Area k-Coverage Optimization Protocol for Heterogeneous Dense Wireless Sensor Networks

Hervé Gokou Diédié
Université Niangui Abrogoua
Abidjan 02 BP 801, Côte d'Ivoire

Boko Aka
Université Niangui Abrogoua
Abidjan 02 BP 801, Côte d'Ivoire

Michel Babri
Institut National Polytechnique Félix Houphouët-Boigny
Yamoussoukro BP 1093, Côte d'Ivoire

***Abstract*—Detecting redundant nodes and scheduling their activity is mandatory to prolong the lifetime of a densely-deployed wireless sensor network. Provided that the redundancy check and the scheduling phases both help to preserve the coverage ratio and guarantee energy efficiency. However, most of the solutions usually proposed in the literature, tend to allocate a large number of unnecessary neighbor (re)discovery time slots in the duty-cycle of the active nodes. Such a shortcoming is detrimental to battery power conservation. In this paper, we propose a crossing points-based heuristic to fast detect redundant nodes even in heterogeneous networks; then, an integer linear program and a local exclusion based strategy to respectively, formulate and solve the sensing unit scheduling problem. Simulations show that the resulting localized asynchronous protocol outperforms some state-of-the-art solutions with respect to coverage preservation and network lifetime enhancement.**

***Keywords*—Coverage; optimization; wireless sensor network; scheduling; GRASP**

I. Introduction

Wireless sensor networks (WSNs) are composed of small electronic resource-limited devices that are capable to measure physical quantities in their environment. To collect information in remote or hostile areas, such networks require a random deployment of large number of nodes. However, this high node density leads to a redundantly monitored Field of Interest (FoI) that favors energy wastes and decreases network lifetime. Therefore, it is mandatory to work towards detecting and turning off all redundant nodes while preserving the area coverage ratio required by the underlying application.

Solutions for this kind of redundancy check are often categorized into deterministic and probabilistic ones [1]. Deterministic methods require that nodes have an exact knowledge of their positions whereas probabilistic ones try to relax this constraint.

As for putting redundant nodes to sleep state, it requires scheduling their duty-cycle. Techniques that are often used for that purpose can also be categorized into deterministic and probabilistic ones. Deterministic solutions are based on self-inactivation or sequential activation techniques. Whereas, probabilistic ones use more or less complex probability laws to calculate states transitions durations.

The above-mentioned coverage optimization processes lead to two well-known NP-hard problems, namely, the *minimum set cover problem* [2], [3] and the *sleep scheduling problem* [4]. Most of the solutions usually proposed for the first process have relatively good accuracy and precision ratios. While solutions for the second process tend to allocate a large number of unnecessary neighbor (re)discovery time slots in the duty-cycle of the active nodes. Such a shortcoming causes message overhead.

In this paper, we use a crossing points-based technique to detect redundant nodes in heterogeneous networks. Then, we formulate the scheduling problem as a special case of the general Maximum Set Packing problem using an integer linear program.

We propose a local mutual exclusion based scheduling scheme from a metaheuristic referred to as GRASP (Greedy Randomized Adaptative Search Procedure) that helps to reduce active nodes' *neighbor discovery frequency* and balance their energy depletion. The resulting asynchronous localized protocol increases network lifetime and preserve the coverage ratio. A second contribution of this paper is a sleep scheduling strategy that takes into account energy wastes due to state transition.

The rest of the paper is organized as follows. In Section II, we review some major related solutions recently proposed in the literature. Then, we detail our contribution in Section III. We evaluate its performance by analysis and simulation in Section IV. The results we obtained are discussed in Section V. Finally, we conclude the paper in Section VI.

II. Related Work

Coverage optimization in WSNs consists of two steps, namely, redundancy detection and sensing unit sleep scheduling. In this paper, we focus on area k-coverage, i.e. we assume that the underlying application requires that every points in the Field of Interest (FoI) be covered by at least k sensors. Where $k \geq 1$.

A. Redundancy Check

Area coverage redundancy check is generally reduced to the minimum set cover problem which was proven to be NP-hard [2], [3]. Existing solutions can be categorized into deterministic and probabilistic ones [1]. The former are often geometric and use techniques, such as *virtual grid* [5], *sponsored sector* [6], *perimeter-coverage* [7], *intersection points* [8], *voronoï tessellation* [9]. Unfortunately, they require that nodes have an exact knowledge of their location. While probabilistic methods try to relax this constraint.

Liu *et al.* [10] proposed the use of a virtual grid. To this end, each sensor node divides its coverage into virtual square

grids. If all these grids are covered by its neighbors a node is said to be redundant. Virtual grid technique has a low time complexity but are often space consuming. Chenait *et al.* [11] suggest the use of the sponsored sector technique. Each node has to create three sectors with a $\frac{2\pi}{3}$ central angle and check if each of them is covered by at least k neighbors. However, they applied this strategy only on a homogeneous network. Chen *et al.* [12] chose the perimeter-coverage scheme to evaluate the k-coverage of sensors. These nodes have to verify that each arc it shares with its neighbors is totally covered. Gupta *et al.* [13] used a similar approach for heterogeneous 3D networks. Unfortunately, when using this strategy, each node needs to have at least three neighbors. Jabeur *et al.*[14] proposed a dynamic strategy referred to as *bully approach* where sensor-nodes must compete to offer their services and get rewarded. In the relocation scheme, nodes with low *actual redundancy* force sensors with high *actual redundancy* to be the redundant sensors. This strategy requires that nodes be able to control their mobility. Xing *et al.* [8] proposed the intersection points method. A node is redundant if all the crossings inside its sensing disk are covered. However, this strategy was applied using an algorithm with a $\mathcal{O}(n^3)$ time complexity. Moreover, Liu *et al.* [15] showed that the above strategy is based on a necessary but not sufficient criterion. Diédié *et al.* [16] gave an additional condition to this technique. It consists in building the Maximal Redundancy Zone of each node, namely the convex hull of all crossings. Therefore, a node is redundant if it is located inside this zone and the above mentioned condition is met. Chang *et al.* [17] use the *weighted voronoï diagram* method. Weight metric is based on nodes residual energy. In order to minimize coverage redundancy, each node has to build its *voronoï cell* and adjust its sensing range while avoiding *holes*.

There are also several probabilistic solutions in the literature. They are based on probabilistic sensing models [18], [19]. Each node must find a condition that helps the area under its surveillance be also covered by at least k neighbors with a probability greater than a predefined threshold. Gupta *et al.* [20] proposed to use a probabilistic scheme in a heterogeneous network. Yang *et al.* [21] showed a relationship between two points in FoI implying that if one of them is covered with a probability greater than a value denoted by εe^{kd} the other one is covered with a probability no less than ε. The problem is formulated as an integer linear program and solved with a greedy approximation solution. Tian *et al.* [22] proposed a solution based on a sensing model derived from the Neyman-Pearson theory [23]. Unfortunately, the strategy is not fully-distributed since it is initialized by a central node.

