Heuristic Search Algorithm (HSA) for Enhancing the Lifetime of Wireless Sensor Networks

Tripat jot S. Panag, J. S. Dhillon

Abstract—The lifetime of a wireless sensor network can be effectively increased by using scheduling operations. Once the sensors are randomly deployed, the task at hand is to find the largest number of disjoint sets of sensors such that every sensor set provides complete coverage of the target area. At any instant, only one of these disjoint sets is switched on, while all other are switched off. This paper proposes a heuristic search method to find the maximum number of disjoint sets that completely cover the region. A population of randomly initialized members is made to explore the solution space. A set of heuristics has been applied to guide the members to a possible solution in their neighborhood. The heuristics escalate the convergence of the algorithm. The best solution explored by the population is recorded and is continuously updated. The proposed algorithm has been tested for applications which require sensing of multiple target points, referred to as point coverage applications. Results show that the proposed algorithm outclasses the existing approaches.

Keywords—Coverage, disjoint sets, heuristic, lifetime, scheduling, wireless sensor networks, WSN.

I. INTRODUCTION

A Wireless Sensor Network (WSN) consists of a large number of densely deployed sensors which work collectively to provide information about the target field. The applications of wireless sensor networks include monitoring the battlefields, exploring the deep oceans, tracking the animals in their natural habitats etc. [1]-[8]. Most of these applications require the sensor nodes to be placed in hostile environments, making the replacement of their batteries difficult or practically impossible. The lifetime of a WSN with limited energy resources can be increased if these resources are used in a controlled and intelligent way. This is an important topic of research [1], [2], [8]-[11].

The first step in setting up of a WSN is the deployment of sensor nodes. Many methods [12]-[17] have been proposed to deploy the sensors in a planned manner in order to achieve complete coverage of the monitored area with minimum number of sensors. It has also been proposed to dispatch a set of mobile sensor nodes to satisfy the coverage and connectivity requirements [7], [16], [18], [19]. For applications where the area to be monitored is too large or where the batteries of the sensor nodes cannot be replaced, random deployment is the preferred choice. It is achieved by spreading the sensors randomly all over the monitored area [9], [10], [20]. In random deployment, the number of sensors required to achieve complete coverage is more than the planned methods. As the technological advancements have considerably reduced the cost of sensor nodes [1], [8], [10], random deployment has become economically feasible. Practically, the number of sensor nodes deployed is much more than the number required for achieving complete coverage. This adds redundancy. Redundant deployment helps in prolonging the lifetime of a WSN. Many methods [21]-[29] have been proposed to intelligently use this redundancy for setting up of energy efficient wireless sensor networks. Redundant deployment also allows identification of redundant backbones for data communication tasks and helps setting up of minimum k-connected, m-dominating set wireless sensor networks [30].

With redundant deployment, more sensors than one monitor any portion of the area under surveillance. This implies that the region can be monitored by switching on any one of them and switching off the rest. Thus only a subset of the deployed sensors needs to be active to completely monitor any area [8]-[11], [31]-[35] at a time. Therefore, the sensors are divided into disjoint sets, each providing a complete coverage of the monitored area. Many methods [2], [8]-[11], [31], [34]-[45] have been proposed to divide the sensors into such mutually exclusive sets. Instead of using any one subset till its energy exhausts and then switching on to the other, it is better to use them in a cyclic manner in order to increase the lifetime of the wireless sensor network. This follows from the fact that the battery life doubles, if it is used in short bursts separated by significant off time than in a continuous mode of operation [46]. Using various disjoint sets one after the other in a cyclic manner is referred to as scheduling. For maximizing the lifetime of a WSN, scheduling requires that the sensors be divided into the maximum number of sets subject to the condition that each of these sets provides complete coverage of the target area.

