Sensor device scheduling-based cuckoo algorithm for enhancing lifetime of cluster-based wireless sensor networks

Mazin Kadhum Hameed

Department of Software,
University of Babylon,
Babylon, Iraq
Email: it.mazen.kadhum@uobabylon.edu.iq

Ali Kadhum Idrees*

Department of Computer Science,
University of Babylon,
Babylon, Iraq
Email: ali.idrees@uobabylon.edu.iq
*Corresponding author

Abstract: This study proposes the Sensor Device Scheduling-based Cuckoo Algorithm (SeDeSCA) for enhancing the lifespan of cluster-based WSNs. SeDeSCA consists of two phases: clustering and scheduling. The WSN is clustered into clusters using the DBSCAN algorithm in the first phase. The scheduling phase is periodic and consists of three steps: cluster head polling, scheduling decision-based optimisation, and covering. The sensors in each cluster choose their cluster head, which executes the Cuckoo Algorithm (CA) to select a suitable schedule of sensors that will achieve sensing during the current period. The aim of scheduling is to minimise energy consumption and ensure sufficient coverage for the monitored area while maximising network lifespan. The third step is to cover the area of interest with the sensors that are scheduled to be active during this period. The simulation results show that SeDeSCA improves the network lifespan and coverage ratio, as well as the lifespan of WSNs.

Keywords: wireless sensor networks; cuckoo algorithm; DBSCAN; lifetime enhancement; scheduling algorithms.

Reference to this paper should be made as follows: Hameed, M.K. and Idrees, A.K. (2022) 'Sensor device scheduling-based cuckoo algorithm for enhancing lifetime of cluster-based wireless sensor networks', *Int. J. Computer Applications in Technology*, Vol. 68, No. 1, pp.58–69.

Biographical notes: Mazin Kadhum Hameed received his MSc degree in Computer Science from the University of Babylon, Iraq in 2012. Currently, he is working toward the PhD degree at the University of Babylon. His research interests include wireless sensor networks, optimisation and scheduling.

Ali Kadhum Idrees received his BSc and MSc degrees in Computer Science from the University of Babylon, Iraq in 2000 and 2003, respectively. He also received his PhD degree in Wireless Networks in 2015 from the University of Franche-Comte (UFC), France. Since 2018, he has been a Full Professor in Computer Science at the University of Babylon, Iraq. He has published over 41 research papers in international conferences, journals and book chapters. His research interests include wireless networks, WSNs, WBSNs, internet of things (IoT), distributed computing, fog computing, data mining, deep learning and optimisation in communication networks. He has actively served as the Technical Program Committee Member for various networking conferences and Reviewer for several international journals.

1 Introduction

Sensors are devices of a smaller size that perform the sensing, processing, and transmission of data within wireless networks, and they are considered to be energy and cost efficient. WSNs are fundamental in many varying types of application, like security, battlefield shrivelling, air traffic controlling, bio-detection, environmental monitors, industrial automation and smart grids (Akyildiz et al., 2002; Gungor et al., 2010). The monitoring of these applications needs multiple sensors to be

deployed within the environment. The coverage of the sensor can be defined as the areas that the sensor reaches to be monitored or sensed, as more than one target can be covered at once (Cardei and Wu, 2006).

Replacing or charging the battery of a sensor is very hard because of the sensor size and that they are used in areas that cannot be reached easily (Shih, 2009). This explains the importance of the network lifespan expansion, particularly in WSNs. To do so, a number of factors should be considered, such as deploying sensors optimally (Han et al., 2009; Krause et al., 2011; Yang et al., 2011), sleep scheduling (i.e.) changing the sensor mode from active to sleep (Heet al., 2010; Ammari and Das, 2011; Ashouri et al., 2012; Jia et al., 2013), and maintaining the balance of load (Chen et al., 2014).

In a WSN, each distantly deployed sensor node is power-driven by a minor battery. Depending on power necessities of the favourite application, a battery can function for days, months or even years. On the other hand, this energy source is expectedly restricted (Nahar et al., 2019). There are numerous meta-heuristic methodologies that are broadly implemented for solving several engineering problems such as scheduling, storage and energy efficiency (Prakash and Viswanathan, 2020).

Generally, the deployment of sensor nodes could be classified into two sorts: random and deterministic deployment (Kulkarni and Venayagamoorthy, 2010; Mini et al., 2013). The former overly suits unfamiliar or hardly accessible regions. Yet, their disadvantage lies in the fact that the located nodes might or might not actually provide proper coverage for the targets, as any node without deployment near the targets could probably not capture any data of use. This issue could be solved by moving the nodes closer to the targets after being deployed. As the region itself could not be accessed, the nodes need to be capable of moving themselves.

With deterministic deployment, on the other hand, the regions are familiar for the placement of the sensor. For this case, the sensor network lifespan could be expanded during either the deployment or scheduling phases. In the former, the sensor node is put in its ideal location for increasing the network lifespan, thereby guaranteeing its coverage (Bojkovic and Bakmaz, 2008). Several biologically inspired algorithms are suggested for obtaining the ideal node placement, including, Genetic algorithm, Artificial Bee Colony (ABC) algorithm, Particle Swarm Optimisation (PSO) and Ant Colony Optimisation (ACO).

The sensor coverage could be divided into two forms, namely area and target coverage. Whenever the sensor network covers a particular region as a whole, it is considered to be area coverage. Otherwise, if it only covers a defined target within a particular region, it could be considered as a target (or point) coverage (Shih et al., 2009). Target coverage, in its turn, could also be subcategorised into simple, *k*- and *Q*-coverage. The first form of coverage involves monitoring each target through minimally one sensor node, which implies that the target stays uncovered even after the node's failure. This situation could be solved in *k*-coverage, as at least '*k*' number of sensors monitors the target. As for *Q*-coverage, multiple sensor nodes monitor multiple targets.

Therefore, the energy can be effectively made use of through scheduled nodes to monitor targets, so as to make sure that no network congestion or re-transmitted packets exist. This will decrease the amount of energy consumed additionally when communicating over the network (Hao et al., 2015). Network lifespans in target coverage may be extended by means of relying on scheduled sensors. In sensor sets, a number of sensors are clusters and they may or may not have duplicates. The active mode of the set implies that the element sensor nodes can be active, whereas the remaining elements are inactive. The network lifespan can be essentially maximised through the maintenance of load balancing. There are a number of variables that determine the availability of energy for the sensors, such as the distance to the base station, sense range and how much transmission energy is consumed. A balance can be obtained between sensor load based on the usable residual energy (Abraham and Greg, 2014; Yuan and Duan, 2008).

Several scheduling algorithms are proposed in the literature. Some of them centralised algorithms that perform the scheduling in a centralised way. This type of algorithm is not scalable and consume high energy when the network is large. The other type of algorithm is distributed that can be scalable for a large network but they cannot provide optimal solutions for extending the coverage lifetime in the network. In our proposed approach, we have proposed a hybrid scheduling approach that exploits the advantages of the centralised and distributed algorithms to provide a better schedule of sensor nodes for extending the coverage lifetime of the network. Our scheduling algorithm applied clustering to control the network topology and to divide the network into clusters. In each period, one cluster head will be elected in each cluster. Each cluster head in each cluster will execute the proposed metaheuristic scheduling algorithm to solve our coverage optimisation model in a distributed way. This hybrid metaheuristic scheduling way contributed to enhance the coverage network lifespan and reduce the energy consumption of the WSN.