B. Sleep Scheduling

Sensing unit scheduling is the second phase of the coverage optimization process. Solutions to this problem can also be categorized into deterministic and probabilistic ones.

Deterministic solutions are based on *self-inactivation* or *sequential activation* techniques [1]. When using the first method, each sensor node has to discover its neighborhood and perform redundancy check then enters into Sleep state, if it is redundant. To help mitigate coverage hole probability, a redundant node has to start a random backoff timer. After the latter timer expiration, it broadcasts a SLEEP message then enters into Sleep state if it has not already received such a message. Among the recent authors that use this technique are Idrees *et al.* [24], Jamali and Hatami [25], Zhang *et al.* [26], Gupta *et al.* [13] and Shi *et al.* [27].

However, when using sequential activation approach, nodes are mostly-off, set a timer and become active with a certain probability. If redundant, they broadcast ACTIVE message before entering into Sleep state. If a node receives a ACTIVE message it adjusts its timer and perform redundancy check. Zhang and Hou [28] are among the first authors who used that technique. Recently, He *et al.* [29] or More and Wagh [30] also applied such a strategy.

Probabilistic scheduling solutions are based on states transitions of which durations are chosen using more or less complex Probability laws. The state transition is similar to the one used by *sequential activation* strategy. Dioungue and Thiare [31] used the Weibull distribution to select all the *sentinels* i.e. nodes that should wake up and remain active when all their neighbors are in sleep state. Shen *et al*. [32] proposed to define sleep state's duration using exponential law. However, the process used to calculate the required node average density is costly. Farinelli *et al*. [33] used an agent-based Learning Automata strategy to help sensor-nodes to coordinate their sense/sleep schedules. Authors proposed a linear program to find a schedule that maximizes the total utility (i.e. social welfare) of agents.

III. Proposed solution

A. Motivation and Objectives

This work is aimed at energy efficiency and coverage ratio preservation. To this end, redundancy check process must be precise and accurate while being executed regularly from fresh information. Strategy often used consists in merging redundancy check and neighbor discovery into a single process. However, this approach is costly when applied with *self-inactivation* or *sequential activation* sleep scheduling techniques, as discussed in the previous section. Indeed, nodes have to check their redundancy each time they receive respectively a SLEEP and a ACTIVE message; hence, a large number of time slots is allocated for unnecessary neighbor (re)discovery processes in the duty-cycle of active nodes. Moreover, many redundancy checks are based on old information. Such shortcomings increase message overhead and the risk of having coverage holes.

Our goal is to minimize unnecessary redundancy check and neighbor discovery periods. We also aim at providing a strategy to define a Sleep schedule that helps to balance the amount of energy expended during states transitions.

B. Assumptions

We make the following assumptions:

- For each sensor u, we have $r_c(u) = 2 \times r_s(u)$ where $r_c(u)$ et $r_s(u)$ are respectively its communication and sensing ranges.

- Nodes have knowledge of their positions using localization schemes similar to the ones discussed by Holger and Willig [34] or Mao and Fidan [35].

- Network is heterogeneous i.e. nodes have different ranges; since they have different residual energy and are able to adjust their communication ranges.

- This process takes place in the plane.

C. Description

We detail in this section our protocol referred to as CGSCP (Coverage Greedy Scheduling Coordination Protocol). It consists of two phases: redundancy check and sensing unit sleep scheduling.

D. Redundancy Check

We give some important definitions for a better understanding of our strategy.

Definition 1 (Redundancy)**.** *A node u is redundant with respect to a subset of neighbors denoted by N_i if the area denoted by $cov(u)$ that u covers is identical with or included inside the area denoted by $cov(v)$ obtained from the union of all areas covered by each member v of N_i. Formally, u is redundant with respect to N_i iff $cov(u) \subseteq \bigcup_{v \in N_i : N_i \subseteq N(u)} cov(v)$. Where $N(u)$ denotes the set of node u's neighbors.*

Definition 2 (m-redundancy)**.** *A node u is m-redundant if it is redundant with respect to at least m subsets of neighbors.*

Definition 3 (Maximum Redundancy Zone)**.** *The Maximum Redundancy Zone (MRZ) is the region delimited by the convex hull deriving from the cloud of the intersection points between a subset of neighbors, as depicted by Fig.1. Points located on this hull will be referred to as* ***Border Points****; whereas the others will be referred to as* ***Interior Points****.*

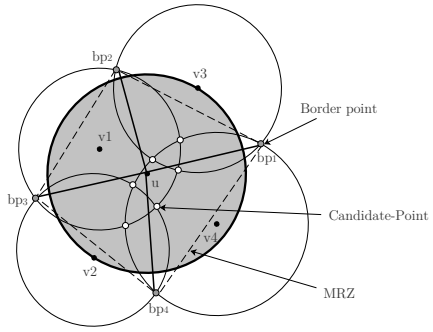


Fig. 1. The Maximum Redundancy Zone (MRZ) of node u deriving from the intersection points between its neighbors v_1, v_2, v_3 and v_4.

Area k-coverage problem has been proven to be NP-hard [36]. Therefore,we have to design an approximated solution.

Each node must apply the following intersection points-based heuristic in order to check its redundancy.

- Step 1: Discover the vicinity and select neighbors.

- Step 2: If no neighbor found then not redundant, stop. Instead, if at least two neighbors found go to Step 4.

- Step 3: If located inside this neighbor then redundant, otherwise not redundant, stop.

- Step 4: Derive the MRZ from neighbors' intersection points.

- Step 5: If not located inside the MRZ then not redundant, stop.

- Step 6: If at least one border point is covered then not redundant, stop.

- Step 7: If covered by one neighbor and the latter is adjacent to all the other neighbors then redundant, stop.

In the example shown in Fig. 1 node u is redundant with respect to neighbors v_1, v_2, v_3 and v_4 since u is located inside the MRZ, it does not cover any border point and it is covered by neighbors v_1 and v_4. Each of them are adjacent to the other neighbors.

It is noteworthy to mention that Steps 5 and 6 can be combined. To this end, node has to derive its RMRZ (Relative MRZ) from the set of points composed of its position, the intersection points with its neighbors and their mutual crossing points. Therefore, node is outside the MRZ if its position or at least one of the intersection points with its neighbors is on the RMRZ.

Methods we use to determine intersection points' coordinates and to construct nodes' RMRZ are detailed in our previous work [16].

We propose to estimate each node u's k-coverage by assessing its m-redundancy. Therefore, we have to search its vicinity denoted by $N(u)$, for $m (m \geq k)$ subsets of neighbors with respect to which node u is redundant. This problem can be formulated using the following program:

$$\max m \tag{1}$$

$$st: \bigcup_{i=1}^{m} N_i \tag{2}$$

$$(d(u,p) \leq r_s(u)) \Rightarrow (\exists v \in N_i, d(v,p) \leq r_s(v)) \tag{3}$$

$$, \forall p \in A, \forall i = 1, ..., m$$

$$(|N_i| \leq |N_j|) \Rightarrow (|N_i \cap N_j| < |N_i|), \tag{4}$$

$$i, j \in [1, m], \ i \neq j$$

Our goal is (1) i.e. finding the number of subsets of neighbors denoted by m that satisfies the constraints expressed by (2) - (4).

