A Novel Random Scheduling Algorithm based on Subregions Coverage for SET K-Cover Problem in Wireless Sensor Networks

Zahid Muhammad[1], Abhishek Roy[2], Chang Wook Ahn[3], Ruchi Sachan[1] and Navrati Saxena[1*]
[1] Electrical and Computer Engineering Department, Sungkyunkwan University, Suwon, South Korea
[e-mail: zahid571, ruchi93, navrati@skku.edu]
[2] System Design Lab., Networks Division Samsung Electronics, Suwon, South Korea
[e-mail: abhishek.roy@samsung.com]
[3] Department of Electrical Engineering & Computer Science, Gwangju Institute of Science and Technology (GIST), South Korea
[e-mail: cwan@gist.ac.kr]
*Corresponding author: navrati@skku.edu

Received August 29, 2017; revised December 10, 2017; accepted January 15, 2018; published June 30, 2018

Abstract

This paper proposes a novel Random Scheduling Algorithm based on Subregion Coverage (RSASC), to solve the SET K-cover problem (an NP-complete problem). SET K-cover problem distributes the set of sensors into the maximum number of mutually exclusive subsets (MESSs) in such a way that each of them can be scheduled for lifetime extension of WSN. Sensor coverage divides the target region into different subregions. RSASC first sorts the subregions in the ascending order concerning their sensor coverage. Then, it forms the subregion groups according to their similar sensor coverage. Lastly, RSASC ensures the K-coverage of each subregion from every group by randomly scheduling the sensors. We consider the target-coverage and area-coverage applications of WSN to analyze the usefulness of our proposed RSASC algorithm. The distinct quality of RSASC is that it utilizes less number of deployed sensors (33% less) to form the optimum number of MESSs with the higher computational speed (saves more than 93% of the time) as compared to the existing three algorithms.

Keywords: WSN, SET K-cover, Subregions, MESSs, RSASC

1. Introduction

Wireless sensor networks (WSNs) are widely being used in a variety of applications such as military surveillance, industrial process automation, traffic control, environmental monitoring, physical security and medical care. The lifetime of a WSN (the time during which it senses the region of interest under some coverage constraint) is a key performance indicator to determine its efficiency. Energy consumption must be restricted to achieve the maximum operating time of WSN because battery-powered sensors are difficult to recharge or replace [1-5].

Coverage ensures that each point of the target region must be completely sensed by WSN. Position and sensing range of sensors are used as inputs to the coverage finding algorithms [6]. Coverage applications are usually categorized as target coverage, area coverage and barrier coverage [2], [3], [7]. Target-coverage focuses on sensing different points in a certain area. In the area-coverage application, deployed sensors sense the whole monitored area, whereas barrier-coverage focuses on observing a subregion defined by two parallel curves.

Sensor placement in the target region is usually done by using two approaches, i.e., random approach and deterministic approach [8]. In random approach, sensors are randomly placed for surveillance application in hard-to-reach areas such as in very high-temperature or very low-pressure area. In deterministic approach, a prior detail of target area is already known to deploy each sensor one by one at a specific position. However, this approach of sensor deployment is time-consuming due to a large number of sensors and it cannot be adopted for inaccessible areas. In such circumstances, a large number of sensors are randomly thrown from airplane to fulfill the coverage constraint of the target region.

Scheduling is an effective approach for extending the operational time of WSN by controlling the sensors actions [9]. A sensor has two operational states; an active state and a sleep state (also known as energy saving state). In an active state, a sensor performs all of its functions like sensing, data processing, and communication. Such operations dissipate comparatively a larger amount of energy. Different from being in an active state, a sensor consumes less amount of energy in the sleep state. Sensors can switch from sleep to active state or vice versa if needed. If a MESS fulfills the coverage constraint, remaining MESSs are forced to switch their sleep state in order to save energy [10, 11]. In such situations, how to extend the operational time of a network under coverage constraint is an important research topic [12].

Several researches such as [4], [12-25] exist on energy efficient WSN. For example, the work in [16], proposed an improved version of Stable Election Protocol (SEP), termed as Prolong-SEP (P-SEP) to extend the steady duration of Fog-assisted WSN by conserving the stable energy utilization. P-SEP permits uniform dispersal of sensors, plan for the nomination of new cluster head, and lifetime extension of WSN, particularly prior to the failure of the first sensor. P-SEP examined two-rank heterogeneity of sensors, termed as advanced and normal sensors which hold the chance of becoming the cluster heads. P-SEP outperforms the traditional SEP, modified SEP and efficient modified SEP. The authors in [17], suggested an effective routing scheme based on the evaluation of existing routing approaches to achieve K-covered WSN and reliable data transfer with reasonable fault acceptance. They assumed that each sensor is well informed of its remaining energy as well as that of its neighboring sensors. They initially classified all sensors in terms of coverage and communicative sensors. Finally, they re-classified some sensors in terms of clustering and dynamic sensors. The authors in [18], proposed a new resource (re)assignment design which permits energy-consious service function chaining for software defined network. The work in [18], suggested heuristic approaches with suitable time complexity to search the near-optimal solution for the deployment of virtual network functions, their assignment to flows and flow routing.

The above works focus on the routing and scheduling to achieve the energy efficient WSN. SET K-cover problem is the main issue which needs to be focused. **The motivation and the objective** of this paper is to achieve optimality with reduction in computational complexity for improving the lifetime of WSNs. The novelity of our algorithm helps to resolve the SET K-cover problem in the minimum time for the target-coverage and area-coverage applications of WSN. SET K-cover allocates the deployed sensors into the optimum number of MESSs under coverage constraint for extending the network`s lifetime. Due to a large number of deployed sensors, solving the SET K-cover problem is a difficult job which increases the computational complexity.

Our main contribution in this paper is to propose a novel Random Scheduling Algorithm based on Subregion Coverage (RSASC) to search an optimum number of MESSs while satisfying the coverage constraint for both the target-coverage and area-coverage applications. The proposed RSASC works in the following way.

a) It first distributes the target region into different subregions based on the sensor coverage using the algorithm in [13].
b) Then, it combines subregions into different groups in such a way that at least one sensor is common in the sensing coverage of all subregions.
c) After formation of different subregion groups, it selects each subregion one by one from each subregion group and schedules their sensors coverage by randomly assigning a number from one to α inclusive, where α is a pre-defined upper bound on the optimal number of MESSs.
d) Similarly, the same process (above three steps) is repeated for other subregion groups.

This approach of scheduling based on coverage of each subregion remarkably reduces the computational time for searching the maximum number of MESSs under coverage constraint. The distinct feature of the RSASC algorithm is that it uses the less number of deployed sensors (on average 33% less than the existing algorithms) to form the optimum number of MESSs while satisfying the coverage constraint. Moreover, the computational time taken by RSASC is much lesser than that of the existing algorithms. On the average, RSASC saves more than 93% of the running time as compared to the existing algorithms.

The rest of our paper is organized as follows. Section 2 presents a brief related work. Section 3 provides the problem definition and discussions on upper bound, the division of target region into subregions and coverage calculation of a sensor. Section 4 explains the proposed RSASC in detail including the formation of subregion group, representation of candidate solution, fitness function, and the computational complexity. Section 5 presents the comparative analysis of RSASC with the existing algorithms. Section 6 is reserved for the conclusion and future work.

2. Related Work

In the literature, finding the optimum number of MESSs in a WSN is known as “SET K-Cover” or "Disjoint Set Covers" problem [13], [14] which are proven to be the non-deterministic polynomial NP-complete problems. Numerous heuristics have investigated this maximization problem. Cardie and Du [14] considered the target-coverage problem for the surveillance application of energy efficient WSN. They proposed a mixed integer programming based heuristic termed as, "maximum covers using mixed integer programming (MCMIP)" to distribute the set of deployed sensors among optimum number of MESS while satisfying the coverage constraint for a set of known target points. MCMIP applies the brute-force search method to find the maximum number of MESS. However, MCMIP is infeasible for practical application because its computational complexity exponentially increases with the number of deployed sensors. Slijepcevic and Potkonjak [13] proposed a greedy deterministic approach, termed as "the most constrained minimum constrained covering (MCMCC)" heuristic to prolong the operational time of WSN. The computational complexity of MCMCC heuristic is polynomial time. However, it searches the near-optimal number of MESS in some cases (e.g., 80% of the best solution).

Some studies have also investigated the SET K-Cover problem using meta-heuristics. Genetic algorithm (GA) [27], [28] is a population-based searching algorithm which has been efficiently used for solving many NP-complete problems [29]. Lai et al. [30] proposed a GA for the maximum disjoint set covers, termed as, GAMDSC to find the optimal number of MESS ($\boldsymbol{\alpha}$) for target-coverage application of WSN. However, GAMDSC failed to search the optimal solution. GAMDSC consists of conventional genetic operations and scattering operation to ensure the presence of one critical sensor in each MESS. Computational time for GAMDSC lies in between MCMCC and MC-MIP.

Hu et al. [31], suggested a mixture of GA with schedule transition operations entitled as "Hybrid Genetic Algorithm with Schedule Transformation (STHGA)" to optimize the network's duration. STHGA adopts the forward encoding scheme based on three schedule transformation operations i.e., mixed schedule transformation (MST), forward schedule transformation (FST) and critical schedule transformation (CST) operations. MST changes the schedule number of a redundant sensor from one MESS to another MESS. Redundant sensors are randomly searched for K_2 times in MST. FST is used to improve the coverage of an incomplete MESS. In FST, redundant sensors are randomly searched for K_1 times to assign an incomplete schedule number where K_1 and K_2 are two predefined parameters in [31]. CST ensures the coverage of critical subregions (a subregion covered by a minimum number of sensors) by at least one sensor in an incomplete MESS. In all of the above-mentioned meta-heuristics, each gene maps to a sensor, whereas its value indicates a schedule number which determines its association to a particular MESS. Moreover, STHGA searches an optimal solution in shorter computational time with better solution quality as compared to GAMDSC and MCMCC.

Lin et al. [32], proposed a harmony search algorithm with multiple populations and local search (HSAML) to prolong the lifetime of dynamic heterogeneous WSN with energy harvesting sensors. HSAML considered a dynamic scenario for WSNs with $\boldsymbol{n_1}$ ordinary and $\boldsymbol{m_1}$ energy-harvesting sensors in which each active sensor can be malfunctioned at any time. HSAML was executed for NI times, where NI is the predefined limit for the iterations, to find the maximum number of MESS and a local search operation for the inclusion of energy harvesting sensors in an active MESS. Authors in [32], validates the performance of HSAML over the traditional HSA algorithm. However, the problem with HSAML is that it does not guarantee the optimal number of MESS since there is no condition of obtaining the $\boldsymbol{\alpha}$ MESSs on its execution.