This paper gives the following contributions:

i) This paper suggests Sensor Device Scheduling-based Cuckoo Algorithm (SeDeSCA) for Enhancing Lifetime of Cluster-based WSNs. This technique achieves two efficient algorithms: clustering and then scheduling the sensor devices in WSN. First, the WSN is clustered into clusters using the distributed DBSCAN approach. The scheduling phase splits the lifespan into periods, and it achieves its goal by three steps. In the first step, the cluster head polling implemented in a distributed way inside each cluster. In the second step, the cluster head will execute Cuckoo Algorithm to optimise the coverage lifespan of the network to produce the optimal schedule of sensor devices that are responsible for monitoring the cluster region in the

next step. In the third step, each sensor device will receive a packet from the cluster head to inform it to stay active or sleep until the beginning of the next period.

ii) The Cuckoo Algorithm (CA) is applied to optimise the network lifespan while maintaining adequate coverage for the monitoring area. CA is employed to solve the proposed coverage optimisation model to decrease the number of devices that are not covered and minimise the number of active sensor devices in each period. The scheduling algorithm-based CA is responsible for providing the best schedule of sensor devices in each period instead of using optimisation solvers. This can decrease the execution time and maximise network lifespan.

iii) The C++ custom simulator has been used in performing extensive experiments. The results indicate that the suggested SeDeSCA technique can enhance the coverage and extend the network lifespan in comparison with other methods like DESK (Yu et al., 2012), GAF (Xu et al., 2001) and PeCO (Idrees et al., 2015).

The remaining parts of this research can be outlined in the following way: Section 2 refers to the related works, Section 3 presents a review of the CS and DBSCAN algorithms and Section 4 will describe the proposed algorithm. Sections 5 and 6 present and discuss the results of experiment as well as the concluding remarks, respectively.

2 Related works

There are a number of works that are related to the present study in some form or another. Nath and Gibbons (2007) developed Sleep Scheduling algorithms, namely the Connected *K*-Neighbourhood (CKN) and the enhanced Energy Consumed uniformly – Connected *K*-Neighbourhood (EC-CKN) algorithms, respectively (Nath and Gibbons, 2007).

These algorithms shut down the lower power nodes with the maintenance of the network's connectivity and sufficient routing latency. With the CKN, nodes have k-connections. This implies that whenever the k-connection is satisfied (the active nodes are more than k), then the node decides to shut itself down. Otherwise, the node stays active. As a distributed SS algorithm, all node life spans can be extended in an efficient way and thereby improve the network's overall life time. Active nodes are identified in CKN using the rank values. The first phase of the CKN algorithm is performed for all periods randomly, with variation in the number of active nodes involved. A drawback of using this algorithm is there is no guarantee of the energy being consumed uniformly (Yuan et al., 2011). On the other hand, in EC-CKN, the rest of the node energy depends on the CKN, so the energy consumption in the network is balanced while maintaining the *k*-connectivity.

Tian and Georganas (2005, 2002) introduced a form of coverage that depends on the node-scheduling mechanism characterised by its energy efficiency. Throughout the auto-scheduling phase, the node examines the off-duty eligibility rule which checks if the nodes' sensing area is covered by any neighbouring sense regions. An eligible node could switch off its functionality, enabling the others to perform the detection process within a certain period of time. Hsin and Liu (2004) introduced a generic, duty-cycling scheduling technique. It makes use of the stochastic theory. In Zhao and Wu (2010), they analysed the coverage and duty cycling characteristics as by the scheduling algorithms. Wang et al. (2019) made judgements upon the sensor node state using the redundancy algorithm according to the neighbour node locations. Whenever a node is found to be redundant, it is switched into sleep mode.

Koubaa et al. (2007) presented the Super-frame Duration Scheduling (SDS) algorithm, provided in a Time Division Beacon Frame Scheduling (TDBS) approach. The main concept of the latter approach is scheduling beacon frames throughout the idle times of adjacent clusters for avoiding an eventual inter-cluster beacon-data collision. This enables clusters to take a sequential order according to the super-frame duration in a periodical major cycle. Yet, such an approach provides no specifications for a mechanism for prioritising a particular traffic pattern.

Indora and Singh (2018) proposed the SEED algorithm, whereby network fields have three divisions: the CH in the higher-energy zones will communicate to the base station to conserve energy in lower-energy areas CHs. This algorithm has a better performance than alternative energy-efficiency mechanisms.

As for (Yu et al., 2012), an algorithm is suggested for scheduling sensor devices in sub-divided grids in the areas of concern, according to the device's geographical position. In (Xu, 2001), the DESK scheduling algorithm is introduced, which has been applied onto all sensor devices. The decisions that have been made are according to the locally available information from neighbour devices through the perimeter coverage model. The studies conducted in Idrees et al. (2015, 2016) proposed two coverage algorithms for the extension of WSN lifespans. The first is DiLCO and makes use of an optimisation solver. The coverage optimising mode depends on essential points when producing the best schedule during the rounds. The other algorithm suggested the maintenance and improvement of coverage and network lifespan, using the perimeter coverage model.

There are a number of algorithms introduced for solving scheduling issues in the WSN. They can be divided into the distributed and centralised scheduling approach. The first one has a rapid performance, as they rely upon locally available information, however no optimal device scheduling can be provided. The second type can suggest an optimal scheduling solution, yet it takes a longer time to be executed when the WS network is larger. A combined type of algorithm involves a global distribution, but is locally centralised and uses an optimisation solver. This type improves the WSN lifetime, but their drawback is a rather high time consumption.

In this paper, a Sensor Device Scheduling-based Cuckoo Algorithm (SeDeSCA) to enhance the life span in cluster-based networking, as the network is divided into partitions of time. Each of these periods optimise the network lifetime in light of the distributed DBSCAN clustering, CH distributing polling and CA sensor schedules. Through this approach, the WSN life span is prolonged with the maintenance of sufficient coverage over the given zone of monitoring.

3 Scientific background

3.1 Cuckoo algorithm

CS is classified as a stochastic algorithm whose mechanism is inspired by the remarkable behaviour of certain cuckoo species (Yang and Deb, 2010a, 2010b), as they tend to keep the eggs in nests of other birds. This algorithm contributed to the efficiency of global searching within solution domains in a more significant manner than any alternative optimising algorithm (Mlakar et al., 2016, 2017).

A *STANDARD CS*: A more simplified form of CS is presented in Ouaarab et al. (2014) eggs represent solutions and the nests are the individuals of the population. Any solution which does not function properly is replaceable by a newer alternative, and the same goes for any abandoned nests. The number of nests equals the population size, and it is represented by three rules. The standard CS algorithm is mainly determined by three rules (Yang and Deb, 2010a, 2010b):

1 Every cuckoo bird will lay a single egg per time within any randomly chosen nest;

2 Nests that have higher quality eggs will continue to the following generation;

3 The hosting bird (or nest owner) could identify the alien egg with a probability of $P_\alpha \in (0, 1)$, after which it might dispose the egg or leave the nest itself to create another.

The standard CS is considered to be of higher efficiency for its simpler structures, fewer variables and relatively easier form of implementing. For a mathematical point of view, the definition of a nest position is as follows:

$$X_i^{(k)},\ i \in \{1, 2, ..., NP\} \tag{1}$$

where NP and k stand for the numbers of cuckoo nests (population size) and generations, respectively.

