table 3 summarizes the time complexity of the three algorithms, showing how PCLA has a lower time complexity than state-of-the-art partial-coverage solutions.

6. Performance evaluation

In this section, we provide a comprehensive performance evaluation of the proposed solution. We first detail the experimental setup we leveraged for the evaluation, also providing details about the implementation of the methods we choose as a comparison (Section 6.1). Then we present experimental results obtained through simulation, showing that PCLA performs better than state-of-the-art partial-coverage solutions in terms of working-node ratio and network lifetime (Section 6.2).

6.1. Evaluation setup

We used the WSN simulator (Stein, 2016) to evaluate the proposed approach through simulation. PCLA has been compared to a method based on that proposed in Donghyun et al. (2009)—whose implementation is detailed in the following—and the CpPCA-CDS algorithm based on the DFS and introduced in Wu et al. (2008); hereafter we will refer to those two approaches simply as CDS and DFS, respectively.

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) 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 (Mostafaei et al., 2015)). All nodes' sensing and communication ranges are assumed to be equal. To compute the network lifetime we used an approach similar to the Maximum Set Cover (MSC) described in Mostafaei and Meybodi (2013), Mostafaei et al. (2015), assuming a relative consume of energy of w for each node. Simulation parameters and coverage requirements used in our simulations are summarized in Table 4. All the obtained results have been averaged over 10 simulation runs with heterogeneous topologies.[1]

Fig. 5(a) shows an example of the simulation environment we used to simulate the algorithms for the proposed evaluation. The figure shows an example with a small number of nodes in the network in order to better clarify the problem (Stein, 2016). Points represent nodes. The nodes selected to monitor the portion of the network area (i.e. nodes inside Ψ set of PCLA) are shown in Fig. 5(b) (red circles). As shown in the figure, the selected nodes are connected to each other.

CDS-based partial coverage. We describe here the CDS-based algorithm we adopted in our simulations as a basis for the comparison.

We obtained the CDS-based algorithm extending state-of-the-art algorithms presented in Donghyun et al. (2009) such that a partial coverage algorithm is devised. In more details, we took advantage of the *minimum weighted* version method proposed in Donghyun et al. (2009) to make a CDS starting from the CG of the network.

The algorithm works as follows: first a CDS is constructed leveraging the initial algorithm presented in Donghyun et al. (2009); then some non-CDS nodes are used in the deployed network to meet partial coverage. Therefore also this algorithm has two phases: (i) constructing a CDS and (ii) adding nodes to meet partial coverage. To construct a CDS upon the CG of the network, the algorithm first selects a random node and adds this node to CDS set (Y). Then, this node selects one of its neighbor nodes and does the same process (i.e. adds the selected node to CDS set). The remaining neighbors of this node will be added to Ω set. This process continues until the total amount of CDS and non-CDS nodes reaches $|V|$. At this point the second phase starts where the algorithm adds a node to the active set if the node generates a coverage increment.[2] If the covered area of the selected node is already covered by other active nodes, this node switches to the inactive state to save its energy.

Algorithm 3 shows the pseudo-code of this new algorithm.

Table 3
Time complexity of the analyzed algorithms.

Algorithm	PCLA	CDS	DFS
Time complexity	$O(N)$	$O(N^{1.6} + Diam)$	$O(N + \|E\|)$

Table 4
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	
α		0.1	
w		0.2	
N	31	63	105
D_ϑ	1.5	3.0	5.0

(b) Coverage requirements.

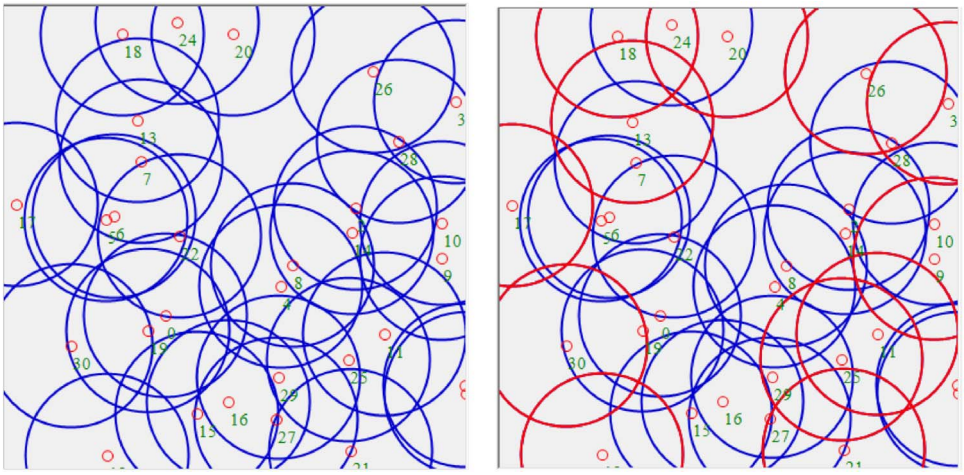
Parameter	Values		
P_s	0.6	0.8	1.0

Algorithm 3. Implemented Version of CDS Algorithm (Donghyun et al., 2009)

