Scalable Offline Optimization of Industrial Wireless Sensor Networks

Luigi Palopoli, *Member, IEEE*, Roberto Passerone, *Member, IEEE*, and Tizar Rizano

***Abstract*—Sensor networks are increasingly used to control and monitor industrial and manufacturing processes. In this paper, we consider the problem of optimizing a cost function for wireless sensor networks of this kind under energy consumption constraints. We focus, in particular, on the problem of coverage optimization through scheduling. Following existing approaches, we use a mixed integer linear program formulation. We show how to use partitioning techniques to decompose the problem into separate subproblems, solved individually, overcoming the exponential complexity typical of integer linear programming, while minimizing the loss in optimality. In addition, we evaluate the achieved degree of optimality by computing relatively tight bounds with respect to the optimal solution. Finally, we employ simple but effective heuristics to further improve our solution. The results show that our procedure is very efficient and scalable, and is able to find solutions that are very close to optimal. These characteristics make our approach a perfect fit for large and fixed deployments of wireless sensors, typical in factory automation and industrial applications. To show the generality of the approach, we apply our methodology to three different models of varying complexity.**

***Index Terms*—Design methodology, modeling, optimization methods, partitioning algorithms, wireless sensor networks (WSNs).**

I. Introduction

AMONG embedded devices, wireless sensor networks (WSNs) are emerging as one of the most interesting innovations in terms of potential areas of application. They have an increasing impact in fields such as security, health care, disaster management, agricultural monitoring, and building automation. This paradigm is of particular interest in an industrial setting (witnessed by the WISA system from ABB [1]), because a dense deployment of sensors enables engineers to more accurately control and optimize process parameters in manufacturing and to more effectively manage logistics [2], [3]. The most relevant feature of a WSN is that it is a dynamic distributed system, in which complex tasks are performed through the coordinated action of a large number of small autonomous devices (nodes). A problem of paramount importance for a WSN is maximizing the lifetime of the network. The lifetime is typically bounded by the amount of energy stored in each node upon its deployment. This is true especially when the system is deployed in a remote or harsh environment and/or when maintenance and battery replacement is costly. For instance, shutting down a production line for battery replacement could have a significant impact on productivity. The objective is, therefore, to design distributed algorithms for data processing and resource management that attain an *optimal tradeoff* between functionality, robustness and energy consumption.

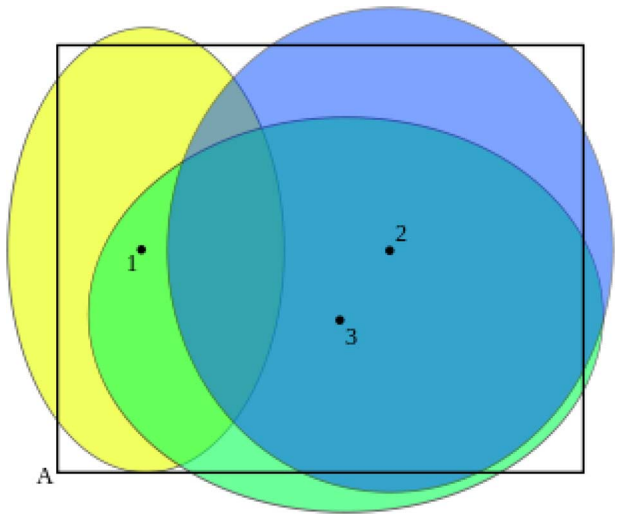


Fig. 1. A topology where wake-up scheduling matters. The square represents the area of interest, the points represent the nodes and the ellipses their sensing range.

A popular way for pursuing this result is the application of "duty-cycling." The idea is to keep a node inactive for a long period of time when its operation is not needed, and then to awake it for a short interval to perform its duties (e.g., to sense the surrounding environment). The operation is repeated periodically and the period is called *epoch* (in our terminology). Clearly, the duty-cycle (i.e., the fraction of time the node is active) is directly linked to the amount of energy spent in every epoch, and hence to the lifetime.

In this paper, we focus on the problem of determining a periodic sequence of wake-ups that maximizes a cost function related to the requested functionality. Because both the epoch and the awake interval are fixed, the problem can be interpreted as a maximization of the system functionality *given* a desired lifetime. Our strategy is to take advantage of the redundancy in the network to improve the efficiency of executing the required task. Consider, for instance, the example in Fig. 1, where three nodes cover, with overlaps, a certain area. In this case, ensuring that nodes 2 and 3 are *not* active at the same time is beneficial, since they share a large portion of the area. In cases like this, a

Manuscript received August 27, 2010; revised December 03, 2010 and February 14, 2011; accepted February 14, 2011. Date of publication March 24, 2011; date of current version May 06, 2011. Paper no. TII-10-08-0221.

The authors are with the Dipartimento di Ingegneria e Scienza dell'Informazione, University of Trento, via Sommarive 14, 38128 Povo di Trento (TN), Italy (e-mail: luigi.palopoli@unitn.it; roberto.passerone@unitn.it; tizar.rizano@unitn.it).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2011.2123904

random schedule can miss considerable opportunities for optimization. In general, since a point in the target area may be monitored by multiple nodes, it is possible to identify a schedule of the nodes that maximizes the average of the covered area. This problem is the representative example of a much larger class in which the cost function is somehow related to the sensing action of the node when it is awake. Several methods have been developed to solve this and similar problems. In general, these can be classified as *centralized* techniques, which make use of global information about the deployment, and *distributed* techniques, which are typically limited by network connectivity, but can more easily adapt to changes in the network. Our work falls in the category of centralized offline techniques, which are the most appropriate for static deployments in relatively controlled environments, such as applications in industrial automation or building management. In this case, an offline method may provide a higher degree of optimization and avoid costly (also in terms of energy consumption) message exchanges required to perform the online computation. This problem may be solved exactly as a mixed integer linear program (MILP), but this approach is impractical for networks with a large number of nodes due to the inherent exponential computational complexity. Instead, we are interested in near-optimal optimization algorithms that are able to scale well with the size of the deployment.

We propose a methodology for solution of large scale optimization problems for WSN centered around the following three ideas: (i) partition the problem into smaller subproblems which are more easily handled by optimization algorithms, thus drastically reducing complexity; (ii) compute bounds on the degree of suboptimality that we achieve; and (iii) further optimize the result using simple but effective heuristics. The main partitioning task is carried out by adapting standard VLSI partitioning techniques for cell placement [4].

The methodology is first instantiated on the coverage maximization problem [5], inspired by previous work of Murphy *et al.* [6]. After discussing the state-of-the-art, Section III briefly recalls the problem formulation and shows how it can be solved. In Section IV, we show that the applicability of the methodology is in fact more general. In particular, we distill a fundamental assumption that underlies the methodology and show two different instances of optimization problems that fall within the assumption range, together with a closed-form expression for the bound. In the first of these extensions, we explicitly consider the probability of failure for the nodes. In the second one, we maximize the minimum coverage achieved across the epoch. The latter problem is particularly interesting when a guaranteed coverage is requested to the WSN at any time. The different steps of the methodology are displayed on a simple example in Section V. An extensive set of experimental data presented in Section VI demonstrates the scalability of the method, while the computed bounds show that there are classes of problems for which it usually identifies a near optimal solution. Finally, Section VII discusses our future work.

II. State-of-the-Art

The problem of optimizing energy constrained industrial WSNs has been approached from the perspective of the communication protocols [7], [8]. Our viewpoint is complementary, and we optimize the schedule for the sensing operation, rather than for communication.

Slijepcevic *et al.* are among the first to address the problem of maintaining *full coverage*, while minimizing power consumption through an active/sleep schedule [9]. Nodes are partitioned into disjoint sets, where each set of nodes completely covers the monitored area. The sets are activated one at a time in a repeating sequence, thus inducing a schedule. The maximum lifetime is obtained using a valid partition with the largest number of sets, a problem which is shown to be NP-complete.[1] The authors propose a quadratic time heuristic, which starts by dividing the monitored area in *fields* covered by exactly the same nodes. The fields covered by the least number of nodes are said to be *critical*. The heuristic covers the most critical fields first, recomputes the criticalities, and then proceeds by selecting a new set of nodes. The authors claim to significantly improve over a simulated annealing approach, both in terms of lifetime and runtime of the optimization. However, no exact solution or bound is derived in the paper.

The work proposed by Cardei *et al.* is the most related to ours in terms of optimization approach [10]. The objective of the work is to maximize the network lifetime by scheduling the activity of the nodes, while maintaining *full coverage* of a *finite set of points*. This is achieved by dividing the sensors into sets which are activated sequentially. Unlike previous work, a sensor may be part of different sets, and the active duration for each set is computed optimally by the algorithm to maximize the lifetime. The optimization problem is formulated as an MILP. To cope with complexity, the authors develop two heuristics, the first based on a linear relaxation, the second similar to that proposed by Slijepcevic and Potkonjak, where each node is only partially assigned to a set for a duration equal to a lifetime granularity parameter. The authors do provide an exact formulation, but only present results for the two heuristics and no bounds.

A different solution strategy is advocated by Alfieri *et al.* [11]. The authors formulate the problem as an MILP and propose a solution strategy that uses two coordinated optimization problems. With the first one, they generate tentative subsets of nodes that have to be switched on at the same time. The second is a linear program that computes the total time every subnet has to be active. The authors also propose a greedy heuristic, amenable to a distributed implementation, but its result can be as low as 40% of the optimal.