New solutions $x_i^{(k+1)}$ could be generated through the use of global and local random walk combined in a balanced manner through Lévy flight (Yang and Deb, 2010a, 2010b). The global random walk is provided by:

$$x_i^{(k+1)} = xi^{(k)} + \alpha \oplus levy(\lambda) \tag{2}$$

where $\oplus$ denotes entry-wise multiplication, σ represents a step-size in relation to the issue scale. This could often be measured by means of the following calculation (Indora and Singh, 2018; Wang and Zhong, 2015):

$$\alpha = \alpha_0 \times \left(xi^{(k)} - xbest^{(k)}\right) \tag{3}$$

where α_0 represents the scaling factor; $xbest^{(k)}$ is the most suitable solution at present time, and $levy(\lambda)$ represents a randomly chosen number from a Lévy distribution:

$$levy(\lambda) \sim t - 1, (1 \le \lambda \ge 3) \tag{4}$$

In implementation, the calculation of *Lévy* (λ) could be simply done in the following manner (Yang and Deb, 201a, 2010b0; Mantegna, 1994; Li et al., 2018):

$$\begin{aligned} s &= \frac{u}{|v|^{1/\beta}} \\ \sigma_u &= \frac{\Gamma(1+\beta)\sin(\pi\beta/2)}{\pi\left[(1+\beta/2)\right]2^{(\beta-1)/2}\beta} 1/\beta,\ \ \sigma_v = 1 \end{aligned} \tag{5}$$

where s forms the simulating value of *Lévy* (λ); u and v represent two randomly assigned umbers that follow the normal distribution with a zero mean and zero $\sigma^2{}_u$ and $\sigma^2{}_v$ deviations, respectively; β represents a Lévy distribution variable, set at 1.5 (Mantegna, 1994); and Γ is a Gamma function.

As for the CS algorithm, the Lévy flight has been employed in global exploration having properties of a random nature, whereas a cross-over operator is made use of with local exploitation which has a mutation of the present solution. An update of the optimal solution is made after every iterating process. Algorithm 1 illustrates the CS algorithm for solving the optimisation problems.

B Binary cuckoo search: In the continuously-valued CS, the nest updates its position to a real value within the potential search space as limited by any issue constraint. The UC issue represents a direct optimisation with 0–1 deciding parameters that represent the ON/OFF unit mode. Thus, the real-valued CS requires a revision so that it could possibly suit a binary issue. BCS acquires its concept from the Binary Particle Swarm Optimisation (BPSO) algorithm (Kennedy and Everhart, 1997), as this usually makes use of a sigmoid function (see Figure 1) for restricting any of the novel solutions to a binary value (Gherboudj et al., 2012; Ouaarab et al., 2014).

Algorithm 1: Cuckoo search algorithm

1: **begin**
2: Objective function $f(x),\ x = (x_1, x_d)T$
3: Generate initial population of n host nests $x_i\ (i = 1, 2, n)$
4: **while** (t <MaxGeneration) or (stop criterion)
5: Get a cuckoo randomly by Levy flights evaluate its
6: Quality/fitness *Fi*

7: Choose a nest among *n* (say, *j*) randomly
8: **if** $\left(Fi > Fj\right)$,
9: replace j by the new solution;
10: **end**
11: A fraction (pa) of worse nests are abandoned and new ones are built;
12 Preserve the best solutions(or nests with quality solutions);
13 Rank the solutions and find the current best
15 **end while**
16 Post-process results and visualisation
17 end

$$S\left(xi^{(k)}\right) = 1/\left(1+e^{-xi^{(k)}}\right) \tag{6}$$

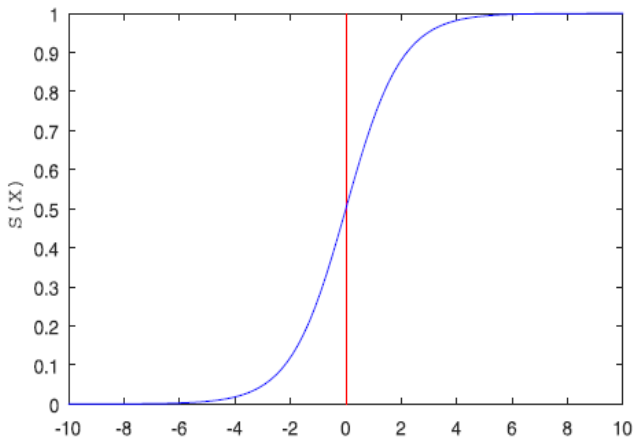
In other words, a binary solution $xi^{(k+1)}$ could be obtained by means of a typical updating equation:

$$xi^{(k+1)} = \begin{cases} 1 \text{ if } S\left(xi^{(k)}\right) > r \\ 0\, xi^{(k)} \text{ otherwise} \end{cases} \tag{7}$$

where *r* stands for a number of uniform attributions in (0, 1).

Whenever new solutions are created, (6) and (7) employed in mapping the search processes from continuous to binary spaces, following the global and local searches in CS which took place in advance.

Figure 1 Sigmoid function



3.2 DBSCAN clustering

Easter et al. (1996) proposed a data clustering algorithm known as Density-Based Spatial Clustering of Applications with Noise (DBSCAN). The DBSCAN algorithm is a density-based clustering non-parametric algorithm: given a lot of focuses in some space, it assortments composed points that are firmly stuffed composed (points with numerous close by neighbours), stamping as exceptions focuses that lie alone in low-thickness locales (whose closest neighbours are excessively far away). The DBSCAN is one of the most well-known clustering algorithms and furthermore most referred to in scientific literature (Sharma et al., 2012).

Considering the focuses on some spaces to be grouped, let ε be a boundary indicating the sweep of an area as for some point. With the end goal of DBSCAN clustering, the points are delegated core points, (thickness) reachable points and anomalies, as follows:

- A point *D* is a core point if at least minPts points are inside separation ε of it (counting *D*).
- A point *k* is directly reachable from *d* if point *k* is inside separation ε from core point *d*. Points are just supposed to be directly reachable from core points.
- A point *k* is reachable from d if there is a way d_1, d_n with $d_1 = d$ and $d_n = k$, where each d_{i+1} is directly reachable from d_i. Note this suggests the initial point and all points on the way should be core points, with the conceivable special case of *k*.
- All points not reachable from some other point are anomalies or clamour points.

Presently on the off chance that *d* is a core point; at that point it shapes a group along with all pointes (core or non-core) that are reachable from it. Each cluster contains at least one core point; non-core points can be a piece of a group, however they structure its 'edge', since they can't be utilised to arrive at more points.

Reachability is definitely not a symmetric connection: by description, non-core points can only reachability by the core points. The inverse isn't accurate, so a non-core point might be reachable, however nobody can be reached from it. In this manner, a further thought of connectedness is expected to officially characterise the degree of the clusters found by DBSCAN. If there is a point *l* with the end goal that both *d* and *k* are reachable from *l*, then a points *d* and *k* are thickness associated. Thickness connectedness is symmetric.

A cluster then fulfils two things:

1 All points inside the cluster are commonly thickness-associated.
2 If a point is thickness-reachable from some purpose of the cluster, it is a piece of the cluster also (Schubert et al., 2017; Sander et al., 1998).