Input:
$CG = (V, E)$ ▷ Snapshot of the network
P_s ▷ Desired partial coverage
Output:
Υ ▷ Selected nodes
Ω ▷ Unselected nodes
repeat
  Randomly select and activate a node S_i
until S_i has at least one neighbor
$\Upsilon = \Upsilon \cup S_i$
while $|\Upsilon \cup \Omega| < |V|$ **do**
  S_i randomly selects one of its neighbors S_j
  $\Upsilon = \Upsilon \cup S_j$
  $\Omega = \Omega \cup$Unselected neighbors of S_i
  S_i=S_j
end while
repeat
  for all $S_i \in \Omega$ **do**
    if $CI_i > 0$ **then** ▷ CI_i is the Coverage Increment associated to the node S_i
      Activate S_i
      $\Upsilon = \Upsilon \cup S_i$
    else
      Deactivate S_i
    end if
  end for
until Υ satisfies P_s

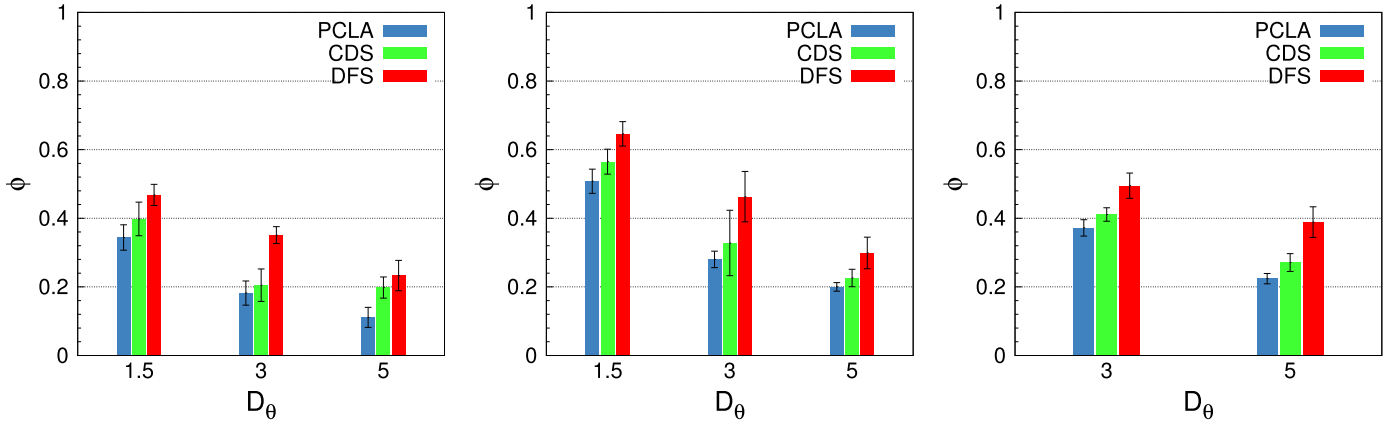
[1] In more details, the position (x,y) of each node follows a *uniform distribution* in [X_{min}, X_{max}] and [Y_{min}, Y_{max}], respectively.

[2] The *coverage increment* (CI_i) is defined as the increment of coverage that would be obtained by adding the node i to Y.



