Paper

Swarm Intelligence-based Partitioned Recovery in Wireless Sensor Networks

Gaurav Kumar and Virender Ranga

Department of Computer Engineering, National Institute of Technology Kurukshetra, Haryana, India

https://doi.org/10.26636/jtit.2018.121817

Abstract—The failure rate of sensor nodes in Heterogeneous Wireless Sensor Networks is high due to the use of low battery-powered sensor nodes in a hostile environment. Networks of this kind become non-operational and turn into disjoint segmented networks due to large-scale failures of sensor nodes. This may require the placement of additional high-power relay nodes. In this paper, we propose a network partition recovery solution called Grey Wolf, which is an optimizer algorithm for repairing segmented heterogeneous wireless sensor networks. The proposed solution provides not only strong bi-connectivity in the damaged area, but also distributes traffic load among the multiple deployed nodes to enhance the repaired network's lifetime. The experiment results show that the Grey Wolf algorithm offers a considerable performance advantage over other state-of-the-art approaches.

Keywords—connectivity restoration, meta-heuristics, relay node placement, wireless sensor networks.

1. Introduction

Heterogeneous wireless sensor networks (HWSNs) employ different types of nodes which differ from each other in terms of capabilities, load assigned to the nodes and coverage areas. HWSNs attract a large number of applications in the field of health, defense, agriculture, forest monitoring, etc. Moreover, HWSNs are well capable of operating in harsh and hostile environments without human intervention. However, it is a challenging task to simultaneously maintain coverage and connectivity in a harsh environment [1], [2].

It is a hard fact that sensor nodes (SNs) are more susceptible to failure in a harsh scenario and may drain battery power within a short span of time if unnecessary loads are assigned to them. Therefore, HWSNs require energy-constrained algorithms to perform operations in harsh environments. Segmentation or network partition is a classic, well-known problem affecting HWSNs. In a segmented network, SNs may not be able to communicate with other sensor relay nodes. This is a distributed problem, where computation is to be performed in different parts of the system and results need to be aggregated for the final action to be taken.

Restoration of lost connectivity in distributed, disconnected HWSNs is an example of diffusing computation, where it starts at one node of the distributed system and slowly transfers towards other parts. On the other hand, segments may be created by using relay nodes (RNs) due to large-scale failures of SNs (i.e. battery power exhausted) or due to a natural disaster. RNs are more powerful than SNs in terms of communication range and reserved battery backup. Therefore, relay node placement (RNP) in HWSNs is a cost effective and best-suited method to solve the network partition problem, simultaneously offering a fault tolerance mechanism. The relay node placement problem (RNPP) is NP-hard [3], [4]. RNs are expensive. Hence, a large number of RNs may increase the overall cost of a network. Each and every position of RNs in 2D renders different optimized solutions. The solution with a minimum RN count could be considered as an optimized solution for RNPP.

There are many techniques to solve RNPP in HWSNs. However, meta-heuristics are recognized as best-suited methods.

Meta-heuristics are problem independent and stochastic in nature to solve NP-hard and optimization problems. Some popular meta-heuristics include Particle Swarm Optimization (PSO) [5], Ant Colony Optimization (ACO) [6], or Genetic Algorithm (GA) [7]. A meta-heuristic may produce a promising solution for a given set of problems besides that it may also give the worst solution for another set. It totally depends on the choice of the meta-heuristic made based on the problem at hand. Therefore, there is a need to find a relevant metaheuristic approach which can produce favorable results for the selected set of problems.

All meta-heuristics performed the search process that was divided into two different phases: exploration and exploitation. Global searching is called exploration and local searching is known as exploitation. Exploration covers the whole solution space by diverging search agents in different directions. Exploitation is a local search and covers only a specific part of the solution space by converging towards a candidate position. Meta-heuristics can be classified into three various classes: swarm intelligence-based (SI), physics-based, and evolutionary algorithms (EAs) [8].

To solve RNPP, we use the Grey Wolf Optimizer (GWO) as a swarm intelligence-based technique enabling to find the optimal location of probable RNs. It is based on the hunting behavior of grey wolves.

The paper is organized as follows. Section 2 shows the related work concerned with RNPP. The system model and the problem statement are discussed in Section 3. Section 4 is completely devoted to GWO explanation. The proposed solution is described in Section 5. Section 6 discusses the pseudo code. Some well-established proposed solutions are compared with proposed solution in Section 7. Finally, the paper is concluded in Section 8.

2. Related Work

RNPP-related techniques used for repairing segmented HWSNs can be classified into two different categories, based upon the behavior of RNs within the network. The first category of approaches is related to the deployment of static RNs in the damaged portion of HWSNs. The second category is used for the deployment of mobile relay nodes which can relay data received from a group of sensor nodes or from nearby neighboring RNs to the base station (BS), which may be either mobile or stationary. The mobility of BS can be taken into consideration for improving transmission efficiency.

2.1. Deployment of Static Relay Nodes

Recently, Lee *et al.* [9] tackled RNPP by deploying RN using Steiner Points (SPs) and the convex hull approach. During the first phase, they find a convex hull on all disjoint segments, where each segment is represented by a representative node. Each representative node denotes the whole segment area as a single point in 2D. In the second phase, they proposed to find minimal sub-Steiner trees for every three neighboring terminal nodes. This procedure is followed repeatedly until two terminal nodes are left.

In [10], Lanza-Gutierrez *et al.* proposed six different multi-objective meta-heuristics (ABC, firefly algorithm, evolutionary algorithm with NBI-Tchebyff approach, non-dominated sorting genetic algorithm-II, strength pareto-evolutionary algorithm 2, variable neighborhood search). They evaluated all proposed solutions based on three objectives (average sensitivity area, network reliability, average energy cost) using the six meta-heuristics referred to above. Each objective is bound with an objective function that is being used as input for the meta-heuristics at hand.

The concept of the local search approximation algorithm is introduced by Ma *et al.* [11] and is also known as LSSA. They proposed a novel, connectivity aware, approximation-based approach for two-tiered HWSNs. They formed, with the help of a local search, a local set cover for different groups of sensor nodes. After that, they calculated a set cover for RNs based on the local set cover. They extended the same for the double relay node set cover.

The authors of [12] solve RNPP in static hybrid HWSNs with the help of PSO and integer, planning the average path length between sensors. They improve the efficiency of relay deployment because of a restricted search space of the integer, instead of the real number. Efficient deployment of RNs and BS may enhance the efficiency of the proposed solution.

Lloyd *et al.* [13] have proposed a solution to solve 1-tier as well as 2-tier RNPPs. Their proposed solution is used to find the optimal path for the single tier, and for RNs between every pair of sensors. The second approach is proposed for a 2-tier RNPP. Time complexity for the 1-tier solution proposed is shown to be a 7-approximation, and for 2-tier it is shown as $(4.5+\varepsilon)$-approximation and $(5+\varepsilon)$-approximation.

The authors of [14] proposed a game theory-based approach for RNPP. This approach is supposed to have a complete knowledge about the network (the number of failed nodes, the number of segmented parts and location of partitioned segments). Each segment is used as a player in the game in which each node is used as a payoff function. Game theory is a centralized approach. Therefore, each partition must know about the payoff function of all other partitions. At last, each player shares their payoff results to restore lost connectivity within the segmented network.