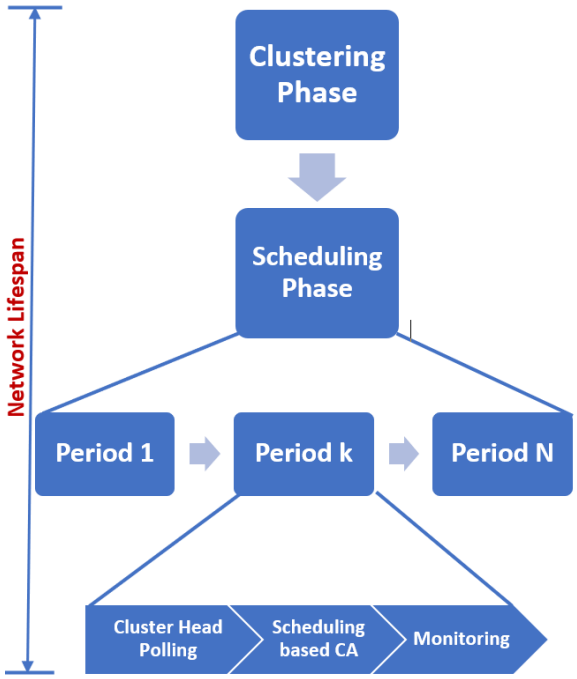
The construction of clusters requires the DBSCAN to draw an unlabelled object d in a random manner, so as to perform the ε-range query on d. In case d turns out to be a core object, the ε-range query will be executed for all $k \in N(d)$ for expanding the clusters. This will continue until no core objects are found anymore. Furthermore, a cluster label will be assigned to d and its density-connected objects, whereas the unlabelled ones will undergo processing so as to expand novel clusters.

4 The proposed SeDeSCA technique

In this article, a Sensor Device Scheduling-based Cuckoo Algorithm for Enhancing Lifespan of Cluster-based Wireless

Sensor Networks; called (SeDeSCA) technique is suggested. It consists of two phases: Clustering and scheduling. Figure 2 shows the proposed SeDeSCA technique.

Figure 2 SeDeSCA techniques



4.1 Clustering phase

The sensor devices of WSN are clustered into small clusters by DBSCAN algorithm. This distributed DBSCAN clustering algorithm is applied on every sensor device and these devices are cooperated based on the DBSCAN algorithm to form several clusters in the network area of interest.

Algorithm 2 explains the Distributed DBSCAN algorithm that will be executed in every sensor node sj.

Algorithm 2: Distributed DBSCAN (sj)

Input: N: number of neighbor nodes, Sr: sensing range, minNodes: minimum number of nodes to create cluster.

Output: sj.rejon: the cluster number for node sj.

```
while REj ≥ Ethr do
  If sj Receive MemberPacket from si then
    Mark sj as member to the Core si;
    Update REj;
  end
    sj.rejon ← 0;
   for each node si in N do  // i ∈ N and i ≠j
      nbrNodes ← nbrNodes + CORE Objective Function (sj, si, Sr);
     if CORE Objective Function return 1 then
     Send MemberPacket to the sensor node i;
        Update REj;
     end
     if nbrNodes ≥ minNodes then
        save the information
     if (((sj.rejon = 0) Or (sj.rejon ≠ 0)) and (r==0)) then
        sj.rejon ← sj.rejon +1;
       Call Cluster(sj);
     end
    else if ((sj.rejon = 0) Or (r ≠ 0)) then
        sj.rejon ← sj.rejon +1;
        Call Cluster1(sj);
     end
     else if ((sj.rejon ≠ 0) Or (r ≠ 0)) then
        Call Cluster2(sj);
     end
    end
  end for
end while
return sj.rejon;
```

The CORE Objective Function will reply with 1 and $r = 0$ in case the sensor node i found in the sensing range (Sr) and does not belong to another cluster. If not, the CORE Objective Function returns 0 and $r = 1$. The cluster functions in putting neighbouring nodes within the sj of the same clusters, and sending the sj MemberPacket to the sensor node i to inform it that it becomes a member in the same cluster of sj. The function Cluster1 put any neighbour node within the sensing range of sj and it has not assigned to any cluster in the same cluster of sensor node j. As for Cluster 2, is also puts neighbouring nodes in the sj. However no clusters are assigned to the node *j*. When each of the Cluster, Cluster1 and Cluser2 achieve their functions, an undate will be made for the energy that remains for the sensor node j by sending a MemberPacket to the sensor node i to inform it that it becomes a member in the same cluster of sj.

4.2 Scheduling phase

The scheduling stage starts with every period, after the clustering phase ends. There are three steps involved during each period, namely selecting the CH, scheduling the nodes, activities according to the Cuckoo Algorithm and the monitoring step.

A *Cluster head selection*: The production of clusters is followed by exchanging information between one core point and another in the clusters. The sensor nodes send messages to all other nodes that share the same cluster. Fundamental pieces of information are involved, such as the rested power, status, position, members' number and overall number of devices within clusters.

As for this stage, all nodes will carry the same information about the other nodes within the shared cluster. This requires each node to realise equation (8), by means of the information referred to earlier. All devices within this cluster will execute the same equation. The nodes that obtain the most favourable results will be assigned as the CH. The execution of the equation occurs in form of distribution, after which the CH will be identified.

$$FitVal_j = \frac{E_{remaining}}{E_{initial}} + \left(1 - \sum_{j \in N} \left| S_j(x,y) - S_i(x,y) \right| \right) + \frac{S_j(Members)}{Cluster(Members)} \quad (8)$$

where $E_{remaining}$ is the residual energy of the node j, $E_{initial}$ is the initial energy value of node j, N is the number of nodes in the current cluster, $S_j(x,y)$ and $S_i(x,y)$ refer to the locations of nodes S_j and S_i, respectively. S_j (Members) indicates the number of nodes members of node j, Cluster (Members) indicates the whole number of nodes in the cluster.

All CHs in the network cluster are independently assigned and selected according to distribution.

B *Activity scheduling-based cuckoo algorithm (CS)*: This stage of procedure involves the formulation of the optimising model that treats the scheduling issues. It is followed by executing the Cuckoo Algorithm so as to obtain the best solution. This is done by forming the optimal sensor device scheduling to monitor the current period.

A mathematical scheduling model has been employed in the optimisation of the network life span and coverages. There are two objectives taken into consideration in the formulation of this optimising model: the minimisation of uncovered zones and number of active sensors after the CA decisions.

The inspiration for this model has been taken from (Sander, 1998), after which a number of alterations have been made in terms of lowering the number of active nodes, decreasing the energy consumption, and improving the network life time. Given that A indicates the coverage of CHs in each cluster, its definition can be stated in the following way:

$$A_{ji} = \begin{cases} 1 \text{ if point centre } j \text{ is covered by sensor } i \\ 0 \text{ Otherwise} \end{cases} \quad (9)$$

For $1 \le j \le N$ and $1 \le i \le N$, where N is the number of sensor devices inside the cluster. The (S) refers to the solution parameter that can be either 0 or 1 according to the status of the sensor device. Its definition is stated below:

$$S_i = \begin{cases} 1 \text{ if sensor } i \text{ is Active} \\ 0 \text{ Otherwise} \end{cases} \quad (10)$$

The coverage probability P_j of the centre point j can be stated as follows:

$$P_j = 1 - \prod_{i=1}^{N} \left(1 - \left(A_{ji} * S_i\right)\right) \quad (11)$$