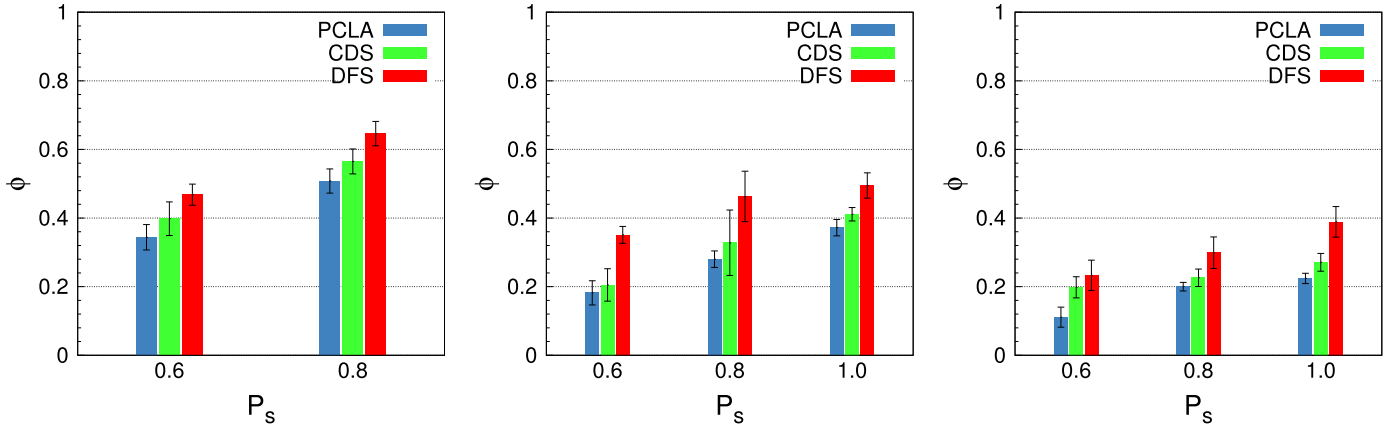
(a) WSN deployed in one of our simulations (b) Set of selected nodes in one of our simulations.

Fig. 5. Results of an example simulation. The circles around the nodes indicate their sensing range, while the numbers their IDs. Nodes selected by PCLA are depicted with red circles. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



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

Fig. 6. Impact of average region coverage degree (D_ϑ) on working-node ratio (ϕ) 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.



(a) $D_\vartheta = 1.5$ (b) $D_\vartheta = 3$ (c) $D_\vartheta = 5$

Fig. 7. Impact of coverage requirement (P_s) on Working-Node Ratio (ϕ) for different values of D_ϑ. Increasing coverage requirements has a detrimental impact on the performance of all the algorithms, with ϕ exposing an increasing trend on average for increasing values of P_s. This impact is mitigated by deploying a larger number of nodes, i.e. for greater values of D_ϑ.

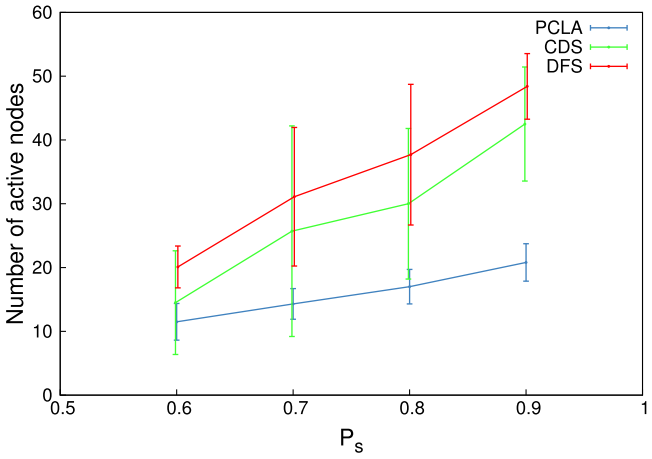


Fig. 8. Number of active nodes for different coverage requirements for a large network (N=200, R_s = 50 m). As expected performance decreases for higher P_s. PCLA exhibits a better trend and outperforms other algorithms.

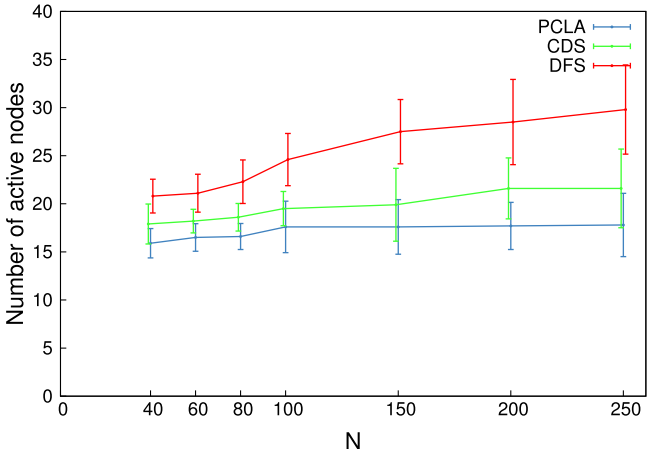


Fig. 9. Number of active nodes for different network sizes (P_s=0.8, R_s = 50 m). When more nodes are available, all the algorithms settle to slightly worse solutions in terms of number of active nodes.