3. Problem Definition and Discussions

3.1 Problem Definition

Suppose that a group of $\boldsymbol{N}$ sensors, $\boldsymbol{S} = \{\boldsymbol{S_1}, \boldsymbol{S_2}, \dots, \boldsymbol{S_N}\}$, is randomly positioned in an $\boldsymbol{L} \times \boldsymbol{W}$ target region to cover a collection of $\boldsymbol{K}$ immobile targets, $\boldsymbol{T} = \{\boldsymbol{T_1}, \boldsymbol{T_2}, \dots, \boldsymbol{T_K}\}$ or a complete $\boldsymbol{L} \times \boldsymbol{W}$ target region. The objective of this problem is to search the optimal number of mutually exclusive subsets of sensors (MESS), $\boldsymbol{\alpha}$, subject to the following constraints:

a) Each subset $\boldsymbol{D_i} = \left\{\boldsymbol{S_i^1}, \boldsymbol{S_i^2}, \dots, \boldsymbol{S_i^{|D_i|}}\right\} \subseteq \boldsymbol{S}$, where $\boldsymbol{D_i}$ indicates specific disjoint subset, whose members ensure complete coverage of $\boldsymbol{K}$ fixed target points or whole target region for target-coverage or area-coverage application, respectively. Each $\boldsymbol{D_i}$ becomes active to sense $\boldsymbol{K}$ target points or whole target region for particular $\boldsymbol{i^{th}}$ schedule, $\boldsymbol{i} = \{\boldsymbol{1}, \boldsymbol{2}, \dots, \boldsymbol{\alpha}\}$ where $|\boldsymbol{D_i}|$ is the total number of sensors in the disjoint subset for $\boldsymbol{i^{th}}$ schedule.

b) Each sensor must be the member of only one specific disjoint subset and satisfy **Eq. 1**,

$$\boldsymbol{D_i} \cap \boldsymbol{D_j} = \boldsymbol{\phi}, \quad where\ \boldsymbol{i} \neq \boldsymbol{j}, \qquad \forall \boldsymbol{i}, \boldsymbol{j} = \{\boldsymbol{1}, \boldsymbol{2}, \dots, \boldsymbol{\alpha}\} \tag{1}$$

Fig. 1 depicts the $\boldsymbol{N}$ randomly deployed sensors to achieve the complete coverage of $\boldsymbol{L} \times \boldsymbol{W}$ target area. Many circles in the **Fig. 1** represent the sensing coverage of sensors. For our convenience, in this paper, we assume that each sensor has circular sensing coverage. However, in practical situations, it may take any irregular appearance.

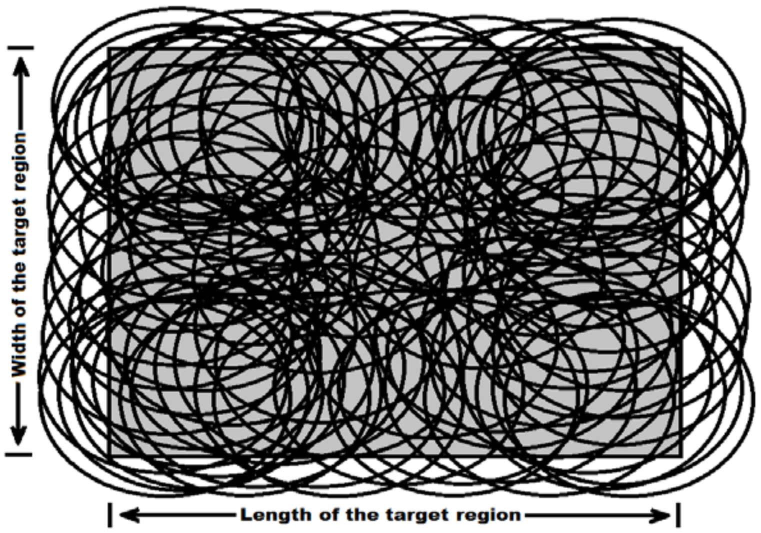

Fig. 1. Example showing the sensor deployment for area-coverage application

3.2 Discussion

The maximum number of MESSs ($\boldsymbol{\alpha}$) and the individual time during which the sensors ensure the full coverage of target region, calculate the operational time of WSNs. However, in this paper, we assume that the duration of each MESS is same. The number of deployed sensors, their sensing range, and positions in the target region decides the number of MESSs.

3.2.1 Lower and Upper Limits on Schedule Numbers

Fig. 2 (a) to find the upper limit ($\boldsymbol{\alpha}$). It is a matter of fact that determining the upper bound ($\boldsymbol{\alpha}$) is also a famous K-coverage problem [33]. It can be seen from the **Fig. 2 (a)** that 10 sensors distribute the target area into a set of $\Delta = \boldsymbol{20}$ subregions, The upper limit denoted as $\boldsymbol{\alpha}$, is the maximum number for which a WSN can be scheduled. The Eq. 2 determines the least number of sensors that sense a specific point or subregion for the target-coverage or area-coverage case, respectively. In **Eq. 2**, $|S_{\beta_j}|$ represents how many sensors monitor a particular β_j target or subregion. Whereas, Δ shows the total number of targets or subregions created in the region for target-coverage or area-coverage application, respectively. $\Psi = \{\beta_1, \beta_2, \dots, \beta_{20}\}$. Similarly, **Fig. 2 (b)** depicts the sensor coverage for each subregion

$$\alpha = \min_{j=\{1,2,\dots,\Delta\}} \left(|S_{\beta_j}|\right) \tag{2}$$

In the area-coverage problem, a target area is distributed into different subregions as shown in **Fig. 2 (a)** to find the upper limit (α). It is a matter of fact that determining the upper bound (α) is also a famous K-coverage problem [33]. It can be seen from the **Fig. 2 (a)** that 10 sensors distribute the target area into a set of $\Delta = 20$ subregions, $\Psi = \{\beta_1, \beta_2, \dots, \beta_{20}\}$. Similarly, **Fig. 2 (b)** depicts the sensor coverage for each subregion. The upper limit in this example is two because the least number of sensors that cover a subregion is two (e.g., sensors S_1 and S_3 cover β_1). Similarly, we can find the upper bound (α) for an area-coverage application using **Eq. (2)**. In this paper, we assume that randomly deployed sensors must provide the complete coverage at least once for both problems. Therefore, the lower limit for each case considered in this paper is one.

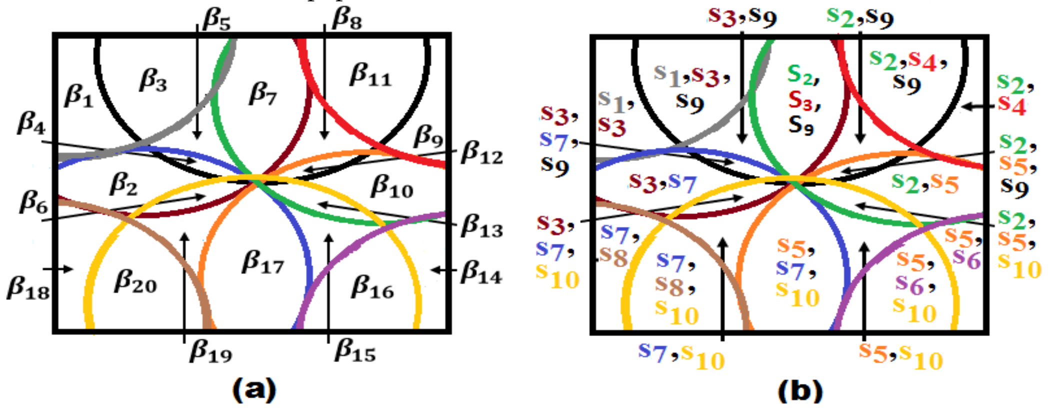

Fig. 2. (a) Target area distribution into a set of $\Delta = 20$ subregions, $\Psi = \{\beta_1, \beta_2, \dots, \beta_{20}\}$ by 10 sensors shown by different colored circles. (b) Sensors coverage for each subregion.

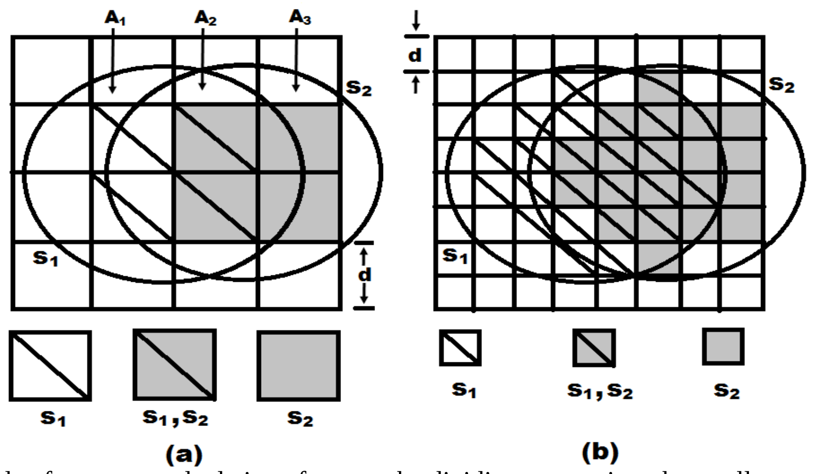

Fig. 3. Example of coverage calculation of sensors by dividing an area into the small square boxes with a grid size d. A_1, A_2 and A_3 indicate the coverage areas of three subregions created by two sensors S_1 and S_2. (a) and (b) show the effect of grid size (**d**) on coverage calculation.

3.2.2 Sensor Coverage Calculation

Fig. 3 indicates the sensing coverage of two sensors $\boldsymbol{S_1}$ and $\boldsymbol{S_2}$ in terms of subregions and small square boxes in the target area. Sensor $\boldsymbol{S_1}$ covers the subregion areas $\boldsymbol{A_1}$ and $\boldsymbol{A_2}$ while $\boldsymbol{S_2}$ covers subregion areas $\boldsymbol{A_2}$ and $\boldsymbol{A_3}$. A square box in the sensing region of interest is supposed to be covered if its four corners lie within the sensing range of sensor as shown in **Fig. 3 (a)** and **(b)**.

In **Fig. 3 (a)**, the entire monitored region is distributed into 16 square boxes according to the specific grid size. Similarly, in **Fig. 3 (b)**, a target area is distributed into 64 square boxes since grid size is one-fourth of the grid size considered in **Fig. 3 (a)**. It can be seen from the figure that coverage approximation of sensor is better in **Fig. 3 (b)** as compared to **Fig. 3 (a)** due to smaller grid size. In **Fig. 3 (a)**, two sensors cover six out of 16 boxes, which is 37.5% of the target area whereas, in **Fig. 3 (b)**, same sensors cover 31 out of 64 boxes, which is 48.44% of the target region. Thus, each subregion can be considered as a target after calculating their coverage in terms of the square box [13]. Then, the subregions can be used for sensor coverage calculation.

4. Proposed Random Scheduling Algorithm based on Subregion Coverage

In this paper, we propose a novel Random Scheduling Algorithm based on Subregion Coverage (RSASC) for maximizing the operational time of WSN considering the target-coverage and area-coverage applications. In this section, we first distribute the subregions into groups. Secondly, we present the representation of candidate solution for our problem. Thirdly, we implement the RSASC step-by-step including the initialization and the evaluation of fitness function. Finally, we present the time complexity of each algorithm.

4.1 Formation of Subregion Groups

The formation of subregion groups plays a vital role in searching the optimal number of MESS. Assume that set $\boldsymbol{\Psi} = \{\boldsymbol{\beta_1}, \boldsymbol{\beta_2}, \ldots, \boldsymbol{\beta_\Delta}\}$ represents the set of subregions and $\boldsymbol{Q_i}$ is the set of sensors, which senses the $\boldsymbol{i^{th}}$ subregion in set $\boldsymbol{\Psi}$. We form the $\boldsymbol{P}$ disjoint groups of $\boldsymbol{\Delta}$ subregions as follows.

a) Sort the subregions in ascending order with respect to their sensor coverage. For example, $\boldsymbol{\Delta = 20}$ subregions in **Fig. 2 (a)** are sorted in ascending order as tabulated in **Table 1** according to their respective sensor coverage shown in **Fig. 2 (b)**.