The initial objective of this model is the increase of covering rate over a particular area, through lowering uncovered ratio $\left(1 - P_j\right)$ in the following way:

$$P_j = 1 - P_j \quad (12)$$

Secondly, the number of points covering the same node within the sensing field is to be minimised, as follows:

$$L_i = \begin{cases} 0 \text{ if point } j \text{ is not covered} \\ \sum_{i=1}^{N} \left(A_{ji} * S_i\right) \end{cases} \quad (13)$$

The issue of the optimisation model is formed as follows:

$$\text{minimise } \cdot \sum_{j=1}^{N} P_j + \vartheta \cdot \sum_{j=1}^{N} L_j \quad (14)$$

$$\text{Subject to } \sum_{i=1}^{N} \left(A_{ji} * S_i\right) = 1 + L_j - P_j \quad \forall j \in N \quad (15)$$

$$S_i \in \{0,1\} \quad \forall i \in N \quad (16)$$

$$P_j, L_j \ge 0 \quad \forall j \in N \quad (17)$$

All the CHs are optimised using the energy efficient mechanism-based scheduling CS. The CS optimisation obtains the best cover sets of active nodes according to their centres only. They are in charge of the sensing process when being monitored. The local random walk is stated as follows:

$$xi^{(k+1)} = \begin{cases} xi^{(k)} + ra \text{ if } r'a > P\alpha \\ xi^{(k)} \quad \text{otherwise} \end{cases} \quad (18)$$

where ra and $r'a$ are two random numbers in range (0, 1).

As for the suggested algorithm, the following assumptions stay true:

- Sensor coverages are circular formed, and all sensors share the same covering with radius Rs.
- No sensing occurs across walls (boundaries or obstacles).
- No change occurs in sensing quality within Rs, and its value stays zero when outside it, based on a binary model.

The following algorithm represents the scheduling CS used to obtain the best node scheduling, which remains active throughout the monitoring of the present period.

Algorithm 3: Scheduling-based CS

Input: POP_S: is the population size
Output: Gbest: is the best solution(nest)

```
Initialise population of POP_S solutions(nests);
Population transformed into 0 or 1 by (7);
Evaluate each solution(nest) using (14);
Update best solution Gbest;
While Stopping criteria is not satisfied do
    For i = 1 to POP_S
        Generate new solution (nest) y_i (new) via (2);
        Evaluate y_i (new) via (14);
        If fit(y(new) < fit ( y_i (k));
            y_i (k+1) = y_i (new);
        Else
```

12: $y_i(k+1) = y_i(k)$;
13: **End If;**
14: **End For;**
15: **For** i = 1 to POP_S
16: Generate a new solution y_i (new) via (18);
17: Evaluate Individual y_i (new) via (14);
18: **If** fit(y_i (new)) < fit (y_i (k));
19: $y_i(k+1) = y_i$ (new)
20: **Else**
21: $y_i(k+1) = y_i(k)$;
22: **End If;**
23: **End For;**
24: Update the best solution Gbest;
25: **End While;**
26: **return** Gbest;

This type of CS is described as being evolutionary, as it optimises the performance globally, according to the special breeding technique of the cuckoo bird (Yuan and Duan, 2008), besides the Levy flight which is a pattern those birds use when looking for their food. To begin with, the population initialisation takes place, consisting of more than one possible solution that is generalised randomly. They are improved by generation, reaching the maximal number or any alternative condition. One solution is the improving procedure, taking place through the application of the Levy flight. This enables the disposal of the solution that is of least benefit, keeping the more favourable ones.

There is a matrix of k rows and n columns in the initial population, which represents the number of nests and sensors in a respective order. The CS can be outlined in the following way.

a) *Generation of initial cuckoo population*: The nest in the initial population presents a probable solution for the sensor scheduling, so as to cover the clusters overall. The values associated with the initial population are real values, based on the CS algorithm.

b) Representation of solution: The Cs aims to identify the optimisation scheduling to monitor the regions in the steps that follow. Nests are sensor nodes schedules, involving a number of eggs. The eggs can have one of two values: (1), which implies that the device is active, or (0), which represents the state of the device being idle. The population is converted from continuous to discreet, using the sigmoid function.

c) *Fitness function*: After evaluating all entities, the fitness function is applied using equation (14), so as to assign fitness values. The suggested algorithm implies that lower fitness values are more favourable, as they indicate that the entity is more likely to remain active and survive.

d) *Generation of new nests* $x_i^{(k+1)}$: The standard CS deploys each of the global and local random walk with a random combination They are given equations (2) and (18), respectively.

e) *Evaluation of new nests*: The evaluation of the new individuals is obtained through equation (14).

f) *Immigration*: After evaluation, the algorithm is applied in two stages: the replacement of nests by a new solution obtained through the random walk and Levy flight, and the calculation of the $P\alpha$ fraction for the least suitable nest in order to replace them.

g) *Updating the Gbest*: The global best will be changed based on the newer best solution that has been obtained using the Cs.

C *Monitoring*: After the production of the ideal sensor scheduling using this CS, the CH will inform all devices about their status during the next round (monitoring). The message will be either (0), indicating that it should remain idle, or (1), which implies that the sensor devices has to stay active during the following stage.

5 Performance evaluation and analysis

In this section, the evaluation is made regarding the efficiency of the SeDeSCA technique through executing multiple experiments with the use of C++ custom simulator. Table 1 presents the parameters applied in the simulating process.

Fifty executions have been performed by means of different WSN topologies. The presented results indicate the average rate of these executions. Five network sizes from 100 to 300 nodes have been used in the simulation process, deploying nodes in a controlled manner over a sensing area of (50×25) m^2 for ensuring a full coverage for the present area of interest. The suggested protocol uses the energy model discussed in Idrees et al. (2016).

Table 1 The variables applied in the simulating process

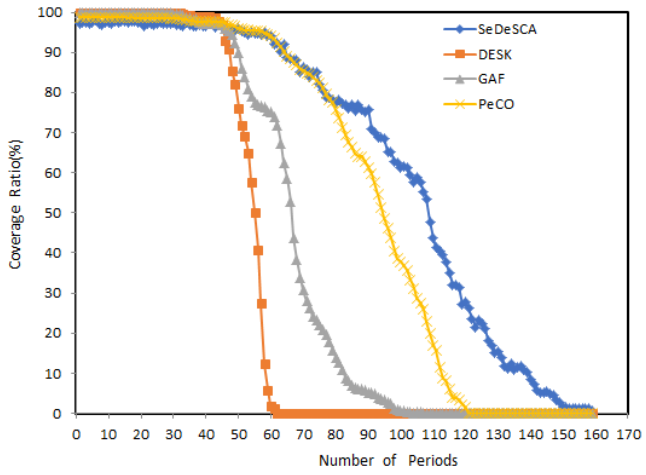
Parameter	*Value*
Field of the Sensing	(50*25) m^2
WSN size	100, 150, 200, 250 and 300 nodes
Range of the initial Energy	500–700 joules
Rs	5 m
Rc	10 m
POP_S	30
δ	0.05
Θ	0.95

The range [500–700] involves a random initialisation of sensor node energy. The evaluation of the SeDeSCA protocol is based on the same performance metrics that have been used in Idrees et al. (2016), including Coverage Ratio, Active Sensors Ratio, Network Lifetime, and Energy Consumption. There are three additional methods applied in this comparison: the DESK (Yu et al., 2012), GAF (Xu et al., 2001) and PeCO (Idrees et al., 2016).