Equation 2 requires that only node u's neighbors are concerned. Whereas (3) helps ensuring that node u is redundant with respect to each of these subsets; i.e. any point p in the area of interest A covered by u $(d(u,p) \leq r_s(u))$ must also be covered by at least a member v of subset N_i. Equation 4 requires more detailed explanations. Indeed, it means that when a subset of nodes is involved in a redundancy case, its members must not be inserted into another subset. For instance, if node u is redundant with respect to subsets $N_1 = \{1, 3, 7\}, N_2 = \{1, 5, 7, 3\}$ and $N_3 = \{1, 7, 4\}$, N_2 will be ignored since it includes N_1. In other words, u is not redundant with respect to N_2 because it is already redundant with respect to N_1. Instead, N_3 can be included in a feasible solution since it has only two elements in common with N_1.

We must relax (4) since we are solving the m-redundancy problem in order to evaluate nodes'k-coverage. Hence, we

replace (4) by (5). The latter requires that the subsets of neighbors be disjoint.

$$N_i \cap N_j = \emptyset\,, i, j \in [1, m]\,,\ i \neq j \tag{5}$$

Therefore, area k-coverage problem becomes similar to a well-known NP-hard problem referred to as *General Maximum Set Packing* [2], [37]. We formulate it with an integer linear program.

$$x_{ij} = \begin{cases} 1 & \text{if neighbor } i \text{ is inserted into subset } j, \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

$$y_i = \begin{cases} 1 & \text{if redundant with respect to subset } i \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

Note that for $n\,(n \geq 1)$ neighbors, there are at most n disjoint subsets of neighbors.

$$\max \sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij} + \sum_{j=1}^{n} y_j \tag{8}$$

$$st: \sum_{i=1}^{n} x_{ij} \ \leq\ 1,\ \forall j = 1, ..., n \tag{9}$$

$$\sum_{j=1}^{n} y_j \ \geq\ k \tag{10}$$

$$x_{ij} \in \{0, 1\}\ \ \forall i = 1, ..., n, \forall j = 1, ..., n \tag{11}$$

$$y_j \in \{0, 1\}\ \ \forall j = 1, ..., n \tag{12}$$

Equation 8 is the objective-function. Equation 9 assures that a neighbor is inserted into only one subset. Equation 10 requires that the number of subsets with respect to which the node is redundant, be greater than or equal to k.

To solve this problem, we use a scheme based on a metaheuristic referred to as **GRASP (Greedy Randomized Adaptative Search Procedure)**, proposed by Feo and Resende [38], [39]. The latter scheme is formally detailed in Algorithms 1 - 3.

Algorithm 1 Evaluation of node u's k-coverage

Input: $k, nitrmax,\ N(u), p$
Output: s^*
$s^* \leftarrow \emptyset$
$c^* \leftarrow 0$
$nitr \leftarrow 0$
$RCL \leftarrow \emptyset$ ▷ Restricted Candidate List
while $(k > |s^*|) \wedge (N(u) \neq \emptyset) \wedge (nitr < nitrmax)$ **do**
  $s \leftarrow$ Generate random greedy solution $(N(u), p, RCL)$
  $s \leftarrow$ Local Search(s) ▷ Neighbouring solutions of s*
  $c \leftarrow f(s)$ ▷ Calculate cost see Eq.8
  if ($c^* < c$) **then**
    $s^* \leftarrow s$
    $c^* \leftarrow c$
  end if
  $nitr \leftarrow nitr + 1$
end while
return s^*

Algorithm 2 Greedy random solution generation by node u

Input: $N(u), p, LRC$
Output: s^*
$n \leftarrow 0$
repeat
  Choose randomly $e \subseteq N(u)\ : 1 \leq |e| \leq |N(u)|$
  if Redundancy-Check(e) **then** ▷ see Algorithm 1
    $LRC \leftarrow LRC \cup \{e\}$
    $N(u) \leftarrow N(u) \setminus e$
    $n \leftarrow n + 1$
  end if
until $(n = p) \vee (N(u) = \emptyset)$

Sort in ascending order the LRC ▷ According to length
$s^* \leftarrow \{e \in LRC \mid\ \nexists f \in LRC : |e| < |f|\}$
return s^*

Algorithm 3 Local search by node u

Input: $s^*, N(u), p$
Output: s^*
$s \leftarrow \{e \in s^* \mid\ \nexists f \in s^* : |e| > |f|\}$ ▷ get the smallest element
$n \leftarrow 0$
OK $\leftarrow$ **false**
while $(n < p) \wedge (N(u) \neq \emptyset) \wedge \neg$OK **do**
  Choose randomly $e \subseteq N(u)\ : 1 \leq |e| \leq |N(u)|$
  if Redundancy-Check(e) **then** ▷ see Algorithm 1
    $s^* \leftarrow s^* \setminus s$
    $s^* \leftarrow s^* \cup \{e\}$
    $N(u) \leftarrow N(u) \setminus s$
    OK $\leftarrow$ **true**
  end if
  $n \leftarrow n + 1$
end while
return s^*

E. Sleep Scheduling

The second phase of CGSCP consists in scheduling the sensing unit's activity. We formulate it as a *local mutual exclusion problem* [40], i.e. a localized version of the well-known mutual exclusion problem [41], [42], [43]. Indeed, we believe that a node enters into the *critical section* when it begins to check its redundancy. *Redundancy* is a relative notion that has a local scope as *Sleep scheduling*.

Our mutual exclusion scheme aims at minimizing the number of active nodes while preserving the coverage degree required by the underlying application. In other words, preventing coverage *holes* by avoiding two redundancy-dependent nodes to enter into Sleep state simultaneously.

We propose an heuristic-based solution since scheduling problem also was proven to be NP-hard [4].

Definition 4 (Node's state). *Any node u can only have the following states :*

- *Active(**ACT**) ,*
$state(u) = ACT \iff [(r_s(u) > 0) \wedge (\nexists v \in N(u) : state(v) = SLP)]$

- *Discovery (**DSC**) ,*
$state(u) = DSC \iff [(r_s(u) > 0) \wedge (\nexists v \in N(u) : state(v) = DSC \vee state(v) = FRZ)]$

- *Frozen (**FRZ**)* ,
$state(u) = FRZ \iff [(r_s(u) > 0) \wedge (\exists v \in N(u) : state(v) = SLP)]$

- *Sleep (**SLP**)*
$state(u) = SLP \iff [(r_s(u) = 0) \wedge (\breve{N}(u) \neq \emptyset)]$

Where $N(u)$ and $\breve{N}(u)$ respectively denote the set of node u's one-hop neighbors and the family of subsets involved in its k-coverage.