Table 1. Sorted subregions with their coverage and subregion groups for example depicted in **Fig. 2 (a) and (b)**

Sorted subregions with their coverage		Subregion groups	Sensor coverage groups
$\beta_1 = \{S_1, S_3\}$	$\beta_3 = \{S_1, S_3, S_9\}$	$G_S^1 = \{\beta_1, \beta_2, \beta_5, \beta_3, \beta_4, \beta_6, \beta_7\}$	{2,2,2,3,3,3,3}
$\beta_2 = \{S_3, S_7\}$	$\beta_4 = \{S_3, S_7, S_9\}$	$G_S^2 = \{\beta_8, \beta_9, \beta_{10}, \beta_{11}, \beta_{12}, \beta_{13}\}$	{2,2,2,3,3,3}
$\beta_5 = \{S_3, S_9\}$	$\beta_6 = \{S_3, S_7, S_{10}\}$	$G_S^3 = \{\beta_{14}, \beta_{15}, \beta_{16}, \beta_{17}\}$	{2,2,3,3}
$\beta_8 = \{S_2, S_9\}$	$\beta_7 = \{S_2, S_3, S_9\}$	$G_S^4 = \{\beta_{18}, \beta_{19}, \beta_{20}\}$	{2,2,3}
$\beta_9 = \{S_2, S_4\}$	$\beta_{11} = \{S_2, S_4, S_9\}$		
$\beta_{10} = \{S_2, S_5\}$	$\beta_{12} = \{S_2, S_5, S_9\}$		
$\beta_{14} = \{S_5, S_6\}$	$\beta_{13} = \{S_2, S_5, S_{10}\}$		
$\beta_{15} = \{S_5, S_{10}\}$	$\beta_{16} = \{S_5, S_6, S_{10}\}$		
$\beta_{18} = \{S_7, S_8\}$	$\beta_{17} = \{S_5, S_7, S_{10}\}$		
$\beta_{19} = \{S_7, S_{10}\}$	$\beta_{20} = \{S_7, S_8, S_{10}\}$		

b) For each $\boldsymbol{G_s^j}$ subregion group where $\boldsymbol{j \in P}$, first sorted subregion is selected as first member, e.g. for first group $\boldsymbol{G_s^1}$, $\boldsymbol{\beta_1}$= $\{\boldsymbol{S_1}, \boldsymbol{S_3}\}$ is selected as first member. After that add every sorted subregion into $\boldsymbol{G_s^j}$ if it satisfies the Eq. (3).

$$\boldsymbol{Q_1^j} \bigcap \boldsymbol{Q_i^j} \neq \boldsymbol{\phi}, \ \forall \boldsymbol{i} \in \boldsymbol{\Psi} \ and \ \forall \boldsymbol{j} \in \mathbf{P} \tag{3}$$

It can be seen from the sorted subregions set $\boldsymbol{\Psi_s}$ **in** Table 1 that $\boldsymbol{\beta_2}$, $\boldsymbol{\beta_5}$, $\boldsymbol{\beta_3}$, $\boldsymbol{\beta_4}$, $\boldsymbol{\beta_6}$, and $\boldsymbol{\beta_7}$, satisfy the Eq. (3) because either $\boldsymbol{S_1}$ or $\boldsymbol{S_3}$ is common in their sensor coverage. Therefore, $\boldsymbol{G_s^1} = \{\boldsymbol{\beta_1}, \boldsymbol{\beta_2}, \boldsymbol{\beta_5}, \boldsymbol{\beta_3}, \boldsymbol{\beta_4}, \boldsymbol{\beta_6}, \boldsymbol{\beta_7}\}$.

c) After formation of $\boldsymbol{G_s^1}$, eliminate all subregions of $\boldsymbol{G_s^1}$ from the sorted subregion set $\boldsymbol{\Psi_s}$ to form new groups. For example, after eliminating the members of $\boldsymbol{G_s^1}$, the number of subregions in set $\boldsymbol{\Psi_s}$ is decreased from 20 to 13.

We form the other subregion groups $\boldsymbol{G_s^{j+1}}$ untill set $\boldsymbol{\Psi_s} = \boldsymbol{\phi}$ by repeating the above three steps. For example, we form the second subregion group $\boldsymbol{G_s^2}$ as follows. $\boldsymbol{\beta_8}$= $\{\boldsymbol{S_2}, \boldsymbol{S_9}\}$ from set $\boldsymbol{\Psi_s}$ is selected as first member. Other members of the second subregion group are $\boldsymbol{\beta_9}$, $\boldsymbol{\beta_{10}}$, $\boldsymbol{\beta_{11}}$, $\boldsymbol{\beta_{12}}$, and $\boldsymbol{\beta_{13}}$. Thus, $\boldsymbol{G_s^2} = \{\boldsymbol{\beta_8}, \boldsymbol{\beta_9}, \boldsymbol{\beta_{10}}, \boldsymbol{\beta_{11}}, \boldsymbol{\beta_{12}}, \boldsymbol{\beta_{13}}\}$ and the number of subregions in set $\boldsymbol{\Psi_s}$ is further reduced from 13 to 7. Similarly, we create the third and fourth subregion as $\boldsymbol{G_s^3} = \{\boldsymbol{\beta_{14}}, \boldsymbol{\beta_{15}}, \boldsymbol{\beta_{16}}, \boldsymbol{\beta_{17}}\}$ and $\boldsymbol{G_s^4} = \{\boldsymbol{\beta_{18}}, \boldsymbol{\beta_{19}}, \boldsymbol{\beta_{20}}\}$. It can be seen from all the subregion groups that sensor coverage for first subregion of each group will always satisfy the Eq. (4).

$$\boldsymbol{Q_1^i} \bigcap \boldsymbol{Q_1^j} = \boldsymbol{\phi}, \ \forall \boldsymbol{i}, \boldsymbol{j} \in \mathbf{P} \tag{4}$$

After formation of subregion groups, each subregion group must satisfy the Eq. (5) and (6). Eq. (5) says that the number of sensors covering the first subregion of each group will always be greater than or equal to the value of an upper bound ($\boldsymbol{\alpha}$). It is clear in each subregion group that the number of sensors which covers the first subregion is $\boldsymbol{\alpha} = \mathbf{2}$. Fig. 4 (a) illustrates the formation of subregion groups according to above-mentioned approach whereas Fig. 4 (b) indicates the %age coverage of each subregion group e.g., %age coverage of group $\boldsymbol{G_s^1} = \frac{\mathbf{7}}{\mathbf{20}} \cong \mathbf{35}\%$ as set $\boldsymbol{G_s^1}$ comprises of seven out of 20 subregions. Similarly, we can form the groups for any value of the $\boldsymbol{\alpha}$ subregions created in the target area. Algorithm 1 presents the complete pseudo-code for the formation of subregion groups.

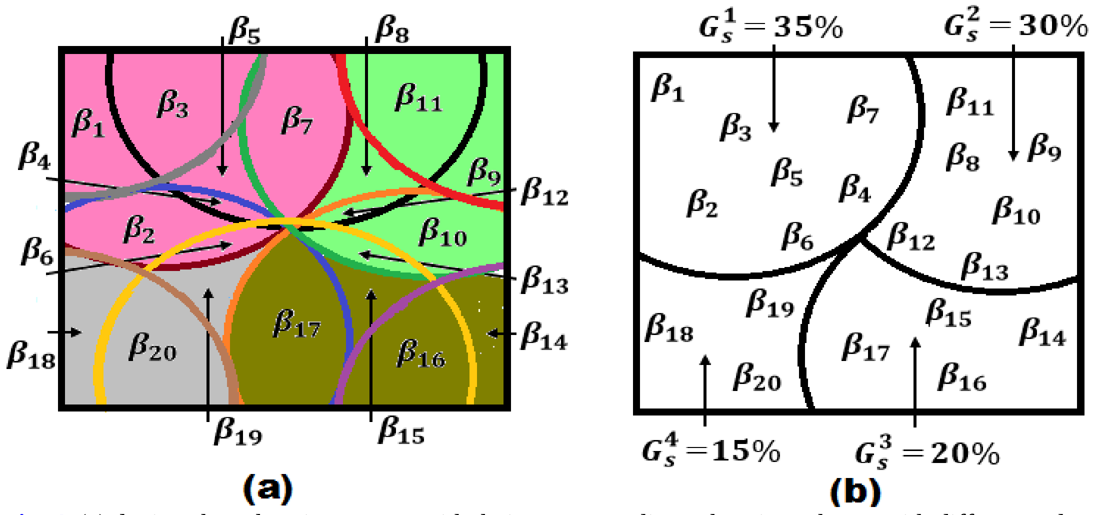

Fig. 4. (a) depicts the subregion groups with their corresponding subregions shown with different colors for example illustrated in Fig. 2. (b) indicates the %age coverage of each subregion group with respect to their total number of subregions.

$$\boldsymbol{Q_1^i} \geq \boldsymbol{\alpha}, \quad \forall \boldsymbol{i} \in \mathbf{P} \tag{5}$$

$$\left|\boldsymbol{Q_v^i}\right| \geq \left|\boldsymbol{Q_1^i}\right| \geq \boldsymbol{\alpha}, \quad \forall \boldsymbol{v} > 1, \text{and } \forall \boldsymbol{i} \in \mathbf{P} \tag{6}$$

4.2 Representation of Candidate Solution

RSASC is not a population-based algorithm. Therefore, the candidate solution consists of $\boldsymbol{N}$ dimensions and each dimension indicates a sensor. The candidate solution represented by $\boldsymbol{X}$ is given in **Eq. (7)**.

$$\boldsymbol{X} = \{\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_N}\} \tag{7}$$

Where $\boldsymbol{x_N} \in \{\mathbf{1}, \mathbf{2}, \ldots, \boldsymbol{\alpha}\}$, each dimension $\boldsymbol{x} \in \boldsymbol{X}$ in **Eq. (7)** represents a sensor and the value of $\boldsymbol{x}$ shows the particular schedule or MESS. A group of dimensions in $\boldsymbol{X}$ having the same value, which completely covers all target points or whole target region forms a MESS. Therefore, for an optimum solution, $\boldsymbol{N}$ sensors must be grouped into $\boldsymbol{\alpha}$ complete MESSs.

4.3 Initialization

RSASC initially allocates a zero to each dimension in candidate solution X, i.e., $\boldsymbol{X} = \{\mathbf{0}, \mathbf{0}, \ldots, \mathbf{0}\}$, meaning that all $\boldsymbol{N}$ randomly deployed sensors are switched off. STHGA initializes the population of $\boldsymbol{m}$ chromosomes as one, i.e., $\boldsymbol{X} = \{\mathbf{1}, \mathbf{1}, \ldots, \mathbf{1}\}$ meaning that all deployed sensors are in active state. A chromosome in GA is a candidate solution, which consists of $\boldsymbol{N}$ elements known as genes. In each of the GA based algorithm, each gene represents a sensor and its value shows membership of particular MESS. Different from RSASC and STHGA, GAMDSC initializes each gene of $\boldsymbol{m}$ chromosomes with different random numbers in the range $[\mathbf{1}, \boldsymbol{\alpha}]$ inclusive. In HSAML, a harmony (candidate solution) consists of $\boldsymbol{N}$ elements known as musicians, whereas their values represent the unique IDs of the deployed sensors. Each harmony consists of randomly stored sensor IDs.