The same problem (i.e., maximizing the lifetime and full coverage of a specified set of points) is addressed by Liu *et al.* [12]. The authors propose a two step procedure. In the first step, they find an upper bound of the maximal lifetime by solving a linear program which considers different types of costs (including the cost of sensing and of communication). As a result, they find a specification on the total time a node should be active, which is used in the second step to generate a schedule and a route of the messages to the base station.

[1]We can solve this problem with our algorithm by optimizing the coverage for an increasing number of slots. Since nodes wake up once per epoch, each slot is a disjoint set of nodes. Since we find the optimal solution, we can achieve full coverage whenever possible. The algorithm terminates as soon as full coverage can no longer be achieved. The maximum number of slots for which this happens is bounded by the number of nodes, which is a linear factor for each particular topology. This shows that our problem is also NP-complete: if it were polynomial, we could solve the full coverage problem in polynomial time.

Several factors distinguish our technique from existing work. First, we do not aim at maintaining *full coverage* of an area or of a set of points (e.g., as in [10]), but rather at establishing a periodic schedule that guarantees the largest *average coverage* of an area over a scheduling period, or *minimum coverage* for each slot, given a specified lifetime of the system. By doing so, we are able to compute the optimal tradeoff between average coverage and lifetime. More importantly, we develop a generic methodology that can be applied to models of varying complexity by adjusting the form of the objective function and the constraints. To address the importance of the system reliability, we could impose a minimum coverage constraint in the problem. The constraint guarantees that a percentage of the monitored area is always covered during the lifetime of the system.

As for many approaches, and following [5], we too formulate the problem as an MILP, which has exponential complexity. Our objective is to make this technique scalable to a large and dense number of nodes. To do so, instead of considering a continuous relaxation as in [10], we partition and solve separate problems in a way that minimizes the potential loss in optimality. The amount of partitioning can be controlled to strike the desired tradeoff between optimality and computational complexity. The idea of applying partitioning techniques is also employed by Guo *et al.* [13] who use a statistical approach to count targets in an area while avoiding double-counting. In addition to considering a different problem (maximization of average coverage), the distinctive feature of our work is that we use full information on the partitioning process to compute relatively tight bounds with respect to the optimal solution, even when this is not known.

Our technique assumes a given topology for the network, motivated by our chosen industrial field of application. Achieving the best spatial sensor placement has also been studied [14]. Solution techniques include integer programming [15], greedy heuristics [16]–[18] and virtual force methods [19]. These works complement our temporal distribution and can be applied in parallel to achieve further improvements.

III. The Proposed Methodology

Our goal is to *efficiently* solve large scheduling problems in the context of wireless sensor networks that involve the optimization of some cost function. Our focus on this paper lies, in particular, on coverage problems. These problems are typically characterized by a complexity that grows exponentially in the number of nodes. To avoid the exponential growth, we solve a number of smaller problems obtained through a careful partitioning of the initial setup. Because this approach may lead to a suboptimal global solution, we are interested in deriving bounds to characterize the quality of the result. In this section, we present the general approach and show how the bounds can be computed in a generic way. In the next sections we will consider more specific models, and validate our technique through experimental results.

We will present our methodology by way of a concrete example, which we have used in our previous work [5], and that we briefly summarize here. We assume that a set of *sensor nodes* $\mathcal{N}$ monitors a set of points $\mathcal{P}$. The two sets are related by a binary coverage relation $\mathcal{R} \subseteq \mathcal{N} \times \mathcal{P}$, where $(n,p) \in \mathcal{R}$ if and only if n sees (or covers) p. We denote by $r(p)$ the set of nodes that cover p. Points can be given a different *weight* (certain points may be more important than others) by a function $w : \mathcal{P} \rightarrow \mathbb{R}$. For instance, if a point p represents an entire region, then $w(p)$ could be the area of that region [5]. The identification of points or regions, and the determination of their weight, is orthogonal to the technique that we present in this paper, and can be done using one of the many methods described in the literature [5], [9], [20], [21].

The system operates according to a periodic schedule. The period, called the *epoch*, is denoted by E. In our problem, the lifetime of the system is determined by the *awake* or *activation interval* I of a node, i.e., the interval during which a node is awake in the epoch. The epoch and interval of the system are determined based on the system time constant. In particular, the duration of the epoch corresponds to the maximum interval between two consecutive monitoring of each point. Consequently, the epoch should be chosen no greater than the minimum interval between two events to be monitored.

At any time t, the set of nodes that are awake at that time defines a covering function $S(t)$, equal to the sum of the weights of all the points covered by the nodes that are awake (clearly, the same point is not counted multiple times). The objective of the optimization is to maximize the sum over the epoch of the covered area, i.e., the integral

$$S = \int_0^E S(t). \tag{1}$$

Below, we will refer to this cost function simply as "coverage." For future purposes, it is also useful to introduce the restriction $S_{\mathcal{Q}}(t)$ that considers only a subset $\mathcal{Q} \subseteq \mathcal{P}$ in the computation of the covering function. The integral of this function over the epoch will be denoted as $S_{\mathcal{Q}}$.

We assume the epoch is discretized into an integer number of slots and that nodes are switched on and off only at slot boundaries. It can be shown that this discretization does not impair optimality [5]. Without loss of generality, we will consider slots of size 1. Under this assumption, we can set up our problem as a Mixed Integer Linear Program (MILP). To do so, we introduce a set of binary *coverage* variables $C_{p,k}$ that are equal to 1 if point p is covered in slot k

$$C_{p,k} = \begin{cases} 1, & \text{point } p \text{ is covered during slot } k \\ 0, & \text{otherwise} \end{cases}. \tag{2}$$

Using the coverage variables $C_{p,k}$ and the w function, the function S to be optimized can be computed as

$$S = \sum_{k=0}^{E-1} \sum_{p \in \mathcal{P}} C_{p,k} \cdot w(p). \tag{3}$$

Our objective is to maximize S, whose value depends on the schedule. To model the schedule, we introduce a set of binary

scheduling variables $x_{n,k}$ that are equal to 1 if node n is active in slot k

$$x_{n,k} = \begin{cases} 1, & \text{if } n \text{ is awake in slot } k \\ 0, & \text{otherwise} \end{cases}. \quad (4)$$

The optimization problem can then be stated as follows [5]:

$$\max \sum_{k=0}^{E-1} \sum_{p \in \mathcal{P}} C_{p,k} \cdot w(p) \quad (5)$$

subject to

$$C_{p,k} \leq \sum_{n \in r(p)} x_{n,k}, \quad \forall p \in \mathcal{P}, \forall k \in [0, E-1] \quad (6)$$

$$C_{p,k} \geq x_{n,k}, \quad \forall p \in \mathcal{P}, \forall k \in [0, E-1], \forall n \in r(p) \quad (7)$$

$$\sum_{k=0}^{E-1} x_{n,k} = I, \quad \forall n \in \mathcal{N} \quad (8)$$

$$x_{n,k} \in \{0,1\}, \quad C_{p,k} \in \mathbb{R} \cap [0,1]. \quad (9)$$

In this formulation, constraints (6) and (7) are instrumental to the computation of the $C_{p,k}$ variable, while constraint (8) enforces that each node stays awake for an interval I for each epoch. The ratio I/E is the duty cycle which determines the energy consumed by each node, and, therefore. the lifetime of the system. Because our cost function is proportional to the average coverage (through the epoch), we define this problem as *deterministic average coverage optimization* (DACO).

Although the $C_{p,k}$ variables can be relaxed to be continuous [as indicated by constraint (9)], this formulation displays severe scalability issues that motivated our heuristic approach. In the sequel, we will denote by $\mathbf{P}_{\mathcal{N},\mathcal{P}}$ the coverage problem defined over the sets $\mathcal{N}$ of nodes and $\mathcal{P}$ of points.

Likewise, we denote by $\mathbf{P}^{\star}_{\mathcal{N},\mathcal{P}}$ the schedule of the nodes corresponding to the optimal solution. For the optimal schedule, we will denote by $\mathbf{cP}^{\star}_{\mathcal{N},\mathcal{P}}$ the value of the integral S and by $\mathbf{cP}^{\star}_{\mathcal{N},\mathcal{P}}|_{\mathcal{Q}}$ (with $\mathcal{Q} \subseteq \mathcal{P}$) the value of the integral $S_{\mathcal{Q}}$.