Our sensing unit scheduling scheme is actually based on the *k*-coverage evaluation process as described in the previous section. Indeed, once deployed, each node must choose its next neighbor discovery and *k*-coverage evaluation time randomly and uniformly in interval $[t_{min}; t_{max}]$, then start a back-off timer. When the latter expires, node enters into Discovery state and broadcasts a HELLO message. Each neighbor should reply with a WELCOME message containing its residual neighborhood discovery time denoted by Δt_{discov}. A node v in Discovery state returns to Active state when receiving a HELLO message from a neighbor u with a greater ID. Therefore, after updating its neighbor table, and checked that it is k-redundant, node u must calculate its sleep time duration denoted by t_{sleep} using (13). If the latter is greater than a threshold denoted by th_{sleep}, node u has to send a SLEEP message that includes its t_{sleep} to a subset of neighbors with respect to which it is k-redundant. This subset is the one with the lowest cardinality chosen among the m subsets discovered by the redundancy check process. th_{sleep} is defined so as to overcome delays and the amount of energy expended during states transitions.

In (13) Er , Ei, $\breve{N}$ and N respectively denote nodes' residual energy, their initial energy, the family of subsets with respect to which they are redundant and their neighbors set.

$$t_{sleep} = \widetilde{t} - \Big((\alpha \times \frac{Er}{Ei}) + (\beta \times (1 - \frac{|\breve{N}|}{|N|})) + \gamma\Big) \qquad (13)$$

α and β, γ are three weighting coefficients such as $\alpha+\beta = 1$ while γ is randomly chosen in interval $[0, 1-(\alpha+\beta)]$. $\widetilde{t}$ is the lowest Δt_{discov} provided by the neighbors with respect to which a node is redundant.

It is worth noting that $\widetilde{t}$ must take account of the average message response time denoted by t_{reply}.

When a node u in Active state receives a SLEEP message from a neighbor v, it resets its next neighbor discovery time and enters into Frozen state. Node u has to postpone its next discovery time by $(t_{sleep}(v) - t_{reply})$ units of time, if its residual discovery time denoted by $\Delta t_{discov}(u)$ is lower than neighbor v's sleeping time, namely, $t_{sleep}(v)$.

It is also noteworthy to mention that after receiving a SLEEP message a node in Frozen state must increase by 1 unit its *fixation counter* denoted by δf. The latter is helpful for nodes to count neighbors that are in Sleeping state.

At the end of its sleeping time, a node enters into Active state, chooses at random its next neighbor discovery time and broadcast a AWAKE message to its one-hop neighbors. After receiving such a message, nodes in Frozen state have to decrease their *fixation counter* by 1 unit and enter into Active state if their fixation counter's value reaches 0. The sensing unit scheduling process is formally described in Algorithms 4 and 5. States transitions are depicted in Fig. 2.

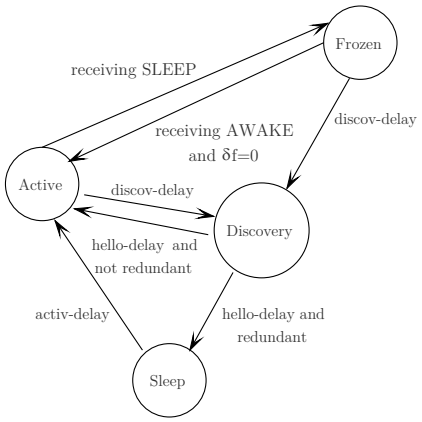


Fig. 2. State transition diagram of CGSCP.

Algorithm 4 Sensing unit scheduling by node u

Input: *Ethr*, t_{reply},α,β,γ,$\widehat{r}_s$, t_{min}, t_{max}, th_{sleep}

Er← Estimate residual energy
ACTIV-delay ← 0
while (*Er* > *Ethr*) **do** ▷ Residual energy is enough
  if (*ACTIV-delay* = 0)∨(Current-Time() = *ACTIV-delay*) **then**
    $r_s(u) \leftarrow \widehat{r}_s$
    $state(u) \leftarrow ACT$
    $t_{discov}(u) \leftarrow U(t_{min}; t_{max})$ ▷ Uniform Law
    DISCOV-delay ← Current-Time() + $t_{discov}(u)$
    $\delta f \leftarrow 0$
    Send AWAKE to $v, \forall v \in \breve{N}(u)$
  end if
  if (Current-Time() = *DISCOV-delay*) **then** ▷ Critical Section
    $state(u) \leftarrow DSC$
    Choose $\Delta t_{discov}(v) : \forall v, w \in N(u) \Rightarrow (\Delta t_{discov}(v) > \Delta t_{discov}(w))$
    $\Delta t_{discov}(u) \leftarrow U(t_{min}; t_{max}) + \Delta t_{discov}(v)$
    DISCOV-delay ← Current-Time() + $\Delta t_{discov}(u)$
    HELLO-delay ← Current-Time() + t_{reply}
    Broadcast HELLO $(id_u, \Delta t_{discov}(u), state(u))$
  end if
  if (Current-Time() = *HELLO-delay*) **then**
    $state(u) \leftarrow ACT$
    Check *k*-coverage($N(u)$) ▷ see Algorithm 1
    if $(\breve{N}(u) \neq \emptyset)$ **then**
      $t_{sleep}(u) \leftarrow$ Sleeptime (α,β,γ,Er, Ei) ▷ see Eq. 13
      if $(t_{sleep}(u) > th_{sleep})$ **then**
        $\widehat{N}(u) \leftarrow \min(\breve{N}(u))$
        Send SLEEP($t_{sleep}(u)$) to v, $\forall v \in \widehat{N}(u)$
        $r_s(u) \leftarrow 0$
        $state(u) \leftarrow SLP$
        ACTIV-delay ← Current-Time() + $t_{sleep}(u)$
      end if
    end if
  end if
  Handle Scheduling messages ▷ see Algorithm 5
  Er← Estimate residual energy
end while

Algorithm 5 Scheduling messages Handling by nœud u

1: **Receive** $message$ de v
2: **switch** $message$ **do**
3: **case** HELLO
4: **if** $(state(u) = ACT) \vee (state(u) = FRZ) \vee ((state(u) = DSC) \wedge (id_v > id_u))$ **then**
5: $\Delta t_{discov}(v) \leftarrow \Delta t_{discov}(v) - t_{reply}$ ▷ Considering latency
6: $N(u) \leftarrow$ Neighbor table Update $(id_v, \Delta t_{discov}(v), state(v))$
7: $\Delta t_{discov}(u) \leftarrow$ Current-Time() - *DISCOV-delay*
8: **Send** WELCOME $(id_u, \Delta t_{discov}(u), state(u))$ to v
9: **end if**
10: **if** $(state(u) = DSC) \wedge (id_v > id_u)$ **then**
11: $state(u) \leftarrow ACT$
12: **end if**
13: **case** WELCOME
14: **if** $(state(u) = DSC)$ **then**
15: $\Delta t_{discov}(v) \leftarrow \Delta t_{discov}(v) - t_{reply}$ ▷ Considering latency
16: $N(u) \leftarrow$ Neighbor table Update $(id_v, \Delta t_{discov}(v), statut(v))$
17: **end if**
18: **case** SLEEP
19: **if** $(state(u) \neq FRZ) \wedge (\Delta t_{discov}(u) < t_{sleep}(v))$ **then**
20: *DISCOV-delay* $\leftarrow$Current-Time() + $(t_{sleep}(v) - t_{reply})$
21: $state(u) \leftarrow FRZ$
22: **end if**
23: $\delta f \leftarrow \delta f + 1$
24: **case** AWAKE
25: **if** $(state(u) = FRZ)$ **then**
26: $\delta f \leftarrow \delta f - 1$
27: **if** $(\delta f = 0)$ **then**
28: $state(u) \leftarrow ACT$
29: **end if**
30: **end if**
31: **end switch**