4.4 Fitness Function

For a target-coverage application, the fitness function ($\boldsymbol{F}$) for a candidate solution $\boldsymbol{X}$ is defined as

$$\boldsymbol{F} = \sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{\alpha}} \left[\left(\frac{\left| \bigcup_{\boldsymbol{n}=\mathbf{1}\ \forall \boldsymbol{x_n}=\boldsymbol{i}}^{\boldsymbol{N}} \boldsymbol{C_n} \right|}{\boldsymbol{K}} \right) = \mathbf{1} \right] \tag{8}$$

In Eq. (8), $\boldsymbol{C_n}$ shows the coverage of $\boldsymbol{n^{th}}$ sensor and $\boldsymbol{C_n} \in \{\mathbf{1}, \mathbf{2}, \ldots \boldsymbol{K}\}$ for target-coverage application. Similarly, for an area-coverage application, the fitness function ($\boldsymbol{F}$) is defined as

$$\boldsymbol{F} = \sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{\alpha}} \left[\left(\frac{\left| \bigcup_{\boldsymbol{n}=\mathbf{1}\ \forall \boldsymbol{x_n}=\boldsymbol{i}}^{\boldsymbol{N}} \boldsymbol{C_n} \right|}{\Delta} \right) = \mathbf{1} \right] \tag{9}$$

In Eq. (9), $\boldsymbol{C_n} \in \{\mathbf{1}, \mathbf{2}, \ldots \Delta\}$ for area-coverage application. For every $\boldsymbol{i^{th}}$ MESS, the coverage of all dimensions in $\boldsymbol{X}$ having same value $\boldsymbol{i}$ is unionized in terms of target points or subregions. The number of unionized target points or subregions must be equal to $\boldsymbol{K}$ or Δ for target-coverage and area-coverage applications, respectively, in order to ensure complete coverage for $\boldsymbol{i^{th}}$ MESS. Therefore for each $\boldsymbol{i^{th}}$ MESS, the statement presented in square brackets must be equal to 1 if the condition is fulfilled, and 0 otherwise. The coverage $\boldsymbol{C}$ of ten sensors in terms of subregions is depicted in **Fig. 2**. Worst case time complexity of fitness evaluation is $\boldsymbol{O}(\boldsymbol{N} \times \boldsymbol{K})$ for target-coverage application and $\boldsymbol{O}(\boldsymbol{N} \times \Delta)$ for area-coverage application, where $\boldsymbol{N}$ shows the number of sensors, $\boldsymbol{K}$ is number of target points for target-coverage problem and Δ is the total number of subregions formed in monitored region for area-coverage application. Similarly, HSAML [32], STHGA [31], and GAMDSC [30] use same equations to evaluate the fitness function for this SET K-Cover problem. Therefore, the computational complexity of three algorithms for calculating the fitness function is same.

Algorithm 1. Formation of Subregion Groups

```
1:  Initialize the set of subregions $\boldsymbol{\Psi}$
2:  Initialize the cell $\boldsymbol{C}$ which stores the sensor coverage for each $\boldsymbol{\beta_i}$ in set $\boldsymbol{\Psi}$
3:  Initialize the set $\boldsymbol{\Psi}_s$ of sorted subregions in ascending order with respect to their
    sensor coverage
4:  Modify the sensor coverage cell $\boldsymbol{C}$ to $\boldsymbol{C_s}$ with respect to $\boldsymbol{\Psi}_s$
5:  Initialize $\boldsymbol{j} = \boldsymbol{0}$
6:  Initialize the total number of subregions in $\boldsymbol{\Psi}_s$ i.e. $\boldsymbol{\Delta} = |\boldsymbol{\Psi}_s|$
7:  while $|\boldsymbol{\Psi}_s| > 0$
8:      Initialize $\boldsymbol{i} = \boldsymbol{1}$
9:      $\boldsymbol{j} = \boldsymbol{j} + \boldsymbol{1}$
10:     Initialize $\boldsymbol{Q_1^j} = \boldsymbol{C_s}\{\boldsymbol{1}\}$
11:     Update $\boldsymbol{C_s}$ i.e., $\boldsymbol{C_s} = \boldsymbol{C_s} - \boldsymbol{C_s}\{\boldsymbol{1}\}$
12:     Store $\boldsymbol{i^{th}}$ subregion from set $\boldsymbol{\Psi}_s$ to group $\boldsymbol{G_s^j}$ i.e. $\boldsymbol{G_s^j(i)} = \boldsymbol{\Psi}_s(\boldsymbol{i})$
13:     for $\boldsymbol{k} = \boldsymbol{2}$ to $\boldsymbol{\Delta}$
14:         Initialize $\boldsymbol{Q_k^j} = \boldsymbol{C_s}\{\boldsymbol{k}\}$
15:         $\boldsymbol{sensors} = \boldsymbol{Q_k^j} \cap \boldsymbol{Q_1^j}$
16:         if $|\boldsymbol{sensors}| > 0$
17:             $\boldsymbol{i} = \boldsymbol{i} + \boldsymbol{1}$
18:             $\boldsymbol{G_s^j(i)} = \boldsymbol{\Psi}_s(\boldsymbol{i})$
19:             Update $\boldsymbol{C_s}$ i.e., $\boldsymbol{C_s} = \boldsymbol{C_s} - \boldsymbol{C_s}\{\boldsymbol{k}\}$
20:         end if
21:     end for
22:     Update $\boldsymbol{\Psi}_s$ i.e., $\boldsymbol{\Psi}_s = \boldsymbol{\Psi}_s - \boldsymbol{G_s^j}$
23:     Update $\boldsymbol{\Delta} = |\boldsymbol{\Psi}_s|$
24: end while
```

4.5 Proposed Random Scheduling Algorithm based on Subregion Coverage

Random Scheduling Algorithm based on Subregion Coverage (RSASC) schedules those sensors which cover subregions of every group step by step. For every subregion in each group, a set of sensors that cover a particular subregion is selected to schedule in two steps. In the first step, all those sensors which are already scheduled, are eliminated from a set of unscheduled sensors. Similarly, assigned schedule numbers are also eliminated from a set of unassigned schedule numbers. In the second step, a randomly selected sensor from a set of unscheduled sensors is scheduled by randomly assigning a schedule number from a set of unassigned schedule numbers. These two steps are repeated for every set of sensors covering a subregion in each group.

Suppose $\boldsymbol{U} = \{\boldsymbol{1}, \boldsymbol{2}, \ldots, \boldsymbol{\alpha}\}$ shows a set of schedule numbers and a set of sensors $\boldsymbol{Q_i} = \{\boldsymbol{S_1}, \boldsymbol{S_2}, \ldots, \boldsymbol{S_q}\} \subseteq \boldsymbol{S}$ covers a subregion $\boldsymbol{\beta_i}$ in subregion group $\boldsymbol{G_s^j}$. Note that the number of sensors covering a subregion will always be greater than or equal to the number of schedules i.e., $\boldsymbol{q} \geq \boldsymbol{\alpha}, \forall \boldsymbol{j} \in \boldsymbol{P}$. From Eq. 8, let $\boldsymbol{X} = \{\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_N}\}$ be the solution where each element $\boldsymbol{x_i}$ indicates an $\boldsymbol{i^{th}}$ sensor in set $\boldsymbol{S}$ and its value indicates a particular schedule number, i.e. $\boldsymbol{x_i} \in \boldsymbol{U}$.

For each subregion in every group, RSASC first sort out the set of already scheduled sensors and their assigned schedule numbers. Suppose $\boldsymbol{E} \in \boldsymbol{Q_i}$ and $\boldsymbol{V} \in \boldsymbol{U}$ denote a set of already scheduled sensors and their assigned schedule numbers, respectively. Similarly, a set of unscheduled sensors, $\boldsymbol{H} \in \boldsymbol{Q_i}$ and unassigned schedule numbers $\boldsymbol{Z} \in \boldsymbol{U}$ can be easily determined by applying the set difference operation using Eq. 10 and 11, respectively.

$$\boldsymbol{H} = \boldsymbol{Q_i} - \boldsymbol{E} \tag{10}$$

$$\boldsymbol{Z} = \boldsymbol{U} - \boldsymbol{V} \tag{11}$$

Algorithm 2. Random Scheduling Algorithm based on Subregion Coverage (RSASC)

```
1:  Find the upper limit ($\boldsymbol{\alpha}$)
2:  Distribute $\boldsymbol{\Delta}$ subregions into P groups using Algorithm 1.
3:  Generate a set $\boldsymbol{U}$ for $\boldsymbol{\alpha}$ unique schedule numbers selected randomly from 1 to $\boldsymbol{\alpha}$
    inclusive using $\boldsymbol{randperm}(\boldsymbol{\alpha}, \boldsymbol{\alpha})$
4:  Initialize the candidate solution X = zeros(1,N)
5:  Initialize $\boldsymbol{r} = \mathbf{0}$, $\boldsymbol{y} = \mathbf{1}$ and $\boldsymbol{repeat} = \mathbf{1000}$ for RSASC repetitions
6:  while $\boldsymbol{y} > 0$ and $\boldsymbol{r} < repeat$
7:      for each subregion group $\boldsymbol{j} = \mathbf{1}$ to P
8:          for each $\boldsymbol{i}$ subregion in group $\boldsymbol{j}$
9:              Initialize a set $\boldsymbol{Q_i}$ for sensors that cover $\boldsymbol{\beta_i}$ subregion in group $\boldsymbol{G_s^j}$ group
10:             Initialize the set $\boldsymbol{E}$ to store already scheduled sensors
11:             Initialize the set $\boldsymbol{V}$ to store assigned schedule numbers for set $\boldsymbol{E}$
12:             for each $\boldsymbol{k}$ sensor in set $\boldsymbol{Q_i}$
13:                 if $\boldsymbol{k}$ is already scheduled i.e. $\boldsymbol{X}(\boldsymbol{k}) > 0$
14:                     Store $\boldsymbol{k}$ in $\boldsymbol{E}$ and schedule $\boldsymbol{X}(\boldsymbol{k})$ in set $\boldsymbol{V}$
15:                 end if
16:             end for
17:             Initialize the set $\boldsymbol{H}$ to store unscheduled sensors using Eq. 11
18:             Initialize the set $\boldsymbol{Z}$ to store unassigned schedule numbers using Eq. 12
19:             if $|\boldsymbol{H}| \geq |\boldsymbol{Z}|$
20:                 for each schedule number $\boldsymbol{z} \in \boldsymbol{Z}$
21:                     Assign $\boldsymbol{z}$ to a unique sensor $\boldsymbol{h} \in \boldsymbol{H}$ i.e., $\boldsymbol{X}(\boldsymbol{h}) = \boldsymbol{z}$
22:                 end for
23:             else
24:                 Set the condition $\boldsymbol{y} = \mathbf{1}$ for repeating RSASC
25:             end if
26:         end for
27:     end for
28:     Increment the repetition number for RSASC i.e., $\boldsymbol{r} = \boldsymbol{r} + \mathbf{1}$
29: end while
30: Evaluate the fitness function
```

Then, RSASC randomly selects a distinct schedule number one by one from set $\boldsymbol{Z}$ and assigns to one of the randomly selected distinct sensor from set $\boldsymbol{H}$. This process is repeated for all $|\boldsymbol{Z}|$ schedules. In most situations, the number of unscheduled sensors is greater than or equal to the number of unassigned schedule numbers i.e., $|\boldsymbol{H}| \geq |\boldsymbol{Z}|$ because the number of sensors covering an $\boldsymbol{i^{th}}$ subregion is always greater than or equal to the value of $\boldsymbol{\alpha}$ i.e.,

$|\boldsymbol{Q_i}| \geq \boldsymbol{\alpha}$. However if $|\boldsymbol{H}| < |Z|$, RSASC repeats its process from the beginning. **Algorithm 2** provides the complete pseudo-code for proposed RSASC.