Many methods [9], [10], [31], [47] have been proposed for scheduling of sensors in a WSN. Cardie and Du [10] proposed a “maximum covers using mixed integer programming (MC-MIP)” algorithm. It is based on a branch and bound method to make an implicit exhaustive search and guarantees to find an optimal solution to the problem. However, the execution time of MC-MIP increases exponentially as the number of sensors and targets becomes larger [9]. Another heuristic, named
“most constrained minimally constraining covering (MCMCC)” heuristic has been proposed by [31]. It uses a greedy deterministic approach. Hu et al. [9] suggest that it works much faster than MC-MIP for large scale problems, but it does not guarantee to find an optimum solution.

Genetic algorithms (GAs) [48] are population based search algorithms that simulate biological evolution process and have successfully been used to solve a wide range of NP-hard optimization problems [49]-[53]. Lai et al. [47] proposed a GA named “genetic algorithm for maximum disjoint set covers (GAMDSC)”. It encodes each gene in the chromosome as an integer index of the set that the sensor has joined. GAMDSC is suitable only when the number of targets and sensors is small. Hu et al. [9] proposed a “schedule transition hybrid genetic algorithm (STHGA)” that uses a combination of genetic algorithm and schedule transition operations to address the problem of finding the largest number of disjoint complete sets. STHGA starts by placing all the sensors in a single set so that it is definitely a complete set. It then uses a forward encoding scheme to move the redundant sensors of the first set to the second set. STHGA clubs forward encoding with genetic algorithm and sensor schedule transition operations to complete the second set. It then moves the redundant sensors of these two sets to the third set. The process continues until the maximum number of disjoint complete cover sets has been created. Hu et al. [9] have compared the performance of STHGA with GAMDSC and MCMCC. It has been highlighted that STHGA always achieves the desired results whereas GAMDSC and MCMCC do not always do so. STHGA has also been reported to be faster than the other two. The number of fitness function evaluations made by STHGA is always considerably less than GAMDSC.

In this paper, a heuristic search algorithm (HSA) is proposed to solve the disjoint set covers problem for maximizing the lifetime of a wireless sensor network. The algorithm uses a group of members to explore the entire search space in order to find an optimum solution. An optimum solution is the one which divides the sensors into maximum number of disjoint complete cover sets. The number of dimensions of the search space is equal to the number of sensors deployed. Each dimension in the search space corresponds to a sensor and its value indicates the set in which the sensor is scheduled to be switched on. Thus each dimension of a member can have an integer value in the range from 1 to the maximum number of possible disjoint complete sets (determined by the number of sensors covering a critical field as discussed later).

The HSA starts by randomly initializing all the members. Then the coverage of all the cover sets in each of the members is evaluated. This is followed by fitness function evaluation of each member. The fitness function has been designed to give more weight to the complete sets so that a member having more number of complete sets MUST have higher fitness. Then three moves are applied heuristically one after the other. These moves have been designed to move the redundant sensors from one set to another. The selection of sensors for movement and the determination of new sets is done randomly to explore the entire space. The directed moves aim at improving the convergence of the algorithm. The performance of the proposed HSA has been compared with STHGA [9]. Results show that HSA always finds the optimum solution like STHGA, but at a much faster optimization speed.

II. PROBLEM DEFINITION

Let \( N \) sensors be randomly deployed in a rectangular region \( A \times B \) (Fig. 1). Each sensor is assumed to have a circular coverage area with radius \( R \). Each sensor has a finite lifetime. The aim is to maximize the time for which the area is monitored. Let the number of sensors deployed be too large than the number of sensors required to achieve complete coverage of the area. Thus only a subset of the sensors deployed can provide complete coverage.

![Fig. 1 N sensors with circular coverage deployed in A × B rectangular area](image)

If the deployed sensors are represented by a set \( S = \{s_1, s_2, \ldots, s_N\} \), and the maximum number of disjoint sets that can be created by \( \beta \), the objective is to find the maximum number of subsets \( S_k, k = 1, 2, \ldots, \beta \) of \( S \) so that each subset provides complete coverage of the target area. Further, \( S_i \cap S_l = \phi \); \( k \neq l \) and \( k, l = 1, 2, \ldots, \beta \).

