Sleep-wakeup Scheduling Algorithm for Lifespan Maximization of Directional Sensor Networks: An Advanced Discrete Cuckoo Search Optimization Algorithm

Mirsaeid Hosseini shirvani (✉ mirsaeid_hosseini@yahoo.com)

Department of computer engineering, sari branch, islamic azad university, sari, iran

Research Article

Keywords: Directional Sensor Network (DSN), Target Coverage, Cover Set, Discrete Cuckoo Search Optimization Algorithm (D-CSOA)

Posted Date: September 27th, 2021

DOI: <https://doi.org/10.21203/rs.3.rs-695725/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Sleep-wakeup Scheduling Algorithm for Lifespan Maximization of Directional Sensor Networks: An Advanced Discrete Cuckoo Search Optimization Algorithm

Mirsaeid Hosseini Shirvani✉*1

1) Department of Computer Engineering, Sari Branch, Islamic Azad University, Sari, IRAN

✉*Corresponding Author : mirsaeid_hosseini@iausari.ac.ir

Abstract:

Directional sensor networks are ad hoc networks which are utilized in different applications to monitor and coverage all of the specific targets in the observing fields permanently. These kinds of networks include several configurable directional sensors in which they can be adjusted in one the possible directions along with one of its adjustable ranges. Although the energy harvesting methodology is being applied for these battery-hungry applications, the battery management and network lifetime maximization is still a challenge. This paper formulates the expansion of directional sensor network lifespan to a discrete optimization problem. Several proposals were presented in literature to solve the stated problem, but majority of them are threatened to get stuck in local optimum and led low efficiency. To solve this combinatorial problem, an advanced discrete cuckoo search algorithm is designed and is called several times until the remaining battery associated to alive sensors do not let observe all targets. In each time, algorithm returns an efficient *cover* along with its activation time. A *cover* is a sub set of available sensors capable of monitoring all targets in the observing field. In the determined activation time, the sensors in the *cover* are scheduled in *wakeup* mode whereas others are set in *sleep* mode to save energy. Designing miscellaneous discrete walking around procedures makes to reach a good balance between exploration and exploitation in search space. The proposed algorithm has been tested in different scenarios to be evaluated. The simulation results in variety circumstances proves the superiority of the proposed algorithm is about 19.33%, 14.83%, 13.50%, and 5.33% in term of average lifespan improvement against *H-MNLAR*, *ACOSC*, *GA*, and *HDPSO* algorithms respectively.

Keywords: Directional Sensor Network (DSN), Target Coverage, Cover Set, Discrete Cuckoo Search Optimization Algorithm (D-CSOA)

1. Introduction

Directional sensor networks (*DSNs*) include set of directional sensors with adjustable ranges such as infrared, radar, ultrasonic, video camera sensors which are usually used for sensing & monitoring, control, and surveillance of special environments [1]. These kinds of networks attracted great attentions in research communities because of their pervasive and variety real world applications especially in IoT context. Some of *DSN* applications in different industries are in gas, oil, military, national security, wild life, and forestry fields in which the goal is to monitor and full coverage of predefined set of targets continuously [2-5]. In the other words, some of the industries are so critical in which the need is to observe permanently the whole of targets so that the appropriate reaction against an event is done by the sink base station whether it is in the filed or not [1]. One of the most important features of directional sensors is that they are only activated in the one of the multiple direction capabilities with adjusted sensing range among set of different possible ranges. In fact, it works in a sector-like behavior unlike traditional sensor network that works in a disk-like monitoring capability. Fig. 1 depicts a typical directional sensor S_i which can work in one of its three possible directions (it only observes 120° degree in each direction) and it can be activated in one of its two possible ranges. In example of Fig. 1, if the sensor S_i is activated in the first direction by the first adjusting range, namely $S_i(1,1)$, it can observe the target t_1 . In addition, if it is activated in the second direction by that adjusted range, namely $S_i(2,1)$, it can monitor no target unless it adjusts its range to the second gear, namely $S_i(2,2)$, so that it can observe target t_2 . Note that, this kind of sensor can only be activated in one direction by only one adjusted range at the same time.

Fig. 1. A typical directional sensor with 3 possible directions and 2 adjustable ranges

One of the most important concerns associated to *DSN* is battery management. Although the energy harvesting technology such as solar energy aims to obviate the battery replacing and replenishing of depleted battery problems, the power management or network lifespan

improvement of *DSN* are still its challenges because this type of harvesting technology is either expensive or it does not work properly in the night, foggy, and cloudy environments [6-7]. Since the sensors are pinned to limited battery power, the network lifetime maximization is a great challenge in this type of network. The network lifetime refers to a period between the outset of working the network until the time the first target is out of coverage because of some sensors' battery depletion. Bear in mind that all the targets must be continuously monitored during this period for this reason only an intelligent and dynamic scheduler can figure out this complexity in great extent. The maximization of *DSN*'s lifetime issue is an NP-Hard problem [4, 20]. The maximum network lifetime with sensor adjustable ranges (*MNLAR*) is a well-known NP-Hard problem [8]. The *DSN* comprises n directional sensors that are placed in n different locations to observe m predefined targets continuously. For the sake of view angle and battery limitations, the number of sensor nodes are commonly multiple times in comparison with the number of targets to have reliable field monitorings. Each sensor node has two states active (*wakeup*) and passive (*sleep*) modes. In the *sleep* mode, the sensor's power consumption is negligible in which it can be overlooked. Also, the switching overhead between sensor modes is considered zero. It is favorable to change sensor's state from *wakeup* to the *sleep* mode once it acts the same functionality such as other sensors act in term of target monitoring at the same time. In the other words, it is desirable to reduce sensors' coverage overlap. However, in the *wakeup* mode the battery consumption depends on the adjusted sensing range. The adjusted greater range is, the greater amount of battery consumption is. The heuristic which engages sensors in such a way to monitor all targets must be aware of some clues. Firstly, it must adjust sensor range in optimal manner. The optimal sensing range for a target is the one that if it is reduced the target cannot be observed. In the other words, the range of activated sensor for a target must be adjusted in the minimum radius as possible. Another clue revolves around this fact that if the engaged sensor with this adjusted optimal range can cover other targets or not. If so, it is not necessary to engage additional sensors to cover those more covered targets and the concentration should be on scheduling of other sensors for remaining uncovered targets. In this way, the more active but idle sensors' state can be changed to *sleep* mode. Consequently, more battery is saved toward lifetime prolonging by utilizing efficient engaged sensors. In this regards, an intelligent scheduler algorithm should periodically wake up some sensors and sleep others. Also, it must adjust *wakeup* sensors in optimal adjusted ranges along with activating them in appropriate direction. In other words, the dynamic scheduler algorithm must constitute a cover set (*CS*) in each time interval in which the *CS* can cover all of the predefined targets in the observing field. The *CS* includes set of active (*wakeup*) sensors with optimal adjusted sensors in suitable direction which can cover all of the targets in the field; note that, state of other available sensors except for in the current *CS* are set to *sleep* mode to save energy. For each *CS*, an upper time limit is considered as a threshold. This threshold strongly depends on the weakest sensor in term of remaining battery power in the current *CS* because the battery depletion of that sensor leads some targets to be out of coverage. By exploiting this strategy, the lifetime with high probability is extended. The scheduler utilizes sensors in the current *CS* within the threshold time duration to cover all targets permanently; after the time is elapsed equal to specified amount, the new *CS* is constituted and is utilized; then, the state of sensors in current new *CS* are set to *wakeup* and other unnecessary sensors' are set to *sleep* mode to save battery life. It is iterated until no *CS* can be created. One of the most important clues that the scheduler must

be aware is to prioritize targets in regarding to constituting the new *cover*. If it considers hazard target with high priority sooner than other targets by adjusting the most appropriate sensor with optimal setting, the energy saving is the most effective; this method is used to generate initial population. The hazard target is the one target in which the sum of remaining battery of all sensors that can monitor that hazard target is the least in comparison to other targets. The network lifetime is calculated by the sum of all activation time (that specific time amount) associated to all constituted CSs. If both CS set and its activation time are selected optimally, then the final optimal network lifetime is optimally achieved based on induction rule [9]. It means that, the algorithm returns optimal solution provided all of its sub problems return their optimal sub-solutions. In this line, if the best CS with the most suited upper time limit are selected in each time interval, the final result will definitely be optimal. Since it is a perplex crux, the smart algorithm can figure it out. The main objective of the current paper is to extend network lifetime by exploiting different techniques. Since the search space is very large, this paper presents an advanced meta-heuristic algorithm to find the efficient *cover* along with its activation time for each round. Its objective function is to minimize the *cover*'s battery consumption; then, it returns an efficient *cover* that consumes the least amount of sum battery usage of all utilized sensors and switch off other unnecessary sensors. Also, for the sake of the uniform usage of all sensors' battery, another function is designed and added to objective function which conducts making covers with both minimum total battery usage and distribution of battery consumption. This fitness lets have more opportunity to construct new *cover* for forthcoming rounds because it saves more energy. Several heuristic and meta-heuristic algorithms have been presented in literature to figure out the well-known *MNLAR* problem. An exact and heuristic-based methods were proposed by Cerulli et al. to maximize network lifetime in wireless sensor networks with adjustable sensing ranges [5]. In the suggested work, the network lifetime challenge was modeled to an optimization problem which was solved by different greedy algorithms. The same heuristic methods were presented by Mohamadi et al. for *DSN* environment [8]. Although both proposed algorithms were promising in final results, these algorithms need more operators to explore bigger search space. A genetic-based meta-heuristic algorithm has been presented to solve the *MNLAR* problem with exploring search space efficiently [4]. It works better than the mentioned heuristics, but it suffers from simple crossover and mutation operators. Another PSO-based meta-heuristic algorithm was presented to solve target-barrier coverage problem in *DSN* network with rotatable sensors in special context such as ecological reserve and contamination monitoring [10]. As the search space of the stated problem is discrete in nature, the intrinsic continuous optimization algorithms such as PSO-based, GWO-based, and WOA-based cannot take over these kinds of problems efficiently. On the other hand, the genetic-based algorithms (GAs) are adaptive in both discrete and continuous search space, but all of them are susceptible to get stuck in local optimum [11]. One of the promising optimization algorithms which experiences even bad solutions to open new room for avoiding from local optimum trap is the simulated annealing algorithm (SA) [12]. Although it returns good solutions especially in the discrete search space, it naturally has local search inclination and cannot control large scale searching fields. For the sake of aforementioned reasons, this paper presents an advanced discrete version of cuckoo search optimization algorithm (*D-CSA*) that does not have aforesaid shortcomings to solve well-known *MNLAR* problem. This algorithm takes benefit of several plus points. Firstly, similar to SA, it experiences bad solutions to evade from local optimal