2.2. Deployment of Mobile Nodes

Mobility in HWSNs may be divided into two categories: batch movement and succeeding movement. In batch movement, a group of nodes moves towards another group of nodes for re-connection. The basic idea behind batch movement is to join two partitions by moving towards each other [15]. Succeeding movement is related to the movement of one or few RNs, performed in an awkward fashion, in order to repair the segmented network, e.g. [16] restore connectivity by succeeding movement of RNs to repair wireless sensor and actor networks (WSANs). The authors of [17] try to identify a node the removal of which may lead to network partition. After identification, the suggested algorithm starts connectivity restoration of the partitioned WSN. Wang *et al.* [18] introduced mobility in RNs as well as in BS and try to increase network lifetime in different environments, e.g. static network, WSN with a single mobile sink, and WSN with mobile RN.

The main advantage of deployment of mobile RNs is the ability to collect/send data from/to a large number of sensors. Mobile RNs in HWSNs enhance coverage, connectivity, fault tolerance and lifetime of the network. Akkya *et al.* [19] deployed mobile agents to re-connect segmented partitions of HWSNs. They proposed a mathematical model for deployment of RNs for a minimum traveling distance. The proposed solution is evaluated based on time required to reconnect the partitioned network and the total distance traveled by mobile RNs. The authors of [20] deploy mobile agents to receive/send sensed data from/to sensor/BS. The proposed solution saves energy of SNs by using mobile RNs to transmit the data to BS.

The authors of [21] proposed an algorithm to control the mobility of nodes to reduce energy consumption. The idea behind controlled mobility is based on covering the damaged part of WSN. The mobile RN restores connectivity in the no-connectivity area. Since moving a node for a long time may drain its battery power at a fast rate the proposed solution minimizes the maximum travel distance.

3. System Model and Problem Statement

In this paper, a flat structure of HWSN is considered on which SNs are deployed throughout a specific predefined area by using any node deployment strategy. Here, random deployment is taken. Sink node/BS is positioned at a predefined location to receive aggregated data. All network traffic flows towards BS to get useful information related to the environment being observed. When a large number of SNs fail, a number of disjoint, partitioned segments may be created. Figure 1a shows a segmented HWSN with seven disjoint segments and the damaged area. Thus, proposed approach is proposes to place RNs inside the damaged area to restore lost connectivity. Initially, one RN is assigned to every segment as a representative node which is denoted by Seg_i. Thus, there is a need to deploy at least N_{Seg} RNs for every disjoint segment (suppose we have N_{Seg} number of the disjoint segment).

The problem of placing relay nodes in the segmented area can be described as follows. Initially, segmentation is detected, with N_{Seg} number of disjoint segments. Initially, N_{Seg} is the number of RNs considered, with each of them working as a gateway node for their respective segment. For simplicity, it is assumed that every segment has an RN for the purpose of the experiment. That RN is denoted by Seg_i where $0 \leq i \leq N_{Seg}$. The range of RNs is considered to be as required, and is denoted by the symbol R_r. The range of RNs (R_r) may differ from the range of SNs (R_s) which is usually $R_r \geq R_s$. The proposed algorithm strives to find the near to optimal position and the minimum count of RNs by using the GWO meta-heuristic technique.

The research is based on the following assumptions:

- all SNs, as well as BS, are static,
- all RNs have an equal unit transmission range of R_r,
- HWSN uses its underlying routing protocol to relay data from source to destination,
- X-Y coordinates of all nodes are considered in integer space.

4. The Grey Wolf Optimizer

Grey wolves have a well-organized social hierarchy. Their hunting strategy can be used for solving the optimization problem. To complete the hunting process grey wolves move forward in a planned manner. The authors of [22] suggest various steps which are used by the wolves for hunting. All these are listed below:

- track, trail and trace,
- keep an eye, surround and hassle the target until it comes to rest,
- attack the prey.

In [8] authors give a model of all these procedures which can be used to solve many optimization problems. Presented solution uses the same technique to restore lost connectivity in a partitioned HWSN. As per the social hierarchy of grey wolves, solutions obtained from mathematical modeling are categorized into three fittest solutions. Alpha (α) is considered to be the best solution. Beta (β) and delta (δ) will be the second and the third fittest solution, respectively. The remaining solutions will be considered to belong to the omega (ω) set.

4.1. Surrounding Prey

To model the surrounding phase, the following equations are used to depict the behavior:

$$\vec{D} = \left|\vec{J}\cdot\vec{P_w}(i) - \vec{P}(i)\right| , \tag{1}$$

$$\vec{P}(i+1) = \vec{P_w} - \vec{K}\cdot\vec{D} , \tag{2}$$

where $\vec{D}$ indicates the distance between prey and wolf, $\vec{J}$ and $\vec{K}$ are the coefficient vector which is used to encircle the prey. Both of these play an important role during hunting. $\vec{P_w}$ and $\vec{P}$ is the position vector of the wolf and the prey, respectively, and i denotes an iterator.

The equation coefficient vectors $\vec{J}$ and $\vec{K}$ are given as follows:

$$\vec{J} = 2\cdot\vec{l_1} , \tag{3}$$

$$\vec{K} = 2\vec{k}\cdot\vec{l_2} - \vec{k} , \tag{4}$$

where vector $\vec{k}$ is decreased from 2 to 0, and vectors $\vec{l_1}$, $\vec{l_2}$ are random vectors in [0, 1].

4.2. Attacking Prey

One of the best capabilities of grey wolves is to make an estimation of the location of prey. Firstly, they encircle the prey and then attack it. Alphas are the most dominating wolves within the group and they steer the hunt. In the case of NP-hard problems, the solution space is very large and it is quite difficult to search for optimal solutions in polynomial time. GWO always strive to find three fittest or best solutions. Considering that alpha is the fittest, beta is the second best and delta is the third best solution, the following equations are used to show the behavior of alpha, beta, and delta:

$$\vec{D_\alpha} = \left|\vec{J_1}\cdot\vec{P_\alpha} - \vec{P}\right|, \vec{D_\beta} = \left|\vec{J_2}\cdot\vec{P_\beta} - \vec{P}\right|, \vec{D_\delta} = \left|\vec{J_3}\cdot\vec{P_\delta} - \vec{P}\right| , \tag{5}$$

$$\vec{P_1} = \vec{P_\alpha} - \vec{K_1}\cdot\vec{D_\alpha}, \vec{P_2} = \vec{P_\beta} - \vec{K_2}\cdot\vec{D_\beta}, \vec{P_3} = \vec{P_\delta} - \vec{K_3}\cdot\vec{D_\delta} , \tag{6}$$

$$\vec{P}(i+1) = \frac{\vec{P_1} + \vec{P_2} + \vec{P_3}}{3} . \tag{7}$$

The pseudo code of GWO is shown as Algorithm 1.