In example depicted in **Fig. 2**, set $\boldsymbol{U} = \{\mathbf{1},\mathbf{2}\}$ indicates the set of schedule numbers with $\boldsymbol{\alpha} = \mathbf{2}$ and $\boldsymbol{X} = \{\boldsymbol{x_1},\boldsymbol{x_2},\ldots,\boldsymbol{x_{10}}\}$ indicates the candidate solution with $\boldsymbol{N} = \mathbf{10}$ sensors. Similarly, the proposed RSASC distributes the set $\boldsymbol{\Psi} = \{\boldsymbol{\beta_1},\boldsymbol{\beta_2},\ldots,\boldsymbol{\beta_{20}}\}$ into four subregion groups as tabulated in **Table 1**. After formation of subregion groups, the proposed sensor scheduling using RSASC is $\boldsymbol{x_1} = \boldsymbol{S_1} = \mathbf{1}$, $\boldsymbol{x_2} = \boldsymbol{S_2} = \mathbf{2}$, $\boldsymbol{x_3} = \boldsymbol{S_3} = \mathbf{2}$, $\boldsymbol{x_4} = \boldsymbol{S_4} = \mathbf{1}$, $\boldsymbol{x_5} = \boldsymbol{S_5} = \mathbf{1}$, $\boldsymbol{x_6} = \boldsymbol{S_6} = \mathbf{2}$, $\boldsymbol{x_7} = \boldsymbol{S_7} = \mathbf{1}$, $\boldsymbol{x_8} = \boldsymbol{S_8} = \mathbf{2}$, $\boldsymbol{x_9} = \boldsymbol{S_9} = \mathbf{1}$, and $\boldsymbol{x_{10}} = \boldsymbol{S_{10}} = \mathbf{2}$. Similarly, the proposed RSASC creates two MESS as $\boldsymbol{D_1} = \{\boldsymbol{S_1},\boldsymbol{S_4},\boldsymbol{S_5},\boldsymbol{S_7},\boldsymbol{S_9}\}$ and $\boldsymbol{D_1} = \{\boldsymbol{S_2},\boldsymbol{S_3},\boldsymbol{S_6},\boldsymbol{S_8},\boldsymbol{S_{10}}\}$.

4.6 Computational Cost

We evaluate the computational complexity of RSASC, HSAML[32], STHGA[31], and GAMDSC[30] in terms of how many times each algorithm performs the sensor scheduling operations and how many times an algorithm evaluates the fitness function to find the optimal solution. RSASC distributes $\boldsymbol{K}$ targets or Δ subregions into $\boldsymbol{P}$ groups and then it selects each subregion to schedule at most $\boldsymbol{\alpha}$ sensors, which cover that subregion. In other words, we can say that RSASC schedules at most $\boldsymbol{\alpha}$ sensors for each subregion. Since the proposed RSACS is a repetitive algorithm. Thus, we calculate the computational complexity of RSASC in terms of the number of scheduled sensors as $\boldsymbol{O}(\boldsymbol{r} \times \boldsymbol{\alpha} \times \boldsymbol{K})$, and $\boldsymbol{O}(\boldsymbol{r} \times \boldsymbol{\alpha} \times \Delta)$ for the target-coverage and area-coverage applications, respectively, where **r** shows the number of repetitions. After scheduling the sensors covering the subregions, RSASC evaluates the candidate solution only once. This one-time fitness evaluation greatly reduces the computational time of RSASC.

Different from work [32], a harmony consists of $\boldsymbol{N}$ sensors for the execution of HSAML instead of $\boldsymbol{n_1}$ ordinary and $\boldsymbol{m_1}$ energy-harvesting sensors. Moreover, it is fair to ignore local search operation in HSAML, since it tries to ensure the presence of energy harvesting sensors in an active MESS. Therefore, HSAML schedules $\boldsymbol{N}$ sensors for each harmony. Similarly, HSAML divides the initial population into $\boldsymbol{\mu}$ subpopulations. For each subpopulation, HSAML performs fitness evaluation twice after performing uniform crossover if the probability is less than the pre-defined number i.e., harmony memory consideration rate (HMCR), otherwise, it evaluates a new randomly generated harmony. For every $\boldsymbol{g}$ generation, the above process is repeated for all $\boldsymbol{\mu}$ sub-populations. Therefore, we use $\boldsymbol{g\mu}\{\mathbf{2}\boldsymbol{HMCR} + (\mathbf{1} - \boldsymbol{HMCR})\} \cong \mathbf{15.6}\boldsymbol{g}$ to calculate the fitness evaluations where HMCR=0.95 and $\boldsymbol{\mu} = \mathbf{8}$.

STHGA [31] schedules $\boldsymbol{N}$ sensors in every chromosome of recombination and selection operation. $\boldsymbol{K_2}$ Redundant sensors are scheduled in its MST operation, $\boldsymbol{K_1}$ in FST operation and one sensor in CST operation for each chromosome. We ignore the sensors scheduled in mutation operation. Therefore, we find the computational complexity of STHGA in terms of the number of scheduled sensors as follows; it schedules $\boldsymbol{N}$ sensors for each of the $\boldsymbol{m}$ chromosome in recombination and selection operation and $(\boldsymbol{K_1} + \boldsymbol{K_2} + \mathbf{1})$ sensors in its three schedule transformation operations in the worst-case scenario. Thus, on the average, STHGA schedules $\frac{(\mathrm{mN} + \mathrm{m}\,(K_1 + K_2 + 1))}{(2\mathrm{m} + \mathrm{m}(K_1 + K_2 + 1))} \cong \frac{(\mathrm{N} + 11)}{13}\}$ sensors for each chromosome, if $\boldsymbol{K_1} = \boldsymbol{K_2} = \mathbf{5}$ and $\boldsymbol{m} = \mathbf{3}$ as defined in STHGA [31]. Authors of STHGA calculates the number of fitness function evaluations using $\boldsymbol{m}(\mathbf{1} + (\mathbf{2}\boldsymbol{g} + \lfloor \boldsymbol{g}/\boldsymbol{G_m} \rfloor))$, where $\boldsymbol{m}$ is the population size. However, they did not consider the number of fitness evaluations performed for finding the redundant sensors in their three operations of schedule transformation. STHGA evaluates the fitness function for $\boldsymbol{K_2}$ times in MST operation, $\boldsymbol{K_1}$ times in FST and one time in its CST operation for searching redundant sensors. Thus, STHGA actually evaluates the fitness function total $(\boldsymbol{K_1} + \boldsymbol{K_2} + \mathbf{1})$ times in three schedule transformations for each of the $\boldsymbol{m}$ chromosome. After these three operations, it evaluates $\boldsymbol{m}$ chromosomes in the population once again. Therefore, we use $\boldsymbol{m}(\mathbf{1} + (\mathbf{2}\boldsymbol{g} + \boldsymbol{g}(\boldsymbol{K_1} + \boldsymbol{K_2} + \mathbf{1}) + \lfloor \boldsymbol{g}/\boldsymbol{G_m} \rfloor))$ to calculate the fitness evaluations after $\boldsymbol{g}$ generations, where $\boldsymbol{K_1} = \boldsymbol{K_2} = \mathbf{5}$, $\boldsymbol{m} = \mathbf{3}$ and $\boldsymbol{G_m}$=**100**.

GAMDSC [30] schedules $\boldsymbol{N}$ sensors in every chromosome to generate new population and $\boldsymbol{\alpha}$ sensors in its scattering operation. Therefore, it schedules $(\boldsymbol{N} + \boldsymbol{\alpha})$ sensors for each chromosome. Similarly, GAMDSC calculates the fitness evaluations after $\boldsymbol{g}$ generations as $(\boldsymbol{M} + \boldsymbol{gM})$, where $\boldsymbol{M}$ is the population size. In summary, the computational complexity of RSASC is much lesser as compared to the complexities of HSAML, STHGA, and GAMDSC in terms of the number of scheduled sensors and fitness evaluations.

5. Simulation Results and Analysis

We assume a $\mathbf{500} \times \mathbf{500}$ square area for random deployment of all sensors for both target-coverage and area-coverage application. We calculate the grid size $(\boldsymbol{d})$ using $\boldsymbol{L}/\lfloor \boldsymbol{L}/(\boldsymbol{R}/\mathbf{8}) \rfloor$ to find the coverage of each subregion [31]. For each case of sensor deployment, we assume that the energy of each sensor is same i.e., lifetime. We simulate the RSASC, HSAML [32], STHGA [31], and GAMDSC [30] to analyze the obtained results of each algorithm. We use MATLAB 2011b as a simulation tool to perform each experiment. The maximum fitness evaluation limit for HSAML, STHGA, and GAMDSC is 20100. However, if an algorithm searches the optimal solution $(\boldsymbol{\alpha})$, it also stops. Parameter settings for HSAML, STHGA, and GAMDSC are mentioned in [32], [31], and [30], respectively.

We evaluate the performance of each algorithm in terms of Avg. no. of MESS $(\boldsymbol{\alpha_a})$, Avg. no. of utilized sensors $(\boldsymbol{N_u})$, Success rate $(\aleph)$, Avg. no. of fitness function evaluations $(\boldsymbol{E_a})$, Avg. no. of scheduled sensors $(\boldsymbol{N_{ss}})$, and Avg. running time in milliseconds $(\boldsymbol{T_a})$. We obtain these parameter values from the 100 independent executions of each algorithm. Success rate $(\aleph)$ of RSASC and STHGA is 100% i.e. both algorithms find the optimal solution in each of the 100 independent execution for each case whereas, for HSAML and GAMDSC, it is different. Similarly, the avg. no. of utilized sensors $(\boldsymbol{N_u})$ are different for RSASC and STHGA whereas, HSAML and GAMDSC utilize all deployed sensors in their obtained MESSs.

5.1 Comparison for Target-Coverage Application

In this section, we evaluate the effectiveness of our proposed algorithm with the HSAML, STHGA, and GAMDSC for nine target coverage cases. Six sub-figures in **Fig. 5** depict the comparison of obtained results in terms of $\boldsymbol{\alpha_a}$, $\boldsymbol{N_u}$, $\aleph$, $\boldsymbol{E_a}$, $\boldsymbol{N_{ss}}$, and $\boldsymbol{T_a}$ respectively. It can be seen from the **Fig. 5** that RSASC algorithm outperforms the existing STHGA, HSAML, and GAMDSC algorithms in terms of $\boldsymbol{N_u}$, $\boldsymbol{E_a}$, $\boldsymbol{N_{ss}}$ and $\boldsymbol{T_a}$ while achieving the optimality (i.e., $\boldsymbol{\alpha}$) in all cases. Take test case #.7 as an example, in which $\boldsymbol{N} = \mathbf{300}$ sensors with sensing range $\boldsymbol{R} = \mathbf{150}$ and **K=60** target points are randomly deployed which provides an upper bound $\boldsymbol{\alpha} = \mathbf{21}$ for the maximum number of MESS. RSASC searches the required 21 MESS in avg. computational time of $\mathbf{10^{2.697}} = \mathbf{498}$ ms while scheduling the 174 out 300 sensors whereas STHGA finds the same number of MESS in avg. running time of $\mathbf{10^{3.97}} = \mathbf{9333}$ ms with 300 utilized sensors. Whereas, HSAML and GAMDSC take the avg. running time of $(\mathbf{10^{4.444}} = \mathbf{27789}$ ms and $\mathbf{10^{4.696}} = \mathbf{49630}$ ms, respectively) to find the near optimal solution $(\boldsymbol{\alpha_a} = \mathbf{18.9}$ and $\boldsymbol{\alpha_a} = \mathbf{20.9}$, respectively) while utilizing all 300 sensors.