trap. Secondly, it is population-based algorithm in spite of SA which is a point-wise algorithm; for this reason, it has different agents to quest unexplored search filed globally. Thirdly, it utilizes *levy flight* concept to explore search space uniformly [13]. For this reason semi-random walking around procedures are applied which are the simulation of *levy flight* in discrete space to permute solutions toward global best effectively. In this phase, variety operators are introduced each of which is randomly, unbiasedly, called in spite of other meta-heuristic algorithms which apply limited operators even biasedly. Succinctly, the proposed *D-CSA* is run several times. In each time it returns the most efficient *cover* along with its effective activation time. This scheduler determines the sensors in this *cover* must be in *wakeup* mode and the rest sensors must be scheduled in *sleep* mode. After activation time is elapsed, another *cover* is created following by performing *sleep/wakeup* scheduling on available sensors. This procedure is repeated until the remaining battery associated to available sensors do not let to create another *cover*.

Therefore, the novelty of the current paper is:

- 1- It formulates *MNLAR* issue to a discrete optimization problem which is an NP-Hard problem
- 2- It presents an advanced discrete cuckoo search optimization algorithm (*D-CSA*) to solve this combinatorial problem
- 3- It presents variety discrete *levy flight* walking around procedures for efficient and uniformly permutation of discrete search space toward global best
- 4- It defines different scenarios and settings to validate the performance of proposed *D-CSA* against other existing state-of-the-arts

The rest of this current paper is structured as follows. Section 2 briefly reviews related works. Section 3 is dedicated to problem statement along with sub sections which introduce notations and problem formulation. The proposed algorithm is placed in section 4. Section 5 presents performance evaluation. Section 6 concludes the paper along with future direction.

2. Related Works

One of the most important application of *DSNs* is the targets coverage in specific observing field. In the other word, targets coverage means that the gathering data of predetermined points from environment is continuously done so that the appropriate reaction is performed by the sink surveillance whether he/she is in the field or not. Apart from the *DSN*'s intrinsic constraints in term of memory and processing capacities, the maximization of network lifespan is a great challenge owing to battery power limitation of sensors. Different solutions were proposed in literature to improve lifetime in these networks. The proposals in literature can be classified in three categories: exact, heuristic, and meta-heuristic approaches. The coverage algorithms which maximize network lifetime and observe all targets at the same time are either target-centric or area centric. In the former, the fixed and limited number of targets are distributed in special environment so that all of them must be continuously monitored; the goal is to utilize the minimum number of sensors as possible [14]. In the latter case, the special targets are not considered, but the determined area must be permanently monitored; the goal is to reduce the view overlap associated

to different sensors to extend network lifetime [15]. This paper's concentration is on target coverage and network lifespan maximization. Several exact and heuristics were proposed by Cerulli et al. in 2012 to solve network lifetime extension in wireless sensor networks with adjustable ranges [5]. The authors modeled the issue to a constraint programming problem in which its goal was to find different cover sets (CSs) each of which is activated by appropriate activation time. To solve this problem, different heuristics and an exact algorithm which was based on column generation techniques were introduced [5]. In their application, sensors were only adjustable in different ranges, but they dissipate much battery energy because they were utilized to monitor targets with disk-like fashion not sector-like behavior. Some heuristic algorithms have been proposed by Mohamadi et al. to solve *MNLAR* problem [8]. In each heuristic, the criteria for making CS is different such as remaining the battery power of sensors, the criticality circumstance of targets that are the key notes for decision making. Afterwards, all of the key notes are weighted in the greater heuristic. Although the proposed heuristics were promising in their final results, the more improvement is possible for the sake of utilizing non-uniform battery usage especially in the larger search space. The similar heuristic works have been proposed by Jinglan Jian et al. to maximize full-view target coverage in camera sensor networks [1]. Another heuristic algorithm was proposed to deploy sensors in the wireless network to maximize object coverage subject to considering sensor angle and battery limitations as constraints [16]. In this line, some other heuristics were proposed by Tan and Jarvis to make balance between detection and coverage of targets [17]. A hierarchical target-oriented multi-agent coordination framework (HiT-MAC) was presented by Xu et al. in 2020 to maximize target coverage problem [18]. This problem is formulated to integer linear programming (ILP) problem. To solve this combinatorial problem, the two-level heuristic has been extended. A heuristic-based target coverage-aware algorithm was presented in 2019 to figure out the problem of network lifetime maximization regarding to the sensor angles and their battery limitations [19].

As the exact and heuristic algorithms work such as greedy algorithms regarding to a predetermined criteria, they do not have capability to amend solutions toward global optimization. For this reason, they cannot reclaim the rudimentary solutions gained during their approach to reach global optimal. Also, the majority of solutions lead sub-optimal especially in the large-scale search space. To this end, several meta-heuristic solutions have been presented in literature to solve *MNLAR* problem in which their results are relatively better than heuristic ones. Some heuristics and a meta-heuristic algorithm based on SA algorithm have been propounded for presenting energy-efficient coverage-preserving protocol in wireless networks [21]. The main concentration of these proposals were to switch energy consumption mode between high-energy mode and low-energy mode in suitable time to manage network power consumption. Another meta-heuristic-based algorithm, a genetic-based algorithm was devised to maximize the number of CSs in *DSN* networks while it covers all of the targets at the same time [22]. The proposed GA's operators were conducted in such a way that to meet two objectives. Firstly, the network lifetime is possibly increased by making the more number of CSs. Secondly, the remaining battery of sensors are preserved so that the next CS can be constituted [23]. A multi-objective coverage optimization based on memetic algorithm has been presented for directional sensor networks [24]. The main objective was to extend network lifespan. To this end, the algorithm's operators are conducted to use less sensors in each cover set. Another genetic-based algorithm was proposed by Alibeiki et al. in 2019 with

considering adjustable sensor ranges along with view angle direction [4]. The fitness function of their algorithm was to consider minimum power usage of each CS, but not attention to uniformly energy usage between sensors. So, the further improvement is possible especially in the larger search space. A learning automata-based algorithm was presented by Razali et al. to solve priority-based target coverage problem in *DSN* networks with adjustable sensing ranges [25]. The distributed particle swarm optimization (*D-PSO*) algorithm was designed to reach a near to optimal coverage of closed virtual target-barrier determined in directional sensor networks with rotatable sensors [26]. A bat-inspired algorithm was applied for connected target coverage in WSNs [27]. It searches to find appropriate sensors to activate for sensing targets. Based on the first selection nodes the other nodes between the first and the sink are gradually selected.

This review reveals that several proposals have been presented to solve sensor network lifetime maximization along with target coverage. However, there are some shortcomings such as lack of scheduling for adjustable range in the only appropriate sector direction, uniformly battery usage, lack of making balance between exploration and exploitation in search space, neglecting to attention the discrete search space which leads sub-optimal solutions. To fill the gaps, the current paper proposes an advanced discrete cuckoo search algorithm which is aware of aforementioned points. The results gained from extensive simulations prove this claims.

3. Problem Statement

This section presents formal problem statement. Before doing so, at first some notations and definitions are proposed to ease the follow of the current paper. Afterwards, the *MNLAR* issue is formulated to discrete optimization problem.

3.1. Notations and Definitions

In this sub section, the notations and terms which are applied in the proposed model is depicted in Table 1. The *DSN* network is modeled to a network (N) that is a tuple such as $N=(T, S, D, R)$. The T is set of m targets $T=\{t_1, t_2, \dots, t_m\}$, placed in predetermined locations that must be monitored by the set of n sensors $S=\{S_1, S_2, \dots, S_n\}$.