A. A Scalable Algorithm

To efficiently solve the class of problems described in the previous section, we use an algorithm organized in three phases, as shown in Fig. 2. The first phase consists of partitioning the set $\mathcal{N}$ of nodes into disjoint subsets $\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_m$, such that $\mathcal{N} = \bigcup_{i=1}^{m} \mathcal{N}_i$. We then solve the coverage problem on each partition independently, and combine the solutions to generate a schedule. Due to the exponential nature of the covering problem [9], the cumulative time required to optimize the coverage on the partitions is radically smaller than the one required to solve the problem as a whole. However, by dealing with each subproblem independently, we neglect the interaction between nodes contained in different partitions that takes place through the constraints over the shared points, which are removed in the subproblems. While we partition the nodes in a way that minimizes this interaction, the solution is necessarily suboptimal for the problem as a whole. In the final postprocessing phase, we recombine the individual schedules, and improve the coverage using simple heuristics, reducing the gap from the maximum. Remarkably, we are able to estimate upper bounds for the deviation between the optimal and the suboptimal solutions. Clearly,

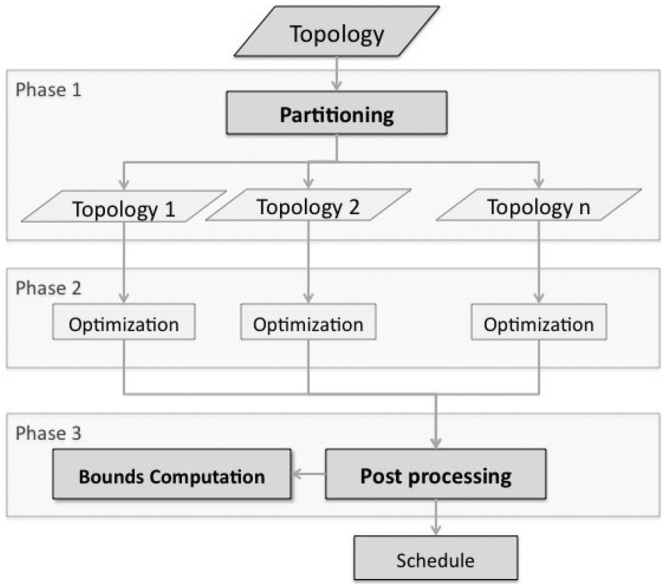


Fig. 2. Overall optimization phases.

the application of the *divide and conquer* approach outlined above greatly improves the scalability of the solution algorithm, as long as the overhead incurred in the first (partitioning) and in the last (recombination) phases can be kept in check. The details of each step, and techniques to compute the bounds, are described below.

B. Partitioning the Problem

The construction of the partition $\{\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_m\}$ out of the set $\mathcal{N}$ can itself be seen as an MILP optimization problem. The choice of which nodes should be placed in each partition is driven by the requirement that interactions between the nodes in the different partitions should be minimized. This way, we can reduce the mismatch between the solution of the entire coverage problem and the aggregate solution of the different subproblems (assumed independent). If two nodes cover disjoint sets of points, then their schedules can be decided independently, since the relative timing of their activations does not affect the total coverage. Conversely, if they share points in their sensing range, then they must be scheduled at different times in order to maximize coverage, since points that are covered by several nodes at the same time are counted only once. In this case, we say that the two nodes interact. The strength of the interaction depends on the weight of the shared points. Since noninteracting nodes can be treated independently, they can be assigned to different partitions. The larger the interaction, instead, the more convenient it is to keep nodes in the same partition, to better account for the shared coverage. Thus, the goal of this phase is to find a partition that minimizes the total interaction between nodes that belong to different sets.

The optimal partitioning problem is well known in the literature, as it applies to communication design and to VLSI placement algorithms [4]. It is easy to express it as an integer linear program [22]. Without loss of generality, consider the case of splitting the set $\mathcal{N}$ into two partitions $\mathcal{N}_1$ and $\mathcal{N}_2$ of equal size. Let v_n be a binary variable, with $n \in \mathcal{N}$, that takes value 0 if

node n belongs to partition $\mathcal{N}_1$ and 1 if it belongs to partition $\mathcal{N}_2$. Equal size partitions can be obtained through the following constraint:

$$\sum_{n\in\mathcal{N}} v_n = \frac{|\mathcal{N}|}{2}.$$

Likewise, for each point p, we introduce two binary variables $V_p^{(1)}$ and $V_p^{(2)}$ that denote if point p is covered by at least one node, respectively, in partition $\mathcal{N}_1$ and $\mathcal{N}_2$. The two variables can be computed as

$$V_p^{(1)} = \bigvee_{n\in\mathcal{N}|(n,p)\in\mathcal{R}} \overline{v}_n, \quad V_p^{(2)} = \bigvee_{n\in\mathcal{N}|(n,p)\in\mathcal{R}} v_n \tag{10}$$

where $\bigvee$ denotes the OR operation and $\overline{v}_n$ denotes the logical negation of v_n. These two expressions can be translated into linear constraints on the variables by standard techniques. We are interested in a partition that minimizes the total weight of the points covered by at least one node from each partition, i.e., the overlapping area if the points represent regions. With our definitions, we need to minimize the following cost function:

$$\sum_{p\in\mathcal{P}} w(p)\left(V_p^{(1)} + V_p^{(2)} - 1\right). \tag{11}$$

Therefore, we obtain a Boolean Linear Program (BLP).

Although the number of variable for this problem is much smaller than for the coverage problem, the asymptotic behavior of solving the partitioning problem this way is still exponential. To avoid this, we can use efficient and effective heuristics for the partitioning problem, such as the algorithm proposed by Fiduccia and Mattheyses [4], initially developed for VLSI standard cell placement, which has linear complexity in the number of points covered by the nodes. As shown in the experimental section, the price to pay in terms of distance from the optimal solution is acceptable.

When the number of nodes is large, partitioning can be applied recursively or the number of partitions can be increased to obtain separate sets of nodes with the desired size. This way, a tradeoff can be established between optimality (fewer partitions) and computational efficiency. We discuss how to combine the schedules for the different partitions in Section III-E.

C. Solving and Bounding the Problem

From the two partitions $\mathcal{N}_1$ and $\mathcal{N}_2$, we can identify three sets of points: $\mathcal{P}_1$ contains the points seen only by nodes in $\mathcal{N}_1$, $\mathcal{P}_2$ contains the points seen only by nodes in $\mathcal{N}_2$, and $\mathcal{P}_{1,2}$ contains the points seen by nodes in both partitions. In fact, as discussed, partitioning is done so that the total weight of the points contained in $\mathcal{P}_{1,2}$ is minimal.

A possible solution for the coverage problem can therefore be found by solving the two problems $\mathbf{P}_{\mathcal{N}_1,\mathcal{P}_1}$ and $\mathbf{P}_{\mathcal{N}_2,\mathcal{P}_2}$, and then combining the two corresponding optimal schedules $\mathbf{P}^\star_{\mathcal{N}_1,\mathcal{P}_1}$ and $\mathbf{P}^\star_{\mathcal{N}_2,\mathcal{P}_2}$. Let $\mathbf{P}^+_{\mathcal{N},\mathcal{P}}$ be the combined schedule obtained when nodes $\mathcal{N}_1$ are scheduled according to $\mathbf{P}^\star_{\mathcal{N}_1,\mathcal{P}_1}$, and nodes $\mathcal{N}_2$ are scheduled according to $\mathbf{P}^\star_{\mathcal{N}_2,\mathcal{P}_2}$, and let $\mathbf{cP}^+_{\mathcal{N},\mathcal{P}}$ be the coverage obtained with this schedule. This solution is feasible for the global problem $\mathbf{P}_{\mathcal{N},\mathcal{P}}$, since the combination of the local constraint simply the satisfaction of the global ones. However, it may be suboptimal: while the solutions of the two subproblems maximize the coverage in $\mathcal{P}_1$ and $\mathcal{P}_2$, respectively, they fail to consider the interaction on the "overlapped" points in $\mathcal{P}_{1,2}$.

The general structure of the problem, however, allows us to compute an upper bound on the distance between the optimal and the suboptimal coverage ($\mathbf{cP}^\star_{\mathcal{N},\mathcal{P}} - \mathbf{cP}^+_{\mathcal{N},\mathcal{P}}$). Given the optimal schedule $\mathbf{P}^\star_{\mathcal{N},\mathcal{P}}$, we can restrict the computation of the coverage to the two sets $\mathcal{P}_1$ and $\mathcal{P}_2$ getting $\mathbf{cP}^\star_{\mathcal{N},\mathcal{P}}|_{\mathcal{P}_1}$ and $\mathbf{cP}^\star_{\mathcal{N},\mathcal{P}}|_{\mathcal{P}_2}$, respectively. Because the subproblems are less constrained, it follows that:

$$\begin{aligned}\mathbf{cP}^\star_{\mathcal{N}_1,\mathcal{P}_1} &\geq \mathbf{cP}^\star_{\mathcal{N},\mathcal{P}}|_{\mathcal{P}_1},\\ \mathbf{cP}^\star_{\mathcal{N}_2,\mathcal{P}_2} &\geq \mathbf{cP}^\star_{\mathcal{N},\mathcal{P}}|_{\mathcal{P}_2}.\end{aligned} \tag{12}$$

The total coverage of $\mathbf{cP}^\star_{\mathcal{N},\mathcal{P}}$ can be found by adding together the contributions of the three sets $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}_{1,2}$

$$\mathbf{cP}^\star_{\mathcal{N},\mathcal{P}} = \mathbf{cP}^\star_{\mathcal{N},\mathcal{P}}|_{\mathcal{P}_1} + \mathbf{cP}^\star_{\mathcal{N},\mathcal{P}}|_{\mathcal{P}_2} + \mathbf{cP}^\star_{\mathcal{N},\mathcal{P}}|_{\mathcal{P}_{1,2}}.$$

Hence, in view of (12), we can write

$$\mathbf{cP}^\star_{\mathcal{N},\mathcal{P}} \leq \mathbf{cP}^\star_{\mathcal{N}_1,\mathcal{P}_1} + \mathbf{cP}^\star_{\mathcal{N}_2,\mathcal{P}_2} + \mathbf{cP}^\star_{\mathcal{N},\mathcal{P}}|_{\mathcal{P}_{1,2}}.$$