5.1 Coverage ratio

The average values of covering ratio for each of the SeDeSCA, DESK, GAF and PeCO with 200 nodes are illustrated in Figure 3. During initial stages, the DESK, GAF and PeCO result in a slightly better coverage rates (99.99%, 99.96% and 98.76%, respectively), as compared to the (97.1%) provided by SeDeSCA. This can be traced back to the fact that SeDeSCA shuts down relatively more redundant nodes than DESK, GAF and PeCO. After the 65th period, the SeDeSCA seems to provide more favourable coverage efficiency than alternative techniques, with the maintenance of a coverage rate of about 80% for several rounds. Such an increase in efficiency is the result of the large quantity of energy that has been saved by SeDeSCA during the initial rounds.

Figure 3 Coverage ratio for WSN consisting of 200 deployment nodes



5.2 Active sensors ratio

The life span of WSNs could be maximised by conserving energy. This is realised by limiting the number of nodes that remain active during the stages. As presented in the diagram below, period one to 15 activated 30.68%, 34.5% and 20.18 nodes for each of DESK, GAF and PeCO, respectively. The SeDeSCA, on the other hand, required the activation of only 19.8% sensor nodes. A gradual increased in activated nodes is observed over time in SeDeSCA, in order to expand the coverage rates, as shown in Figure 4.

5.3 Energy consumption

The current subsection introduces the effect of energy consumption on behalf of the network throughout several statuses of the sensor node (like during the modes of communicating, computing and listening, as well as the active and sleep statuses), for different WSN sizes. An investigation is made of other approaches being compared. Each of Figures 5(a) and 5(b) illustrate the amount of energy consumption for different WSN sizes, in addition to $Lifespan_{95}$ and $Lifespan_{50}$. The *Lifespan* refers to the total time during which the WSN can provide coverage higher than X%. The relative superiority of SeDeSCA in terms of economising could be concluded from the illustration. Both Figures indicate the reduction in the amount of energy consumed by SeDeSCA in comparison with other methods. The rate of energy consumption hits relatively lower for $Lifespan_{95}$ and $Lifespan_{50}$.

Figure 4 Active sensors ratio for WSN consisting of 200 deployed nodes

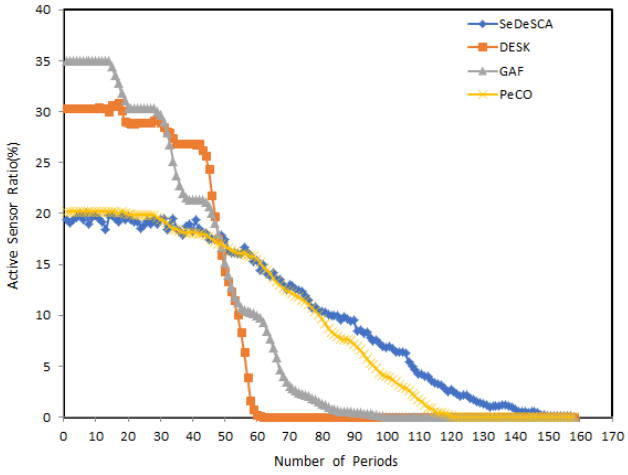
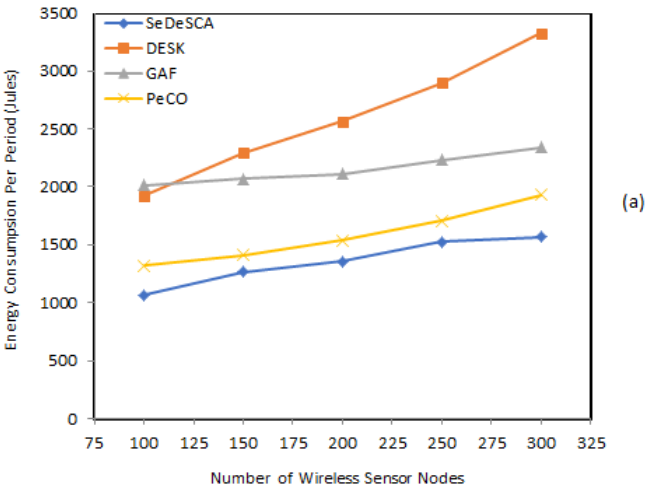
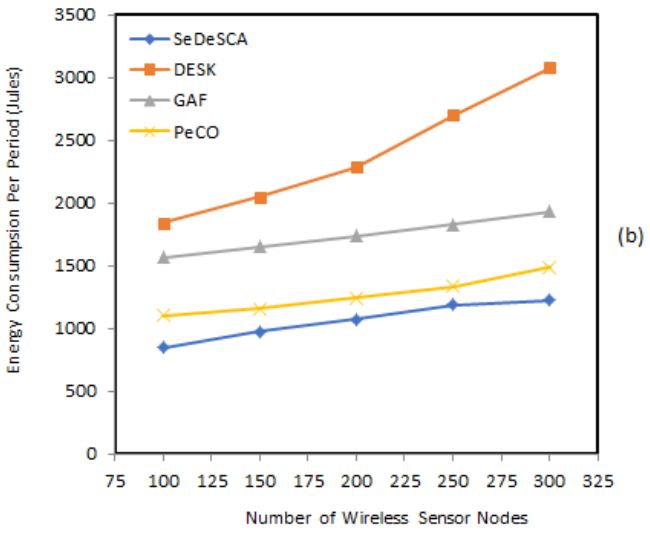


Figure 5 Energy consumption per round for (a) $Lifetime_{95}$ and (b) $Lifetime_{50}$

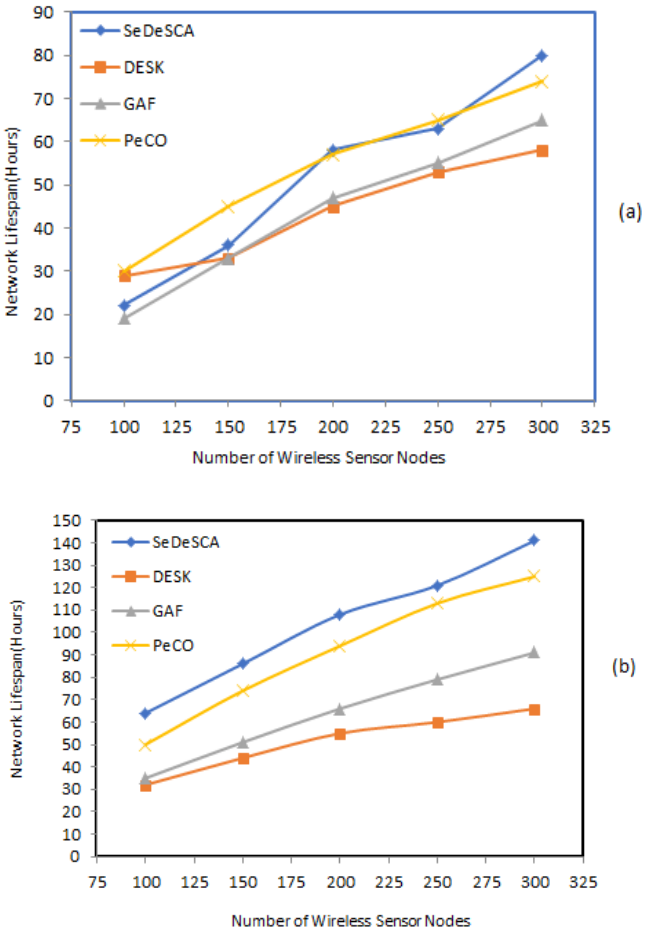




5.4 Network lifespan

By comparing the resulting data, the conclusion can be drawn that SeDeSCA provides a relatively more efficient Lifespan improvement for $Lifespan_{50}$, while SeDeSCA functions optimally for $Lifespan_{95}$. Figures 6(a) and 6(b) show the exhibition of $Lifespan_{95}$ and $Lifespan_{50}$ at different WSN densities, indicating that SeDeSCA enhances the network lifespan remarkably along with the increase in WSN size. The network lifespan for $Lifespan_{95}$ (see Figure 7 (a)) could be enhanced by means of the SeDeSCA protocol up to 15.8%, 15.4% as compared to the DESK and GAF protocols respectively. As for $Lifespan_{50}$ (see Figure 7 (b)), the SeDeSCA technique enhances the network Lifespan up to 12.3%, 50.6% and 38.1% as compared to the PeCO, DESK and GAF protocols, respectively.