IV. Performance Evaluation

To verify and validate our protocol we analyzed its time and messages number complexities. Then we formally proved its mutual exclusion property. We also conducted extensive simulations using OMNeT++ 4.6 simulator [44]. The results were compared to those obtained with some related protocols namely, CCP by Xing *et al*. [8] DiLCO by Idrees *et al*. [24], ERPC by Liu *et al.* [15], the solution by Gupta *et al.* [13] and VGSCA by Liu *et al.* [10] .

Theorem 1 (**Time complexity**). *On an asynchronous fair daemon, in the worst case and in the absence fault, the time complexity of CGSCP is* $\mathcal{O}(n)$.

Proof: In the worst case, topology induced by the network is a complete graph with n nodes and where each of them has $n-1$ neighbors. In the worst case, a node u will randomly choose the greatest neighbor discovery time denoted by $t_{discov}(u)$. Therefore, before the latter occurs, it may receive at most $n-1$ SLEEP messages from its neighbors. After receiving a SLEEP message from a redundant neighbor v, node u will enter into Frozen state for at most $t_{sleep}(v) - t_{reply}$ units of time if v has chosen a sleeping time denoted by $t_{sleep}(v)$ that postpones node u's neighbor discovery time (see Lines 18 - 23 in Algorithm 5). Moreover, each node has to define its next neighbor discovery time according to those of its neighbors (see Lines 15 - 18 in Algorithm 4). In other words, nodes' waiting time before entering into the critical section and their sleeping time duration are linearly dependent on the number of their neighbors. Hence the $\mathcal{O}(n)$ time complexity. ■

Theorem 2 (**Message complexity**). *On an asynchronous and fair daemon, in the worst case and in the absence of fault, message number complexity of CGSCP is* $\mathcal{O}(n)$ *where* n *denotes the number of neighbors.*

Proof: In the worst case, topology induced by the network is a complete graph. Since each of the n nodes has $n-1$ neighbors, CGSCP requires three messages namely, HELLO, SLEEP and AWAKE sent by a redundant node u to respectively, discover its neighborhood, announce its Sleep state and announce its Active state. Node u will receive $n-1$ WELCOME messages in response, hence $\mathcal{O}(n^2)$ messages in the worst case for n redundant nodes. ■

Lemma 1 (**Safety**). *Two redundancy-dependent nodes cannot discover their neighborhood, or enter into Sleep state simultaneously. More formally, let* $G = (V, E)$ *be the graph induced by network ; where* V *and* E *respectively denote the set of nodes and the set of links,* $\forall u, v \in V : \forall v \in N(u), ((state(u) = DSC) \vee (state(u) = SLP)) \Rightarrow ((state(v) = FRZ) \vee (state(v) = ACT))$.

Proof: Proving that nodes have an exclusive access to the critical section consists in showing that two adjacent nodes u and v ($v \in N(u) \wedge u \in N(v)$) cannot discover their neighborhood, let alone enter into Sleep state simultaneously. Indeed, if node u chooses a shorter next neighborhood discovery time denoted by $t_{discov}(u)$, it will certainly enter into Discovery state ($state(u) = DSC$) before its neighbors.

Therefore, when node v receives a HELLO message, it will not enter also into discovery state (see Lines 3 - 12 in Algorithm 5). However, we could have $t_{disc}(u) = t_{disc}(v)$ for two nodes u and v because neighbor discovery times are randomly chosen; but, since their respective IDs namely, id_u and id_v are different ($id_u \neq id_v$), it follows that $state(u) \neq state(v)$. Indeed, let us assume that $id_u > id_v$; after sending a HELLO message, node u will receive WELCOME messages in response and will evaluate its redundancy (see Lines 20 - 22 in Algorithm 4 and Lines 3 - 9 in Algorithm 5). In contrast, v will enter into Active state ($state(v) = ACT$ see Lines 10 - 12 in Algorithm 5). Therefore, node u would probably be the only node to enter into Sleep State ($state(u) = SLP$) (see Lines 23 - 31 in Algorithm 4). Consequently, only node u's redundancy-dependent neighbors ($\widehat{N}(u)$) will finally be in Frozen state ($state(v) = FRZ$) after receiving its SLEEP message (Lines 18 - 23 in Algorithm 5). Formally, $\forall v \in \widehat{N}(u), state(v) = FRZ$ and $\widehat{N}(u) \subseteq N(u)$ then $\forall v \in N(u), ((state(u) = DSC) \vee (state(u) = SLP)) \Rightarrow ((state(v) = FRZ) \vee (state(v) = ACT))$ ■

Lemma 2 (**Liveness**). *Each node will eventually discover its neighborhood or enter into Sleep state.*

Proof: Proving liveness consists in showing that any node can enter into another state after a finite length of time particularly, after been in Frozen state, even if a sleeping neighbor fails. Indeed, according to Lemma 1 after it had randomly chosen its next neighbor discovery time denoted by $t_{disc}(u)$ a node u will always enter into the critical section if $t_{disc}(u)$ is the shortest discovery time and its ID is the smallest after breaking tie. Therefore, two situations may occur:

- Node u is not redundant, it will update its neighborhood table and choose another neighbor discovery time denoted by $t_{discov}(u)$.