Table 1. Nomenclature used in this paper

Notation	Description	Notation	Description
S	Set of sensors	$T[S_i(j, k)]$	All target which can be observed by adjusted sensor S_i
T	Set of targets	CS_i	i -th cover set
D	Set of possible directions	R	Set of possible ranges
S_i	i -th sensor	t_i	i -th target
m	Number of targets	Δ^y	The amount of battery consumption in range y
n	Number of sensors	UT	The unit time of sensor observing when it consumes battery permanently
d_i	i -th direction of sensor	τ_j	Activation time for utilizing i -th cover set
$MaxDir$	Maximum number of available direction	β_j	Battery level of i -th sensor
r_i	i -th adjustable range	$X_{ij}^{d,r}$	Binary variable which indicates the sensor S_i is wakeup in j -th round
$S_i(j, k)$	i -th sensor in j -th direction with k -th adjustable range	y_j	Binary variable which indicates j -th target is under observing

As mentioned earlier, the number of sensors are more than the number of targets. Each sensor can work only in one of its different angles or directions at a time. Each possible direction is determined by $D = \{d_1, d_2, \dots, d_{MaxDir}\}$ where $d_i = i$. For instance, if the $MaxDir$ parameter is 3, each sensor can only observe one of 120° degree of a 360° circular region. Also, each sensor can be activated only in one of its possible ranges. All possible ranges are in set $R = \{r_1, r_2, \dots, r_{MaxRng}\}$ where $r_i = i$. For instance, the $MaxRng$ parameter associated to sensor depicted in Fig. 1 is normalized to 2 (each range covers 30 meters, c.f. in Table 4). Note that, corresponding to the utilized sensor range, the level of battery energy is consumed. Namely, the greater range is set for a sensor, the more and far coverage are gained; also, the more battery is consumed. The notation $S_i(j, k)$ is used to indicate the sensor S_i 's direction is in j -th angle with k -th adjustable range where $1 \leq j \leq MaxDir$ and $1 \leq k \leq MaxRng$. For the sake of simplicity, we take all of the sensors are homogeneous in architecture and same battery consumption. In this study, the coordination information of sensors and targets are determined in advance. Also, the number and position of sensors and targets do not change during the scheduling process. The notation $T[S_i(j, k)]$ is the subset of all targets that can be covered by sensor S_i when it is set in the j -th working direction and is activated in k -th sensing range that is corresponding to k -th battery usage level. It is clear-cut that for each sensor S_i which is set in the j -th working direction, the inequality $T[S_i(j, l_1)] \subseteq T[S_i(j, l_2)]$ is correct where $1 \leq l_1 \leq l_2 - 1$. The sensor $S_i(j, l_1)$ is optimal for a target t_p if either $l_1=1$, or $t_p \in T[S_i(j, l_1)]$ and $t_p \notin T[S_i(j, l_2)]$ $1 \leq l_2 \leq l_1$. In the other words, the sensor S_i is optima for a target t_p if the target is placed in its working direction and can be sensed by the

minimum adjusting range and battery usage level as possible. The main concentration of the current paper is to constitute cover sets (CS s); each cover set CS_i contains some tuples each of which is an applied sensor specifications in terms of utilized direction along with its adjusted range. A cover set CS_i is calculated by Eq. (1).

$$CS_i = \{S_i(j, k) \mid S_i \in S, 1 \leq j \leq \text{MaxDir} \text{ and } 1 \leq k \leq \text{MaxRng} \} \quad (1)$$

The notation $T[CS_q]$ is used to indicate all of the targets which can be observed by cover set CS_q . The $T[CS_q]$ value can be calculated by Eq. (2). If a cover set CS_q can observe and monitor all of the targets placed in the observation environment, this cover set is called a *cover*. This investigation can be compared by Eq. (2). Formally, if $T[CS_q] = T$ (as set of all targets in the field), then the set CS_q is a *cover*.

$$T[CS_q] = \bigcup_{S_i(j,L) \in CS_q} T[S_i(j, L)] \quad (2)$$

The main objective of the current paper is to constitute different *covers*. Each cover is scheduled for some time. The upper time limit as upper threshold for engaging each *cover* is bounded to the battery level of the weakest sensor in term of its residual battery. To utilize sensors efficiently, the regulator parameter (RP) is also applied which is less than that upper threshold. By exploiting this parameter we can uniformly use overall sensors battery. In each round, the current *cover* is utilized to monitor all of the targets in the field for some time. After that the next *cover* must be made and the time which the *cover* must be in use is determined. This time is named coverage time or activation time by a *cover*. The network lifetime is the sum of coverage time of each *cover* until the last interval when it is impossible to constitute another *cover* owing to sensor battery depletion. The chief objective of the current paper is to extend network lifetime. The determinants are how to utilize sensors in which direction by which adjustable range and with what activation time. The wrong selection of each will take the solution's trajectory far from optimal ones. The reason why this paper presents an intelligent scheduling algorithm for selecting optimal adjustable range and direction associated to each sensor in a *cover* along with engagement of them in appropriate coverage time. To normalize the power consumption of each sensor, the notation Δ^y is defined; it means the power consumption ratios between when sensor adjusting range is in y and the least possible range that is 1. For instance, the $\Delta^y=3$ means that the power consumption of the sensor when it is scheduled in y -th range is 3 times more than that of activated in the first level (first possible range). It is clear when the sensor range adjusted in the lowest radius as possible (in the first radius), the power consumption is $\Delta^1=1$ for one unit of time (UT). Then, the power consumption of sensors in different adjusted ranges are normalized in comparison with Δ^1 which is 1. For example, if the sensor S_i is adjusted in the second range, it keeps observing continuously only in $\frac{UT}{3}$ of time since $\Delta^2= 3$.

Fig. 2. A typical directional sensor network (DSN) with 5 sensor and 5 targets

Fig. 2 depicts an illustrative example where in the sensing field there are 5 homogeneous sensors are observing 5 different targets. Each sensor has 3 possible directions d_1 , d_2 , and d_3 along with 2 possible adjustable ranges, namely, 1 and 2. Take $\Delta^1=1$ and $\Delta^2=2$ in this network. In Fig. 2, four sets CS_1 , CS_2 , CS_3 , and CS_4 are four possible *covers* since $T[CS_i] = \{t_1, t_2, t_3, t_4, t_5\} = T$, where $i=1,2,3,4$. The four *covers* CS_1 , CS_2 , CS_3 , and CS_4 are calculated by Eqs. (3-6) respectively.

$$CS_1 = \{S_1(1,1), S_2(2,1), S_3(2,1), S_4(1,2), S_5(1,2)\} \quad (3)$$

$$CS_2 = \{S_1(1,2), S_3(2,1), S_4(1,2), S_5(1,2)\} \quad (4)$$

$$CS_3 = \{S_1(1,2), S_3(2,1), S_4(1,2), S_2(3,2)\} \quad (5)$$

$$CS_4 = \{S_1(1,2), S_3(2,1), S_5(2,2), S_2(3,2)\} \quad (6)$$

At the outset, the energy level of each sensor is normalized to 1 as for full battery level ($\beta_i=1$ for all sensors). Note that, the battery remaining of a sensor S_i in a *cover* after its deployment in time interval τ_j as activation time is measured by Eq. (7); the term Δ^r is the amount of battery usage when the sensor is adjusted in r -th range.

$$\beta_i = \beta_i - \Delta^r \times \tau_j \quad (7)$$

Regarding to Fig. 2, if each *cover* is engaged for 0.5 UT to observe all of the targets, the network lifetime would be 0.5 UT. After that time, it is impossible to create another *cover*. For example, if the *cover* CS_2 is continuously utilized for monitoring all of the targets in the field for 0.5 UT duration, the sensors S_1 , S_4 , and S_5 are depleted after this interval. Then, it is impossible to make

another cover although the sensors S_2 and S_4 were *sleep* mode during the activation time to save energy. So, the network lifetime is 0.5 UT. This situation is happened for other *covers*. Here, we show that how the intelligent scheduling and appropriate time activation can extend network lifespan. If the intelligent scheduler, schedule five *covers* C_1 through C_5 from possible cover sets CS_1 , CS_2 , CS_3 , and CS_4 in the sequence $C_1 = CS_3$, $C_2 = CS_2$, $C_3 = CS_1$, $C_4 = CS_3$, $C_5 = CS_4$ each of which is activated for 0.125 UT; then, the network lifetime is extended to 1.125 UT. In fact, here the *RP* parameter was used that is 0.125 UT. After 5 switching between different *covers*, three sensors S_1 , S_4 and S_5 are depleted. In the other words, the next *cover* cannot be constituted in the sixth round. This simple example proves the intelligent scheduler improves network lifetime 225% in comparison with naïve procedure in this case study.

3.2. Problem Formulation

This section formulates network lifespan maximization with target coverage issue to a discrete optimization problem subject to some constraints. This is an NP-Hard problem. The main objective function is modeled in Eq. (8) where the constraints are formulated in Eqs. (9-11).

$$\left\{ \begin{array}{l} \text{Max } f(t) = \sum_{j=1}^{NCovers} \tau_j \quad (8) \\ \text{Subject to :} \\ \sum_{i=1}^m \sum_{j=1}^{NCovers} \sum_{r=1}^{MaxRng} X_{ij}^{d,r} \times \Delta^r \times \tau_j \leq \beta \quad \text{where } d \in \{1, 2, \dots, MaxDir\}, X_{ij}^{d,r} \in \{0, 1\} \quad (9) \\ \sum_{j=1}^n y_j = n, \text{ where } y_j \in \{0, 1\} \quad (10) \\ \tau_i \geq 0, \forall i = 1, 2, \dots, NCovers \quad (11) \end{array} \right.$$

In this optimization model, the term τ_j is the activation time of each *cover* that is utilized to monitor all targets placed in the observing filed in j -th round. Then, the main objective function is to maximize the sum of all activation times which Eq. (8) indicates. In this model, some constraints are to be met. The nested sigma in Eq. (9) implies that each sensor can observe the environment till its battery power is depleted. The parameter β is used to show the maximum amount of battery that a sensor can use up. In this normalization form, β is normalized to 1. The binary decision variable $X_{ij}^{d,r}$ is applied to imply that sensor S_i is activated only in its d -th direction with r -th adjusting range for j -th round (sensor S_i is activated in j -th *cover* that is utilized in j -th round); in this case, the decision variable $X_{ij}^{d,r}$ is set to 1 otherwise it is considered 0. Note that, when the sensor's range is adjusted to r -th possible range, the amount of battery consumption is Δ^r in its activation time. The term $\Delta^r \times \tau_j$ dubs the amount of sensor battery consumption in τ_j interval. The Eq. (10) emphasizes that all of the targets must be covered. To this end, the binary variable y_i is dedicated for. If the target T_j is under coverage, then the binary variable y_i is set to 1 otherwise it is considered 0. The variable $NCovers$ is the number of *covers* that can be constituted.