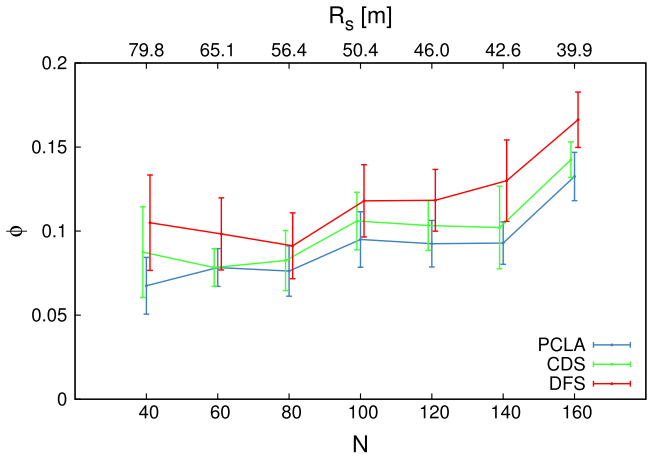


Fig. 10. Effect of varying the number N of nodes, keeping D_ϑ fixed (D_ϑ = 5, P_s=0.6). Increasing the number of nodes—and reducing the sensing ranges accordingly—leads to worse results in terms of ϕ.

6.2. Experimental results

In this section, 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. Detailed results are provided in the following.

Theoretical bounds for both lifetime and coverage exist. They primarily depend by the placement of nodes in the WSN. Regarding the coverage, the lower bound (worst case) is encountered when all the nodes exactly lie in the same place (full overlap). The overall area covered by the WSN is equal to πR_s^2, i.e., to the sensing area of a single node. According to the requirements in terms of the portion to cover (P_s) the WSN could either meet or not the coverage goal. The upper bound (best case) is met when the sensing areas of the nodes do not overlap, while each node falls in the communication range of some other nodes. The overall area covered by the WSN is equal to $N\pi R_s^2$, in this case. For what concerns the lifetime, assuming that each node has a lifetime equal to T, we meet the lower bound (worst case) when all the nodes are required to be active from the beginning. The minimum lifetime of the WSN is therefore T. The upper bound (best case) is encountered when only one node at a time is required to be active. In this case the maximum lifetime is equal to NT.

Impact of the Average Region Coverage Degree and Coverage Requirement. 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 3—D_ϑ may also be (indirectly) considered an input for our simulations, as being function of N, A_ϑ, and R_s. By varying the number of nodes, we have defined three different configurations, corresponding to different resource constraints: D_ϑ = 5, D_ϑ = 3, and D_ϑ = 1.5 (see Table 4a). 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 4b).

Fig. 6 shows through the working-node ratio (ϕ) how the amount of resources needed to reach the required coverage level 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 reported that ϕ exposes a decreasing trend on average, for increasing values of D_ϑ. Note that results for D_ϑ = 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_ϑ = 1.5 to D_ϑ = 5. Indeed, the value of ϕ decreases by 67.7% (60.6%) when P_s=0.6 (P_s=0.8).

Fig. 7 shows the simulation results obtained in terms of working node ratio (ϕ), when varying the coverage requirements (P_s) for the different Average Region Coverage Degrees considered (D_ϑ = 1.5, D_ϑ = 3, and D_ϑ = 5, respectively). As expected, 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_ϑ = 5, PCLA has the lowest performance decrease when passing from P_s=0.8 to P_s=1.0. Indeed, the value of ϕ increases by 12% for PCLA against a 20-percent and 30-percent increase obtained with CDS and DFS, respectively.

These results show how the proposed algorithm is able to better utilize the available resources: (i) PCLA guarantees always better performance than other algorithms; (ii) PCLA leads to markedly better performance than other algorithms when more sensors are available (e.g., D_ϑ = 5).

This evidence can be motivated by the following reasons: (i) PCLA method uses the minimum possible number of nodes as backbone nodes to reach partial coverage requirement and guarantee connectivity between them; (ii) PCLA tries to select the nodes with minimum possible overlap (see Algorithm 2).