Considering that

$$\mathbf{cP}^+_{\mathcal{N},\mathcal{P}} = \mathbf{cP}^+_{\mathcal{N},\mathcal{P}}|_{\mathcal{P}_1} + \mathbf{cP}^+_{\mathcal{N},\mathcal{P}}|_{\mathcal{P}_2} + \mathbf{cP}^+_{\mathcal{N},\mathcal{P}}|_{\mathcal{P}_{1,2}}$$

and that obviously

$$\mathbf{cP}^+_{\mathcal{N},\mathcal{P}}|_{\mathcal{P}_1} = \mathbf{cP}^\star_{\mathcal{N}_1,\mathcal{P}_1}, \quad \mathbf{cP}^+_{\mathcal{N},\mathcal{P}}|_{\mathcal{P}_2} = \mathbf{cP}^\star_{\mathcal{N}_2,\mathcal{P}_2}$$

we obtain

$$\mathbf{cP}^\star_{\mathcal{N},\mathcal{P}} - \mathbf{cP}^+_{\mathcal{N},\mathcal{P}} \leq \mathbf{cP}^\star_{\mathcal{N},\mathcal{P}}|_{\mathcal{P}_{1,2}} - \mathbf{cP}^+_{\mathcal{N},\mathcal{P}}|_{\mathcal{P}_{1,2}}. \tag{13}$$

The value of $\mathbf{cP}^+_{\mathcal{N},\mathcal{P}}|_{\mathcal{P}_{1,2}}$ can be easily computed once the schedule $\mathbf{P}^+_{\mathcal{N},\mathcal{P}}$ is known, i.e., after the two individual subproblems have been solved. The situation is different for $\mathbf{cP}^\star_{\mathcal{N},\mathcal{P}}|_{\mathcal{P}_{1,2}}$, since we do not have the global optimal solution. We can however bound it by first solving the problem $\mathbf{P}_{\mathcal{N},\mathcal{P}_{1,2}}$, and observing that

$$\mathbf{cP}^\star_{\mathcal{N},\mathcal{P}}|_{\mathcal{P}_{1,2}} \leq \mathbf{cP}^\star_{\mathcal{N},\mathcal{P}_{1,2}}. \tag{14}$$

The problem $\mathbf{P}_{\mathcal{N},\mathcal{P}_{1,2}}$ can itself be of high complexity, but is defined on a lower number of points, and therefore more practical. If still too complex, this second problem itself can be partitioned, and an upper bound can be computed recursively, as long as the number of shared points decreases from one iteration to the next. Section VI shows that the bounds that we compute using this technique are tight for the majority of the cases.

Remark 1: An interesting open point in the procedure outlined above is whether the shared points $\mathcal{P}_{1,2}$ should be neglected in the optimization of each of the partition (as suggested in the discussion above), included in one of them or included in

both. We carried out a large set of tests for each of these possibilities, but the results on the convenience of one of these possibility appear inconclusive: we could not identify conditions on the topology that produce a clear "winner" and the differences were more often than not very small (below a few percent). For implementation ease, in this paper, we opted for including the shared points in the optimization problems defined for all partitions.

D. Closed-Form Bound

While the expression of the bound given in (13) is generic, the particular structure of the model may suggest other ways of computing the bounds that lend themselves to a closed-form solution. In our particular example, an upper bound for $\mathbf{cP}^{\star}_{\mathcal{N},\mathcal{P}}|_{\mathcal{P}_{1,2}}$ can be found considering the best case situation where every point in $\mathcal{P}_{1,2}$ is covered by all the nodes that have it in their range, but in separate slots (disjoint awake intervals)

$$\mathbf{cP}^{\star}_{\mathcal{N},\mathcal{P}}|_{\mathcal{P}_{1,2}} \leq \sum_{p\in\mathcal{P}_{1,2}} \min\left\{\frac{E}{I}, |r(p)|\right\} w(p).$$

The upper bound B of $\mathbf{cP}^{\star}_{\mathcal{N},\mathcal{P}} - \mathbf{cP}^{+}_{\mathcal{N},\mathcal{P}}$ can, therefore, be computed as

$$B = \sum_{p\in\mathcal{P}_{1,2}} \min\left\{\frac{E}{I}, |r(p)|\right\} w(p) - \mathbf{cP}^{+}_{\mathcal{N},\mathcal{P}}|_{\mathcal{P}_{1,2}}.$$

In practice, the bound assumes that the optimal solution will do as well as the subproblems on the individual partitions (which is optimistic), and will do the absolute best on the overlaps (which is also optimistic). Our experiments show that, because the overlaps are minimized, the bounds are in practice very tight (see Section VI).

E. Final Optimization

The obvious way to combine schedules computed independently for each partition is simply to synchronize them at the beginning of the epoch. If we define the "phase" of a schedule as the time in which it is started in the epoch, this corresponds to running the separate schedules "in phase." This choice is, however, arbitrary. Recall, in fact, that a schedule is periodic and that we evaluate the coverage over the entire epoch. Coverage is therefore invariant to translations of the schedule on the time axis. Indeed, the solution returned by the ILP solver is only one of several equivalent solutions that can be obtained by shifting the awake interval of all nodes repeatedly one slot to the right or to the left.

Coverage on points that are shared between nodes of different partitions, however, is not optimal, and is therefore affected by changing the relative phase of the schedules. One way to improve the solution is therefore to recompute the total coverage under all possible shifts, and run the schedules, possibly out of phase (starting at different times in the epoch), for the best result. In practice, we need only recompute the coverage of the points shared by the partitions (the "overlaps"), since, as pointed out, the coverage on the partitions themselves is constant. Thus, the complexity of this computation is linear in the number of overlapping points times the number of slots in the epoch. This heuristic is particularly simple and fast, but provides excellent results. In addition, since the coverage on the individual partitions is not affected by the operation, we can recompute tighter bounds on the solution that take the new schedule on the overlaps into account.

The extension to several schedules, obtained from a recursive application of the partitioning procedure, is not totally straightforward, since the number of possible relative shifts grows exponentially with the number of partitions. For instance, while with two partitions one has to test only E shifts (i.e., the number of slots), with four partitions the number of tests grows to E^3. In general, the number of shifts to test is given by E^{T-1}, where T is the number of partitions. For this work, we do not investigate efficient ways to optimize this step and simply recombine the schedules in the reverse order of partitioning (from the leaves to the root). This choice is justified by the consideration that adjacent partitions are also those that likely interact the most.

IV. Extended Models

The general methodology shown in Fig. 2 is applicable to a variety of models. Indeed, the definition of the methodology is underpinned by one simple assumption.

Assumption 1: Consider the problem of finding a global schedule for a set of nodes that satisfies all the constraints (i.e., it is feasible). Suppose the nodes are partitioned into disjoint sets, for which separate subproblems are defined. Then, every collection of schedules that are feasible for each subproblem is also feasible for the global problem.

The assumption is obviously satisfied by the DACO problem. Indeed, the constraints that are satisfied by the solution of each subproblem imply the ones in the global problem [(6) to (9)], or in any case the existence of a consistent assignment for the decision variables. In fact, the class of scheduling problems for which Assumption 1 is satisfied is much larger. In this section, we propose two representative examples.

A. Maximizing the Minimum Coverage

The solution of the DACO problem maximizes the total area covered throughout the epoch given a desired lifetime. Because each node is switched on once in the epoch, it is implicitly guaranteed that each point is covered at least once. In some applications such a guarantee may be insufficient. Instead it may be desirable to have a uniform coverage across the slots. This can be achieved by changing the cost function (5), and requiring that the *minimum* coverage in all slot be maximized

$$\max \min_{k\in[0,E-1]} \sum_{p\in\mathcal{P}} C_{p,k} w(p). \tag{15}$$

This problem can be rephrased as a BLP using an auxiliary variable μ and introducing some additional constraint

$$\max \mu \tag{16}$$

$$\mu \leq \sum_{p\in\mathcal{P}} C_{p,k} w(p), \qquad \forall k. \tag{17}$$

The additional constraints do not invalidate the compliance of the problem with Assumption 1. Therefore, the proposed methodology is entirely applicable. In the remainder of this

paper, we will refer to this problem as Deterministic Minimal Coverage Optimization (DMCO).

Remark 2: Compared to DACO, DMCO offers higher temporal reliability on the coverage. Indeed, the DACO problem could, in principle, produce schedules in which most of the coverage is concentrated in certain slots, while in others the coverage could be very low. This is impossible with DMCO. Additional guarantees on the coverage of "sensitive" points could be attained by inserting additional constraints or by attaching a different weight to the points to emphasize the relative importance of some of them. These slight modifications of the problem, can be easily addressed by simple adaptations of the algorithm.