4. Proposed Discrete Cuckoo Search Algorithm for Lifetime Maximization in DSN

This section presents the proposed meta-heuristic algorithm which solves lifetime maximization problem in DSN networks. Since the search space is intrinsically discrete, the well-reputed continuous optimization algorithms such as PSO, GWO, WOA, etc. do not yield efficient solutions even from their discrete versions because their discretized solutions are either inefficient or meaningless regarding to the stated problem of this paper. On the other hand, the majority of meta-heuristic algorithms get stuck in local optimum. For aforementioned reasons, the cuckoo search algorithm is selected because it evades from local optimum trap similar to simulated annealing (SA) algorithms, but it has inclination to global optimization in contrast to SA. In the other words, it examines even bad solutions to find new rooms for both evading from local optimal trap and finding global optimal. The canonical Cuckoo Search Algorithm (CSA) is drawn in Algorithm 1 [13].

Algorithm 1. Cuckoo Search Algorithm (CSA)
1: Objective function for d -dimensional space $f(x)$, $x=(x_1, x_2, \dots, x_d)^T$ 2: Produce initial population of n host nests x_i where $i=1,2,\dots,n$ 3: While ($t < \text{MaxGeneration}$) or (stop criteria) Do 4: Get a cuckoo randomly by Levy flights 5: Evaluate the quality of this solution by fitness F_i 6: Select a random nest among n , call it j 7: if $F_i > F_j$ then 8: Replace j by the new solution 9: end-if 10: A fraction (P_a) of the worst solutions are left and the new ones are built 11: Keep the best solutions 12: Rank the solutions based on the fitness value and find the current best 13: end-while 14: return the best solution

One of the most important features of cuckoo search algorithm is to apply *levy flight* concept. It is beneficial to pave the way of trajectory toward exploring the search space uniformly. Engaging the continuous *levy flight* and discretizing the gained solution lead inefficient solution or meaningless regarding to our stated problem. To assimilate this point in discrete search space, diverse exploring procedures are defined each of which is randomly called to make the sense of randomness and *levy flight* concept in its discrete version. In the next, two subsections are dedicated to explain about the proposed algorithm. The first subsection defines problem encoding, fitness function, and termination criteria. The second subsection clarifies the proposed algorithm in detail.

4.1.Preliminaries: encoding, fitness, termination criteria

One of the most important parts of each meat-heuristic algorithm is how to encode the stated problem (or phenotype) in artificial intelligence domain (or genotype). The encoding phase has significant impact on overall cost especially its encoded problem length. For this reason, the problem is encoded such as in Refs. [4, 28] to be as short as possible. In cuckoo search context, each candidate solution is modeled to an *egg*. Each *egg* is encoded such a chromosome in GA. Regarding to Fig. 2, the intention is how to construct a candidate solution (*egg*). Here, each

candidate solution is a candidate *cover*. To present an efficient encoded solution, the target is in preference instead of utilized sensors because the number of utilized sensors depending on their adjusted ranges and residual battery energy may differ in each *cover*, but the number of targets are constant in each *cover*. To construct an *egg*, each target must be monitored by an adjusted sensor. Also, it may be possible to monitor other targets by the current engaged sensor this is the reason the number of utilized sensors in a *cover* differs in comparison with other *covers*. For instance, the encoded solution of *cover* CS_3 of *DSN* of Fig. 2 is depicted in Fig. 3.

Target	t_1	t_2	t_3	t_4	t_5
Utilized Sensor	[1,1,2]	[1,1,2]	[3,2,1]	[4,1,2]	[2,3,2]

Fig. 3. Solution encoding for cover

The notation [2,3,2] in t_5 's column means that the sensor S_2 is activated in its first direction by the second adjusted range to monitor target t_5 . During the algorithm process, there may be a possibility to have impairment in encoded solution. For instance, the encoded solution may have one activated sensor in more than one direction or the sensor is activated in more than one range; all of mentioned circumstances are impossible. Therefore, the encoded solution is considered impaired and must be repaired.

The fitness function is very important part of each meta-heuristic algorithm because it directly steers solutions toward the stated problem's objective functions. So, it needs intricate fitness function to gain promising and efficient solutions. In the stated problem of this paper, selection of the best effective *cover* must has the lowest sum of battery consumption in a *cover*; in the other side, it needs taking uniform usage of sensors' battery because the unbalanced battery usage may cause the situation in near future so that all sensors with high rate of battery remaining cannot have full target coverage owing to some sensors battery depletion. As mentioned earlier, the objective function is to extend network lifetime. This is done by considering lower battery consumption in a *cover*. Eq. (12) calculates the energy usage of each $Cover_p$.

$$E(Cover_p) = \sum_{j: S_j(i, L_j) \in Cover_p} \Delta^{L_j} \quad (12)$$

It indicates that the *cover* among several *covers* which consumes the lowest sum of battery usage is the most favorable. If two different *covers* in *egg* population have the same amount of fitness value based on Eq. (12), the distribution factor (*DF*) is defined and applied in fitness function. This parameter is used to show in how extent the *cover* uses up battery uniformly. The *DF* value is calculated by Eq. (13).

$$DF(Cover_p) = \frac{\sum_{q=1}^m \sum_{j: S_j \in NS(t_q)} \Delta^{L_j}}{m} \quad (13)$$

Where $NS(t_q)$ is the set of sensors of a candidate *cover* that can observe target t_q ; it is calculated by Eq. (14).

$$NS(t_q) = \{S_j(i, L_j) \in Cover_p \mid t_q \in T[S_j(i, L_j)]\} \quad (14)$$

The parameter L_j is the S_j 's adjusted sensor range in $Cover_p$ which can cover target t_q . Succinctly, the parameter DF indicates the average battery usage for each target. So, the *cover* which has the lower DF value is favorable. For instance, if two targets can be monitored by either one sensor in wider adjusting range or two sensors with lower adjusting ranges, the second option is more efficient because this method uniformly uses up the whole battery and implicitly extends network lifetime. With closer look on the covers made in Fig. 2, all of the covers CS_1 , CS_2 , CS_3 , and CS_4 have the fitness values equal to 7 respectively based on Eq. (12), but in term of DF parameter in Eq. (13) the *cover* CS_1 is the most efficient because it utilizes more sensors with lower adjusted ranges instead of the less number of sensors with high adjusted ranges which indirectly preserves sensors life.

Therefore, the final fitness function for each *cover* is calculated via Eq. (15) where coefficient ω_1 and ω_2 are used to indicate the importance of each determinant. In this regards, importance of both determinants are considered the same, namely, $\omega_1 = \omega_2 = 0.5$. In addition, both $DF(.)$ and $E(.)$ values are normalized in the same range.

$$Fitness(Cover_p) = \omega_1.DF(Cover_p) + \omega_2.E(Cover_p) \quad (15)$$

Therefore, the fitness function returns sum of battery usage by sensors in a *cover* along with considering the lower DF value which indirectly uses up battery uniformly. In the other words, a *cover* that uses up the minimum battery power along with low ranges is returned in each round.

The termination criteria of the main algorithm is to reach the point so that it is not possible to construct new *cover* for the sake of insufficient remaining battery power. So, it iterates until it cannot make new *cover*. In each iteration, the meta-heuristic algorithm is run *MaxGen* times for maximum generations times; then a *cover* along with its activation time is returned.

For initial population, half of the solutions are created in such a way the target that is in the hazard situation is selected with high priority in process of constructing a solution. The hazard target is a target which sum of the remaining battery associated to the only sensors monitoring that target is the least amount. Therefore, the sensor that must monitor the hazard target must be determined optimally. This point is applied in semi-random population generation. The rest solutions are randomly generated.

4.2. Main Algorithm Presentation and Description

The main algorithm which solves *MNLAR* problem is Algorithm 2. Algorithm 2 is a discrete cuckoo search algorithm for solving *MNLAR* problem. It inputs the *DSN* information in terms of targets & sensors specifications and algorithm settings. Then, it returns the optimal scheduling solution which is a set of several tuples (*cover* no., *cover*, activation time) for each *cover*. The main proposed algorithm for solving *MNLAR* problem is elaborated and depicted in Algorithm 2.

Algorithm 2. Discrete Cuckoo Search Algorithm for *MNLAR* Problem (*D-CSA-MNLAR*)**Input:**

$N = \langle T, S, D, R \rangle$ as network specification
 n as number of sensors;
 m as number of targets;
 $MaxDir$ as maximum number of sensor directions;
 $MaxRng$ as maximum number of sensor adjustable ranges;
 information of target positions and coordination;
 information of sensors' location and coordination;
 β_i as the battery level of each homogeneous sensor S_i ;
 $MaxGen$ as maximum generation;
 $PopSize$ as size of cuckoo population
 P_α as the fraction of *inferior* solutions in the cuckoo population

Output:

The optimal scheduling including different *covers* along with their activation time and network's maximum lifespan as well

```

1: OptimalSchedulingSolution  $\leftarrow \phi$  ;
2: Lifespan  $\leftarrow 0$  ;
3: NCover  $\leftarrow 0$  ; (* number of rounds or number of covers made so far *)
4: For  $i \leftarrow 1$  To  $n$  Do
5:    $\beta_i \leftarrow 1$  ; (* The initial battery power associated to each sensor's normalized to 1 *)
6: End-For
7: While  $\cup_{S_i(j,L) \in S, j \in \{1, \dots, MaxDir\}, L \in \{1, \dots, MaxRng\}} T[S_i(j, L)] = T$  Do
8:   Initialize the cuckoo nest population in which every nest is a candidate solution
9:   Calculate the fitness value for each nest in population based on Eq. (15).
10:  Rank the entire population based on fitness value
11:  For  $G \leftarrow 1$  to  $MaxGen$  Do
12:    Partition the population into inferior and superior nests based on fitness value respectively by  $P_\alpha$  and  $(1 - P_\alpha)$  probability
13:    For all pair of solutions ( $Sol_i, Sol_j$ ) in inferior nests Do
14:      [ $Sol_p, Sol_q$ ]  $\leftarrow$  DiscreteLevyFlight ( $Sol_i, Sol_j, Population, PopSize$ ) based on Algorithm 3
15:       $Sol_i \leftarrow Sol_p$ 
16:       $Sol_j \leftarrow Sol_q$ 
17:       $Fitness(Sol_i) \leftarrow Fitness(Sol_p)$ 
18:       $Fitness(Sol_j) \leftarrow Fitness(Sol_q)$ 
19:    End-For-inferior
20:    Again, Partition the current updated population into inferior and superior nests
21:    For all pair of solutions ( $Sol_i, Sol_j$ ) in superior nests Do
22:      [ $Sol_p, Sol_q$ ]  $\leftarrow$  DiscreteLevyFlight ( $Sol_i, Sol_j, Population, PopSize$ ) based on Algorithm 3
23:      Pick two random solutions (random nests, call them  $Sol_{r1}$  and  $Sol_{r2}$ ) from the whole population
24:      If  $Fitness(Sol_p)$  is better than  $Fitness(Sol_{r1})$  Then
25:         $Sol_{r1} \leftarrow Sol_p$ 
26:         $Fitness(Sol_{r1}) \leftarrow Fitness(Sol_p)$ 
27:      End-if
28:      If  $Fitness(Sol_q)$  is better than  $Fitness(Sol_{r2})$  Then
29:         $Sol_{r2} \leftarrow Sol_q$ 
30:         $Fitness(Sol_{r2}) \leftarrow Fitness(Sol_q)$ 
31:      End-if
32:    End-For-superior
33:  End-For-Generation
34:  Rank solutions in Population and determine the best cover so far as  $Best_{Cover}$ 
35:   $AT \leftarrow$  Set appropriate activation time for  $Best_{Cover}$  solution based on Eq. (16)
36:   $Lifespan \leftarrow Lifespan + AT$ ;
37:   $NCover \leftarrow NCover + 1$ ; (* The new cover has been created *)
38:  OptimalSchedulingSolution  $\leftarrow OptimalSchedulingSolution \cup \{ (NCover, Best_{Cover}, AT) \}$ 
39:  (* Battery update *)
40:  For  $i \leftarrow 1$  to  $n$  Do
41:    If the sensor  $S_i$  is activated in this current round Then
42:       $\beta_i \leftarrow \beta_i - X_{iNCover}^{d,r} \times \Delta^r \times AT$ 
43:      If  $\beta_i$  is not sufficient to engage sensor  $S_i$  Then
44:        Dismiss the sensor  $S_i$  from  $S$  that is the set of sensors
45:    End-if

```



```

46:   End-if
47: End-For-each-sensor
48: End-While
49: Return OptimalSchedulingSolution, Lifespan, and NCover
50: End-Algorithm 2

```

Algorithm 2 is iterated till the remaining battery of available sensors do not suffice to make another *cover*. This iteration is between lines 7 through 48. The number of rounds that the algorithm can make new *cover* is then specified by *NCover* variable which is not determined in advance. In the main loop, an optimal *cover* is found to be added into rudimentary scheduling. This optimal *cover* is made by running lines 11 to 33. Before doing so, the initial population is created and all of the fitness values are calculated for each solution (each *egg*). Then, the ranking of *eggs*, representative of candidate solutions, are done based on determined fitness values. In the outset of *For*-loop which creates an optimal *cover*, the solutions are partitioned into two nest categories *superior* and *inferior* in line 12. The term P_α determines the fraction of population belonging to *inferior* category. The rest that are $(1-P_\alpha)$ belongs to *superior* category. The prominent point of the proposed algorithm revolves around two parts. At first, for each solution in *inferior* nests the *DiscreteLevyFlight* algorithm is called to explore discrete search space uniformly. To do so, handful discrete walking around procedures are defined each of which is randomly called. By applying of one this procedure, the new solution is created. This new solution is substituted by the old one. This part works such as the SA algorithm does. In fact, it experiences the bad solutions to evade from getting stuck in the local optimum, but there may be a concern in which the new generated solutions worsen the objective function. This concern can be obviated in the second part because in the second part, for each solution in the *superior* nests the *DiscreteLevyFlight* algorithm is called; then, the new generated solution is compared with one of the random solution selected from the entire population. If the new generated solution is better than the selected random solution, that random solution is abandoned from the population and the new solution is placed instead. In the second part it is evolution-based in which the solutions are conducted in global optimum. This way is to make a good balance between exploration and exploitation in search space. After the execution of several generations, the best so far solution, $Best_{Cover}$, is opted from ranking of solutions based on their fitness values in the current population which line #34 of Algorithm 2 shows. In line #35, the upper threshold time for activation time (*AT*) of best *cover* so far, $Best_{Cover}$, is determined. This value depends on the weakest sensor in term of its remaining battery in the gained *cover*. This activation time can be obtained via Eq. (16).

$$AT = \min \left\{ \frac{\beta_j}{\Delta^{r,j}} \mid S_j \in Best_{Cover} \right\} \quad (16)$$

This activation time is added to the *lifespan* variable which determines the life time of the network. The variable *NCover* that indicates the number of made *covers* so far is added by one. Then, the tuple (*NCover*, $Best_{Cover}$, *AT*) is added to *OptimalSchedulingSolution*. One of the most prominent part of cuckoo search algorithm is how to apply *Levy flight* procedure. To do so, the novel *DiscreteLevyFlight* algorithm is called for both *inferior* and *superior* nests. In contains handful walking around procedures to explore discrete search space uniformly and efficiently. These procedures are named *Crossover-one-point*, *Crossover-two-point*, *Shuffle-Odd-Even*, and

Mutation-two each of which is called randomly because the unbiased artificial algorithm yields more efficient results. This *Levy flight* procedure is conducted in such a way that it inputs a pair solutions; then, it returns a new pair solutions. All of walking around procedures act by changing the input solutions. If during the changes the new solutions are infeasible, the minor modification converts them to make feasible solutions. The infeasibility of solutions mainly backs to the fact that each sensor is utilized in more than one direction or by more than one adjusted radius. The novel *DiscreteLevyFlight* procedure is elaborated in Algorithm 3.

Algorithm 3. *DiscreteLevyFlight* procedure

Input:

Population : all of the solutions;
PopSize as size of cuckoo population;
Sol_i, *Sol_j* : two consecutive solutions in *Population*;
n as number of sensors;
m as number of targets;
 P_{α} as the fraction of *inferior* solutions in the cuckoo population

Output:

Sol_p, *Sol_q* : two new solutions;

```

1: Draw a random integer  $R \in [1..4]$  for calling one of four walking around procedures randomly ;
2: if  $R = 1$  then
3: (* Crossover-one-point is performed *)
4: Draw another random integer  $Rn \in [1..m]$  in which the parameter  $m$  is the size of encoded solution
5:  $Temp1 \leftarrow$  Concatenate ( $Sol_i[1..Rn]$ ,  $Sol_j[Rn+1..m]$ )
6:  $Temp2 \leftarrow$  Concatenate ( $Sol_j[1..Rn]$ ,  $Sol_i[Rn+1..m]$ )
7:  $Sol_p \leftarrow$  Check&Correct ( $Temp1$ )
8:  $Sol_q \leftarrow$  Check&Correct ( $Temp2$ )
9: else if  $R = 2$  then
10: (* Crossover-two-point is performed *)
11: Draw two random integers  $Rn1$  and  $Rn2$  whereas  $1 \leq Rn1 < Rn2 \leq m$  and both of them belong to  $[1..m]$ ;
12:  $Temp1 \leftarrow$  Concatenate ( $Sol_i[1..Rn1]$ ,  $Sol_j[Rn1+1..Rn2]$ ,  $Sol_i[Rn2+1..m]$ )
13:  $Temp2 \leftarrow$  Concatenate ( $Sol_j[1..Rn1]$ ,  $Sol_i[Rn1+1..Rn2]$ ,  $Sol_j[Rn2+1..m]$ )
14:  $Sol_p \leftarrow$  Check&Correct ( $Temp1$ )
15:  $Sol_q \leftarrow$  Check&Correct ( $Temp2$ )
16: else if  $R = 3$  then
17: (* Shuffle-Odd-Even is performed *)
18:  $Temp1 \leftarrow Sol_i$  ;
19:  $Temp2 \leftarrow Sol_j$  ;
20: For  $K \leftarrow 1$  to  $m$  Do step by +2
21:    $Temp1[K] \leftarrow Sol_j[K]$  ;
22:    $Temp2[K+1] \leftarrow Sol_i[K+1]$  ;
23: End-For
24:  $Sol_p \leftarrow$  Check&Correct ( $Temp1$ )
25:  $Sol_q \leftarrow$  Check&Correct ( $Temp2$ )
26: else (* $R = 4$ *)
27: (* Mutation-two is performed *)
28: Draw two random integers  $Rn1$  and  $Rn2$  both of them belong to  $[1..m]$ ;
29:  $Temp1 \leftarrow$  Change  $Sol_i$  solution by modification of  $Sol_i[Rn1]$ 's meme with a valid meme ;
30:  $Temp2 \leftarrow$  Change  $Sol_j$  solution by modification of  $Sol_j[Rn2]$ 's meme with a valid meme ;
31:  $Sol_p \leftarrow$  Check&Correct ( $Temp1$ )
32:  $Sol_q \leftarrow$  Check&Correct ( $Temp2$ )
33: end-if
34: Return  $Sol_p$  and  $Sol_q$  ;
35: End-Algorithm 3

```

When *DiscreteLevyFlight* procedure is called, at its first line a random integer number from [1..4] interval is drawn in which the number is respectively for calling *Crossover-one-point*, *Crossover-two-point*, *Shuffle-Odd-Even*, and *Mutation-two* procedures. One important thing to mention is that all of the walking around procedures in *levy flight* are conducted in such a way to decrease the exploring costs. For this end, regarding to the style of problem encoding in which for each corresponding target, set of sensors can cover that target; the reason why operators of proposed walking around algorithms utilize corresponding gene from other side's solution. To do so, a pair of solutions creates new pair of solutions. After applying each of walking around algorithms, the new born solutions may violate the rules; for this reason, after each production, the *Check&Correct(.)* procedure is called to repair impaired solutions. The fault gene in a solution can be simply replaced with a valid gene.