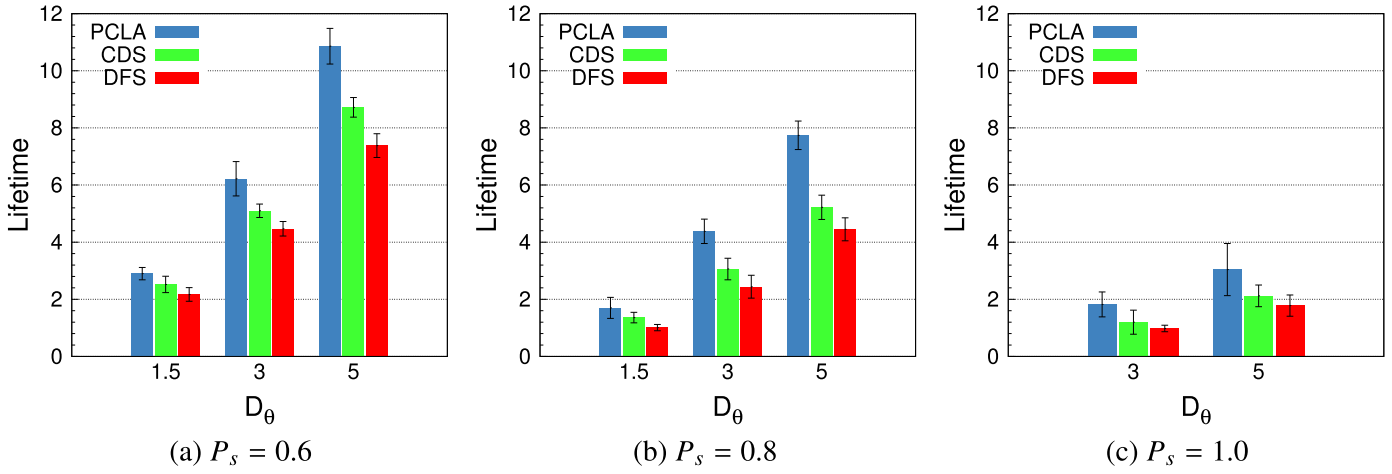


Fig. 11. Impact of average region coverage degree (D_ϑ) on network lifetime for different values of P_s. The adoption of PCLA prolongs the network lifetime in all the circumstances taken into account.

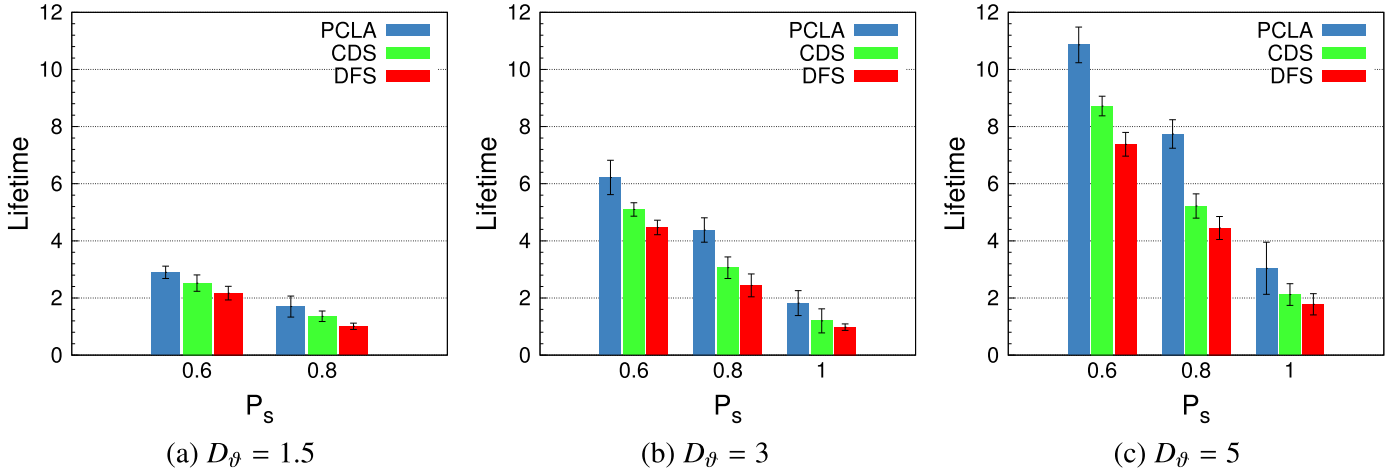


Fig. 12. Impact of coverage requirement (P_s) on the network lifetime for different values of D_ϑ. Increasing coverage requirements has a negative impact on the lifetime of all the algorithms.

Scalability Analysis. 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.

Fig. 8 reports the number of the active nodes ($|\Psi|$) considering a network 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 ones of CDS and DFS when increasing the coverage constraints. The improvement obtained by using PCLA respect to CDS (DFS) raises 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. The motivations behind these results can be summarized as follows: (i) more active nodes are required to meet the stricter coverage constraints when the value of P_s increases; (ii) when the overlap between nodes raises—e.g., in networks dense as much as the ones we considered in this analysis— PCLA performs a more efficient selection of the nodes to activate checking continuously for their suitability, as long as partial coverage requirement is met.