Algorithm 1. Grey wolf optimizer algorithm

```
1: procedure GWO(initial population)
2:     Initialize the population of grey wolf agents $\vec{P_d}(d =
3:     1, 2, \dots, n)$
4:     Initialize all coefficient vectors $\vec{k}$, $\vec{K}$ and $\vec{J}$
5:     $\vec{P_\alpha} \leftarrow$ First best possible solution
6:     $\vec{P_\beta} \leftarrow$ Second best possible solution
7:     $\vec{P_\beta} \leftarrow$ Third best possible solution
8:     while ($i <$ max number of iterations) do
9:         for each search agent do
10:            Update possible location of current search
11:            agent by using Eq. (7)
12:        end for
13:        Update coefficient vectors $\vec{k}$, $\vec{K}$ and $\vec{J}$
14:        Calculate the fitness of all search agents
15:        Update position vectors $\vec{P_\alpha}$, $\vec{P_\beta}$ and $\vec{P_\delta}$
16:        $i \leftarrow i + 1$
17:    end while
18:    return $\vec{P_\alpha}$                ▷ Return the best solution
19: end procedure
```

5. The Proposed Solution

We have considered a well-connected network in the simulation, which is converted into segmented portions after a large-scale failure of SNs. It leads to the creation of multiple, disjoint segments within the network. A 2D vector of the locations of disconnected segments is used as population size for the proposed solution and generates the probable location of prey. This is an iterative process and each iteration has its own solution, because the number of segments keeps changing continuously. The proposed GAIN solution is mapped with the GWO algorithm, as: all disconnected segments are treated as grey wolves. By using the current location of wolves, the probable location of prey is found. The location of prey is continuously updated with coefficient vectors, as discussed in Section 4. The observed location of prey is used for relay node placement between different segments to recover lost connectivity. The proposed solution is executed in different phases as shown below:

- locate the position of initial RNs,
- neighbor discovery,
- populating RNs,
- termination phase.

In the first phase, RNs are considered as representative nodes for all disjoints segments. Their locations are observed as discussed in Subsection 5.1. The transmission range of RNs (i.e. R_r) is considered to be the radius of the circle made by the coverage field of any RN. Therefore, any RN can cover a distance of $2R_r$ in all directions. The second phase is concerned with finding the neighboring segments with the help of the deployed RNs. In the third phase, the proposed algorithm strives to populate RNs in the damaged region by using different rounds. In each round, one best RN position is returned by the presented solution. The calculated location is abbreviated as the current relay node location for the respective round. The relay placed at the said location is called current relay. The current relay node strives to find representative nodes within its range. Now, nodes within the range of each other become neighboring nodes. All connected nodes in the segment are denoted by only one representative node for the simplicity of the algorithm. In the fourth phase (i.e. termination phase), the algorithm terminates when condition $N_{Seg} \leq 2$ is met, where N_{Seg} represents a number of disjoint segments in the network. Finally, if only one segment is left, connectivity has been restored successfully. If two segments are left, then there is a need to populate some additional RNs on the basis of the Euclidean distance between them, as discussed and shown in Eq. (8). The detailed description of the proposed GAIN solution is described in the following subsections.

5.1. Locate the Position of Initial RNs

Whenever the BS observes a sudden decrease in the amount of information sensed from the deployed network, it identifies a large-scale failure of SNs. The proposed solution is required to find relevant positions for populating representative nodes Seg_i where i represents the number of segments. Considering all segments have their own representative RN as a gateway for the segmented part, any communication within the segments would be possible only through these RNs. Figure 1a represents an example of a partitioned network with its seven representative nodes denoted by Seg_1 to Seg_7. The solution takes X-Y locations of all representative nodes as an input and produces a relay count. In addition, to identify the minimum distance between any two points, GAIN uses Euclidean distance given by:

$$ED_{(x_1,y_1)(x_2,y_2)} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}\,, \tag{8}$$

where $ED_{(x_1,y_1)(x_2,y_2)}$ denotes the Euclidean distance between two points (x_1, y_1) and (x_2, y_2).

5.2. Neighbor Discovery

In this phase, we consider a set of representative nodes obtained from the first phase. After that, each segment discovers its neighboring segments, as explained in Algorithm 2. After placing initial RNs, some of the segments find other segments to be within their transmission range and list those segments as their neighboring segments. Figure 1a shows an example of seven initial segments. Each segment is denoted by a single RN and it is referred to as a representative node for all nodes of the respective segment. Each representative RN is denoted by $Seg_i(x, y)$, where Seg_i is the segment number with its coordinate (x, y). As calculated in the simulation, X-Y coordinates of all segments are as Seg_1: $(50, 12)$, Seg_2: $(79, 23)$,

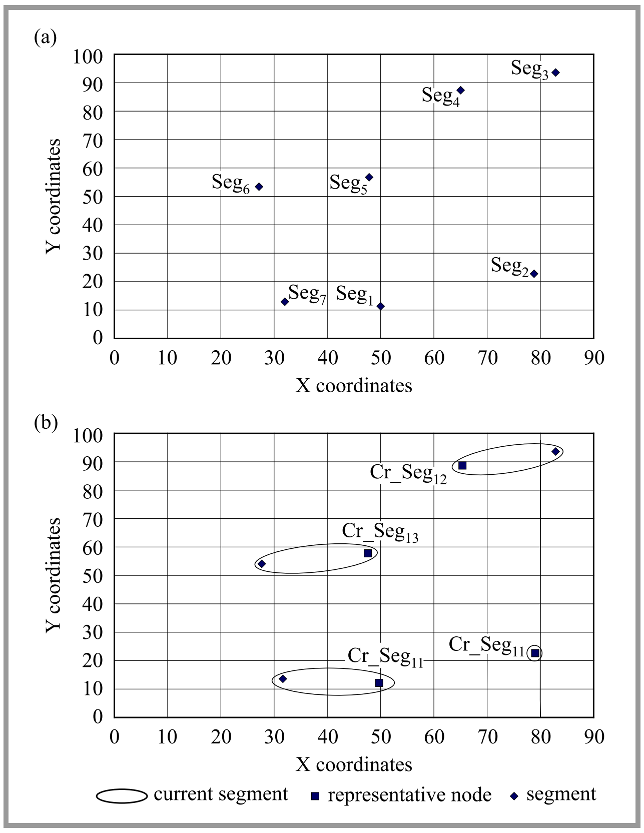

Fig. 1. Representing initial two steps of GAIN: (a) showing initial position of each segment Seg_i, and (b) showing neighboring segments where $R_r = 15$ m.

$Seg_3: (83, 94)$, $Seg_4: (65, 88)$, $Seg_5: (48, 57)$, $Seg_6: (27, 54)$ and $Seg_7: (32, 13)$. After the placement of representative nodes within every segment, the proposed solution will increase the communication range of the respective segments due to the large communication range of RNs, albeit some of the segments do not require additional relays to establish communication between disjoint segments. The Euclidean distance between two segments (representative nodes) can be calculated as two points (x_1, y_1) and (x_2, y_2) in the X-Y plane. These points will be the neighbors of each other when these two points satisfy equality $R_r \leq ED_{(x_1,y_1)(x_2,y_2)}$. In this way, neighboring points will be combined into a single segment which is shown in Fig. 1b. It can be observed that Seg_1 joins Seg_7, Seg_3 joins Seg_4, Seg_5 joins Seg_6 and Seg_2 have no neighboring segment and converted it into current segments $Cr_{Seg_{14}}$, $Cr_{Seg_{12}}$, $Cr_{Seg_{13}}$, $Cr_{Seg_{11}}$, respectively.