Bounding the distance from the optimum. By applying our methodology to the DMCO problem we are able to compute a suboptimal solution and a bound, as described in Section III-C. For the bounds, this requires solving an optimization problem defined on the points shared between the partitions. Although this problem is simpler than the original one (since there are fewer points and nodes involved) and it can be solved by recursively applying the partitioning methodology proposed here, its solution is time demanding if compared with the closed-form bound identified for the DACO problem. Unfortunately, a tight closed-form bound is not easy to find in this case. Nonetheless, it is possible to simplify the solution of this new optimization problem by continuous relaxation (i.e., converting the binary variables to reals bounded in the [0, 1] interval). Indeed, as shown in (14), we are in this case interested in finding an upper bound for $\mathbf{cP}^{\star}_{\mathcal{N},\mathcal{P}_{1,2}}$, which is obviously provided by the continuous relaxation

B. Addressing Failures

One of the most relevant distinctive features of a WSN is the intrinsic lack of reliability of the nodes. Not only is it possible that a node fails to detect the event of interest, but it could also fail to report it (e.g., due to a packet drop). A possible way for modeling this limitation is by introducing a probability q_n for each node n to "correctly" sense the point during its wake-up interval. As a result, in a given slot the area is covered with some probability (depending on the number of nodes that are on). A reasonable cost function to maximize is in this setting the *expected value* of the covered area. Since this quantity changes in the different slots, we can reasonably maximize its average in the epoch. This problem is henceforth referred to as expected average coverage optimization (EACO), which is proportional to the sum of the expected coverage through the epoch.

For slot k, let $C_{p,k}$ represent the *probability* of covering the point p in that slot. For the sake of simplicity, we restrict to the case that two nodes a and b cover a point p. If the probabilities are independent, then $C_{p,k}$ is given by the probability that a senses p, plus the probability that b senses p, minus the probability that both a and b sense p

$$C_{p,k} = x_{a,k} \cdot q_a + x_{b,k} \cdot q_b - x_{a,k} \cdot x_{b,k} \cdot q_a \cdot q_b. \tag{18}$$

This equation straight forwardly descends from the axiomatic theory of probability. Using this construction, we can obtain the EACO problem by simply replacing (6) and (7) with the equations that for each point p and each slot k express the probability $C_{p,k}$, and by changing the meaning of $C_{p,k}$ in the cost function (3) (which for DACO is a binary variable, while for EACO is a real variable representing a probability).

In order to cast the problem as a BLP, we can use standard techniques. For instance, the nonlinear constraint associated with the probability in (18) can be "linearized" by an additional variable $R_{p,k} \geq 0$ that replaces the term $x_{a,k} \cdot x_{b,k}$

$$C_{p,k} = x_{a,k} \cdot q_a + x_{b,k} \cdot q_b - R_{p,k} q_a \cdot q_b \tag{19}$$

$$R_{p,k} \leq x_{a,k} \tag{20}$$

$$R_{p,k} \leq x_{b,k} \tag{21}$$

$$R_{p,k} \geq x_{a,k} + x_{b,k} - 1 \tag{22}$$

This way, $R_{p,k} = 1$ if and only if $x_{a,k} = x_{b,k} = 1$, and 0, otherwise.

This idea can easily be generalized to the case of more than two nodes covering a point (we omit the details for the sake of brevity). This way, we finally come up with a BLP that obviously falls within the range of Assumption 1. Hence, it is possible to apply the methodology presented in this paper.

Bounding the distance from the optimum. Contrary to the DMCO problem, for the EACO problem it is possible to identify a closed-form bound, which is proven very tight in our experiments.

We define $\mathcal{F}(n,p,k)$ as the event that node n that is active in slot k successfully covers point p, which is in its sensing range. We denote by $\mathcal{P}(\mathcal{F}(n,p,k))$ the probability of the event. This probability is simply given by

$$\mathcal{P}\left(\mathcal{F}(n,p,k)\right) = x_{n,k} \cdot q_n.$$

In this setting, the probability $C_{p,k}$ is the joint probability for all events $\mathcal{F}(n,p,k)$ associated with the nodes that cover p and are active in slot $k (n \in \mathcal{N}_{p,k})$

$$C_{p,k} = \mathcal{P}\left(\bigcup_{n \in \mathcal{N}_{p,k}} \mathcal{F}(n,p,k)\right). \tag{23}$$

An upper bound for $\mathbf{cP}^{\star}_{\mathcal{N},\mathcal{P}}|_{\mathcal{P}_{1,2}}$ in the estimated average (EACO) model can be found by applying Boole's Inequality

$$C_{p,k} \leq \mathcal{P}\left(\bigcup_{n \in \mathcal{N}_{p,k}} \mathcal{F}(n,p,k)\right) \leq \sum_{n \in \mathcal{N}_{p,k}} \mathcal{P}\left(\mathcal{F}(n,p,k)\right). \tag{24}$$

In plain words, the joint probability of a set of events is bounded by the sum of their probabilities. Therefore, the expected value of the coverage of point p in k is upper-bounded by $\sum_{n \in \mathcal{N}_{p,k}} \mathcal{P}(\mathcal{F}(n,p,k)) w(p)$ and its total expected covered area through the epoch is upper-bounded by $\sum_{k=0}^{E-1} \sum_{n \in \mathcal{N}_{p,k}} \mathcal{P}(\mathcal{F}(n,p,k)) w(p)$. As for the DACO problem, the bound can be further refined considering that a point cannot, in any case, be covered for more times than the number of available slots. Hence, an upper bound for the coverage of the points shared by two partitions $(\mathbf{cP}^{\star}_{\mathcal{N},\mathcal{P}}|_{\mathcal{P}_{1,2}})$ can be computed as follows:

$$\mathbf{cP}^{\star}_{\mathcal{N},\mathcal{P}}|_{\mathcal{P}_{1,2}} \leq \sum_{p \in \mathcal{P}_{1,2}} \min\left\{\frac{E}{I}, \overline{C_p}\right\} w(p)$$

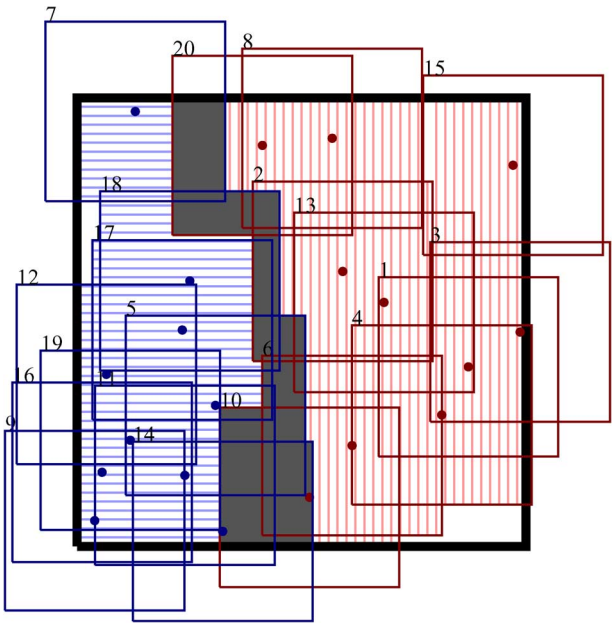


Fig. 3. The WSN topology with 20 nodes. The red areas are the regions covered by nodes in the first partition, the blue areas are the regions covered by nodes in the second partition, and the gray area are regions covered by nodes in both partition.

where

$$\overline{C_p} = \sum_{k=0}^{E-1} \sum_{n \in \mathcal{N}_{p,k}} \mathcal{P}\left(\mathcal{F}(n,p,k)\right).$$

The bound B can, therefore, be computed as follows:

$$B = \sum_{p \in \mathcal{P}_{1,2}} \min\left\{\frac{E}{I}, \overline{C_p}\right\} w(p) - \mathbf{cP}^{+}_{\mathcal{N},\mathcal{P}}|_{\mathcal{P}_{1,2}}. \quad (25)$$

V. Example

In order to clarify the most significant aspects of our methodology, it is useful to show its concrete application on an example. In this section we show how to solve the EACO problem for a simple topology consisting of 20 nodes that are deployed in a 100 × 100 unit region, as shown in Fig. 3. In the example, nodes 1–7, 9, 11–17, 19 have probability of success 1, nodes 8 and 18 have very low probability of success 0.2, and nodes 5 and 20 have probability of 0.5. Each node has sensing range of 20 units and is placed randomly in the region. For simplicity, the sensing region of each node is represented as a square centered in the node deployment position. As a first phase, a spatial scan algorithm can be used to identify rectangular regions, each one covered by a set of nodes [5]. Such regions can be seen as the set of points $\mathcal{P}$ in the EACO formulation, and the area of each region p is used as the weight function $w(p)$.

The first step of the methodology is to partition the set of nodes in two sets, minimizing the area of the shared regions (points). In this case, it is sufficient to use two partitions, since a partition of ten nodes is easily manageable by standard MILP solvers. The two partitions identified by the Fiduccia and Mattheyses algorithm share 39 regions out of 137, and are shown in dark inFig. 3.

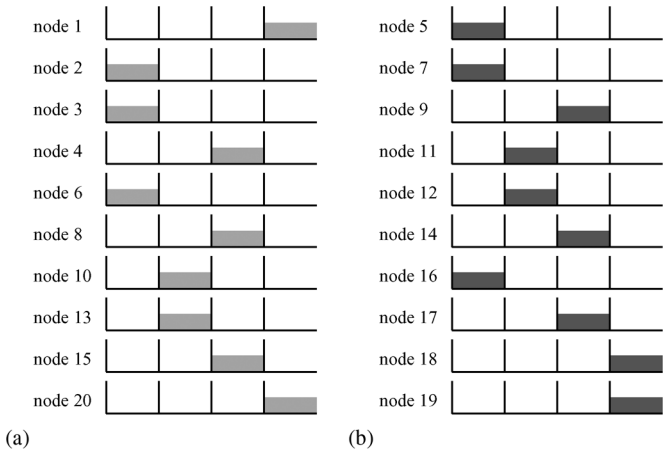


Fig. 4. The schedule of the topology after partition. (a) The schedule for the first partition. (b) The schedule for the second partition.