To this end, u chooses $t_{discov}(u) : \forall v \in N(u), t_{disc}(u) > \Delta t_{disc}(v)$ (see Line 15 Algorithm 4). Where $\Delta t_{disc}(v)$ is the

residual time before node v's next neighbor discovery. This information was piggybacked in the WELCOME message that neighbor v sent to node u. The latter cannot prevent anymore its neighbors to also enter into critical section;

- Node u is redundant, therefore it enters into Sleep state $(state(u) = SLP)$ and set its sleeping state time duration denoted by $t_{sleep}(u)$ according to the shortest neighbor discovery time and the threshold denoted by th_{sleep} . Formally, $t_{sleep}(u) : \forall v \in N(u), th_{sleep} < t_{sleep}(u) < \Delta t_{disc}(v)$. If $t_{sleep}(u)$ cannot meet this condition, node u will return immediately to Active state $(state(u) = ACT)$. Instead, if node u can enter into Sleep state, its redundancy-dependent neighbors will enter into Frozen state. (see Lines 18 - 23 in Algorithm 5) $\forall v \in \widehat{N}(u), state(v) = FRZ$. The latter will return to Active state $(state(v) = ACT)$ after receiving a AWAKE message sent by node u at the end of its sleeping time. In the worst case, nodes that are in Frozen state $(state(v) = FRZ)$ will return to Active state when their neighbor discovery time occurs. In other words, let $G = (V; E)$ be the graph induced by the network, where V and E denote respectively the set of nodes and the set of links. Formally, $\forall u \in V, (\Delta t_{disc}(u) = 0) \Rightarrow (state(u) = ACT)$. ■

Lemma 3 (Concurrency). *Two redundancy-independent nodes can discover their neighborhood or enter into Sleep state simultaneously.*

Proof: Proving concurrency consists in showing that two non-adjacent nodes u and v ($v \notin N(u) \wedge u \notin N(v)$) can enter into their critical section freely with no conflict. Therefore, two events may occur:

- Nodes u and v have at least one neighbor in common $(N(u) \cap N(v) \neq \emptyset)$. If their neighbor discovery times respectively denoted by $t_{disc}(u)$ and $t_{disc}(v)$ are such that $t_{disc}(u) > t_{disc}(v)$ and if we have $(state(u) = SLP) \Rightarrow (\exists w \in N(u) \cap N(v) : (state(w) = FRZ))$, node v will enter into its critical section freely $(state(v) = DSC \vee state(v) = SLP)$ without node w having to enter into another state. The same event occur if we have $((t_{disc}(u) = t_{disc}(v)) \wedge (state(u) = SLP) \wedge (state(v) = SLP) \Rightarrow (\exists w \in N(u) \cap N(v) : (state(w) = FRZ))$;

- Nodes u and v do not have any neighbor in common $(N(u) \cap N(v) = \emptyset)$ then it is obvious that no decision made by node u will affect node v and vice versa. Therefore, nodes u and v can enter into their critical section simultaneously. Let $G = (V; E)$ the graph induced by the network topology; where V and E respectively denote the set of nodes and the set of links. Formally, $\forall u, v \in V, (v \notin N(u) \wedge u \notin N(v)) \Rightarrow ((state(u) = state(v)) \vee (state(u) \neq state(v)))$. ■

Theorem 3. *CGSCP provides a local mutual exclusion for the sensing unit sleep scheduling problem.*

Proof: CGSCP allows a safe sensing unit scheduling and a weak fairness. Proof is given by Lemma 1 and 2. CGSCP allows a mutual exclusion; Lemma 3 showed that this process is fully localized. ■

Corollary 1. *In the absence of fault, CGSCP does not create any coverage hole.*

TABLE I. SIMULATION PARAMETERS

Parameter	Value
Deployment area	500 m X 500 m
Number of sensors	100 to 1000
Sensors' initial energy (Ei)	0,2 J
Self-discharge per second	0,1 μJ
Energy threshold ($Ethr$)	100 μJ
E_{elec}	50 nJ/bit
e_{fs}	10 nJ/bit/m^2
e_{amp}	0,0013 nJ/bit/m^4
d_0	87 m
Message length (l)	2000 bits
U_{sup}	2,7 V
I_{sens}	25 mA
t_{sens}	0,25 ms
Data length (b)	200 bits
$nitrmax$	100 to 200

TABLE II. ENERGY LOST DURING STATE TRANSITION

	Active	Sleep
Active	-	0.4mW
Sleep	0.4mW	-

TABLE III. STATE TRANSITION DELAYS

	Active	Sleep
Active	-	2μs
Sleep	2μs	-

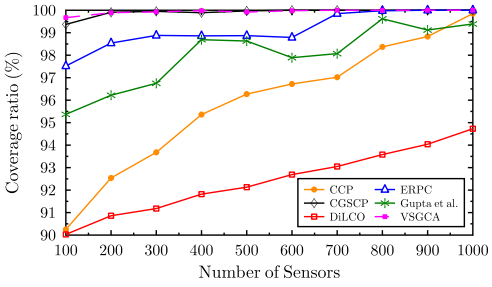
For simulation campaigns, we used the radio energy consumption model by Heinzelman *et al.* [45] and a sensing unit energy consumption model by Halgamuge *et al.* [46].

Tables I to III summarize parameters we used for the simulation campaigns we conducted with respect to three metrics, namely, k-coverage efficiency, energy efficiency and network lifetime. Each experiment was repeated 10 times as nodes population was increased.

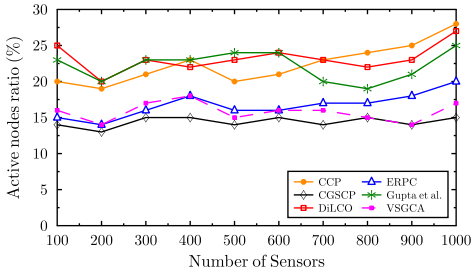
V. RESULTS AND DISCUSSION

A. k-Coverage Efficiency

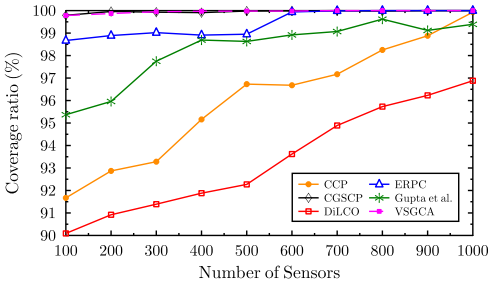
Fig. 3a shows that among the 6 evaluated protocols, only CGSCP and VSGCA help to preserve virtually 100% of the coverage ratio in the case of 1-coverage despite the increase in the number of sensor-nodes. However, as shown in Fig. 3b, CGSCP requires to keep active on average about 15% of the deployed nodes. This trend continues in the case of 4-coverage as depicted in Fig. 3c and Fig. 3d. The number of nodes kept active logically increases according to parameter k. However, CGSCP is the protocol that keeps the least number of active nodes. These performances are due to the detection process. Indeed, the strategy we used has a good accuracy and precision ratio even for heterogeneous networks [16]. DiLCO and the solution by Gupta *et al.* are the worst performing protocols since they are essentially based on the scheduling process with a less precise random redundancy detection process. Therefore, one can conclude that deterministic strategies provide the best k-coverage ratio.