This step adds an extra boost to the proposed algorithm to solve the network partition problem, and it is really helpful in reducing the number of deployed RNs. It is observed in the simulation that the total number of relay nodes would be always greater than or equal to the initial number of segments. This relation can be shown as $Count\_Relay \geq N_{Seg}$ where $Count\_Relay$ denotes the total number of relays requiring restoring connectivity within a disconnected WSN. After completion of the neighbor discovery phase, some of the segments combine with their neighboring segments and the rest of them are termed as current segments. The current segments are represented by $Cr_{Seg_{ij}}$, where i denotes the iteration number and j simply denotes the current segment number. Each segment may or may not have its neighbors, for e.g. Seg_2 does not have any neighboring nodes and segments Seg_1, Seg_4, Seg_5 have one neighboring node, i.e. Seg_7, Seg_3, Seg_6, respectively, as depicted in Fig. 1b.

5.3. Populating RNs

The third step of the proposed solution is executed in rounds. In each round, RN position is calculated and the observed position is represented by RN_i, where i denotes the respective iteration number. The proposed GAIN solution generates a convex hull over the representative nodes of the current segments $Cr_{Seg_{ij}}$. Due to the random and stochastic nature of RNs, their positions come out of the transmission range of all the segments. To remove this side effect, we incorporate the convex hull algorithm that uses the Graham scan algorithm [23]. The proposed solution continuously checks whether the calculated point lies inside the convex hull or not. If it lies inside the convex hull, then we proceed to the next iteration, otherwise the algorithm discards the candidate position and puts it into the

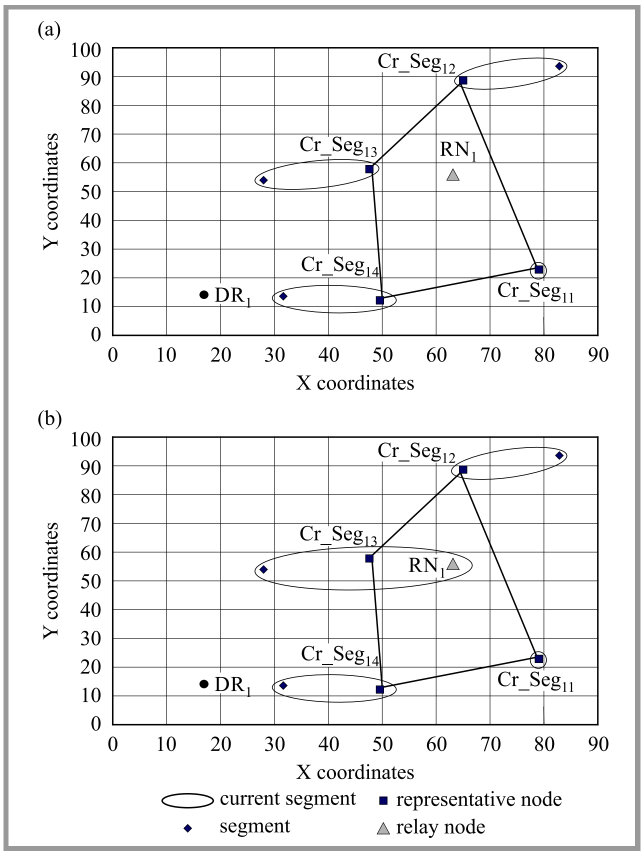

Fig. 2. First iteration of placing relay node placement by GWO: (a) shows a convex hull made by current segments, the first RN $RN_1(63, 56)$ and the discarded relay $DR_1(17, 14)$, and (b) current segment Cr_{Seg_3} is within the range of RN RN_1.

category of discarded RNs, represented by DR1, as shown in Fig. 2a.

The first iteration scenario has been shown in Fig. 2b. Here, one discarded RN position $DR_1(17,14)$, which lies outside the convex hull and one accepted RN position $RN_1(63,56)$ which lies inside the convex hull are shown. Sometimes only one, and sometimes a higher number of executions is required. Therefore, time-related complexity may be increased. However, GAIN shows better results compared with state-of-the-art solutions. The proposed algorithm strives to find segments within the range of RN RN_1. Again, the solution uses Eq. (8), where coordinates (x_1,y_1) represent the position of the RNs and (x_2,y_2) shows the coordinates of current segments. If any two current segments satisfy relation $ED_{(x_1,y_1),(x_2,y_2)} \leq 2R_r$, then both of these segments are considered within the range of each other. The same scenario can be seen in Fig. 1b, which shows that RN_1 is within the communication range of current segment $Cr_{Seg_{13}}$, so these two segments combine into a single segment.

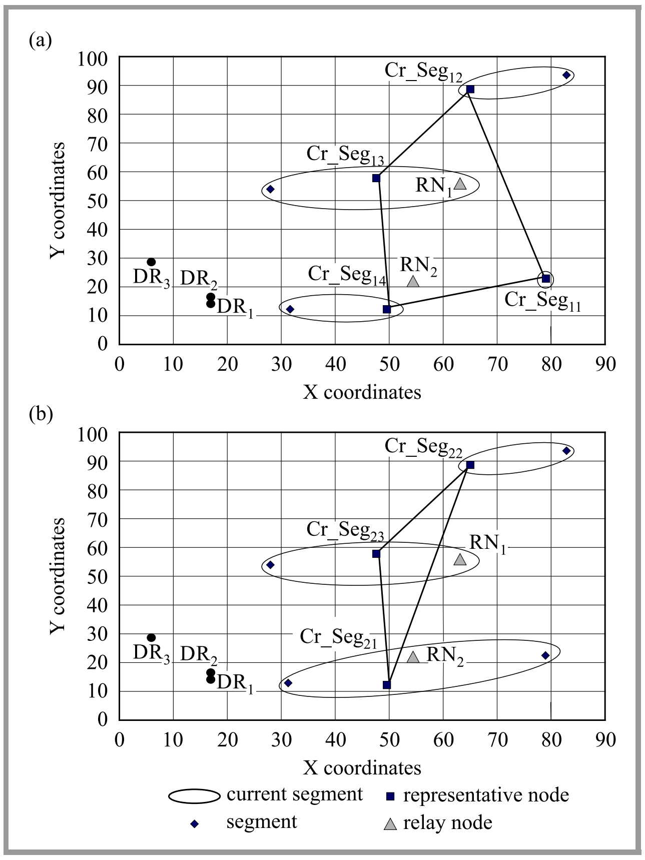

Fig. 3. Second iteration of placing relay node placement by GWO: (a) shows discarded relays $DR_2(17,16)$ and $DR_3(6,28)$ and second RN $RN_2(55,21)$, and (b) RN RN_2 combines current segments Cr_{Seg_4} and Cr_{Seg_1}

We obtain two discarded relays $DR_2(17,16)$ and $DR_3(6,28)$, after the second iteration of the proposed solution due to its random nature (as shown in Fig. 3a). In the third attempt, GAIN is able to find a relevant position for RN $RN_2(55,21)$ which lies inside the convex hull of the respective current segments. The solution considers α as the best solution, so it tries to produce an output which is favorable for the first input in the population. But this side effect can be removed by using the convex hull approach. GAIN forces RNs to populate inward the damaged area. Figure 3b shows that current segments Cr_{11} and Cr_{14} combine into a single current segment Cr_{21}.