If the drawn random integer is one, the *Crossover-one-point* procedure is called that is similar to crossover of genetic algorithm. Fig. 4 depicts two input solutions Sol_i and Sol_j for *Crossover-one-point* procedure by considering crossover single point on 3. After one-point crossover is performed two solutions $Temp_1$ and $Temp_2$ are created in which the first one is invalid. Fig. 5 illustrates two new generated solutions. The solution $Temp_1$ is infeasible because the sensor S_3 cannot be activated in two directions at the same time. Afterward, the *Check&Correct(.)* procedure is called to repair the contingent impaired solution. Therefore, a candidate sensor such as S_2 for monitoring of target t_5 is randomly added to make feasible and valid solution. The valid solutions are shown in Fig. 6.

Sol_i						Sol_j					
Target	t_1	t_2	t_3	t_4	t_5	Target	t_1	t_2	t_3	t_4	t_5
Utilized Sensor	[1,1,2]	[1,1,2]	[3,2,1]	[4,1,2]	[5,1,2]	Utilized Sensor	[1,1,2]	[1,1,2]	[2,1,2]	[4,1,2]	[3,3,2]

Fig. 4. Input Solutions Sol_i and Sol_j for *Crossover-one-point* procedure by taking point 3 as cross point

$Temp_1$ (invalid solution)						$Temp_2$ (valid solution)					
Target	t_1	t_2	t_3	t_4	t_5	Target	t_1	t_2	t_3	t_4	t_5
Utilized Sensor	[1,1,2]	[1,1,2]	[3,2,1]	[4,1,2]	[3,3,2]	Utilized Sensor	[1,1,2]	[1,1,2]	[2,1,2]	[4,1,2]	[5,1,2]

Fig. 5. Two new semi-perfect generated Solutions

valid solution Sol_p						valid solution Sol_q					
Target	t_1	t_2	t_3	t_4	t_5	Target	t_1	t_2	t_3	t_4	t_5
Utilized Sensor	[1,1,2]	[1,1,2]	[3,2,1]	[4,1,2]	[2,3,2]	Utilized Sensor	[1,1,2]	[1,1,2]	[2,1,2]	[4,1,2]	[5,1,2]

Fig. 6. Two new checked and corrected solutions

The same behavior is done for *Crossover-two-point* procedure. Figs. (7-9) are dedicated for illustrative example of this procedure by considering 1 and 3 as cross points.

Sol_i						Sol_j					
Target	t_1	t_2	t_3	t_4	t_5	Target	t_1	t_2	t_3	t_4	t_5
Utilized Sensor	[1,1,2]	[1,1,2]	[3,2,1]	[4,1,2]	[5,1,2]	Utilized Sensor	[1,1,2]	[1,1,2]	[2,1,2]	[4,1,2]	[3,3,2]

Fig. 7. Input Solutions Sol_i and Sol_j for *Crossover-two-point* procedure by taking 1 and 3 as cross points

$Temp_1$ (valid solution)						$Temp_2$ (invalid solution)					
Target	t_1	t_2	t_3	t_4	t_5	Target	t_1	t_2	t_3	t_4	t_5
Utilized Sensor	[1,1,2]	[1,1,2]	[2,1,2]	[4,1,2]	[5,1,2]	Utilized Sensor	[1,1,2]	[1,1,2]	[3,2,1]	[4,1,2]	[3,3,2]

Fig. 8. Two new semi-perfect generated Solutions

valid solution Sol_p						valid solution Sol_q					
Target	t_1	t_2	t_3	t_4	t_5	Target	t_1	t_2	t_3	t_4	t_5
Utilized Sensor	[1,1,2]	[1,1,2]	[2,1,2]	[4,1,2]	[5,1,2]	Utilized Sensor	[1,1,2]	[1,1,2]	[2,1,2]	[4,1,2]	[3,3,2]

Fig. 9. Two new checked and corrected solutions

The solution $Temp_2$ is infeasible because the sensor S_3 cannot be activated in two directions at the same time (see target t_5). Afterward, the *Check&Correct(.)* procedure modifies it to the accurate solution by engaging random sensor $S_2(1,2)$ instead of $S_3(2,1)$ for observing the target t_3 .

In this line, Fig. 10 depicts two input solutions Sol_i and Sol_j for *Shuffle-odd-even* procedure that is relevant to when Algorithm 3 draws number 3. The *Shuffle-odd-even* procedure shuffles between two solutions in which it takes odd genes from itself and takes corresponding even genes from other side. After doing so over two solutions of Fig. 10, two new solutions are created, but both of them are invalid that Fig. 11 illustrates. However, by incorporating minor changes, new valid solutions are produced. For instance, $Temp_1$ is invalid because both sensors S_1 and S_5 have been utilized in two different directions. With closer look at when $S_1(1,2)$ is utilized for target t_2 , the simple changes $S_1(1,1)$ for target t_1 to $S_1(1,2)$ can cover both targets t_1 and t_2 at the same time so that it does not need additional sensor. For another target t_5 , the impossible $S_5(1,2)$ is substituted by adjusted $S_2(3,2)$. Therefore, the valid solutions Sol_p and Sol_q are created with the same behavior that are depicted in Fig. 12.

Sol_i						Sol_j					
Target	t_1	t_2	t_3	t_4	t_5	Target	t_1	t_2	t_3	t_4	t_5
Utilized Sensor	[1,1,1]	[2,2,1]	[3,2,1]	[4,1,2]	[5,1,2]	Utilized Sensor	[1,1,2]	[1,1,2]	[2,1,2]	[5,2,2]	[3,3,2]

Fig. 10. Input Solutions Sol_i and Sol_j for *Shuffle-odd-even* procedure

$Temp_1$ (invalid solution)						$Temp_2$ (invalid solution)					
Target	t_1	t_2	t_3	t_4	t_5	Target	t_1	t_2	t_3	t_4	t_5
Utilized Sensor	[1,1,1]	[1,1,2]	[3,2,1]	[5,2,2]	[5,1,2]	Utilized Sensor	[1,1,2]	[2,2,1]	[2,1,2]	[4,1,2]	[3,3,2]

Fig. 11. Two new semi-perfect generated solutions

valid solution Sol_p						valid solution Sol_q					
Target	t_1	t_2	t_3	t_4	t_5	Target	t_1	t_2	t_3	t_4	t_5
Utilized Sensor	[1,1,2]	[1,1,2]	[3,2,1]	[5,2,2]	[2,3,2]	Utilized Sensor	[1,1,2]	[1,1,2]	[2,1,2]	[4,1,2]	[3,3,2]

Fig. 12. Two new checked and corrected solutions

The last walking around algorithm of *levy flight* concept is the *Mutation-two* procedure; it is called when Algorithm 3 draws 4 as random integer. Figs. (13-14) depict the illustrative example of the application of *Mutation-two* procedure. In the Fig. 13, a pair of solutions are given which was taken from either *inferior* or *superior* nests. Then, a random gene is selected for each (a sensor that can cover a target) and is substituted by another possible gene (another adjusted sensor that can cover the same target). For instance in solution Sol_i , the third gene of this solution is selected that is for monitoring of target t_3 . This gene, $S_3(2,1)$, is substituted by another possible adjusted sensor $S_2(1,2)$ which is randomly selected. If the invalid solution is gained, the *Check&Correct(.)* procedure modifies this invalid solution to a valid one similar to previous examples. Fortunately, both new generated solutions are valid which are depicted in Fig. 14.

Sol_i						Sol_j					
Target	t_1	t_2	t_3	t_4	t_5	Target	t_1	t_2	t_3	t_4	t_5
Utilized Sensor	[1,1,2]	[1,1,2]	[3,2,1]	[4,1,2]	[5,1,2]	Utilized Sensor	[1,1,2]	[1,1,2]	[2,1,2]	[5,2,2]	[3,3,2]

Fig. 13. Input Solutions Sol_i and Sol_j for *Mutation-two* procedure

valid Sol_p						valid Sol_q					
Target	t_1	t_2	t_3	t_4	t_5	Target	t_1	t_2	t_3	t_4	t_5
Utilized Sensor	[1,1,2]	[1,1,2]	[2,1,2]	[4,1,2]	[5,1,2]	Utilized Sensor	[1,1,2]	[1,1,2]	[2,1,2]	[4,1,2]	[3,3,2]

Fig. 14. Two feasible solutions

4.3 Time Complexity of the proposed algorithm

To measure the time complexity of the main proposed algorithm that is Algorithm 2, the effective statements must be taken into consideration. The sub-procedures of *levy flight* in Algorithm 3 have at most $O(m)$ where the parameter m is the length of encoded solutions or the number of targets in the observing field because all of the walking around procedures *Crossover-one-point*, *Crossover-two-point*, *Shuffle-Odd-Even*, and *Mutation-two* take at most $O(m)$. Therefore, the Algorithm 3 takes $O(m)$ as time complexity. Algorithm 2, the main proposed algorithm, iterates $NCover$ times which *while*-loop does. In each loop, the main cuckoo search is executed to generate an optimal cover; then, it is added to rudimentary scheduling. This part is iterated $MaxGen$ times. The *levy*

flight walking around procedures are called in this loop. Thus, its complexity must be multiplied by $O(m)$. In addition, after each cover is constituted, remaining battery of all sensors must be checked; therefore the $O(n)$ should be added to the complexity in this inner loop where the parameter n is for the number of available sensors. Overall, the time complexity of proposed Algorithm 2 is $O(N_{Cover} \times (MaxGen \times m + n))$ which is a rational cost.