Fig. 9 shows how the number of active nodes changes with the size of the network, for network sizes ranging from 40 to 250 nodes. When more nodes are available, all the algorithms settle to slightly worse solutions in terms of active nodes. This is due to the fact that in dense networks the overlap between active nodes increases and cannot be avoided. Nevertheless, PCLA outperforms the CDS and DFS algorithms, mitigating this detrimental effect.

We have also investigated whether PCLA performs better when leveraging more nodes with a limited sensing range, or—*vice versa*—fewer nodes with higher sensing capabilities. In other words, we have considered the effects of varying the number N of nodes, while keeping D_ϑ fixed. To this end, in each simulation we have modified the value of the sensing range R_s depending on the value of N and according to the definition of D_ϑ (see Section 3). The outcome of this analysis is shown in Fig. 10. As shown in the figure, keeping the value of D_ϑ fixed leads to limited variations of ϕ, which therefore has proved to mainly depend on D_ϑ and P_s. Interestingly, better performance in terms of ϕ can be achieved for lower values of N and larger sensing range.

Lifetime analysis. As network lifetime—i.e., the time span from the networks initial deployment to the first loss of coverage—is a critical performance index for WSN, our evaluation also took it into consideration. Figs. 11 and 12 compare 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 the analysis, we note that a larger Average Region Coverage Degree 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%). The reason for this lifetime enhancement can be found in the working principle implemented by the inner loop of PCLA. Indeed, nodes in Ψ with higher C_i are selected, thus requiring less of them to meet coverage requirements. In some cases only backbone nodes can reach the desired coverage requirement. This allows to rely on a reduced number of active nodes, and guarantees a higher number of idle nodes that can be selected by PCLA in the subsequent cycles.

7. Conclusion

In this work, we have addressed the problem of partial coverage in WSNs and have proposed PCLA, a solution based on Learning Automata. Our algorithm finds a convenient set of sensors to activate that is able to cover the desired portion of the region of interest and to preserve the connectivity among active nodes.

According to experimental results, PCLA outperforms state-of-the-art algorithms. Indeed, it exposes lower time complexity and better 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, when making coverage constraints more strict, PCLA performance decreases more slowly than those of the other solutions evaluated. Accordingly, the benefit obtained by using PCLA instead of the other algorithms increases when increasing the value for P_s. These results still hold for larger networks thus guaranteeing the scalability of the approach.

References

Akbari Torkestani, J., 2013. An adaptive energy-efficient area coverage algorithm for wireless sensor networks. Ad Hoc Netw. 11 (6), 1655–1666, (⟨http://dx.doi.org/http://dx.doi.org/10.1016/j.adhoc.2013.03.002⟩. URL ⟨http://www.sciencedirect.com/science/article/pii/S1570870513000310⟩).

Ammari, H., Das, S., 2012. Centralized and clustered k-coverage protocols for wireless sensor networks. IEEE Trans. Comput. 61 (1), 118–133. http://dx.doi.org/10.1109/TC.2011.82.

Atzori, L., Iera, A., Morabito, G., 2010. The internet of things: a survey. Comput. Netw. 54 (15), 2787–2805.

Botta, A., de Donato, W., Persico, V., Pescapé, A., 2016. Integration of cloud computing and internet of things: a survey. Future Gener. Comput. Syst. 56, 684–700, ⟨http://dx.doi.org/10.1016/j.future.2015.09.021⟩ URL ⟨http://www.sciencedirect.com/science/article/pii/S0167739×15003015⟩.

Botta, A., de Donato, W., Persico, V., Pescapé, A., 2014. On the integration of cloud computing and internet of things. In: Proceedings of the IEEE International Conference on Future Internet of Things and Cloud (FiCloud), pp. 23–30.

Demirbas, M., Chow, K., Wan, C., 2006. Insight: Internet-sensor integration for habitat monitoring. In: International Symposium on a World of Wireless, Mobile and Multimedia Networks, (WoWMoM), pp. 6 pp.–558. http://dx.doi.org/10.1109/WOWMOM.2006.52.

Donghyun, K., Yiwei, W., Yingshu, L., Feng, Z., Ding-Zhu, D., 2009. Constructing minimum connected dominating sets with bounded diameters in wireless networks. IEEE Trans. Parallel Distrib. Syst. 20 (2), 147–157. http://dx.doi.org/10.1109/TPDS.2008.74.