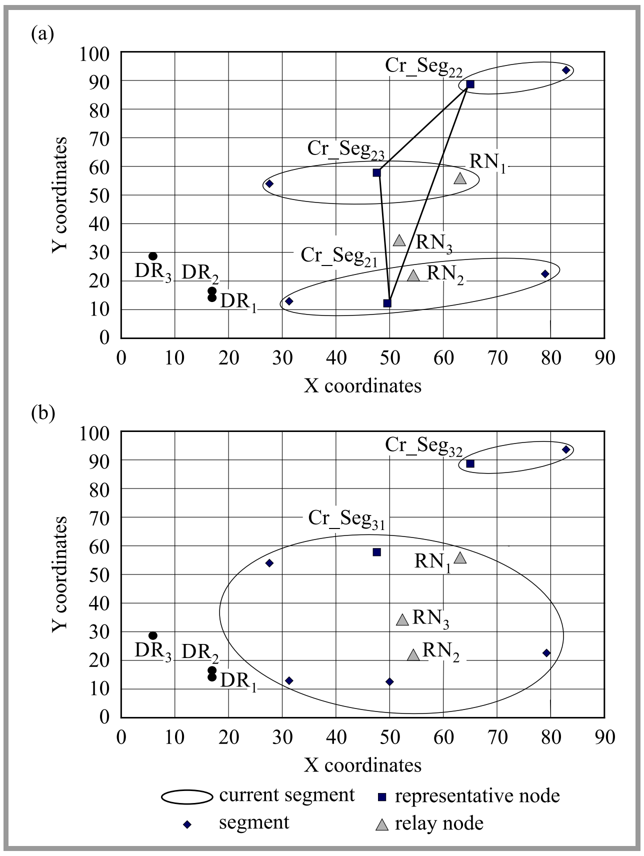

Fig. 4. Third iteration of placing relay node by GWO: (a) location of RN $RN_3(52,34)$ is found using GWO, and (b) RN RN_3 combines current segments Cr_1 and Cr_3.

The proposed solution generates three current segments after the execution of the second iteration – Cr_{21}, Cr_{22}, Cr_{23}. In the third round, GAIN produces another RN position $RN_3(52,34)$ which is able to communicate with current segments Cr_{21} and Cr_{23} by satisfying equation Eq. (8) and tries to restore lost connectivity as shown in Fig. 4a. Here, RN_3 will act as a gateway or bridge between both these segments. Figure 4b represents only two current segments Cr_{31} and Cr_{32}. Therefore, it is considered as a terminating condition.

5.4. Termination Phase

Termination phase is the last step of the proposed algorithm. The solution always checks for the remaining number of disconnected segments, because it is an iterative approach and it reconnects segments in the same manner. If two or fewer segments are left, the algorithm enters its termination

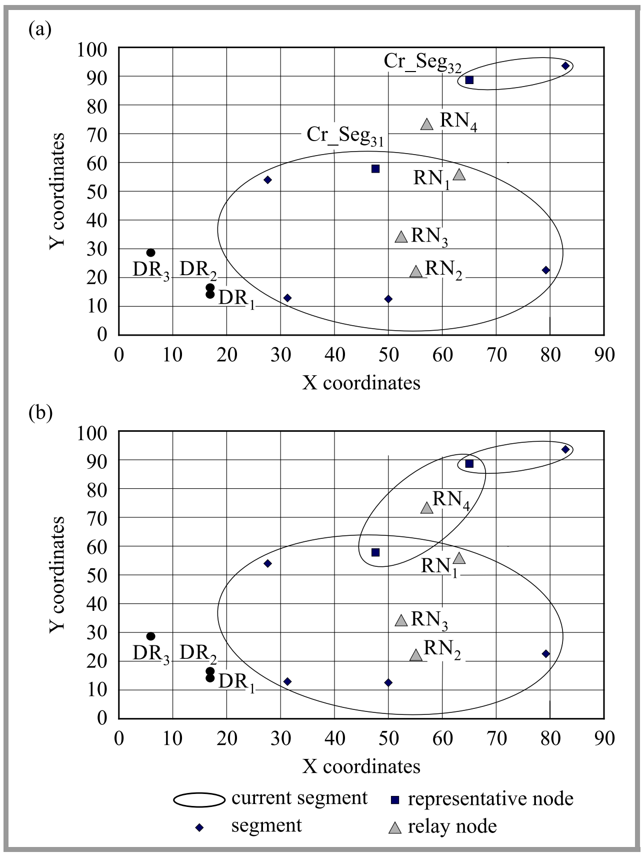

Fig. 5. Termination condition because only two current segments are left: (a) fourth RN $RN_4(57,73)$ has been placed manually, and (b) RN RN_4 combines both of the remaining segments.

phase. In the termination phase, it is considered that if one segment is left only, then connectivity has been restored. If two segments are left, then the algorithm finds the number of RNs required for restoration of connectivity by using the equation listed in the GAIN Algorithm 2 (lines 33–36). The relevant positions of RNs can be found by using the Euclidean distance between last two segments. Figure 5a shows that RN RN_4 has been placed between current segments $Cr_{Seg_{31}}$ and $Cr_{Seg_{32}}$. The position is somewhere along the line that joins these two segments. Figure 5b shows that disconnected segments can communicate with each other using RN RN_4 as a bridge.

5.5. Discussion on Degree of Novelty

The proposed algorithm has the power to generate an efficient and well-connected network topology. GAIN populates RNs in the inward direction of the damaged area. This generates a simple and efficient topology. The final topology generated is shown in Fig. 6. Some important observations may be inferred from the resulting topology, and are explained below:

- Connectivity is an important factor of any network topology. It shows its ability to handle failures. If network connectivity is high, then it can bear a large number of failures, up to a threshold limit. The resulting topology shown in Fig. 6 indicates that nodes on the edge of the network or terminal nodes are characterized by high connectivity (bi-connectivity). When a failure occurs, it does partition the network due to its high degree of connectivity.
- Distributed traffic load is the backbone of any network topology, because any network which has a centralized traffic load may suffer from continuous failures of the central part. Presented topology has no central point of failure, however. Instead of centralized load shown in Fig. 6, it is characterized by traffic load distributed among all nodes within the network. Therefore, it helps enhance the overall lifetime of the network.