5. Performance Evaluation

This section presents performance assessment of proposed discrete cuckoo search optimization algorithm (*D-CSA*) which solves *DSN* lifespan maximization problem. It is compared against other state-of-the-arts in term of network lifetime which is the most prominent evaluation metric in this field.

5.1. Experimental Settings

To have better performance assessment, several scenarios that are conducted in two directions are presented. At first the number of targets is fix and the number of sensors in observing field is gradually increased; these are scenarios 1 through 3 drawn in Table 2. At second, the number of sensors is fix and the number of targets is gradually increased; these are scenarios 3 through 5 in Table 2. For proposal scalability testing, the sixth scenario which significantly increases both number of targets and sensors is considered. Table 2 draws considered scenarios.

Table 2. Scenarios Description

Scenario no.	Number of Sensors	Number of Targets
1	30	20
2	40	20
3	50	20
4	50	30
5	50	\ 40
6	100	50

As discussed in related work section, brilliant papers in this domain were selected to be compared with current proposal. The selected papers are based on one heuristic, two meta-heuristics, and one hybrid meta-heuristic approaches. After examination in some scenarios by miscellaneous algorithms, the most efficient approaches have been selected for each category. In this regard, a heuristic which integrates some greedy algorithms to a weighted objective function is selected; this is called heuristic *MNLAR (H-MNLAR)* [8]. Other comparative algorithms which are meta-heuristic-based are genetic algorithm (*GA*) and ant colony optimization (*ACO*) algorithm. The efficient *GA* algorithm in the stated problem domain has been selected from [4]. Since a coverage problem is one of the most critical issues in wireless sensor networks domain, this issue has been formulated into an optimization problem and was solved by an ant colony optimization for sweep coverage (*ACOSC*) problem [29]; this algorithm is customized based on the circumstance of the current paper. In the last, a hybrid discrete PSO (*HDPSO*) algorithm has been selected and customized for the stated problem [30]. The reason behind it revolves around the fact that the stated problem is discrete in nature; so, the fast, discrete-based, and at the same time a hybrid algorithm which balances exploration and exploitation in search space is favorable the reason why *HDPSO* algorithm has been selected from literature to be customized based on the stated problem. Table 3 shows parameter settings of comparative algorithms. All of the scenarios are executed in the same circumstance to have a fair and efficient comparison.

Table 3. Parameter Settings of Comparative Algorithms

Algorithms	Specific parameters	Population /Swarm Size	Max Iterations/ Generations
H-MNLAR [8]	$\gamma_1 = \gamma_2 = \gamma_3 = 0.33$, other parameters are drawn from dataset	NA	NA
GA-based [4]	Crossover Percentage: 70% Mutation Percentage: 5%	50~150 depending on Scenario	50~150 depending on the scenario
ACOSC [29]	$\tau_0 = 0.5$, $\alpha=0.45$, $\rho=0.35$, $\gamma=0.30$, Number of Ant (NA)=25		
HDPSO [30]	Binary vectors X and V are randomly set, $V = P_1V_1 \oplus P_2V_2 \oplus \dots P_nV_n$ where $\sum_{i=1}^n P_i = 1$; P_i is gained its value based on inertial, local, and global best fitness during each round		
Proposed D-CSA	$MaxGen \in [30\sim50]$, $\beta_i = 1$ (normalized full battery) $P_\alpha = 0.75$ (the inferior probability in whole nest), $R \in [1\sim4]$, The rest data is drawn from dataset		

In this line, Table 4 shows simulation parameters in terms of the area of monitoring district, number of sensor directions, the maximum number of adjustable ranges along with energy usage level of each, the number of directional sensors, and the number of targets in the monitoring field.

Table 4. Simulation Description

Parameter	Value
Monitoring district	1000 × 1000 (m ²)
Number of Targets	20~50
Number of Sensors	30~100
Max direction	1~3 (each 120 ⁰)
Max Adjustable Range	1=(30 m), 2=(60m), 3=(90 m)
Battery Usage level	$\Delta^i=i$, $i=1,2,3$

5.2.Experimental Results Analysis

The proposed algorithm designated for solving DSN's lifetime maximization problem, which at the same time monitors all targets in the observing filed continuously, is run in all of the defined scenarios along with all comparative algorithms in the same circumstance and fair conditions. Then, the experimental results are presented in a descriptive statistics, namely, the *min-Max* range, *average*, and standard deviation (*STD*) values gained via different executions of each comparative algorithm that are separately reported based on each scenario. To this end, Table 5 is dedicated to show *min-Max* range and the *average* value of lifetime that is gained from the output of comparative algorithms.

Table 5. Performance comparison of different comparative algorithms in terms of *min-Max range and average value*

Scenario	<i>min-Max (Lifespan in term of UT)</i>					<i>Average (UT)</i>			
	<i>H-MNLAR (fix)</i>	<i>GA</i>	<i>ACOSC</i>	<i>HDPSO</i>	<i>D-CSA</i>	<i>GA</i>	<i>ACOSC</i>	<i>HDPSO</i>	<i>D-CSA</i>
1	1.24	1.11-1.43	1.13-1.42	1.38-1.51	1.43-1.54	1.33	1.31	1.44	1.46
2	1.45	1.28-1.58	1.40-1.66	1.63-1.73	1.76-1.82	1.45	1.51	1.65	1.79
3	1.76	1.85-1.98	1.75-1.85	1.90-2.25	2.15-2.30	1.93	1.82	2.13	2.25
4	2.50	2.47-2.60	2.47-2.58	2.53-2.60	2.55-2.65	2.55	2.53	2.58	2.63
5	2.85	2.95-3.15	2.95-3.05	3.05-3.25	3.14-3.60	3.08	2.99	3.17	3.47
6	3.45	3.55-3.70	3.47-3.75	3.83-4.05	4.05-4.23	3.61	3.57	3.91	4.14

As Table 5 reports, the range which *D-SCA* produces is better than others; also, its averages value in all scenarios outperform other state-of-the-arts. These results show that the heuristic *H-MNLAR* has the lowest value although its time complexity is the lowest [8]. It only competes with GA in the second scenario where the GA average value is the same as this heuristic value. Also, the *ACOSC* marginally beats this heuristic in the fourth scenario. Closer look proves that the GA and *ACOSC* competes each other whereas the *HDPSO* competes *D-CSA*, but both *HDPSO* and *D-CSA* beats both *ACOSC* and GA.

In this line, Table 6 is dedicated to inform *STD* value along with the amount of relative percentage deviation (*RPD*) of average value relevant to proposed algorithm against others [30]. This *RPD* value is used to show the amount of improvement.

Table 6. Performance comparison of different comparative algorithms in terms of *STD* value along with average *RPD*

Scenario	<i>STD (Lifespan)</i>				Average <i>RPD</i> (%)			
	<i>GA</i>	<i>ACOSC</i>	<i>HDPSO</i>	<i>D-CSA</i>	<i>H-MNLAR</i>	<i>GA</i>	<i>ACOSC</i>	<i>HDPSO</i>
1	0.23	0.26	0.12	0.10	18%	10%	11%	1%
2	0.31	0.29	0.13	0.09	23%	23%	18%	8%
3	0.19	0.25	0.17	0.15	28%	17%	24%	5%
4	0.16	0.18	0.08	0.08	5%	3%	4%	2%
5	0.28	0.26	0.31	0.23	22%	13%	16%	10%
6	0.36	0.43	0.27	0.25	20%	15%	16%	6%
Total improvement (%)					19.33%	13.50%	14.83%	5.33%

Table 6 informs that all average values associated to meta-heuristics and hybrid algorithm are better than the heuristic algorithm. Also, the proposal algorithm has better outcome against other comparative algorithms. In term of *STD*, the proposed algorithm has the minimum value. It proves that designing and conducting miscellaneous discrete walking around procedures in the proposed cuckoo search algorithm explores search space uniformly and yields good balance between exploration and exploitation in search field. In all scenarios, after *D-CSA* which is in the first and best place the *HDPSO*, *GA*, *ACOSC*, and *H-MNLAR* are respectively in the next ranking from best to worst based on *RPD* measurement except for in the second scenario where the *ACOSC* is the third place and *GA* is in the fourth place. Totally, the proposed algorithm has the amount of 5.33%, 13.50%, 14.83%, and 19.33% dominance in term of average lifetime maximization against *HDPSO*, *GA*, *ACOSC*, and *H-MNLAR* algorithms respectively.

Also, the Fig. 15 illustrates the average value of different executions associated to each algorithm in each scenario.

Fig. 15. The average lifetime comparison between comparative algorithms separated in each scenario

As Fig. 15 depicts, the *D-CSA* beats all of the comparative algorithms in all cases. In the second scenario, the *H-MNLAR* and *GA* gain the same result, but in other scenarios the *H-MNLAR* returns the worst outcome in comparison to others. On the other hand, the average value of *GA* is better than *ACOSC*; in this vein, the *HDPSO* after *D-CSA* beats others. When the number of targets and sensors are significantly increased the difference values between proposed *D-CSA* against comparative algorithms is noticeable. It proves the high potential scalability of the proposed algorithm. In term of average elapsed time in all scenarios related to each comparative algorithm, the comparison is set in Table 6.