The second step is to carry out the optimization for each partition separately. The result is a schedule for the nodes of each partition that maximizes the sum of average coverage over the slots. The schedules produced in this way are shown in Fig. 4. For instance, node 1 and node 20 are scheduled in the same slot because they do not share any point, while node 4 is scheduled in a different slot from node 1 since their sensing area is largely overlapping.

The recombined schedule for the two partitions is shown in Fig. 5. Fig. 5(a) is the combined schedule of partition 1 and partition 2. The total expected area covered by this schedule over four slots is 10688.09 (the average is 2672.02) for partition 1 and 9476.35 (2369.09 for each slot) for partition 2. Accounting for overlaps, the total average for the two partitions is 4886.90.

To account for the interaction on the shared points, we apply the postprocessing step described in Section III-E: the schedule of partition 1 is rigidly shifted with respect to the schedule of partition 2 to increase the coverage. The new schedule is shown in Fig. 5(b). The average coverage after postprocessing grows to 4972.5, a 1.7% improvement over the coverage before postprocessing. Using the closed-form bound shown in Section IV-B, we can approximate the distance of the postprocessing coverage from the optimal coverage which is 265.42 (1.32%). The actual distance of the postprocessing coverage from the optimal coverage is 181.460254 (0.9%). For the sake of completeness, we report in Fig. 5(c) the optimal schedule resulting from the solution of the EACO problem as a whole.

VI. Experimental Results

The different phases of the methodology described in this paper have been implemented by a combination of scripts and Java programs.

As input, we use a file describing the topology of the nodes on a test area and their sensing range (as shown in Fig. 3). The first step is to partition the nodes. To this end, we construct a graph $G = (V, E)$. The set of vertices V corresponds to the set of nodes $\mathcal{N}$ in the system. Edges, instead, are used to model

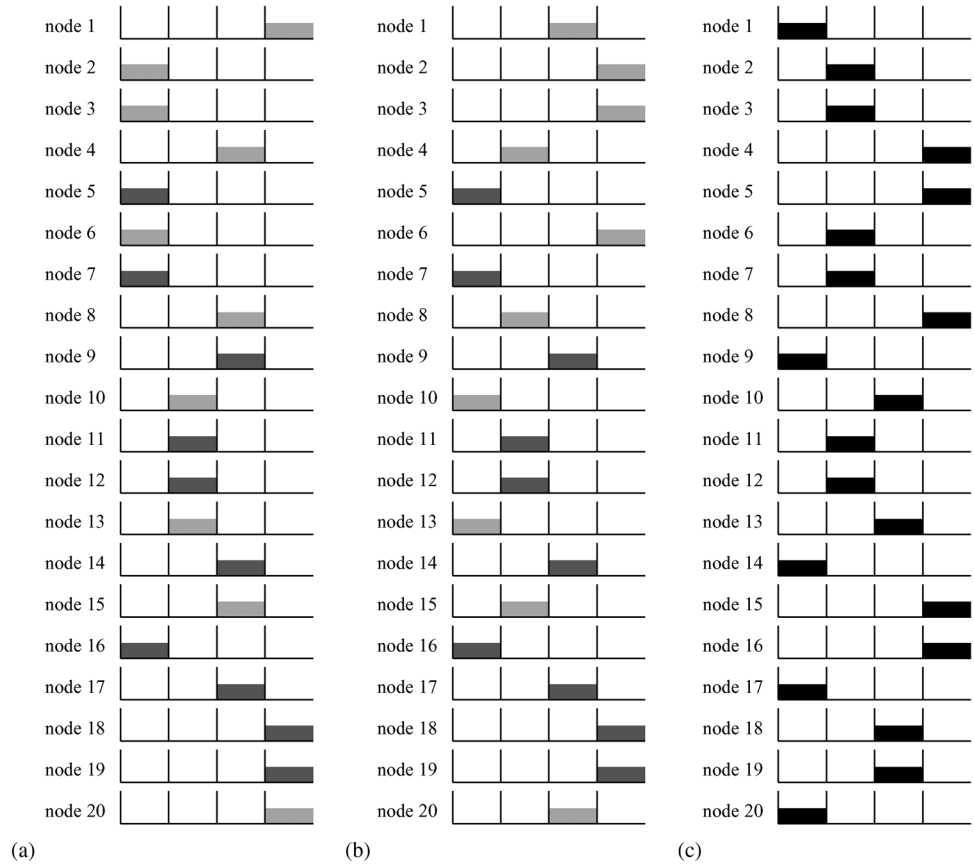


Fig. 5. The schedule of the topology. (a) The combined schedule of the first and second partition. (b) The sub optimal schedule of the topology after post optimization. (c) The optimal schedule for the topology.

the overlaps between the sensing ranges of the nodes: an edge exists between two nodes n_i and n_j whenever they cover a common point. Edges are weighted by the sum of the weights of the common points in the sensing range of the nodes, as a measure of the degree of overlap. For instance, if nodes n_i and n_j have points p_1 and p_2 in common, with $w(p_1) = w_1$ and $w(p_2) = w_2$, then n_i and n_j would be connected by an edge with weight $w_1 + w_2$. Our objective is to partition the set of nodes so that the overlap between the partitions is minimized. This can be accomplished by partitioning the vertices into two sets of equal size, so that the sum of the weights of the edges that go across the partitions is minimum. This formulation matches exactly the one used by min-cut placers in VLSI design, for which several tools are available. In this work, we have used the freely available MLPart [23], part of the Capo placer, which is an efficient implementation of the Fiduccia–Mattheyses algorithm.

Any standard MILP solver can be used to carry out the optimization (second phase of the algorithm). The results reported in this section were obtained using CPLEX (by IBM Inc.). Finally, our software also performs the optimal rephasing of the schedules (third phase) described in Section III-E.

We have evaluated the effectiveness of our approach on several random topologies, ranging from a few tens of nodes to a thousand. The first thing we tested with our experiments was the performance and the scalability of the methodology. Topologies were randomly generated with uniform distribution. Experiments with other distributions show consistent results. Points are taken to be the regions covered by the different groups of nodes (see [5] and [9]) and their weight is set equal to the area of the region they represent. To test different densities (average number of nodes that see a point), we changed the sensing range of the nodes for each topology. We formulated and solved the DACO, the DMCO and the EACO problems for each of the topologies thus obtained. The results are reported in Table I.

Each row in the table refers to a family of five random topologies. For each family of topologies we report the computation time, including the partitioning overhead, and the bound averaged through the different topologies of each family. For the DACO and EACO problems, we report the closed-form bound, and for the DMCO we computed the bound by continuous relaxation of the optimization problem defined on the shared points. For the DACO and EACO problems, in our experiments, the closed-form bounds were considerably tighter and easier to obtain than the ones based on continuous relaxation on the shared points (which are omitted for brevity). As it is possible to see, the problem is tractable even for a very large number of points and nodes. The EACO and DACO problems are solved very efficiently even on a 32 bits INTEL I686 processor with 4 GB of RAM. The similarity in the computation time for the EACO and the DACO problems is due to the partitioning overhead that dominates over the optimization of the partitions. Decreasing the number of partitions, however, quickly makes the EACO and the DMCO problems intractable (we set a time out at two hours for the optimization time). A thorough tradeoff

TABLE I
PERFORMANCE RESULTS, LARGE TOPOLOGIES

Top.	Nodes	Points	Partitions	Time (s) (Dev)	Bound (Dev) %
DACO					
1	300	$\approx$ 1800	16	7.16 (0.44)	2.95 (0.48)
2	300	$\approx$ 1800	32	10.66 (0.65)	4.27 (0.70)
3	500	$\approx$ 3000	16	11.52 (1.36)	2.48 (0.41)
4	500	$\approx$ 3000	64	24.31 (1.65)	4.90 (0.65)
5	700	$\approx$ 2900	64	19.03 (0.38)	2.99 (0.14)
6	1000	$\approx$ 2800	128	32.80 (0.15)	2.76 (0.26)
EACO					
1	300	$\approx$ 1800	16	timeout	-
2	300	$\approx$ 1800	32	19.95 (5.02)	2.51 (0.43)
3	500	$\approx$ 3000	16	timeout	-
4	500	$\approx$ 3000	64	27.70 (4.12)	2.81 (0.43)
5	700	$\approx$ 2900	64	25.61 (2.78)	1.32 (0.11)
6	1000	$\approx$ 2800	128	33.30 (0.13)	1.16 (0.25)
DMCO					
1	300	$\approx$ 1800	16	timeout	-
2	300	$\approx$ 1800	32	45.03 (12.51)	17.66 (1.74)
3	500	$\approx$ 3000	16	timeout	-
4	500	$\approx$ 3000	64	49.10 (34.49)	17.37 (1.43)
5	700	$\approx$ 2900	64	191.55 (11.55)	17.17 (2.21)
6	1000	$\approx$ 2800	128	33.66 (0.11)	13.45 (1.62)

TABLE II
TIGHTNESS OF BOUNDS

Nodes	Points	Partitions	Bound %	Act. Dist. %	Diff. %
DACO					
300	$\approx$ 1800	16	2.95	2.78	0.17
300	$\approx$ 1800	32	4.27	3.87	0.40
500	$\approx$ 3000	16	2.48	2.11	0.38
500	$\approx$ 3000	64	4.90	4.20	0.70
700	$\approx$ 2900	64	2.99	2.32	0.67
1000	$\approx$ 2800	128	2.76	2.50	0.26
EACO					
1000	$\approx$ 2800	128	1.16	1.13	0.04
DMCO					
300	$\approx$ 1900	32	17.66	7.59	10.07
500	$\approx$ 1500	64	17.37	7.58	9.79
700	$\approx$ 2900	64	17.17	4.99	12.18
1000	$\approx$ 2800	128	13.45	3.76	9.69

analysis between partitioning and optimization is reported in Section VI-A3. The solution is reasonably close to the optimal one, its bound ranging from 1% to 5%. The solution of the DMCO is much more demanding (although still tractable if sufficiently partitioned). Also, the bound of the solution was comparatively higher (from 13% to 17%).