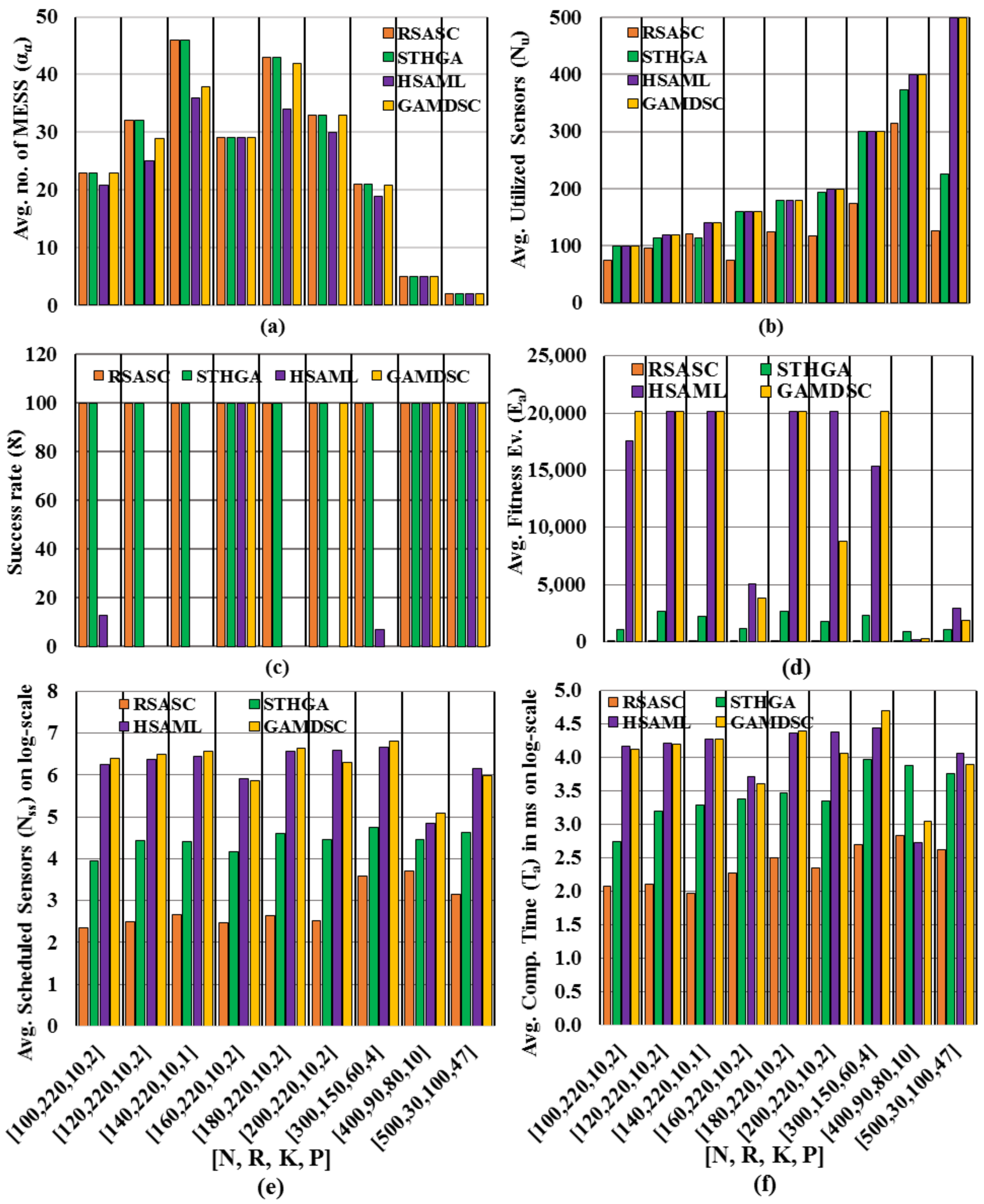

Fig. 5. Results obtained from RSASC, HSAML, STHGA, and GAMDSC for nine target-coverage cases. The x-axis is same for each sub-figure. The data in the square brackets on the x-axis of sub-figure (e) and (f), respectively, represents the value for $\boldsymbol{N}$, $\boldsymbol{R}$, $\boldsymbol{K}$, and $\boldsymbol{P}$, e.g., in 8th case $[\mathbf{N}, \mathbf{R}, \mathbf{K}, \mathbf{P}] = [400,90,80,10]$. Whereas, $\boldsymbol{N}$ is the number of deployed sensors, $\boldsymbol{R}$ shows the sensing range, $\boldsymbol{K}$ indicates the number of targets, and $\boldsymbol{P}$ is the number of target groups created using proposed RSASC. (a) Avg. no. of MESS ($\boldsymbol{\alpha_a}$). (b) Avg. no. of utilized sensors ($\boldsymbol{N_u}$). (c) Success rate (ℵ). (d) Avg. no. of fitness evaluations ($\boldsymbol{E_a}$). (e) Avg. no. of scheduled sensors on log-scale ($\boldsymbol{N_{ss}}$). (f) Avg. running time (milliseconds) on log-scale ($\boldsymbol{T_a}$).

The reason why RSASC takes less running time as compared to STHGA, HSAML, and GAMDSC is that it randomly schedules the $\boldsymbol{\alpha}$ sensors covering each target points and it is the single solution based algorithm, which evaluates the fitness function only for once. Whereas, STHGA, HSAML, and GAMDSC are population-based random search algorithms, which evaluate the fitness function so many times to search an optimal solution. For the same case #.7, RSASC distributes $\boldsymbol{K = 60}$ target points into $\boldsymbol{P = 4}$ groups. In each group, it selects each target point one by one to randomly schedule at most $\boldsymbol{\alpha}$ sensors, which cover that particular target point. Since there are 60 target points and $\boldsymbol{\alpha = 21}$. Therefore, our proposed RSASC schedules total 1260 sensors in its operation. After sensor scheduling operation, RSASC evaluates the candidate solution only once. STHGA finds the optimal solution in 2307 fitness evaluations. As we have already discussed in Section 4.6 that STHGA approximately schedules $\boldsymbol{(\mathrm{N}+11)/13}$ sensors in each chromosome. Therefore, STHGA schedules approximately 55191 sensors since $\boldsymbol{N = 300}$ for test case #.7. The increase in the number of scheduled sensors (55191 > 1260) as well as fitness function evaluations (2307>1) increases the computational time of STHGA as compared to our proposed scheme. For the same test case, HSAML searches approximately 19 MESS in 20100 fitness evaluations while scheduling total $\boldsymbol{4593900 \cong 4.5}$ million sensors at the rate of $\boldsymbol{N = 300}$ sensors per harmony. Similarly, GAMDSC also finds the near-optimal solution ($\boldsymbol{\alpha_a = 20.9}$) in 20100 fitness evaluation with total $\boldsymbol{6452100 \cong 6.45}$ million sensors at the rate of $(N + \alpha) = 171$ sensors per chromosome. In summary, RSASC gives high-quality results in shorter running time as compared to the existing STHGA, HSAML and GAMDSC for the target-coverage application.

5.2 Comparative Analysis for Area-Coverage Application

This section analyzes the performance of each algorithm for 10 area-coverage cases in terms of $\boldsymbol{\alpha_a}$, $\boldsymbol{N_u}$, $\aleph$, $\boldsymbol{E_a}$, $\boldsymbol{N_{ss}}$, and $\boldsymbol{T_a}$, respectively, as depicted in six sub-figures of **Fig. 6**. Area-coverage is a bigger scenario as compared to the target-coverage in which subregions (Δ) can be treated as targets $(\boldsymbol{K})$. The results in **Fig. 6** show that RSASC outperforms the existing STHGA, HSAML, and GAMDSC in terms of $\boldsymbol{N_u}$, $\boldsymbol{E_a}$, $\boldsymbol{N_{ss}}$ and $\boldsymbol{T_a}$ while searching the optimum number of MESS $(\boldsymbol{\alpha})$ in all cases. However, the performance of STHGA is far better than the HSAML and GAMDSC in all test cases except cases #. 5, and 10, where both give the better result than STHGA due to the smaller value of $\boldsymbol{\alpha}$. For example, in case #. 9 as given in **Fig. 6**, $\boldsymbol{N = 900}$ sensors with sensing range R=90 are randomly deployed to monitor whole target region. These 900 sensors distribute the target region into $\Delta = \boldsymbol{2024}$ subregions and provide an upper bound of $\boldsymbol{\alpha = 12}$. For the same test case, the computational time taken by RSASC is $\boldsymbol{10^{3.37} = 2345}$ ms with 391 utilized sensors. Whereas STHGA takes $\boldsymbol{10^{4.951} = 89267}$ ms with 415 utilized sensors to search an optimal solution of $\boldsymbol{\alpha = 12}$. In contrast to RSASC and STHGA, HSAML and GAMDSC find the near-optimal solution in $\boldsymbol{10^{5.265} = 183}$ seconds and $\boldsymbol{10^{5.013} = 103}$ seconds, respectively, with all 900 utilized sensors.

In comparison with the three algorithms, our proposed RSASC is a non-iterative algorithm, which evaluates the fitness function only once. Moreover, the formation of subregion groups helps the RSASC to perform fewer sensor scheduling operations. This one-time fitness function evaluation and fewer number of sensor scheduling operations extraordinarily reduce its computational time. For the same case #.9, RSASC distributes the $\Delta = \boldsymbol{2024}$ subregions into $\boldsymbol{P = 13}$ groups. Each subregion from every group is selected one by one to schedule the $\boldsymbol{\alpha}$ sensors from their sensor coverage. Thus, RSASC performs total $\boldsymbol{2024 \times 12 = 24288}$

operations to search an optimal solution of $\boldsymbol{\alpha} = \mathbf{12}$. STHGA schedules total $\cong$ $\mathbf{416046}$ sensors in 5937 evaluations at the rate of 70 sensors in each evaluation. HSAML is able to find $\boldsymbol{\alpha_a} = \mathbf{10.9}$ after scheduling 18900000 ($\cong$ $\mathbf{18.9}$ million) sensors. Similarly, GAMDSC evaluates the fitness function 20100 times to obtain the near-optimal solution ($\boldsymbol{\alpha_a} = \mathbf{11.9}$) while scheduling 20341200 ($\cong$ $\mathbf{20}$ million) sensors with an average of 1012 sensors per chromosome.