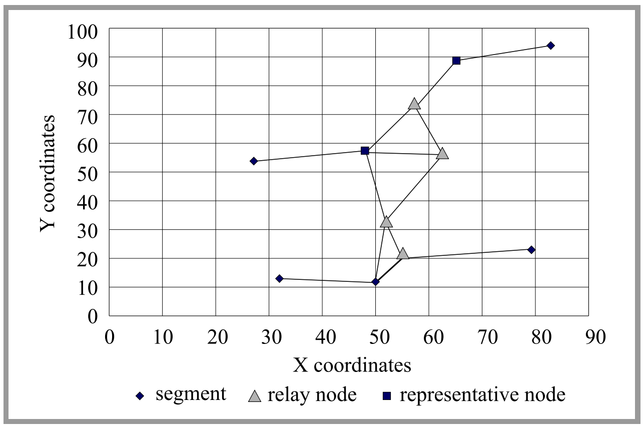

Fig. 6. Resulting topology calculated with GAIN.

6. Discussion on Pseudo Code of Proposed Solution

In this section, we explain the pseudo code of the proposed GAIN solution, which is shown in Algorithm 2. This pseudo code is divided into different parts based on the phases of GAIN. GAIN maintains a 2D array S for the coordinates of representative nodes and an integer variable, *Count_Relay* to hold the total number of RNs. The integer variable N_{Seg} shows the number of disjoint segments within the partitioned network. Lines 3–9 show the neighbor discovery phase. Line 5 is the main part of the neighbor discovery phase. If Seg_i and Seg_j satisfy this condition, then these two segments become the neighbors of each other.
The iterative process for populating RNs is shown in lines 10–32. Line 12 computes the convex hull over the representative nodes of current segments. Line 13 calculates the candidate position of RN for the current round. Next, the solution checks for the position of the current RN. If it lies inside the convex hull, then it is put into the acceptable list. Otherwise, it is included in the list of discarded RNs. The same scenario is depicted in lines 14–31. Each candidate RN position has some segments within its communication range. If these segments are in the range of the current RN, then it is considered as a new segment, as

Algorithm 2. GAIN pseudocode

```
 1: procedure GAIN(Area, $N_{Seg}$, $R_r$)
 2:     Initialize array S[$N_{Seg}$][2] and $Count\_Relay \leftarrow 0$;
 3:     for $i \leftarrow 0$ to $N_{Seg}$ do
 4:         for $j \leftarrow i+1$ to $N_{Seg}$ do
 5:             if Euclid_Dist($S[i][0], S[i][1], S[j][0], S[j][1]$)$< 2R_r$ then
 6:                 Join segment $j$-th with segment $i$-th and remove it from array S.
 7:             end if
 8:         end for
 9:     end for                                              ▷ Discovery
10:     while S have more than 2 points do
11:         Initialize $Cr_{Relay}[2]$;
12:         $CH \leftarrow$ Compute a convex hull of all points in S
13:         $Cr_{Relay} \leftarrow GWO(S)$;
14:         if check $Cr_{Relay}$ lie inside convex hull $CH$ then
15:             $Count\_Relay++$;
16:             Initialize $C\_Seg\_in\_Range \leftarrow 0$ and 2D vector $Seg\_in\_Range$;
17:             for $i \leftarrow 0$ to $N_Seg$ do
18:                 if Euclid_Dist($S[i][0], S[i][1], Cr_{Relay}[0], Cr_{Relay}[1]$)$< 2R_r$ then
19:                     $Seg\_in\_Range \leftarrow S[i]$ and $Count\_Seg\_in\_Range++$;
20:                 end if
21:             end for
22:             if C_Seg_in_Range==0 then Create a new segment by using current relay
23:             else
24:                 if C_Seg_in_Range==1 then Add current relay to respective segment
25:                 else
26:                     for $i \leftarrow 1$ to $Seg\_in\_Range.size()$ do
27:                         Join all segments together and remove those from S instead of of first one.
28:                     end for
29:                 end if
30:             end if
31:         end if
32:     end while
33:     if S have two point then
34:         $Temp \leftarrow (Euclid\_Dist(S[0][0], S[0][1], S[1][0], S[1][1]) - 2R_r)/(2R_r)$;
35:         $Count\_Relay \leftarrow Count\_Relay + Temp$;
36:     end if  ▷ To obtain the required number of RN to connect last two segments
37: return Count_Relay
38: end procedure
```

explained in line 22 of Algorithm 2. If the segments are in the range of the current RN, then they are added to the list of current RNs of the respective segment (line 24). If the number of segments within the range of the current relay exceeds one, then all these segments are joined together and represented as a single segment (lines 26–27).

Algorithm 3. Euclidean distance between two points in X-Y plain

```
1: procedure Euclid_Dist($x_1, x_2, y_1, y_2$)
2:     $a \leftarrow | x_1 - x_2 |^2$
3:     $b \leftarrow | y_1 - y_2 |^2$
4:     return $\sqrt{a+b}$
5: end procedure
```

For initiating the termination condition, we check the count of remaining segments. If it is lower than 3, then the algorithm enters the termination phase, implemented by means of lines 33–36. These lines calculate the number of remaining RNs requiring joining, by calculating the Euclidean distance between them.

7. Performance Evaluation and Comparison

In this section, the performance and effectiveness of GAIN are discussed. Comparison with state-of-the-art algorithms is also explained. The GAIN algorithm is implemented and simulated in Java. The experiment results show that the proposed solution identified the lowest count of RNs as a number of segments increased. The total number of RNs and the number of disconnected segments are proportional to each other. So, these two factors are very important. Our objective is to find the minimum number of relay counts. Table 1 shows the parameters used in the simulation to analyze its performance.

Table 1
Simulation parameters

Parameter	Value
Area	1000×1000 m
Nodes	100–500
Total number of partition	5–9
Communication range of RNs	50–325 m
Number of placed RNs	5–45

7.1. Performance Metrics

For the purpose of the experiment, we have considered a variable range of RNs with a fixed number of partitioned segments. We have also taken a variable number of partitioned segments for a variable range of RNs. The following listed metrics are used to validate performance:

- Number of segments (N_{Seg}) – a large value of (N_{Seg}) may increase the requirements related with connectivity, which would result in a large number of RNs required – *Count_Relay*. As discussed earlier in Section 5, the total count of relay nodes is always greater than or equal to the number of partitioned segments. Hence, this metric has a direct impact on the minimum number of RNs.

- Transmission range of RN (R_r) – a longer transmission range directly affects the total count of RNs. R_r exerts a direct impact on performance. Our experiment results show that a longer communication range may be considered for getting better results. The minimum number of RNs required to ensure inter-segment connectivity is directly influenced by inter-segment connectivity.
- Total count of RNs $(Count\_Relay)$ – this parameter is directly related to the deployment cost of the network, as a minimum relay count is preferred. It shows the estimated performance advantage of the proposed approach in comparison with other algorithms.

7.2. Comparative Approaches

This section provides an overview of some well-known and recently published algorithms. Considering the metrics described above, our objective is to solve the network partition problem and to propose an efficient approach to the issue at hand. In this section a brief introduction of four other RNPP solutions is discussed. The first algorithm generates a convex hull and calculates the Steiner point for every three neighboring nodes repeatedly, until two segments are left (ORC [9]). The second algorithm uses the Steiner minimum tree with the minimum number of Steiner points to find the location of RNs for restoration of global connectivity in WSN (STP-MSP [24]). The third algorithm forms the minimum spanning tree which is based on a single-tiered RNP (MST-1tRN [13]). The last algorithm studies the problem of the connected single cover, where each SN is covered by a single RN (1CSCP [25]).