Assessing the Tightness of the Bound: From the previous experiments, the point remained open of understanding if a large bound (e.g., for DMCO) is the result of a low quality of the solution or of a high degree of conservativeness of the bound. To investigate the matter we have solved the problem with a direct approach (i.e., without partitioning). With a reasonably low density we could make the test also for large topologies, getting the solution in a few hours instructing the CPLEX solver to stop when it was 0.5% apart from the continuous relaxation. In Table II, we report for each family of topologies the actual distance between the solution produced by CPLEX for the global problem and the one produced by our methodology. For the EACO and the DACO problems, the bound is very tight even applying the closed-form version. For DMCO, the actual distance is less than one half of the one estimated by the bound.

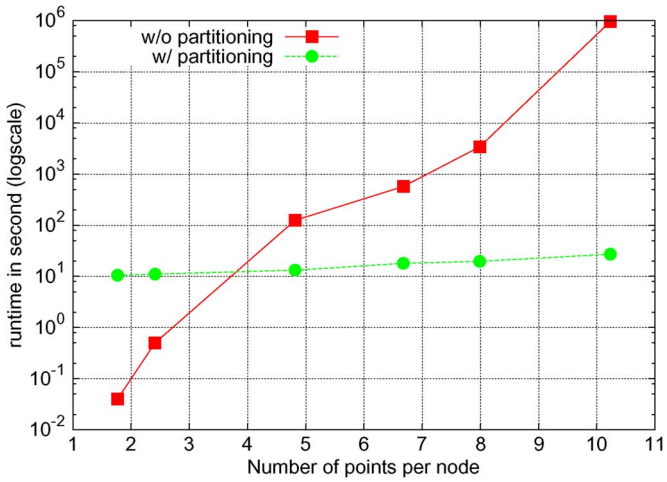


Fig. 6. Time comparison of optimization experiments without partitioning and with partitioning.

This is suggestive of a good performance of the algorithm and of the need for further investigation on the bound. To improve the bound, we tried solving the optimization problem on the shared points exactly (rather than with the continuous relaxation) without significant improvements (not even 1% in the face of a substantial increase of the computation time).

Comparison With the Solution Time of the Global Problem: We compare the computation time of the optimization algorithm with and without partitioning. We present the results for the DACO problem, for which the solution of the global problem was reasonably feasible even for topologies characterized by high density (cutting off the solution when it was 1.5% from the optimum of the continuous relaxation). The results are reported in Fig. 6, where we considered topologies with 300 nodes and different densities. The density is reported on the X axis as the number of points seen on average by a node, while the Y axis shows the computation time in a logarithmic scale, highlighting the difference in orders of magnitude. The greater efficiency of our approach is evident, except for very sparse topologies for which the overhead of partitioning the problem is dominant. The performance gain increases with the density and it reaches between four and five orders of magnitude for ten points per node.

Choosing the Number of Partitions: In this section, we present experiments aimed at providing guidelines for the choice of the number of partitions. As discussed above, the convenience of our algorithm over the solution of the complete problem is very much related to the density of the nodes. Therefore, the evaluation of the number of partitions has to be made for a fixed density. To explore this issue, we have carried out a set of experiments for fixed density and varying number of nodes and partitions. The average density was approximately six points per node.

Fig. 7(a) compares the optimization time and the overhead time, for an increasing number of partitions and a fixed number of nodes, in this case equal to 100. When the number of partitions is small, the optimization time prevails over the partitioning overhead. When the number of partitions is large, the relation between these two components is inverted. The plot also

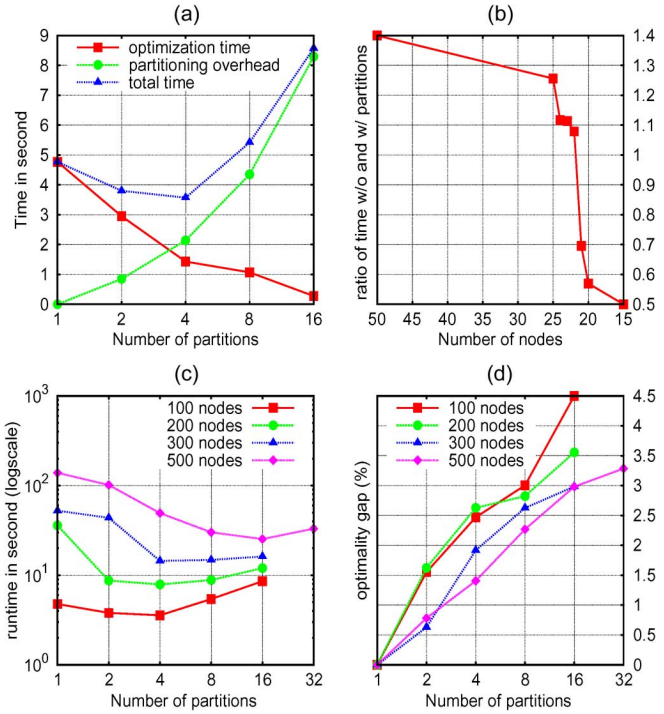


Fig. 7. (a) Comparison between optimization time and the timing overhead for experiments with 100 nodes. (b) The ratio of cumulative runtime without and with partitions for different number of nodes. (c) Cumulative runtime of optimization for 1, 2, 4, 8, 16, and 32 partitions. (d) The optimality gap for experiments with 1, 2, 4, 8, 16, and 32 partitions.

shows the cumulative runtime of the optimization, which, as expected, presents a minimum, located at four partitions for these experiments (25 nodes per partition).

We are interested in characterizing, for a given density of nodes (and also for the particular solver and computer in use), how many times a problem should be partitioned for both efficiency and optimality. We do this heuristically by benchmarking the ratio between the computation time for random topologies when the problem is not partitioned and when it is partitioned in two, for an increasing number of nodes. The results of these experiments are shown in Fig. 7(b). From the plot, we observe that when the number of nodes decreases below about 20–25, the ratio falls below 1, indicating that partitions of about 25 nodes and below should not be further partitioned.

Fig. 7(c) shows additional cumulative performance curves for topologies with 200, 300, and 500 nodes. If we select 25 nodes per partition, our heuristics indicate that the best performance should be achieved for four partitions for 100 nodes, eight partitions for 200 nodes, 12 partitions for 300 and 20 for 500. This is fairly consistent with the experiments, which also show that the curve is rather flat around the minimum (despite the log scale), suggesting that the exact choice of nodes per partition is not extremely critical as long as it is within a reasonable neighborhood of the benchmark. The performance gain at the minimum increases as the number of nodes gets larger, and reaches one order of magnitude for 500 nodes. As discussed before (Fig. 6), these gains increase for topologies of higher density.

Fig. 7(d) also shows the average gap between the coverage achieved with the optimal solution and the coverage obtained by solving the partitioned problem. As expected, this value increases monotonically with the number of partitions, since a larger portion of overlap is neglected as the number of partitions increases. These curves provide data to evaluate a trade-off between computation time and optimality.

Application of K-Way Partitioning: An interesting future work direction is the application of k-way partitioning, instead of recursive bipartitioning to decompose the optimization problem. In principle, this method could allow us to further reduce the area of the overlapped regions, with a better decoupling of the optimization problem defined on the different partitions.

We have made a preliminary evaluation of this possibility using a demonstrative implementation of the hMetis algorithm [24], which is only able to manage small graphs. We ran some tests on the DACO and DMCO problems for topologies with 60 nodes and around 125 points (the resulting partitioning problem was very close to the maximum size the tool could manage). The results were encouraging. For the DACO problem, we executed the algorithm for 20 topologies and in all cases it identified the global optimum (whereas the algorithm with the recursive scheme scored an average distance from the optimum of 0.6%). For DMCO defined on the same set of 20 topologies, the k-way partitioning reached an average distance from the optimum of 4.6%. Even in this case, there was a remarkable improvement over the recursive algorithm, which scored an average distance of 8.5%.