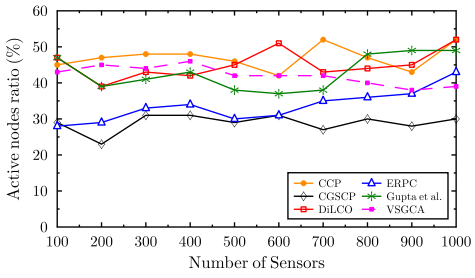
(a)



(b)



(c)



(d)

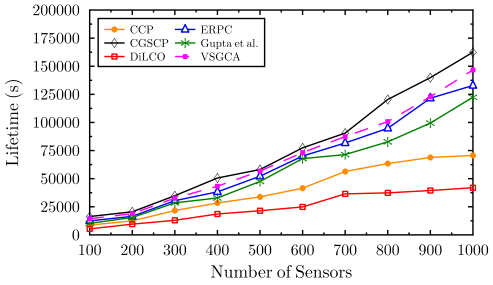
Fig. 3. k-coverage efficiency vs. Number of sensors. (a) coverage ratio for $k = 1$, (b) active nodes ratio for $k = 1$, (c) coverage ratio for $k = 4$, (d) active nodes ratio for $k = 4$.

B. Network Lifetime and Energy Efficiency

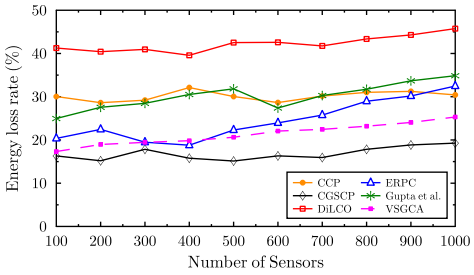
Fig. 4a shows that CGSCP is the protocol that best increases network lifetime i.e. that keeps the longest the coverage ratio above 98%. Fig. 4b shows that this performance requires to spend on average 17% of the initial energy in spite of the increasing of the number of sensor-nodes. This trend continues as parameter k increases. Logically, network lifetime decreases as parameter k grows; since the number of nodes to be kept active depends on parameter k.

CCP and DiLCO are the two protocols of which performances are the lowest. This situation is due to the poor performances of their redundancy detection process.

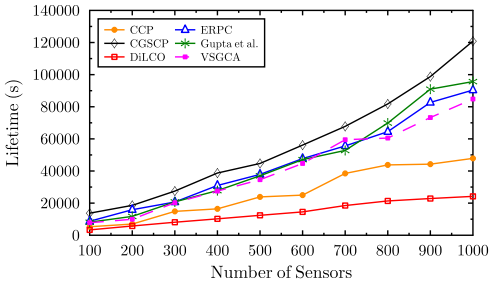
As for CGSCP, these results show the energy-efficiency of the local mutual exclusion strategy. Indeed, it prevents nodes that are in Frozen state to unnecessarily rediscover their neighborhood. Furthermore, this strategy allows load balancing since it guarantees to each node the opportunity of entering into Sleep state.



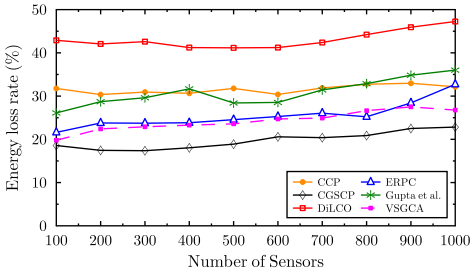
(a)



(b)



(c)



(d)

Fig. 4. Network lifetime & Energy Efficiency with a 98% coverage ratio. (a) Network lifetime for $k = 1$, (b) Energy Efficiency for $k = 1$, (c) Network lifetime for $k = 4$, (d) Energy Efficiency for $k = 4$.

VI. Conclusion

In this paper, we investigated the problem of area k-coverage in a heterogeneous wireless sensor network. We assumed that the underlying application requires that every points in the Field of Interest (FoI) be covered by at least $k(k \geq 1)$ sensor-nodes. We proposed an asynchronous and localized protocol referred to as CGSCP (Coverage Greedy Scheduling Coordination Protocol). The latter use a crossing points-based technique and a local mutual exclusion-based heuristic to, respectively, detect redundant nodes and schedule their duty-cycles. Several simulations have been carried out to evaluate the proposed protocol. Results show that our scheme provides a lower coverage ratio and is more energy-efficient than some recently proposed solutions.

In a future work, we plan to extend our protocol by adding a process that helps to explicitly tolerate both node and link failures. Finally, it would be also interesting to implement the CGSCP protocol in 3D more realistic environment.

References

[1] B. Wang, "Sensor activity scheduling," in *Computer Communications and Networks*, pp. 121–153, Springer London, 2010.

[2] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations*, pp. 85–103, Springer Nature, 1972.

[3] M. R. G. D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*. W. H. Freeman, 1979.

[4] K.-S. Hung and K.-S. Lui, "Perimeter coverage scheduling in wireless sensor networks using sensors with a single continuous cover range," *EURASIP Journal on Wireless Communications and Networking*, vol. 2010, no. 1, p. 926075, 2010.

[5] H. Bai, X. Chen, and X. Guan, "Preserving coverage for wireless sensor networks of nodes with various sensing ranges," in *2006 IEEE International Conference on Networking, Sensing and Control*, pp. 54–59, 2006.

[6] N. G. D. Tian, "A coverage-preserving node scheduling scheme for large wireless sensor networks," *Proc. ACM (WSNA)*, pp. 32–41, 2002.

[7] C.-F. Huang and Y.-C. Tseng, "The coverage problem in a wireless sensor network," *Mobile Networks and Applications*, vol. 10, no. 4, pp. 519–528, 2005.

[8] G. Xing, X. Wang, Y. Zhang, C. Lu, R. Pless, and C. Gill, "Integrated coverage and connectivity configuration for energy conservation in sensor networks," *ACM Transactions on Sensor Networks*, vol. 1, pp. 36–72, aug 2005.

[9] B. Carbunar, A. Grama, J. Vitek, and O. Carbunar, "Coverage preserving redundancy elimination in sensor networks," in *2004 First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004.*, IEEE, 2004.

[10] Y. Liu, L. Suo, D. Sun, and A. Wang, "A virtual square grid-based coverage algorithm of redundant node for wireless sensor network," *Journal of Network and Computer Applications*, vol. 36, pp. 811–817, mar 2013.

[11] M. Chenait, B. Zebbane, L. Boufennara, L. Haouaya, C. Benzaid, and N. Badache, "Lsea: Light weight sector eligibility algorithm for k-coverage in wireless sensor networks," in *2015 9th International Conference on Next Generation Mobile Applications, Services and Technologies*, pp. 294–299, Sept 2015.

[12] Y. Chen, C. Li, J. Yu, H. Zhu, and Y. Sun, "k-perimeter coverage evaluation and deployment in wireless sensor networks," in *Wireless Algorithms, Systems, and Applications*, pp. 50–59, Springer International Publishing, 2015.

[13] H. P. Gupta, S. V. Rao, and T. Venkatesh, "Sleep scheduling protocol for k-coverage of three-dimensional heterogeneous wsns," *IEEE Transactions on Vehicular Technology*, vol. 65, pp. 8423–8431, oct 2016.

[14] N. Jabeur, A. N. S. Moh, and M. M. Barkia, "A bully approach for competitive redundancy in heterogeneous wireless sensor network," *Procedia Computer Science*, vol. 83, pp. 628–635, 2016.