- ORC – this approach seeks to form the minimum Steiner tree on the convex hull to populate RNs. It is an iterative process and is completed in different stages. Authors of ORC [9] proposed a heuristic based on the convex hull and populated RNs inwards from the boundary of the convex hull. In each iteration, the convex hull is formed and Steiner points for every set of three neighboring nodes are found considering Steiner points of the previous iteration as an input for the next iteration to form convex hull. ORC uses the k-LCA approach [26] to solve the Steiner tree problem.
- STP-MSP – this algorithm follows the concepts of the minimum spanning tree (MST) and the Steiner tree point (STP). The combination of these two approaches results in a fully connected WSN. STP-MSP considers P terminals which have no connectivity and it strives to find an MST formed by these P terminal nodes. Regarding a constant R, STP-MSP forms an edge p_1p_2 between two points p_2 and p_2 and inserts $\lceil\frac{|p_1p_2|}{R}-1\rceil$ a number of Steiner points, where R is the transmission range of the relay.
- MST-1tRN – initially, the MST-1tRN algorithm uses a set of sensor nodes, $S=(s_1,s_2,s_3,\dots,s_n)$ and some other constants r and R as the range of SNs and RNs, respectively. It strives to compute MST T_S generated by set S. It then tries to steinerize that MST to obtain an (r,R)-constrained Steiner tree by populating RNs on each edge of MST T_S.
- 1CSCP – the 1CSCP algorithm is totally based on the concept of the connected single cover problem. It seeks to find the location of a minimum number of RNs and ensures that each SN is covered by at least one RN. RNs are placed so that a connected network can be generated by using these RNs, and that BSs be reachable. A connected network of RNs is called a minimum connected single cover graph.

7.3. Simulation Results and Comparison

The simulations prove the concepts used in respective research work. The GAIN algorithm is simulated in the Java environment, with multiple configurations. Each configuration has a different number of segments N_{Seg} and a varying transmission range of RNs R_r. All SNs are randomly placed in the area of interest (i.e. 1000×1000 m). The transmission range of RNs varies from 50 m to 325 m and the value of N_{Seg} assumes 5, 6, 7 and 8. All experiment results are taken into consideration, with their average equaling 30 individual results.

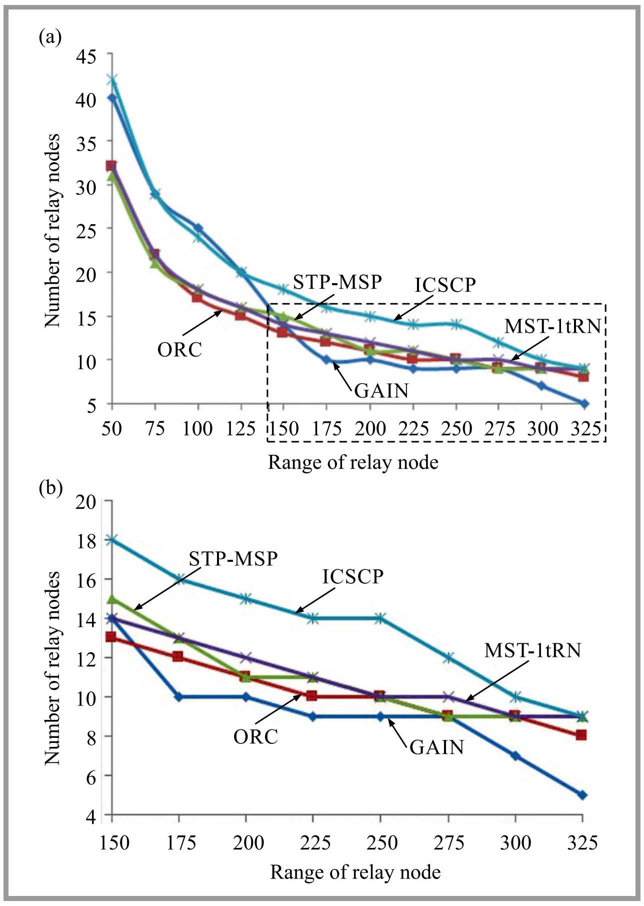

Fig. 7. GAIN vs. other algorithms. The lower figure is a magnified version of the rectangle area in the upper chart.

The performance of GAIN is studied with other four comparative algorithms in terms of the total RN count required to restore lost connectivity of a partitioned WSN, for various ranges of RNs. For all results, shown in Fig. 7, N_{Seg} is taken as a constant value of 5. Figure 7a shows that proximity is an important factor in a disconnect network. It has been observed that the RN count decreases as the range increases. While studying the impact of the large value of R_r, it has been shown that the total count of RNs is largely influenced by the value of R_r. The transmission range of RNs R_r and the number of RNs required to restore connectivity are inversely proportional to each other, which means that as range increases, the count of RNs decreases and vice-versa. The same relation can be depicted in Figs. 7a-b. The performance of GAIN decreases slightly compared to other algorithms for a low transmission range of RNs, due to the random nature of GWO. It tries to find an optimal location by considering all segments at a time. However, in Steiner points-based algorithms, the candidate RN location is found by considering only a few nodes at a time. As seen in Fig. 7a, for the value of R_r from 50 to 150 m, GAIN produces a larger number of RN locations, outperforming ORC. Generally, in a real time application, the value of R_r is to equal 200 or more than 200. Hence, Fig. 7a shows that for the transmission range of 150–325 m, the proposed solution renders better results than ORC, STP-MSP, MST-1tRN, 1CSCP in terms of the total RN count. Figure 7b is a magnified version of Fig. 7a, showing the results in greater detail. For a large value of R_r, GAIN populates RNs inwards from the outer edge of the first convex hull. At the outset, GAIN consistently outperforms all other comparative algorithms, for a large value in the range of 200–325 m.

In the subsequent simulation results, the range of RN is considered to be a constant value, i.e. equals 200 m. Here we simulate GAIN for various numbers of segments, to see the impact of a large-scale failure, where the value of N_{Seg} increases by leaps and bounds. Figure 8a shows the results of four different experiments, with the number of segments varying from 5 to 8. GAIN consistently outperforms other algorithms, because of a large transmission range of RN. Meanwhile, it can be observed that as segmentation increases by one, the required number of RNs increases sharply. Apart from this, we also observed one thing – GAIN can also restore connectivity for a large scale failure, where the number of partitioned segments is large. But other algorithms are not able to populate RNs in such a large-scale type of failure.

Figure 8b shows the experiment results only for GAIN, but with various numbers of disconnected segments considered. This figure depicts the relation between the number of segments and the range of RN. It can be concluded that when the transmission range of RN is high, the number of segments does not affect the final count of RNs (i.e. *Count_Relay*). Despite the large value of N_{Seg} GAIN consistently maintains the minimum number of RNs. Instead of this, when there is a slight change in the value of N_{Seg} and with a low value of R_r, the final count of RN *Count_Relay* increases quickly. This simulation is performed for three different values of N_{Seg} equaling 5, 6 and 7. At last, we conclude, based on the results of the experiments, that GAIN's performance is favorable in comparison with other comparative approaches, for a large value of R_r. GAIN shows a below-benchmark performance for a low value of R_r, due to the randomness of GWO.