Once the sensors are divided into subsets \( S_k, k = 1, 2, \ldots, \beta \), as per the criteria given above, sensors in only one of the subsets shall be activated at any instant of time. After some time that subset of sensors shall be switched off and another subset shall be switched on. Thus sensors in a sensor subset \( S_k \) shall be activated in the \( k^{th} \) schedule.

In the illustration of Fig. 1, each sensor \( s_i \) has been assumed to have a circular coverage area of radius \( R \). In practice, however, the sensor coverage area can be of any shape. In this work, for simplicity the coverage area has been assumed to be circular.

Field: Referring to Figs. 1 and 2 it can be observed that the areas covered by sensors overlap each other, forming separate fields [32]. A field can thus be defined as the area that is covered by the same set of sensors e.g. in Fig. 2 three sensors combine together to form seven fields.
TABLE I

<table>
<thead>
<tr>
<th>Notations</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_k$</td>
<td>Sensor $k$</td>
</tr>
<tr>
<td>$f_i$</td>
<td>Fitness value of member $i$, $i = 1, 2, ..., M$</td>
</tr>
<tr>
<td>$f_{best}$</td>
<td>Fitness value of the best member</td>
</tr>
<tr>
<td>$A$</td>
<td>Width of the target area</td>
</tr>
<tr>
<td>$B$</td>
<td>Length of the target area</td>
</tr>
<tr>
<td>$N$</td>
<td>Number of sensors</td>
</tr>
<tr>
<td>$R$</td>
<td>Sensing range of each sensor</td>
</tr>
<tr>
<td>$M$</td>
<td>Number of members in a population</td>
</tr>
<tr>
<td>$K_i$</td>
<td>Number of sensors selected during the redundant move</td>
</tr>
<tr>
<td>$K_i^C$</td>
<td>Number of sensors selected during the completion move</td>
</tr>
<tr>
<td>$N_{F}$</td>
<td>Total number of fields in the region that are to be covered</td>
</tr>
<tr>
<td>$N_{CF}$</td>
<td>Number of critical fields i.e. the number of fields that are covered by $\beta$ sensors</td>
</tr>
<tr>
<td>$N_{MAX}$</td>
<td>Maximum number of iterations</td>
</tr>
<tr>
<td>$C_i$</td>
<td>Number of complete sets in member $i$, $i = 1, 2, ..., M$</td>
</tr>
<tr>
<td>$C_{best}$</td>
<td>Number of complete sets in the best member</td>
</tr>
<tr>
<td>$V_i$</td>
<td>Coverage percentage of $i^{th}$ incomplete set in member $i$</td>
</tr>
<tr>
<td>$H(i,j)$</td>
<td>Gives the location of the $j^{th}$ member in the $i^{th}$ dimensional search space. Each dimension represents a sensor and its value in the range 1 to $\beta$ indicates the set in which the sensor is scheduled to be switched on</td>
</tr>
<tr>
<td>$S$</td>
<td>Set of sensors deployed in the target area</td>
</tr>
<tr>
<td>$S_i$</td>
<td>Subset of $S$</td>
</tr>
<tr>
<td>$k = 1, 2, ..., \beta$</td>
<td>Number of sensors covering a critical field</td>
</tr>
</tbody>
</table>

Fig. 2 Circular coverage areas of 3 sensors combine together to form 7 fields

**Critical Field**: A field that is covered by the minimum number of sensors is referred to as the critical field and the sensors covering that critical field are called critical sensors.

If a critical field is covered by $\beta$ sensors, then at least one of these $\beta$ sensors must be active at any instant of time to provide complete coverage of the target area. Thus we can have at most $\beta$ number of disjoint sets of sensors which provide complete coverage to the target area.

Let $N$ sensors combine together to form $N_{F}$ fields. Let $F_{b}$, $k=1, 2, ..., N_{F}$ represent the number of sensors that cover the $k^{th}$ field, then

$$\beta = \min \{ F_{k}; k = 1, 2, ..., N_{F} \}$$  (1)

In Fig. 2 the values of $F_{1}$, $F_{2}$, $F_{3}$, $F_{4}$, $F_{5}$ respectively are 1, 2, 3, 2, 1, 2 and 1. Thus the value of $\beta$ is 1.