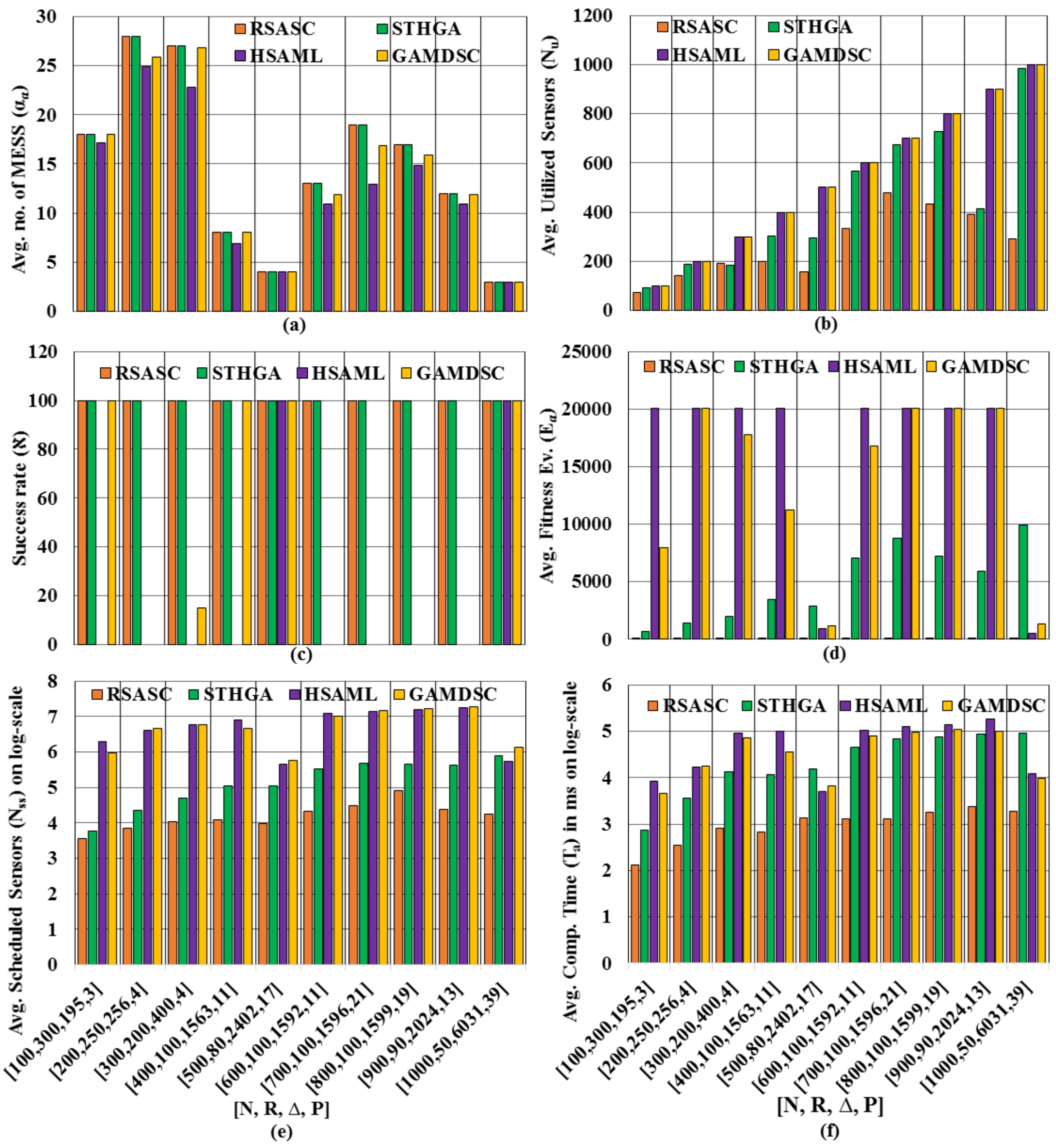

Fig. 6. Results obtained from RSASC, HSAML, STHGA, and GAMDSC for 10 area coverage cases. The x-axis is same for all six sub-figures. The data in the square brackets on the x-axis of (e) and (f), respectively, represents the value for $\boldsymbol{N}$, $\boldsymbol{R}$, Δ, and $\boldsymbol{P}$, e.g., in 5th case $[\mathbf{N}, \mathbf{R}, \Delta, \mathbf{P}] = [500,80,2402,17]$. Whereas, $\boldsymbol{N}$ is the number of deployed sensors, $\boldsymbol{R}$ shows the sensing range, Δ indicates the number of created subregions, and $\boldsymbol{P}$ is the number of target groups created using proposed RSASC. (a) Avg. no. of MESS ($\boldsymbol{\alpha_a}$). (b) Avg. no. of utilized sensors ($\boldsymbol{N_u}$). (c) Success rate (ℵ). (d) Avg. no. of fitness evaluations ($\boldsymbol{E_a}$). (e) Avg. no. of scheduled sensors on log-scale ($\boldsymbol{N_{ss}}$). (f) Avg. running time (milliseconds) on log-scale ($\boldsymbol{T_a}$).

As we have already discussed in Section 5.1 that formation of subregion groups helps the RSASC to search the optimal solution with higher optimization speed. Therefore, for the same case #.9, we analyze the significance of the formation of subregion groups how effectively it guides the RSASC to quickly achieve the maximum number of MESS. Fig. 7 depicts the simultaneous growth in fitness values of all $\boldsymbol{\alpha} = \mathbf{12}$ MESSs after the coverage of each subregion group using the proposed RSASC. The number of subregions in each group is also listed on the right side of the figure. In 1st subregion group $\boldsymbol{G_s^1}$, total 144 out of 2024 subregions are grouped. Similarly, 2nd and 3rd group consist of 161 and 169 subregions, respectively, and so on. After the formation of subregion groups, RSASC selects 1st subregion from $\boldsymbol{G_s^1}$ and randomly schedules at most $\boldsymbol{\alpha} = \mathbf{12}$ sensors one by one which covers the 1st subregion. For the 2nd subregion in $\boldsymbol{G_s^1}$, it first sorts out such sensors from the sensor coverage of 2nd subregion which have already scheduled and their assigned schedule numbers. Then, it randomly assigns remaining schedule numbers in succession to one of the remaining sensors to ensure the coverage of 2nd subregion for each MESS. This process is repeated for all 144 subregions in $\boldsymbol{G_s^1}$. Similary, the above process is repeated for all $\boldsymbol{P} = \mathbf{13}$ subregion groups.

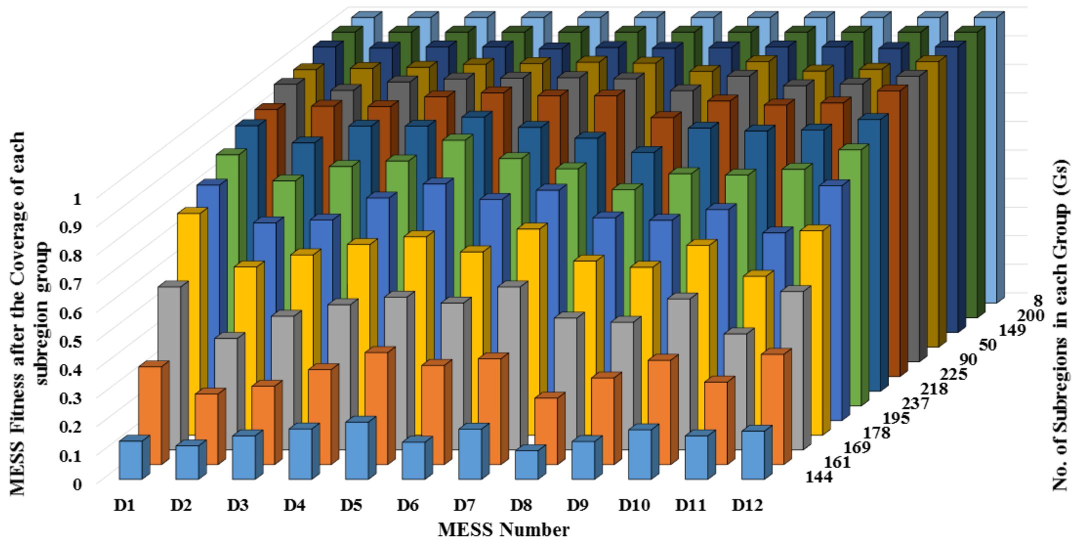

Fig. 7. Simultaneous growth in fitness value of each MESS after the coverage of each subregion group using the proposed RSASC for the 9th test case shown in Fig. 7 in which $\boldsymbol{N} = \mathbf{900}$ sensors with sensing range $\boldsymbol{R} = \mathbf{90}$ divides the target region into $\Delta = \mathbf{2024}$ subregions. The proposed RSASC distributes the $\Delta = \mathbf{2024}$ subregions into $\boldsymbol{P} = \mathbf{13}$ groups. The right side of the figure also lists the number of subregions in each group.

It can be seen from the Fig. 7 that how fitness value of each MESS grows simultaneously after the coverage of each subregion group. The fitness of each MESS gets improved continuously when the coverage of subregion groups from $\boldsymbol{G_s^1}$ to $\boldsymbol{G_s^7}$ is fulfilled by using RSASC. After ensuring the coverage of subregion group $\boldsymbol{G_s^8}$, first MESS ($\mathbf{D_{12}}$) is constituted which completely covers the target area. Whereas, at the same time, the fitness of other MESSs reaches to more than 90%. Another MESS ($\mathbf{D_9}$) achieves its 100% fitness value after the coverage of $\boldsymbol{G_s^9}$ while other MESSs improve their fitness. Three new schedule numbers $\mathbf{D_1}$, $\mathbf{D_3}$, and $\mathbf{D_{10}}$ achieve their 100% fitness after the coverage of $\boldsymbol{G_s^{10}}$ to $\boldsymbol{G_s^{11}}$. All remaining seven MESSs achieve their 100% fitness when RSASC covers the group $\boldsymbol{G_s^{12}}$ and $\boldsymbol{G_s^{13}}$.

5.3 Effect of Redundant Sensors on Optimization Difficulty of RSASC

In addition to the number of sensors N, their positions and sensing ranges, redundant sensors ($\boldsymbol{N_R}$) also play a vital role to calculate the optimization difficulty of an algorithm for this set K-cover problem. Since both, RSCAC and STHGA, find the optimal solution with 100% success rate. Therefore, in this subsection, we only analyze the optimization difficulty of RSCAC and STHGA by reducing the number of redundant sensors for the test case #.10 in Fig. 6. Initially, $\boldsymbol{N} = \mathbf{1000}$ deployed sensors divide the target region into $\Delta = \mathbf{6031}$ subregion with an upper bound of $\boldsymbol{\alpha} = \mathbf{3}$. We remove all the redundant sensors step by step and record the results of both algorithms. Table 2 tabulates the seven cases of removed redundant sensors ($\boldsymbol{N_R}$), remaining deployed sensors $\boldsymbol{N}$, the updated number of subregions Δ and the results obtained from both algorithms. We limit the number of repetitions for RSASC to 1000. Similarly, for STHGA, the maximum number of fitness evaluations is set to 20100.

Table 2. Results for the optimization difficulty of RSASC and STHGA considering seven different scenarios of redundancy for area-coverage case #. 10 in Fig. 6.

Cases				RSASC ($\boldsymbol{repeat} = \mathbf{1000}$)			STHGA with $\boldsymbol{m} = \mathbf{3}$			
#.	$\boldsymbol{N_R}$	$\boldsymbol{N}$	Δ	$\boldsymbol{N_u}$	$\boldsymbol{\alpha_a}$	$\boldsymbol{T_a(s)}$	$\boldsymbol{N_u}$	$\boldsymbol{\alpha_a}$	$\boldsymbol{E_a}$	$\boldsymbol{T_a(s)}$
1	100	900	5947	611	3	**2.783**	763	3	5235	297
2	200	800	5772	514	3	**2.176**	578	3	13704	517
3	300	700	5449	414	3	**2.753**	534	3	14796	356
4	400	600	5236	324	3	**2.472**	595	2.96	20100	427
5	500	500	4650	225	3	**2.201**	500	1.99	20100	391
6	600	400	3985	129	3	**2.189**	400	1.99	20100	244
7	761	239	2265	0	0	1278	239	1.96	20100	182

It can be seen from the Table 2 that RSASC finds the optimal solution in six out of seven cases, whereas STHGA is only able to find the optimal solution in three out of seven test cases in the specified limit of fitness evaluations. STHGA performs well as compared to proposed RSASC for an only 7th case where all redundant sensors are removed. Notice that the situation of sensor deployment like case #.7 is rare since a large number of sensors are deployed in random deployment to achieve the coverage of target region. The reason why proposed RSASC gives $\boldsymbol{\alpha} = \mathbf{0}$ after 1000 repetitions is that there might be a condition in which the number of unassigned schedules ($\boldsymbol{Z}$) is greater than the number of unscheduled sensors ($\boldsymbol{H}$). In such situations, it repeats the whole process. However, in contrast with the RSASC, STHGA finds the solution of $\boldsymbol{\alpha} = \mathbf{1.961}$ in 20100 fitness evaluations in 182 seconds. Failure of STHGA to find the optimal solution is obvious, since it relies on redundant sensors to transform their schedules in its three transition operations. In summary, results show that proposed RSASC outperforms the STHGA.

6. Conclusion

In this paper, we propose a novel Random Scheduling Algorithm based on the Subregion Coverage (RSASC) to solve famous SET K-cover problem to maximize the network lifetime under target-coverage and area-coverage constraint. Firstly, the proposed RSASC distributes the Δ subregions into $\boldsymbol{P}$ disjoint groups according to their similar sensing coverage as discussed in Subsection 4.1. Then for each group, RSASC selects a subregion one by one and schedules at most Δ sensors, which cover that subregion. Simulation results show that RSASC outperforms the existing algorithms in terms of greater computational speed (saves on average more than the 93% of the time) as well as the number of utilized sensors while achieving the optimal solution. We will extend this research for lifetime maximization of dynamic WSN. Since, our proposed RSASC achieves the maximum number of MESS with greater computational speed.