Gupta, H.P., Rao, S.V., Venkatesh, T., 2013. Sleep scheduling for partial coverage in heterogeneous wireless sensor networks. In: Proceedings of the Fifth International Conference on Communication Systems and Networks (COMSNETS), pp. 1–10. http://dx.doi.org/10.1109/COMSNETS.2013.6465580.

Hefeeda, M., Ahmadi, H., 2010. Energy-efficient protocol for deterministic and probabilistic coverage in sensor networks. IEEE Trans. Parallel Distrib. Syst. 21 (5), 579–593. http://dx.doi.org/10.1109/TPDS.2009.112.

Karrer, R., Matyasovszki, I., Botta, A., Pescapè, A., 2006. 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, Los Angeles, California, USA, September 29, pp. 26–33. http://dx.doi.org/10.1145/1160987.1160994 URL ⟨http://doi.acm.org/10.1145/1160987.1160994⟩.

Karrer, R., Matyasovszki, I., Botta, A., Pescapè, A., 2007. Magnets – experiences from deploying a joint research-operational next-generation wireless access network testbed. In: Proceedings of the 3rd International ICST Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities, TridentCom, Orlando, Florida, USA, May 21-23 2007, pp. 1–10. http://dx.doi.org/10.1109/TRIDENTCOM.2007.4444714. URL ⟨http://dx.doi.org/10.1109/TRIDENTCOM.2007.4444714⟩

Li, Y., Ai, C., Cai, Z., Beyah, R., 2011. Sensor scheduling for p-percent coverage in wireless sensor networks. Clust. Comput. 14 (1), 27–40. http://dx.doi.org/10.1007/s10586-009-0088-9, URL ⟨http://dx.doi.org/10.1007/s10586-009-0088-9⟩.

Li, S., Shen, H., 2015. Minimizing the maximum sensor movement for barrier coverage in the plane. In: IEEE Conference on Computer Communications (INFOCOM), pp. 244–252. http://dx.doi.org/10.1109/INFOCOM.2015.7218388.

Mao, Y., Wang, Z., Liang, Y., 2007. Energy aware partial coverage protocol in wireless sensor networks. In: International Conference on Wireless Communications, Networking and Mobile Computing WiCom, pp. 2535–2538. http://dx.doi.org/10.1109/WICOM.2007.631.

Mostafaei, H., 2015. Stochastic barrier coverage in wireless sensor networks based on distributed learning automata. Comput. Commun. 55 (1), 51–61, (doi: ⟨http://dx.doi.org/10.1016/j.comcom.2014.10.003⟩. URL ⟨http://www.sciencedirect.com/science/article/pii/S0140366414003326⟩).

Mostafaei, H., Meybodi, M., 2013. Maximizing lifetime of target coverage in wireless sensor networks using learning automata. Wirel. Pers. Commun. 71 (2), 1461–1477. http://dx.doi.org/10.1007/s11277-012-0885-y, (URL ⟨http://dx.doi.org/10.1007/s11277-012-0885-y⟩).

Mostafaei, H., Shojafar, M., 2015. A new meta-heuristic algorithm for maximizing lifetime of wireless sensor networks. Wirel. Pers. Commun. 82 (2), 723–742. http://dx.doi.org/10.1007/s11277-014-2249-2.

Mostafaei, H., Esnaashari, M., Meybodi, M., 2015. A coverage monitoring algorithm based on learning automata for wireless sensor networks. Appl. Math. Inf. Sci. 9 (3), 1317–1325.

Mostafaei, H., Montieri, A., Persico, V., Pescapé, A., 2016. An efficient partial coverage algorithm for wireless sensor networks. In: 2016 IEEE Symposium on Computers and Communication (ISCC) (ISCC2016), Messina, Italy, pp. 501–506.

Narendra, K.S., Thathachar, M.A.L., 1989. Learning Automata: an Introduction. Prentice Hall, ISBN-10: 0486498778; ISBN-13: 978-0486498775.

Qianqian, Y., Shibo, H., Junkun, L., Jiming, C., Youxian, S., 2015. Energy-efficient probabilistic area coverage in wireless sensor networks. IEEE Trans. Veh. Technol. 64 (1), 367–377. http://dx.doi.org/10.1109/TVT.2014.2300181.

Ren, S., Li, Q., Wang, H., Chen, X., Zhang, X., 2007. Design and analysis of sensing scheduling algorithms under partial coverage for object detection in sensor networks. IEEE Trans. Parallel Distrib. Syst. 18 (3), 334–350. http://dx.doi.org/10.1109/TPDS.2007.41.