III. PROPOSED DIRECTED RANDOM SEARCH ALGORITHM

The algorithm being proposed in this paper solves the problem at hand by using random search clubbed with some directive moves. A complete flow chart of the proposed algorithm is shown in Fig. 3. The algorithm uses $M$ members to explore an $N$-dimensional space, where $N$ is the number of sensors deployed. In this section, first the representation and initialization of the members is discussed, followed by the calculation of coverage percentage of a set, evaluation of the population and the directive moves.

A. Representation of Members

The initial population comprises of $M$ members, each having $N$ dimensions. Each member is a candidate solution for the problem. Every dimension of a member corresponds to a sensor. A member is represented as:

$$I(i) = [a_{ij}, a_{i2}, ..., a_{in}]^{T}; \quad (i = 1, 2, ..., M)$$  (2)

where $a_{ij} \in \{1, 2, ..., \beta \}; i = 1, 2, ..., M; j = 1, 2, ..., N$. The value $a_{ij}$ indicates the set or the schedule in which the $j^{th}$ sensor shall be switched on in member $i$. Thus each member divides the $N$ sensors into $\beta$ disjoint sets.

B. Initialization

As the critical field is covered by $\beta$ sensors, the sensors may at the most be divided into $\beta$ disjoint sets so that each of the sets completely covers all the fields. The algorithm starts by randomly placing the sensors in a member into any of the $\beta$ sets as:

$$I(i,j) = \text{roundoff} \left( (1 + (\beta - 1) \times \text{uniran()} \right)$$

$$\quad (i = 1, 2, ..., M; j = 1, 2, ..., N)$$  (3)

$\text{uniran()}$ is a function which returns a uniform random number in the range [0-1]. Thus, the sensors are divided into $\beta$ disjoint sets.

C. Fitness Evaluation

The percentage coverage of each of the sets is to be evaluated which in turn requires the calculation of the percentage coverage of each sensor. As the determination of exact percentage coverage of the sensors is difficult, an approximation is made by dividing the area into grids [9]. A set having hundred percent coverage is termed as a complete set. The aim of the algorithm is to make all the sets complete i.e. to divide the sensors into $\beta$ disjoint complete cover sets.

The fitness of the $i^{th}$ member in the population is evaluated as

$$f_i = C_i + \sum_{j=1}^{(\beta-C_i)} \left( 0.1 \times V_{i,j} \right)$$  (4)

where $C_i$ is the number of complete sets in member $i$, $(\beta-C_i)$ the number of incomplete sets in member $i$, and $V_{i,j}$ the coverage percentage of the $j^{th}$ incomplete set in member $i$. 
The move makes \( K_1 \) efforts to reschedule the sensors. But this does not imply that \( K_1 \) sensors are rescheduled for every member in each of the iterations. The sensor may be rescheduled if and only if it is redundant to its current schedule. A pseudo-code for the redundant move is given in Fig. 4.

\[
\text{FOR } (k = 1, 2, \ldots, K_1) \\
\quad j = \text{roundoff} \left( 1 + (N - 1) \times \text{uniran()} \right) \\
\quad T = I(i,j) \\
\quad \text{IF } (j \text{ is redundant to the set } T) \\
\quad \quad T^{\text{new}} = \text{roundoff} \left( 1 + (\beta - 1) \times \text{uniran()} \right) \\
\quad \quad I(i,j) = T^{\text{new}} \\
\quad \text{END IF} \\
\text{END FOR}
\]

![Fig. 4 Pseudo-code for redundant move](image)

2) Completion Move

The completion move aims at improving the coverage percentage of incomplete sets to make them complete. It moves the redundant sensors from the complete sets to the incomplete sets. This move is also performed \( K_2 \) times for every member in each iteration. The procedure for this move is explained below and a pseudo-code for the same is given in Fig. 5.