Acknowledgement

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education [NRF-2016R1D1A1B03935633]. This work was also supported by the IITP grant funded by the Korea government (MSIT) [No. 2016-0-00133] and by the ICT R&D program of MSIT/IITP [2014-0-00077, Development of global multi-target tracking and event prediction techniques based on real-time large-scale video analysis]. Chang Wook Ahn and Navrati Saxena have equally contributed to this paper.

References

[1] Ian F. Akyildiz, Tommaso Melodia and Kaushik R. Chowdhury, “A survey on wireless multimedia sensor networks,” *Computer Networks: The Int. Jr. of Comp. and Telecom. Netw.*, vol. 51, no. 4, pp. 921--960, March, 2007. Article (CrossRef Link)

[2] Bang Wang, “Coverage problems in sensor networks: A survey,” *ACM Computing Surveys (CSUR)*, vol. 43, no. 4, pp.32:1--32:53, October, 2011. Article (CrossRef Link)

[3] Yourim Yoon and Yong-Hyuk Kim, “An efficient genetic algorithm for coverage deployment in wireless sensor networks,” *IEEE Transactions on Cybernetics*, vol. 43, no. 5, pp.1473-1483, October, 2013. Article (CrossRef Link)

[4] S. Mini, Siba K. Udgata and Samrat L. Sabat, “Sensor deployment and scheduling for target coverage problem in wireless sensor networks,” *IEEE Sensors Journal*, vol. 14, no. 3, pp.636-644, March, 2014. Article (CrossRef Link)

[5] Chun-Wei Tsai, Tzung-Pei Hong and Guo-Neng Shiu, “Metaheuristics for the lifetime of WSN: A Review,” *IEEE Sensors Journal*, vol. 16, no. 9, pp. 2812-2831, May, 2016.
Article (CrossRef Link)

[6] Mrinmoy Bhattacharjee and S. Gupta, “Determining redundant nodes in a location unaware wireless sensor networks,” in *Proc. of IEEE Int. Conf. on Adv. Comm. Contr. and Comp. Tech.*, pp.858-862, 8-10 May, 2014. Article (CrossRef Link)

[7] Dimitrios Zorbas, Dimitris Glynos, Panayiotis Kotzanikolaou and Christos Douligeris, “Solving coverage problems in wireless sensor networks using cover sets,” *Ad Hoc Networks*, vol. 8, no. 4, pp.400-415, June, 2010. Article (CrossRef Link)

[8] Mohammad Younis and Kemal Akkaya, “Strategies and techniques for node placement in wireless sensor networks: A survey,” *Ad Hoc Networks*, vol. 6, no. 4, pp.621-655, June, 2008.
Article (CrossRef Link)

[9] Giuseppe Anastasi, Marco Conti, Mario Di Francesco and Andrea Passarella “Energy conservation in wireless sensor networks: A survey,” *Ad Hoc Networks*, vol. 7, no. 3, pp.537—568, May, 2009. Article (CrossRef Link)

[10] Vijay Raghunathan, Curt Schurgers, Sung Park and Mani B. Srivastava, “Energy-aware wireless micro sensor networks,” *IEEE Signal Processing Magazine.*, vol. 19, no. 2, pp.40-50, March, 2002. Article (CrossRef Link)

[11] Yohong Zhang and Wei Li “Modeling and energy consumption evaluation of a stochastic wireless sensor network,” *EURASIP J. Wireless Communication and Networking*, vol. 2012, pp.282, December, 2012. Article (CrossRef Link)

[12] Zaixin Lu, Wei Wayne Li and Miao Pan, “Maximum Lifetime Scheduling for Target Coverage and Data Collection in Wireless Sensor Networks,” *IEEE Transactions on Vehicular Technology*, vol. 64, no. 2, pp.714-727, February, 2015. Article (CrossRef Link)

[13] S. Slijepcevic and M. Potkonjak, “Power efficient organization of wireless sensor networks,” in *Proc. of IEEE Int. Conf. on Communications*, pp.472-476, 11-14 June, 2001. Article (CrossRef Link)

[14] Mihaela Cardei and Ding-Zhu Du, “Improving wireless sensor network lifetime through power aware organization,” *Wireless Networks*, vol. 11, no. 3, pp.333-340, May, 2005. Article (CrossRef Link)

[15] Joaquin Aparicio, Juan Jose Echevarria and Jon Legarda, “A Software Defined Networking Approach to Improve the Energy Efficiency of Mobile Wireless Sensor Networks,” *KSII Transactions on Internet and Information Systems*, vol. 11, no.6, pp. 2848-2869, April, 2017. Article (CrossRef Link)

[16] Naranjo, P. G. V., Shojafar, M., Mostafaei, H., Pooranian, Z., & Baccarelli, E. “P-SEP: a prolong stable election routing algorithm for energy-limited heterogeneous fog-supported wireless sensor networks,” *The Journal of Supercomputing*, *73*(2), pp. 733-755, 2017. Article (CrossRef Link)

[17] Ahmadi, A., Shojafar, M., Hajeforosh, S. F., Dehghan, M., & Singhal, M. “An efficient routing algorithm to preserve k-coverage in wireless sensor networks,” *The Journal of Supercomputing*, *68*(2), pp. 599-623, 2014. Article (CrossRef Link)

[18] Tajiki, M. M., Salsano, S., Shojafar, M., Chiaraviglio, L., & Akbari, B. “Joint Energy Efficient and QoS-aware Path Allocation and VNF Placement for Service Function Chaining,” *arXiv preprint arXiv:1710.02611*, 2017.

[19] Razieh Sheikhpour and Sam Jabbehdari, “An Energy Efficient Chain-based Routing Protocol for Wireless Sensor Networks,” *KSII Transactions on Internet and Information Systems*, vol. 7, no.6, pp. 1357-1378, June, 2013. Article (CrossRef Link)

[20] Ying Zhou, Lihua Yang, and Longxiang Yang, “Improved Compressed Network Coding Scheme for Energy-Efficient Data Communication in Wireless Sensor Networks,” *KSII Transactions on Internet and Information Systems*, vol. 11, no.6, pp. 2946-2962, June, 2017. Article (CrossRef Link)

[21] Muruganantham ARUNRAJA and Veluchamy MALATHI, “Collective Prediction Exploiting Spatio Temporal Correlation (CoPeST) for Energy Efficient Wireless Sensor networks,” *KSII Transactions on Internet and Information Systems*, vol. 9, no.7, pp. 2488-2511, July, 2015. Article (CrossRef Link)

[22] Z. Muhammad, N. Saxena, I.M. Qureshi and C.W. Ahn, “Hybrid Artificial Bee Colony Algorithm for an Energy Efficient Internet of Things based on Wireless Sensor Network,” *IETE Technical Review*, Vol. 0, No. 0, pp. 1-13, 2017. Article (CrossRef Link)

[23] Zeyu Sun, Yongsheng Zhang, Xiaofei Xing, Houbing Song, Huihui Wang and Yangjie Cao, “EBKCCA: A Novel Energy Balanced k-Coverage Control Algorithm Based on Probability Model in Wireless Sensor Networks,” *KSII Transactions on Internet and Information Systems*, vol. 10, no.8, pp. 3621-3640, August, 2016. Article (CrossRef Link)

[24] Yifei Qi, Peng Cheng, Jing Bai, Jiming Chen, Adrien Guenard, Ye-Qiong Song and Zhiguo Shi, “Energy-Efficient Target Tracking by Mobile Sensors With Limited Sensing Range,” *IEEE Transactions on Industrial Electronics*, vol. 63, no. 11, pp.6949-6961, November, 2016. Article (CrossRef Link)

[25] K. Senthil Kumar and R. Amutha, “An Algorithm for Energy Efficient Cooperative Communication in Wireless Sensor Networks,” *KSII Transactions on Internet and Information Systems*, vol. 10, no.7, pp. 3080-3099, July, 2016. Article (CrossRef Link)

[26] Di Tian, Nicolas D. Georganas, “A coverage-preserving node scheduling scheme for large wireless sensor networks,” in *Proc. of the 1st ACM Int. Workshop on Wireless Sensor Networks and applications*, pp.32-41, September 28-28, 2002. Article (CrossRef Link)

[27] John H. Holland, “*Adaptation in Natural and Artificial Systems*,” MIT Press Cambridge, MA, USA, 1992.

[28] Ruchi Sachan, Zahid Muhammad, Jaehoon (Paul) Jeong, Chang Wook Ahn, and Hee Yong Youn, "MABC: Power-Based Location Planning with a Modified Artificial Bee Colony Algorithm for 5G Networks," *Discrete Dynamics in Nature and Society*, 4353612, pp.13, 2017. Article (CrossRef Link)
[29] Zbigniew Michalewicz, "*Genetic Algorithm + Data Structures = Evolution Programs*," Springer-Verlag, Berlin Heidelberg, 1996. Article (CrossRef Link)
[30] Chih-Chung Lai, Chaun-Kang Ting, and Ren-Song Ko, "An effective genetic algorithm to improve wireless sensor network lifetime for large-scale surveillance applications," *IEEE Congr. on Evolutionary Computations*, 25-28 September, 2007.Article (CrossRef Link)
[31] Xiao-Min Hu, Jun Zhang, Yan Yu, Henry Shu-Hung Chung, Yuan-Long Li, Yu-Hui Shi, and Xiao-Nan Luo, "Hybrid genetic algorithm using a forward encoding scheme for lifetime maximization of wireless sensor networks," *IEEE Transactions on Evolutionary Computations*, vol. 14, no. 5, pp.766-781, October, 2010. Article (CrossRef Link)
[32] Lin, C. C., Chen, Y. C., Chen, J. L., Deng, D. J., Wang, S. B., & Jhong, S. Y. "Lifetime Enhancement of Dynamic Heterogeneous Wireless Sensor Networks with Energy-Harvesting Sensors," *Mobile Networks and Applications*, pp. 1-12, 2017. Article (CrossRef Link)
[33] Chi-Fu Huang, and Yu-Chee Tseng, `The coverage problem in a wireless sensor networks," *Mobile Networks and Applications*, vol. 10, no. 4, pp. 519-528, August, 2005. Article (CrossRef Link)

Zahid Muhammad is a PhD student in the department of Electrical & Computer Engineering at Sungkyuwan University, Suwon, South Korea. He received his B.E. and M.S. degrees from Mehran University of Engineering & Technology and ISRA University in 2006 and 2012, respectively. His research interests are Evolutionary Algorithms, Optimization Techniques and their applications in IoTs and WSNs.

Abhishek Roy is currently working in the Networks Division, Samsung Electronics, South Korea. He received his Ph.D. in 2010 from Sungkyunkwan University. His research interests include mobility and resource management in 4G/5G wireless systems, IoTs, and smart grids. He serves as a Guest Editor and Technical Program Committee member of many international journals and conferences. He has co-authored one book (Taylor & Francis) and published in more than 50 international journals.

Chang Wook Ahn is currently working in the Department of Electrical Engineering and Computer Science, GIST as a Professor. He received his Ph.D. degree in 2005 from Gwangju Institute of Science and Technology (GIST), Korea. He has published over 70 journal and conference papers. His main research areas are evolutionary algorithms, machine learning, and intelligent wireless networks and estimation of distribution algorithms.

Ruchi Sachan got her M.S. Ph.D. degrees from Sungkyunkwan University, Korea in 2014 and 2017, respectively. Her primary area of research interest includes wireless networks, energy consumption as a green networking with applications of evolutionary algorithms and machine learning.

Navrati Saxena is an associate professor and director of Mobile Ubiquitous System Information Center (MUSIC) in Software Department, Sungkyunkwan University, Korea. She got her PhD from University of Trento, Italy. Her research interests involve 4G/5G wireless, IoT and smart grids. She serves in guest editorial and technical program committee of international journals and conferences. She has co-authored one book (Taylor & Francis) and published more than 60 international journals.