Shan, G., Xiaoming, W., Yingshu, L., 2008. p-percent coverage schedule in wireless sensor networks. In: . In: Proceedings of the 17th International Conference on Computer Communications and Networks ICCCN '08, pp. 1–6. http://dx.doi.org/10.1109/ICCCN.2008.ECP.109.

Stein, D.J., 2016. Wireless Sensor Network Simulator, ⟨www.djstein.com/Projects/Files/Wireless%20Sensor%20Network%20Simulator%20v1.1.zip⟩.

Wang, B., 2011. Coverage problems in sensor networks: a survey. ACM Comput. Surv. 43 (4), 1–53. http://dx.doi.org/10.1145/1978802.1978811.

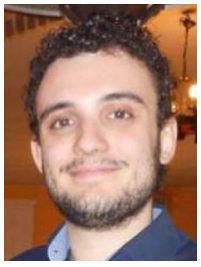
Wu, Y., Ai, C., Gao, S., Li, Y., 2008. p-Percent Coverage in Wireless Sensor Networks, vol. 5258 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, book section 21, pp. 200–211.

Xing, G., Wang, X., Zhang, Y., Lu, C., Pless, R., Gill, C., 2005. Integrated coverage and connectivity configuration for energy conservation in sensor networks. ACM Trans. Sen. Netw. 1 (1), 36–72. http://dx.doi.org/10.1145/1077391.1077394, (URL ⟨http://doi.acm.org/10.1145/1077391.1077394⟩).

Yardibi, T., Karasan, E., 2010. A distributed activity scheduling algorithm for wireless sensor networks with partial coverage. Wirel. Netw. 16 (1), 213–225. http://dx.doi.org/10.1007/s11276-008-0125-2.

Younis, O., Krunz, M., Ramasubramanian, S., 2008. Location-unaware coverage in wireless sensor networks. Ad Hoc Netw. 6 (7), 1078–1097. http://dx.doi.org/10.1016/j.adhoc.2007.10.003, (⟨http://dx.doi.org/10.1016/j.adhoc.2007.10.003⟩. URL ⟨http://www.sciencedirect.com/science/article/pii/S1570870507001576⟩).

Habib Mostafaei is a Ph.D. student at the department of engineering of Roma Tre University. He received the M.S. degree from the Islamic Azad University Arak branch, in 2009 and the B.S. degree from the Islamic Azad University Khoy branch, in 2006, both in software engineering. Prior to the PhD training at Roma Tre University, he was a lecturer at the Computer Engineering Department of Islamic Azad University (2009–2015). He has published a number of peer reviewed journal and Conference papers on wireless networks. He has served as a reviewer for a number of journal and conference papers since 2013. His research interests include wireless networks, Computer Networks, and Software Defined Networking (SDN). Contact him at mostafae@dia.uniroma3.it



Antonio Montieri is a research associate at NM2 srl (Italy). He received his M.S. Laurea Degree in Computer Engineering in 2015 at the University of Napoli Federico II (Italy). Currently, he strictly collaborates with the members of the research group named Traffic of the University of Napoli Federico II, working in the area of computer networks and multimedia. His research interests are in the networking field with focus on network measurements, traffic analysis and classification, IP topology discovery and alias resolution, and cloud network assessment. Antonio is a co-author of various journal and conference publications. He received the Best Poster Award for his poster "A Platform for Monitoring Public Cloud Networks" at the TMA (Traffic Monitoring and Analysis) PhD School 2016. He has served and serves as peer-reviewer for different international journals (IEEE TSC, Elsevier COMNET, Elsevier TNSM, etc.) and conferences (IEEE Globecom, IEEE ISCC, IEEE ITC, IFIP TMA, etc.) in the field of networking. Contact him at montieri@nm-2.com.

Valerio Persico is a Post Doc at the Department of Electrical Engineering and Information Technology of University of Napoli Federico II. His research interests fall in the area of networking and of IP measurements; in particular his past and present work focuses on cloud network monitoring, traffic classification, IP topology discovery, IP path tracing, and IP alias resolution. Valerio is the recipient of the best student paper award at ACM CoNext 2013. Contact him at valerio.persico@unina.it.

Antonio Pescapé is a Full Professor of computer engineering at the University of Napoli Federico II. His work focuses on Internet technologies and more precisely on measurement, monitoring, and analysis of the Internet. Antonio has co-authored more than 200 conference and journal papers and is the recipient of a number of research awards. Contact him at pescape@unina.it.