Randomly select a sensor. If the sensor belongs to a complete set, check whether it is redundant to its current set. If yes, reschedule it to a randomly selected incomplete set.

\[
\text{FOR } (k = 1, 2, \ldots, K_2) \\
\quad j = \text{roundoff} \left( 1 + (N - 1) \times \text{uniran()} \right) \\
\quad T = I(i,j) \\
\quad \text{IF } (\text{the sensors of set } T \text{ provide complete coverage}) \\
\quad \quad x = \text{roundoff} \left( 1 + ((\beta - C_i) - 1) \times \text{uniran()} \right) \\
\quad \quad i^* = \beta - C_i \\
\quad \quad T^{\text{new}} = \text{schedule of the } x^{\text{th}} \text{ incomplete set} \\
\quad \quad I(i,j) = T^{\text{new}} \\
\quad \text{END IF} \\
\text{END FOR}
\]

![Fig. 5 Pseudo-code for completion move](image)

3) Critical Move

The upper limit on the number of disjoint complete sets is fixed by the number of sensors covering the critical fields. In order to classify the sensors into the maximum number of disjoint complete sets, the sensors covering the critical fields must be placed in different sets. In other words, no set can be complete if it does not contain a sensor for each of the critical

![Fig. 3 Flowchart of the proposed HAS](image)
fields. This move tries to displace the multiple critical sensors scheduled in a common set to those incomplete sets which do not cover the corresponding critical field. The critical move is performed for all the incomplete sets of every member in each iteration as given below:

For each critical field, check whether it has been covered in the incomplete set. If not, randomly select a sensor from among the sensors covering that critical field. Check whether the selected sensor is redundant to its current set. If yes, reschedule it to the current incomplete set.

Fig. 6 presents the pseudo-code for the critical move.

```plaintext
FOR (k = 1, 2, ..., (β - C)) /* for every incomplete set in member I */
  T = schedule of the k
  FOR (i = 1, 2, ..., NCF) ; /* for every critical field */
    IF (the
      IF (sensor j is redundant to its current schedule in the member i)
        I(i, j) = T /* reschedule the sensor j to the complete set T */
      END IF
    END FOR
  END FOR
END IF
END FOR
```

Fig. 6 Pseudo-code for critical move

After performing the heuristics, the fitness of the new population is evaluated using the fitness function given by (4). The best-so-far member is then updated. The HSA terminates either when β disjoint complete cover sets are found or when the predefined maximum number of iterations is reached. If none of the termination conditions is met, the HSA again starts with the redundant move for the new population and the process is repeated.

With M members in the population, the number of coverage evaluations as well as the fitness function evaluations made during initialization is M. Further, in every iteration, coverage evaluation is made three times for every member and the fitness function evaluation is done once for every member. Thus for t iterations, the number of coverage evaluations made shall be M(1 + 3t) and the number of fitness function evaluations shall be M(1 + t).

IV. RESULTS AND DISCUSSION

The proposed HSA has been tested with different sensor deployments for point coverage problems. The performance of the HSA has been compared with STHGA [9]. This section presents the analysis and discussion of the results.

The population size for all the experiments has been fixed at M = 5. The parameters K1 and K2 used in the heuristics are set as K1 = K2 = (N / 3) where N is the number of sensors deployed. These parameters have been set empirically. Each case been tested 100 times for the HSA. The parameter settings and other details regarding STHGA can be referred in [9]. The monitored area is a 50 × 50 unit square. The sensor node location coordinates have been randomly generated as float point values in [1], [51]. All the simulations have been carried out on a computer having Intel® Core™ i3 CPU @2.4GHz with 4GB RAM.