Figure 6 Network lifespan for (a) $Lifespan_{95}$ and (b) $Lifespan_{50}$



5.5 Execution time

This section discusses the feasibility of our protocol in a real limited resources wireless sensor networks as well as shows why the proposed technique is better than existing distributed techniques. The average execution time is required to solve our proposed scheduling optimisation problem is reported for several methods and different network sizes. We compute the original execution time on a Lenovo laptop with Intel Core i5 2520 M processor and the MIPS (Million Instructions per Second) rate equal to 50,350. In order to be compatible with the use of a sensor node with Atmel's AVR ATmega103L microcontroller (6 MHz) and a MIPS rate equal to 6 to run the optimizstion resolution, this time is multiplied by 2097.9 $\left(\frac{50350}{4}\times\frac{1}{6}\right)$ and reported on Figure 7.

Figure 7 Execution time (in seconds)

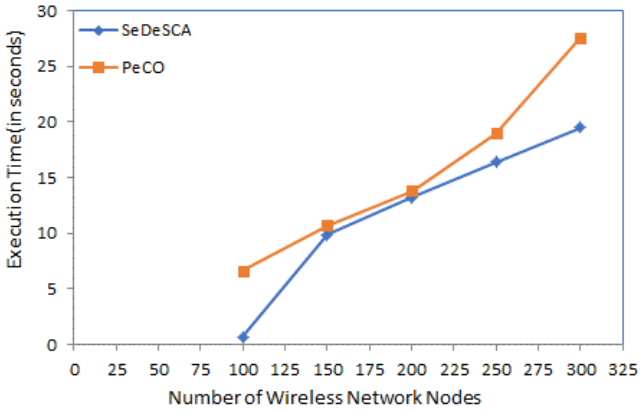


Figure 7 shows that the SeDeSCA technique has lower execution times in comparison with other methods due to utilising a metaheuristic method (Cuckoo Algorithm) to find good approximate solutions for the scheduling optimisation problem instead of using the optimisation solver.

6 Conclusion

The sensor nodes scheduling to cover the area of interest and improve the network lifespan represent a significant problem in WSN. Since the sensor nodes are resource-constrained, therefore it is necessary to propose methods to optimise these resources to improve the lifespan of the WSN. In this paper, a Sensor Device Scheduling-based Cuckoo Algorithm (SeDeSCA) to improve the lifespan of cluster-based WSNs is proposed. The SeDeSCA technique is composed of two stages: First, clustering the WSN into clusters using the DBSCAN method. Second, periodic scheduling that consists of three steps: cluster head election, Scheduling-based Cuckoo Algorithm to produce the best schedule of the sensor nodes to take the mission of sensing during the next step, and monitoring. The sensor nodes in each cluster determine their cluster head. The simulation results indicate that the proposed SeDeSCA technique outperforms the other methods in terms of coverage ratio, active sensor nodes ratio, consumed energy, network lifespan and execution time. In future, we plan to propose a multi-period protocol to provide more than one schedule during one execution. In addition, the coverage optimisation model can be enhanced to include the heterogeneity of the nodes into account.

References

Abraham, S. and Greg, G. (2014) *Operating System Concepts Essentials*, 2nd ed., Wiley.

Akyildiz, I.F., Su, W., Sankarasubramaniam, Y. and Cayirci, E. (2002) 'Wireless sensor networks: a survey', *Computer Networks*, Vol. 38, No. 4, pp.393–422.

Ammari, H.M. and Das, S.K. (2011) 'Centralized and clustered k-coverage protocols for wireless sensor networks', *IEEE Transactions on Computers*, Vol. 61, No. 1, pp.118–133.

Ashouri, M., Zali, Z., Mousavi, S.R. and Hashemi, M.R. (2012) 'New optimal solution to disjoint set K-coverage for lifetime extension in wireless sensor networks', *IET Wireless Sensor Systems*, Vol. 2, No. 1, pp.31–39.

Bojkovic, Z. and Bakmaz, B. (2008) 'A survey on wireless sensor networks deployment', *WSEAS Transactions on Communications*, Vol. 7, No. 12, pp.1172–1181.

Cardei, M. and Wu, J. (2006) 'Energy-efficient coverage problems in wireless ad-hoc sensor networks', *Computer Communications*, Vol. 29, No. 4, pp.413–420.

Chen, C.P., Mukhopadhyay, S.C., Chuang, C.L., Liu, M.Y. and Jiang, J.A. (2014) 'Efficient coverage and connectivity preservation with load balance for wireless sensor networks', *IEEE Sensors Journal*, Vol. 15, No. 1, pp.48–62.

Ester, M., Kriegel, H.P., Sander, J. and Xu, X., (1996) 'A density-based algorithm for discovering clusters in large spatial databases with noise', *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, Vol. 96, No. 34, pp.226–231.

Gherboudj, A., Layeb, A. and Chikhi, S. (2012) 'Solving 0-1 knapsack problems by a discrete binary version of cuckoo search algorithm', *International Journal of Bio-Inspired Computation*, Vol. 4, No. 4, pp.229–236.

Gungor, V.C., Lu, B. and Hancke, G.P. (2010) 'Opportunities and challenges of wireless sensor networks in smart grid', *IEEE Transactions on Industrial Electronics*, Vol. 57, No. 10, pp.3557–3564.

Han, X., Cao, X., Lloyd, E.L. and Shen, C.C. (2009) 'Fault-tolerant relay node placement in heterogeneous wireless sensor networks', *IEEE Transactions on Mobile Computing*, Vol. 9, No. 5, pp.643–656.

Hao, J., Zhang, B., Jiao, Z. and Mao, S. (2015) 'Adaptive compressive sensing-based sample scheduling mechanism for wireless sensor networks', *Pervasive and Mobile Computing*, Vol. 22, pp.113–125.

He, X., Yang, H. and Gui, X. (2010) 'The maximum coverage set calculated algorithm for WSN area coverage', *Journal of Networks*, Vol. 5, No. 6, pp.650–657.

Hsin, C.F. and Liu, M. (2004) 'Network coverage using low duty-cycled sensors: random & coordinated sleep algorithms', *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks*, pp.433–442.

Idrees, A.K., Deschinkel, K., Salomon, M. and Couturier, R. (2015) 'Distributed lifetime coverage optimization protocol in wireless sensor networks', *The Journal of Supercomputing*, Vol. 71, No. 12, pp.4578–4593.