VII. Conclusion

The demand for energy efficiency in wireless sensor networks requires a careful deployment and schedule of the node operations. Typical integer linear program formulations suffer from high computational complexity and do not scale well with the size of the network. It is, therefore, useful to have heuristics that can be applied broadly, and whose quality can be measured. In this paper, we have presented a methodology to decompose a large problem into individual subproblems, and shown how to derive bounds that provide a measure of the degree of optimality that was achieved. Our method is generic and can be applied to a variety of sensor network models. The experimental results show that our procedure is very efficient and scalable, and is able to find solutions that are very close to optimal in many cases.

Our current work is focused on further extending the models and investigating alternative more accurate formulations for the bounds. Our objective is to eventually integrate the problem of coverage with the communication architecture and with the protocol layer, using realistic sensor models [7], [8]. An interesting direction is to integrate the optimal schedule of wake-up time for the nodes with power-aware routing policies in multihop networks [25]. The dynamics of the sensed phenomena could also be accounted for to adapt the solution to the nature of the monitored area, and to adjust the protocol parameters. Our current work is also concentrating on developing efficient online heuristics with provable convergence properties. Likewise, because of partitioning, the offline method lends itself to a parallelized implementation, which could dramatically speed up the search for the optimal solution.

ACKNOWLEDGMENT

The authors would like to thank P. Toldo for his contributions in a preliminary version of this work and the reviewers for their valuable feedback and comments.

REFERENCES

[1] G. Scheible, D. Dzung, J. Endresen, and J.-E. Frey, "Unplugged but connected-design and implementation of a truly wireless real-time sensor/actuator interface," *IEEE Ind. Electron. Mag*, vol. 1, no. 2, pp. 25–34, 2007.

[2] L. Hou and N. W. Bergmann, "System requirements for industrial wireless sensor networks," in *Proc. IEEE Conf. Emerging Technol. Factory Autom. (ETFA2010)*, Bilbao, Spain, Sep. 13–16, 2010, pp. 1–8.

[3] A. Willig, "Recent and emerging topics in wireless industrial communications: A selection," *IEEE Trans. Ind. Informat.*, vol. 4, no. 2, pp. 102–124, May 2008.

[4] C. M. Fiduccia and R. M. Mattheyses, "A linear time heuristic for improving network partitions," in *Proc. 19th Design Autom. Conf.*, Jun. 14–16, 1982, pp. 175–181.

[5] L. Palopoli, R. Passerone, A. Murphy, G. Picco, and A. Giusti, "Solving the wake-up scattering problem optimally," in *Proc. 6th Eur. Conf. Wireless Sensor Networks*, Cork, Ireland, Feb. 2009, pp. 166–182.

[6] A. Giusti, A. Murphy, and G. Picco, "Decentralized scattering of wake-up times in wireless sensor networks," in *Proc. 4th Eur. Conf. Wireless Sensor Networks (EWSN)*, Jan. 2007, vol. 4373, LNCS, pp. 245–260.

[7] A. Bonivento, C. Fischione, L. Necchi, F. Pianegiani, and A. L. Sangiovanni-Vincentelli, "System level design for clustered wireless sensor networks," *IEEE Trans. Ind. Informat.*, vol. 3, no. 3, pp. 202–214, Aug. 2007.

[8] Z. Hanzalek and P. Jurcik, "Energy efficient scheduling for cluster-tree wireless sensor networks with time-bounded data flows: Application to IEEE 802.15.4/ZigBee," *IEEE Trans. Ind. Informat.*, vol. 6, no. 3, pp. 438–450, Aug. 2010.

[9] S. Slijepcevic and M. Potkonjak, "Power efficient organization of wireless sensor networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2001, pp. 472–476.

[10] M. Cardei, M. Thai, Y. Li, and W. Wu, "Energy-efficient target coverage in wireless sensor networks," in *Proc. INFOCOM*, 2005, pp. 1976–1984.

[11] A. Alfieri, A. Bianco, P. Brandimarte, and C. F. Chiasserini, "Maximizing system lifetime in wireless sensor networks," *Eur. J. Oper. Res.*, vol. 127, no. 1, pp. 390–402, Aug. 2007.

[12] H. Liu, X. Jia, P. Wan, C. Yi, S. Makki, and N. Pissinou, "Maximizing lifetime of sensor surveillance systems," *IEEE/ACM Trans. Networking*, vol. 15, no. 2, pp. 334–345, 2007.

[13] S. Guo, T. He, M. Mokbel, J Stankovic, and T. Abdelzaher, "On Accurate and efficient statistical counting in sensor-based surveillance systems," in *Proc. 5th IEEE Int. Conf. Mobile Ad Hoc and Sensor Syst.*, Sep. 29–Oct. 2 2008, pp. 24–35.

[14] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. Srivastava, "Coverage problems in wireless ad-hoc sensor networks," in *Proc. 20th IEEE INFOCOM*, Anchorage, AK, Apr. 2001, pp. 1380–1387.

[15] K. Chakrabarty, S. S. Iyengar, H. Qi, and E. Cho, "Grid coverage for surveillance and target location in distributed sensor networks," *IEEE Trans. Comput.*, vol. 51, no. 12, pp. 1448–1453, Dec. 2002.

[16] N. Bulusu, D. Estrin, and J. Heidemann, "Adaptive beacon placement," in *Proc. 21st Int. Conf. Distrib. Comput. Syst.(ICDCS)*, Phoenix, AZ, Apr. 2001, pp. 489–498.

[17] A. Howard, M. Mataric, and G. Sukhatme, "An incremental self-deployment algorithm for mobile sensor networks," *Auton. Robots*, vol. 13, no. 2, pp. 113–126, 2002.

[18] A. Howard, M. Mataric, and G. Sukhatme, "Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem," in *Proc. 7th Int. Symp. Distrib. Autonomous Robotic Syst. (DARS)*, June 2002.

[19] Y. Zou and K. Chakrabarty, "Sensor deployment and target localization in distributed sensor networks," *ACM Trans. Embedded Comput. Syst.*, vol. 3, no. 1, pp. 61–91, 2004.

[20] D. Tian and N. Georganas, "A coverage-preserving node scheduling scheme for large wireless sensor networks," in *Proc. 1st ACM Int. Workshop Wireless Sensor Networks Appl.*, Atlanta, GA, 2002, pp. 32–41.

[21] C. Huang and Y. Tseng, "The coverage problem in a wireless sensor network," in *Proc. 2nd ACM Int. Conf. Wireless Sensor Networks and Applications (WSNA03)*, Sep. 2003, pp. 115–121.

[22] R. S. Tsay and E. Kuh, "A unified approach to partitioning and placement," *IEEE Trans. Circuits Syst.*, vol. 38, no. 5, pp. 521–533, May 1991.

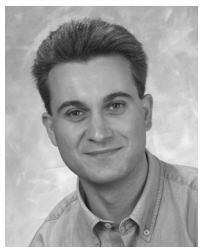
[23] A. E. Caldwell, A. B. Kahng, and I. L. Markov, "Iterative partitioning with varying node weights," *VLSI Design*, vol. 11, no. 3, pp. 249–258, 2000.

[24] G. Karypis and V. Kumar, "Multilevel k-way hypergraph partitioning," in *Proc. 36th Annu. ACM/IEEE Design Autom. Conf.*, New York, 1999, pp. 343–348 [Online]. Available: http://doi.acm.org/10.1145/309847.309954

[25] A. Rogers, E. David, and N. R. Jennings, "Self-organized routing for wireless micro-sensor networks," *IEEE Trans. Syst., Man, Cybern.—Part A*, vol. 35, no. 3, pp. 349–359, May 2005.

Luigi Palopoli (M'03) graduated with a Degree in computer engineering from the University of Pisa, Pisa, Italy, in 1992, and received the Ph.D. degree in computer engineering from Scuola Superiore Sant. Anna, Pisa, in 2002.

He is an Associate Professor of Computer Engineering at the University of Trento. His main research activities are in embedded system design with a particular focus on resource-aware control design and adaptive mechanisms for QoS management. He has coauthored more than 50 scientific papers in this area and he served in the program committees of several conferences, including the Real-Time System Symposium (RTSS), Euromicro ECRTS (ECRTS), and Hybrid Systems Computation and Control (HSCC). He is responsible for the Degree in Embedded Systems at the University of Trento and he has played an important role in many research projects (both national and European).



Roberto Passerone (M'97) received the Laurea degree (*summa cum laude*) in electrical engineering from the Politecnico di Torino, Torino, Italy, in 1994, and the M.S. and Ph.D. degrees in electrical engineering and computer sciences from the University of California, Berkeley, in 1997 and 2004, respectively.

From 1998 to 2005, he was with Cadence Design Systems, Berkeley. Since 2006, he has been with the Dipartimento di Ingegneria e Scienza dell'Informazione, University of Trento, Italy. He is the leader at the University of Trento of the COMBEST European Project, and of the Artist Design Network of Excellence. His research interests include system level design, communication design and formal methods.

Dr. Passerone has been involved as Chair or Co-Chair in several events related to embedded systems.

Tizar Rizano received the B.S. degree (*cum laude*) in computer science from Bandung Institute of Technology, Bandung, Indonesia, in 2004 and double M.S. degrees from the University of Trento, Trento, Italy and RWTH, Aachen, Germany, in 2009. He is currently working towards the Ph.D. degree at the University of Trento.

His main research interests are real-time scheduling and model-based development methodologies for embedded control software.