Experiments have been conducted for different cases with different numbers N of the sensors deployed. The number of targets and the sensing range R of the sensors has been fixed at 10 and 22 respectively. Table II tabulates the results computed by HSA and compares it with the results for STHGA documented by Hu et al. [9]. To facilitate the comparison, the results for the same values of N, R and β have been presented as [9]. Both STHGA and HSA create the maximum number of disjoint sets in all the cases. Similar to STHGA [9], HSA also finds the optima in all the 100 independent runs for each case. The results listed in the table are the mean values. However, HSA reaches the optima much faster than STHGA in all the cases. This is reflected by the considerably smaller average number of fitness function evaluations made by HSA in comparison with STHGA for all the seven cases. Fig. 7 graphically shows the variation in number of fitness function evaluations by both STHGA and HSA for the seven cases listed in Table II. It can be noted that both STHGA and HSA show similar variation trends. However, the number of fitness evaluations for HSA is smaller than STHGA in all the cases.

![Fig. 7 Comparison of number of fitness function evaluations for STHGA and HSA](image-url)
V. CONCLUSIONS AND FUTURE SCOPE

In this paper a directed random search algorithm has been presented to find the maximum number of disjoint complete cover sets of sensors for point coverage in order to maximize the lifetime of a WSN. HSA uses a population to explore the entire solution space in search of the optimum solution. The members are initially randomly scattered all over the search space. Each member explores the search space in its neighborhood and reports the best position acquired by it. Keeping in view, the requirements of the application for which the search algorithm is designed, a set of heuristics has been introduced. These heuristics speed up the convergence of the algorithm towards the optima. The proposed algorithm achieves high quality solutions at fast optimization speeds and outperforms the STHGA which has been proven to be the most suitable algorithm for this problem [9].

The results show that HSA always finds the optima and that too at a high pace. Further research needs to be carried out to analyze the effect of variation in the population size and the number of times the redundant and completion moves are made. Also the suitability of HSA to increase the lifetime of WSNs in area coverage problems needs to be explored.

REFERENCES


<table>
<thead>
<tr>
<th>Sr.</th>
<th>N</th>
<th>R</th>
<th>β</th>
<th>Nsets</th>
<th>avgFE</th>
<th>Nsets</th>
<th>avgFE</th>
<th>Number of Iterations</th>
<th>avgFE</th>
<th>avgCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>90</td>
<td>22</td>
<td>30</td>
<td>30</td>
<td>596</td>
<td>30</td>
<td>30</td>
<td>13.77</td>
<td>73.83</td>
<td>211.50</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>22</td>
<td>23</td>
<td>21</td>
<td>214</td>
<td>23</td>
<td>23</td>
<td>6.07</td>
<td>35.33</td>
<td>96.00</td>
</tr>
<tr>
<td>3</td>
<td>110</td>
<td>22</td>
<td>21</td>
<td>21</td>
<td>156</td>
<td>21</td>
<td>21</td>
<td>3.90</td>
<td>24.50</td>
<td>63.50</td>
</tr>
<tr>
<td>4</td>
<td>120</td>
<td>22</td>
<td>35</td>
<td>35</td>
<td>422</td>
<td>35</td>
<td>35</td>
<td>8.67</td>
<td>48.33</td>
<td>135.00</td>
</tr>
<tr>
<td>5</td>
<td>130</td>
<td>22</td>
<td>41</td>
<td>41</td>
<td>1856</td>
<td>41</td>
<td>41</td>
<td>61.80</td>
<td>314.00</td>
<td>932.00</td>
</tr>
<tr>
<td>6</td>
<td>140</td>
<td>22</td>
<td>44</td>
<td>44</td>
<td>3568</td>
<td>44</td>
<td>44</td>
<td>115.97</td>
<td>584.83</td>
<td>1744.50</td>
</tr>
<tr>
<td>7</td>
<td>150</td>
<td>22</td>
<td>42</td>
<td>42</td>
<td>532</td>
<td>42</td>
<td>42</td>
<td>17.67</td>
<td>93.33</td>
<td>270.00</td>
</tr>
</tbody>
</table>

* avgFE = average of number of fitness function evaluations
** avgCE = average of number of coverage evaluations
@ Nsets = Average of number of disjoint complete sets created