Idrees, A.K., Deschinkel, K., Salomon, M. and Couturier, R. (2016) 'Perimeter-based coverage optimization to improve lifetime in wireless sensor networks', *Engineering Optimization*, Vol. 48, No. 11, pp.1951–1972.

Indora, S. and Singh, D. (2018) 'Sleep-distance based sleep-awake mechanisms in wireless sensor network', *International Journal of Electronics Engineering*, Vol. 10, No. 2, pp.572–576.

Jia, S.G., Lu, L.P., Su, L.D., Xing, G.L. and Zhai, M.Y. (2013) 'An efficient sleeping scheduling for save energy consumption in wireless sensor networks', *Advanced Materials Research*, Trans Tech Publications Ltd., Vol. 756, pp.2288–2293.

Kennedy, J. and Everhart, R.C., (1997) 'Adiscrete binary version of the partical swarm algorithm', *Proceedings of the Conference on Systems, Man and Cybernetics*, pp.4104–4109.

Koubaa, A., Cunha, A. and Alves, M. (2007) 'A time division beacon scheduling mechanism for IEEE 802.15: 4/ZigBee cluster-tree wireless sensor networks', *Proceedings of the 19th Euromicro Conference on Real-Time Systems (ECRTS'07)*, IEEE, pp.125–135.

Krause, A., Rajagopal, R., Gupta, A. and Guestrin, C. (2011) 'Simultaneous optimization of sensor placements and balanced schedules', *IEEE Transactions on Automatic Control*, Vol. 56, No. 10, pp.2390–2405.

Kulkarni, R.V. and Venayagamoorthy, G.K. (2010) 'Particle swarm optimization in wireless-sensor networks: a brief survey', *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, Vol. 41, No. 2, pp.262–267.

Li, S., Bi, F., Chen, W., Miao, X., Liu, J. and Tang, C. (2018) 'An improved information security risk assessments method for cyber-physical-social computing and networking', *IEEE Access*, Vol. 6, pp.10311–10319.

Mantegna, R.N. (1994) 'Fast, accurate algorithm for numerical simulation of Levy stable stochastic processes', *Physical Review E*, Vol. 49, No. 5. Doi: 10.1103/PhysRevE.49.4677.

Mini, S., Udgata, S.K. and Sabat, S.L. (2013) 'Sensor deployment and scheduling for target coverage problem in wireless sensor networks', *IEEE Sensors Journal*, Vol. 14, No. 3, pp.636–644.

Mlakar, U., Fister Jr, I. and Fister, I. (2016) 'Hybrid self-adaptive cuckoo search for global optimization', *Swarm and Evolutionary Computation*, Vol. 29, pp.47–72.

Mlakar, U., Zorman, M., Fister Jr, I. and Fister, I. (2017) 'Modified binary cuckoo search for association rule mining', *Journal of Intelligent and Fuzzy Systems*, Vol. 32, No. 6, pp.4319–4330.

Nahar, K.M., Al-Khatib, R.E.M., Barhoush, M. and Halin, A.A. (2019) 'MPF-LEACH: modified probability function for cluster head election in LEACH protocol', *International Journal of Computer Applications in Technology*, Vol. 60, No. 3, pp.267–280.

Nath, S. and Gibbons, P.B. (2007) 'Communicating via fireflies: Geographic routing on duty-cycled sensors', *Proceedings of the 6th International Conference on Information Processing in Sensor Networks*, pp.440–449.

Ouaarab, A., Ahiod, B. and Yang, X.S. (2014) 'Improved and discrete cuckoo search for solving the travelling salesman problem', *Cuckoo Search and Firefly Algorithm*, Springer, Cham, pp.63–84.

Prakash, B. and Viswanathan, V. (2020) 'A comparative study of meta-heuristic optimisation techniques for prioritisation of risks in agile software development', *International Journal of Computer Applications in Technology*, Vol. 62, No. 2, pp.175–188.

Sander, J., Ester, M., Kriegel, H.P. and Xu, X. (1998) 'Density-based clustering in spatial databases: the algorithm gdbscan and its applications', *Data Mining and Knowledge Discovery*, Vol. 2, No. 2, pp.169–194.

Schubert, E., Sander, J., Ester, M., Kriegel, H.P. and Xu, X. (2017) 'DBSCAN revisited, revisited: why and how you should (still) use DBSCAN', *ACM Transactions on Database Systems (TODS)*, Vol. 42, No. 3, pp.1–21.

Sharma, N., Bajpai, A. and Litoriya, M.R. (2012) 'Comparison the various clustering algorithms of WEKA tools', *Facilities*, Vol. 4, No. 7, pp.78–80.

Shih, K.P., Chen, H.C., Chou, C.M. and Liu, B.J. (2009) 'On target coverage in wireless heterogeneous sensor networks with multiple sensing units', *Journal of Network and Computer Applications*, Vol. 32, No. 4, pp.866–877.

Tian, D. and Georganas, N.D. (2002) 'A coverage-preserving node scheduling scheme for large wireless sensor networks', *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, pp.32–41.

Tian, D. and Georganas, N.D. (2005) 'Connectivity maintenance and coverage preservation in wireless sensor networks', *Ad Hoc Networks*, Vol. 3, No. 6, pp.744–761.

Wang, L. and Zhong, Y. (2015) 'Cuckoo search algorithm with chaotic maps', *Mathematical Problems in Engineering*, pp.1–15.

Wang, Z., Chen, Y., Liu, B., Yang, H., Su, Z. and Zhu, Y. (2019) 'A sensor node scheduling algorithm for heterogeneous wireless sensor networks', *International Journal of Distributed Sensor Networks*, Vol. 15, No. 1. Doi: 10.1177/1550147719826311.

Xu, Y., Heidemann, J. and Estrin, D. (2001) 'Geography-informed energy conservation for ad hoc routing', *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, pp.70–84.

Yang, D., Misra, S., Fang, X., Xue, G. and Zhang, J. (2011) 'Two-tiered constrained relay node placement in wireless sensor networks: computational complexity and efficient approximations', *IEEE Transactions on Mobile Computing*, Vol. 11, No. 8, pp.1399–1411.

Yang, X.S. and Deb, S. (2010a) 'Engineering optimisation by cuckoo search', *International Journal of Mathematical Modelling and Numerical Optimisation*, Vol. 1, No. 4, pp.330–343.

Yang, X.S. and Deb, S. (2010b) 'Cuckoo search via Lévy flights', *World Congress on Nature and Biologically Inspired Computing (NaBIC)*, IEEE, pp.210–214.

Yu, J., Ren, S., Wan, S., Yu, D. and Wang, G. (2012) 'A stochastic k-coverage scheduling algorithm in wireless sensor networks', *International Journal of Distributed Sensor Networks*, Vol. 8, No. 11. Doi: 10.1155/2012/746501.

Yuan, X. and Duan, Z. (2008) 'Fair round-robin: a low complexity packet schduler with proportional and worst-case fairness', *IEEE Transactions on Computers*, Vol. 58, No. 3, pp.365–379.

Yuan, Z., Wang, L., Shu, L., Hara, T. and Qin, Z. (2011) 'A balanced energy consumption sleep scheduling algorithm in wireless sensor networks', *Proceedings of the 7th International Wireless Communications and Mobile Computing Conference*, IEEE, pp.831–835.

Zhao, Y. and Wu, J. (2010) 'Stochastic sleep scheduling for large scale wireless sensor networks', *IEEE International Conference on Communications*, IEEE, pp.1–5.