[15] Y. Liu, J. Pu, S. Zhang, Y. Liu, and Z. Xiong, "A localized coverage preserving protocol for wireless sensor networks," *Sensors*, vol. 9, pp. 281–302, jan 2009.

[16] H. G. Diédié, M. Babri, and S. Oumtanaga, "Redundancy detection protocol for area coverage control in heterogeneous wireless sensor networks," *IJCSI International Journal of Computer Science Issues*, vol. 12, pp. 100–108, March 2015.

[17] C.-T. Chang, C.-Y. Chang, S. Zhao, J.-C. Chen, and T.-L. Wang, "SRA: A sensing radius adaptation mechanism for maximizing network lifetime in WSNs," *IEEE Transactions on Vehicular Technology*, vol. 65, pp. 9817–9833, dec 2016.

[18] M. Hefeeda and H. Ahmadi, "Energy-efficient protocol for deterministic and probabilistic coverage in sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, pp. 579–593, may 2010.

[19] Y. Wang, S. Wu, Z. Chen, X. Gao, and G. Chen, "Coverage problem with uncertain properties in wireless sensor networks: A survey," *Computer Networks*, vol. 123, pp. 200–232, aug 2017.

[20] H. P. Gupta, S. V. Rao, and T. Venkatesh, "Analysis of the redundancy in coverage of a heterogeneous wireless sensor network," in *2013 IEEE International Conference on Communications (ICC)*, IEEE, jun 2013.

[21] Q. Yang, S. He, J. Li, J. Chen, and Y. Sun, "Energy-efficient probabilistic area coverage in wireless sensor networks," *IEEE Transactions on Vehicular Technology*, vol. 64, pp. 367–377, jan 2015.

[22] Y. Tian, X. Yang, Y. Ou, and G. You, "An area coverage control protocol based on probabilistic coverage for wireless sensor networks," in *2016 Seventh International Conference on Intelligent Control and Information Processing (ICICIP)*, IEEE, dec 2016.

[23] V. K. Rohatgi and A. K. M. E. Saleh, "Neyman-pearson theory of testing of hypotheses," in *An Introduction to Probability and Statistics*, pp. 429–462, John Wiley & Sons, Inc, sep 2015.

[24] A. K. Idrees, K. Deschinkel, M. Salomon, and R. Couturier, "Distributed lifetime coverage optimization protocol in wireless sensor networks," *The Journal of Supercomputing*, vol. 71, pp. 4578–4593, nov 2015.

[25] S. Jamali and M. Hatami, "Coverage aware scheduling in wireless sensor networks: An optimal placement approach," *Wireless Personal Communications*, vol. 85, pp. 1689–1699, jul 2015.

[26] B. Zhang, E. Tong, J. Hao, W. Niu, and G. Li, "Energy efficient sleep schedule with service coverage guarantee in wireless sensor networks," *Journal of Network and Systems Management*, vol. 24, pp. 834–858, jan 2016.

[27] B. Shi, W. Wei, Y. Wang, and W. Shu, "A novel energy efficient topology control scheme based on a coverage-preserving and sleep scheduling model for sensor networks," *Sensors*, vol. 16, p. 1702, oct 2016.

[28] H. Zhang and J. Hou, "Maintaining sensing coverage and connectivity in large sensor networks," in *Handbook on Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless, and Peer-to-Peer Networks*, Informa UK Limited, aug 2005.

[29] S. He, D.-H. Shin, J. Zhang, J. Chen, and Y. Sun, "Full-view area coverage in camera sensor networks: Dimension reduction and near-optimal solutions," *IEEE Transactions on Vehicular Technology*, vol. 65, pp. 7448–7461, sep 2016.

[30] A. More and S. Wagh, "Energy efficient coverage using optimized node scheduling in wireless sensor networks," in *2015 1st International Conference on Next Generation Computing Technologies (NGCT)*, IEEE, sep 2015.

[31] D. Diongue and O. Thiare, "ALARM: An energy aware sleep scheduling algorithm for lifetime maximization in wireless sensor networks," in *2013 IEEE Symposium on Wireless Technology & Applications (ISWTA)*, Institute of Electrical and Electronics Engineers (IEEE), sep 2013.

[32] F. Shen, C. Liu, B. Qi, Y. Ji, and D. Caban, "Building effective scheduling algorithms for sensor networks," in *2014 11th International Conference on Information Technology: New Generations*, Institute of Electrical and Electronics Engineers (IEEE), apr 2014.

[33] A. Farinelli, A. Rogers, and N. R. Jennings, "Agent-based decentralised coordination for sensor networks using the max-sum algorithm," *Autonomous Agents and Multi-Agent Systems*, vol. 28, pp. 337–380, may 2013.

[34] K. Holger and A. Willig, "Localization and positioning," in *Protocols and Architectures for Wireless Sensor Networks*, pp. 231–249, Wiley-Blackwell, jan 2006.

[35] G. Mao and B. Fidan, "Introduction to wireless sensor network localization," in *Localization Algorithms and Strategies for Wireless Sensor Networks*, pp. 1–32, IGI Global, 2010.

[36] S. Yang, F. Dai, M. Cardei, J. Wu, and F. Patterson, "On connected multiple point coverage in wireless sensor networks," *International Journal of Wireless Information Networks*, vol. 13, pp. 289–301, may 2006.

[37] E. Hazan, S. Safra, and O. Schwartz, "On the complexity of approximating k-set packing," *computational complexity*, vol. 15, pp. 20–39, may 2006.

[38] T. A. Feo and M. G. Resende, "A probabilistic heuristic for a computationally difficult set covering problem," *Operations Research Letters*, vol. 8, pp. 67–71, apr 1989.

[39] M. G. Resende and C. C. Ribeiro, *Optimization by GRASP*. Springer Nature, 2016.

[40] H. Attiya, A. Kogan, and J. Welch, "Efficient and robust local mutual exclusion in mobile ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 9, pp. 361–375, mar 2010.

[41] E. W. Dijkstra, "Hierarchical ordering of sequential processes," *Acta Informatica*, vol. 1, no. 2, pp. 115–138, 1971.

[42] G. Peterson, "Myths about the mutual exclusion problem," *Information Processing Letters*, vol. 12, pp. 115–116, jun 1981.

[43] P. A. Buhr, D. Dice, and W. H. Hesselink, "Dekkers mutual exclusion algorithm made rw-safe," *Concurrency and Computation: Practice and Experience*, vol. 28, pp. 144–165, sep 2015.

[44] A. Varga, "https://omnetpp.org," 2016.

[45] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Transactions on Wireless Communications*, vol. 1, pp. 660–670, oct 2002.

[46] M. N. Halgamuge, M. Zukerman, K. Ramamohanarao, and H. L. Vu, "An estimation of sensor energy consumption," *Progress In Electromagnetics Research B*, vol. 12, pp. 259–295, 2009.