Table 6. The average elapsed time in all scenarios of each comparative algorithm

Algorithm	Average Execution Time	H-MNLAR:	80.46 s	GA-based :	338.27
ACOSC:	235.14 s	HDPSO :	165.83 s	D-CSA :	188.92

Table 6 shows that *H-MNLAR*, *HDPSO*, *D-CSA*, *ACOSC*, and *GA*-based are from the quickest to the slowest ranking algorithms. The proposed *D-CSA* is in the third place. Since the sensors deployment in reality is after determination of the scheduling, the execution time relevant to comparative algorithms is not very important, but on the other side the quality of algorithm which extends the network lifetime in significant amount is the most favorite feature where the proposed *D-CSA* is the most efficient against others. In the other words, in reality the sleep/wakeup assignment of each sensor is done according to the scheduling plan which was designed in the previous episode.

6. Conclusion and Future Work

This paper formulated lifetime maximization of directional sensor networks to a discrete optimization problem which is well-known NP-Hard problem. To solve this combinatorial problem, an advanced cuckoo search optimization algorithm has been devised. It takes benefit of

randomly applying miscellaneous discrete walking around procedures to assimilate the levy flight concept in continuous version of cuckoo search algorithm. To verify the proposed algorithm, it has been tested in different circumstances. The simulation results proved the significantly dominance of proposed algorithm against other state-of-the-arts in term of the amount of network lifetime improvement. For future work, we envisage to determine the most efficient place of sensors deployment to observe all targets subject to maximize network lifetime by utilizing computational geometry regarding to sensors capabilities in terms of number of possible directions, maximum usable ranges and power consumption of each.

Compliance with ethical standards

Conflict of interest: The authors declare that there is no conflict of interest regarding the publication of this paper.

Ethical approval: This article does not contain any studies with human participants or animals performed by any of the authors.

Informed consent: Informed consent was obtained from all individual participants included in the study.

References:

- [1] J. Jia, C. Dong, Y. Hong, L. Guo, Y. Yu, “ Maximizing full-view target coverage in camera sensor networks”, *Ad Hoc Networks*, Vol. 94, no. 101973, pp. 1-10, 2019.
- [2] O. E. Elnaggar, R. A. Ramadan, M. B. Fayek, “WSN in Monitoring Oil Pipelines Using ACO and GA”, *The 2nd International Workshop on Wireless Networks and Energy Saving Techniques (WNTEST 2015)*, *Procedia Computer Science*, Vol. 52, 2015, pp.1198 – 1205.
- [3] C. Luo, Y. Hong, D. Li, Y. Wang, W. Chen, Q. Hu, “Maximizing Network Lifetime Using Coverage Sets Scheduling in Wireless Sensor Networks”, *Ad Hoc Networks*, Vol.98, no.102037, pp.1-16, 2019. doi: <https://doi.org/10.1016/j.adhoc.2019.102037>.
- [4] A. Alibeiki, H. Motameni, H. Mohamadi, “A new genetic-based approach for maximizing network lifetime in directional sensor networks with adjustable sensing ranges”, *Pervasive and Mobile Computing*, vol. 52,, pp. 1-12., 2019.
- [5] R. Cerulli, R. De Donato, A. Raiconi, “Exact and heuristic methods to maximize network lifetime in wireless sensor networks with adjustable sensing ranges”, *European J. Oper. Res.*, Vol. 220, pp. 58-66, 2012.
- [6] F. Mazunga, A. Nechibvute, “ Ultra-low power techniques in energy harvesting wireless sensor networks: Recent advances and issues”, *Scientific African*, Vol. 11, no. e00720, pp.1-14 , 2021.

- [7] S. Kyi, A. Taparugssanagorn, "Wireless sensing for a solar power system", Digital Communications and Networks, Vol. 6, pp.51-57, 2020.
- [8] H. Mahamadi, S. Salleh, M.N. Razali, "Heuristic methods to maximize network lifetime in directional sensor networks with adjustable sensing ranges", journal of network and computer applications, Vol. 46, pp. 26-35, 2014.
- [9] C., Thomas, H.; Leiserson, C. E.; Rivest, Ronald L. (1990). Introduction to Algorithms (1st ed.). MIT Press and McGraw-Hill. ISBN 0-262-03141-8.
- [10] G. Chen, Y. Xiong, J. She, M. Wu and K. Galkowski, "Optimization of the Directional Sensor Networks With Rotatable Sensors for Target-Barrier Coverage," in *IEEE Sensors Journal*, vol. 21, no. 6, pp. 8276-8288, 15 March 2021, doi: 10.1109/JSEN.2020.3045138.
- [11] Deb, Kalyanmoy; Spears, William M. (1997). "C6.2: Speciation methods". *Handbook of Evolutionary Computation*. Institute of Physics Publishing. S2CID 3547258.
- [12] Saeedi P, "An improved thermodynamic simulated annealing-based approach for resource-skewness-aware and power-efficient virtual machine consolidation in cloud datacenters", Soft Comput. [https:// doi.org/ 10. 1007/ s00500- 020- 05523-1](https://doi.org/10.1007/s00500-020-05523-1).
- [13] X.-S. Yang; S. Deb (December 2009). *Cuckoo search via Lévy flights*. World Congress on Nature & Biologically Inspired Computing (NaBIC 2009). IEEE Publications. pp. 210–214. [arXiv:1003.1594v1](https://arxiv.org/abs/1003.1594v1).
- [14] Ai J, Abouzeid A. "Coverage by directional sensors in randomly deployed wireless sensor networks", J. Combin. Optim, Vol. 11, pp. 21-41, 2006.
- [15] A. Rossi, A. Singh, M. Sevaux, "Lifetime maximization in wireless directional sensor network", European Journal of Operational Research, Vol. 231, pp. 229–241, 2013.
- [16] P. Singh and S. Mini, "A Heuristic to Deploy Directional Sensor Nodes in Wireless Sensor Networks," *2016 8th International Conference on Computational Intelligence and Communication Networks (CICN)*, 2016, pp. 11-15, doi: 10.1109/CICN.2016.11.
- [17] W. M. Tan, S. A. Jarvis, "Heuristic solutions to the target identifiability problem in directional sensor networks", journal of network and computer applications, Vol. 65, pp. 84-102, 2016.
- [18] J. Xu, F. Zhong, Y. Wang, "Learning Multi-Agent Coordination for Enhancing Target Coverage in Directional Sensor Networks", 34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.
- [19] S. Sharmin, F. N. Nur, M. Islam, M. A. Razzaque, M. M. Hassan and A. Alelaiwi, "Target Coverage-Aware Clustering in Directional Sensor Networks: A Distributed Approach," in *IEEE Access*, vol. 7, pp. 64005-64014, 2019, doi: 10.1109/ACCESS.2019.2916407.
- [20] Lu, Z., Wu, W. and Li, W.W., "Target coverage maximization for directional sensor networks", *Int. J. Sensor Networks*, Vol. 24, No. 4, pp.253–263, 2017.
- [21] P. Chaturvedi and A. K. Daniel, "Trust based energy efficient coverage preserving protocol for wireless sensor networks," *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)*, 2015, pp. 860-865, doi: 10.1109/ICGCIoT.2015.7380583.

- [22] Harizan, S., Kuila, P., “ Coverage and connectivity aware energy efficient scheduling in target based wireless sensor networks: an improved genetic algorithm based approach”, *Wireless Netw*, Vol. 25, pp. 1995–2011, 2019. <https://doi.org/10.1007/s11276-018-1792-2>.
- [23] Gil JM, Han YH. A target coverage scheduling scheme based on genetic algorithms in directional sensor networks. *Sensors (Basel)*. 2011;11(2):1888-906. doi: 10.3390/s110201888. Epub 2011 Feb 1. PMID: 22319387; PMCID: PMC3274050.
- [24] A. Wang, Y. Gao, J. Wu, G. Sun, W. Jia, “A novel multi-objective coverage optimization memetic algorithm for directional sensor networks”, *international journal of distributed sensor networks*, Vol. 12, no.7, pp.1-9, 2016.
- [25] Razali, M.N., Salleh, S. & Mohamadi, H. Solving Priority-Based Target Coverage Problem in Directional Sensor Networks with Adjustable Sensing Ranges. *Wireless Pers Commun* 95, 847–872 (2017). <https://doi.org/10.1007/s11277-016-3801-z>.
- [26] G. Chen, Y. Xiong, J. She, M. Wu and K. Galkowski, "Optimization of the Directional Sensor Networks With Rotatable Sensors for Target-Barrier Coverage," in *IEEE Sensors Journal*, vol. 21, no. 6, pp. 8276-8288, 15 March15, 2021, doi: 10.1109/JSEN.2020.3045138.
- [27] Kim, Jaemin, and Younghwan Yoo. “Sensor Node Activation Using Bat Algorithm for Connected Target Coverage in WSNs.” *Sensors (Basel, Switzerland)* vol. 20,13 3733. 3 Jul. 2020, doi:10.3390/s20133733.
- [28] Ramzanpoor, Y., Shirvani, M. & Golsorkhtabaramiri, M. Multi-objective fault-tolerant optimization algorithm for deployment of IoT applications on fog computing infrastructure. *Complex Intell. Syst.* (2021). <https://doi.org/10.1007/s40747-021-00368-z>.
- [29] P. Huang, F. Lin, C. Liu, J. Gao, J. Zhou, “ACO-Based Sweep Coverage Scheme in Wireless Sensor Networks”, *Journal of Sensors*, Vol. 2015.
- [30] M. Shirvani, “A hybrid meta-heuristic algorithm for scientific workflow scheduling in heterogeneous distributed computing systems,” *Eng. Appl. Artif. Intell.*, vol. 90, no. December 2019, p. 103501, Apr. 2020, doi: 10.1016/j.engappai.2020.103501