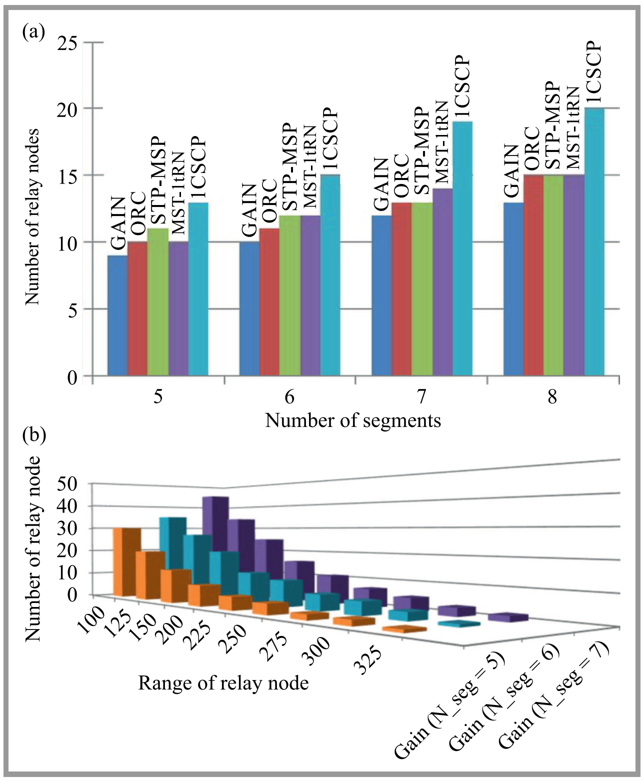

Fig. 8. Comparison of GAIN and other algorithms ($R_r = 200$) (a) and comparison of GAIN results for varying numbers of segments (b). (For color pictures visit www.nit.eu/publications/journal-jtit)

8. Conclusion and Future Scope

Generally, HWSNs operate in a harsh and hostile environment, where SNs are more susceptible to failures. A large-scale failure of SNs results in the creation of a partitioned network. Disjoint segments are no more able to communicate with other. Therefore, we proposed the grey wolf optimizer algorithm to repair the segmented heterogeneous wireless sensor network and to restore lost connectivity within the disjoint HWSN. The proposed approach operates in rounds, and in each round one RN position is returned. If the candidate position is inside the convex hull, then it is considered as an acceptable RN position, otherwise we discard it. When the number of remaining segments falls below three, then the situation is considered to be the terminating condition of GAIN. In the termination phase, the position of RNs is calculated with Euclidean equality.

We evaluate the performance of GAIN and compare it with other well-known algorithms of similar nature. The simulation results have confirmed that our proposed solution outperforms other algorithms and populates a minimum number of nodes for a large communication range of RNs. The resultant topology shows a good connectivity with balanced traffic loads. In the future, we will be working to test the proposed solution on a real testbed.

References

[1] V. Ranga, M. Dave, and A. K. Verma, "Network partitioning recovery mechanisms in WSANs: A survey", *Wireless Personal Commun.*, vol. 72, no. 2, pp. 857–917, 2013 (doi: 10.1007/s11277-013-1046-7).

[2] V. Ranga, M. Dave, and A. K. Verma, "Relay node placement for lost connectivity restoration in partitioned wireless sensor networks", *in Proc. Int. Conf. on Electron. and Commun. Syst. ECS 2015*, Bacelona, Spain, 2015, pp. 170–175.

[3] V. Ranga, M. Dave, and A. K. Verma, "Relay node placement to heal partitioned wireless sensor networks", *J. of Comp. & Elec. Engin.*, vol. 48, no. C, pp. 371–388, 2015 (doi: 10.1016/j.compeleceng.2015.09.014).

[4] G. Kumar and V. Ranga, "Meta-heuristics for relay node placement problem in wireless sensor networks", in *Proc. 4th Int. Conf. on Parallel, Distrib. and Grid Comput. PDGC 2016*, Waknaghat, India, 2016, pp. 375–380 (doi: 10.1109/pdgc.2016.7913180).

[5] J. Kennedy, "Particle swarm optimization", in *Proc. of Int. Conf. on Neural Netw. ICNN'95*, Perth, WA, Australia, 1995, pp. 1942–1948 (doi: 10.1109/ICNN.1995.488968).

[6] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization", *IEEE Comput. Intell. Mag.*, vol. 1, no. 4, pp. 28–39, 2006 (doi: 10.1109/MCI.2006.329691).

[7] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*, 1st ed. Oxford University Press, 1999 (ISBN: 978-0195131598).

[8] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer", *Adv. in Engin. Software*, vol. 69, pp. 46–61, 2014 (doi: 10.1016/j.advengsoft.2013.12.007).

[9] S. Lee and M. Younis, "Optimized relay node placement for connecting disjoint wireless sensor networks", *Comp. Netw.*, vol. 56, no. 12, pp. 2788–2804, 2012 (doi: 10.1016/j.comnet.2012.04.019).

[10] J. M. Lanza-Gutierrez and J. A. Gomez-Pulido, "Assuming multiobjective metaheuristics to solve a three-objective optimisation problem for relay node deployment in wireless sensor networks", *Appl. Soft Comput.*, vol. 30, no. C, pp. 675–687, 2015 (doi: 10.1016/j.asoc.2015.01.051).

[11] C. Ma, W. Liang, M. Zheng, and H. Sharif, "A connectivity-aware approximation algorithm for relay node placement in wireless sensor networks", *IEEE Sensors J.*, vol. 16, no. 2, pp. 515–528, 2016 (doi: 10.1109/JSEN.2015.2456931).

[12] C. Zhao and P. Chen, "Particle swarm optimization for optimal deployment of relay nodes in hybrid sensor networks", in *IEEE Congr. on Evolut. Comput. CEC 2007*, Singapore, 2007, pp. 3316–3320 (doi: 10.1109/CEC.2007.4424899).

[13] E. L. Lloyd and G. Xue, "Relay node placement in wireless sensor networks", *IEEE Trans. on Computers*, vol. 56, no. 1, pp. 134–138, 2007 (doi: 10.1109/TC.2007.250629).

[14] I. F. Senturk, S. Yilmaz, and K. Akkaya, "A game-theoretic approach to connectivity restoration in wireless sensor and actor networks", in *Proc. IEEE Int. Conf. on Commun. ICC 2012*, Ottawa, ON, Canada, 2012, pp. 7110–1714 (doi: 10.1109/ICC.2012.6364848).

[15] P. Basu and J. Redi, "Movement control algorithms for realization of fault-tolerant ad hoc robot networks", *IEEE Network*, vol. 18, no. 4, pp. 36–44, 2004 (doi: 10.1109/MNET.2004.1316760).

[16] A. A. Abbasi, M. Younis, and K. Akkaya, "Movement-assisted connectivity restoration in wireless sensor and actor networks", *IEEE Trans. on Parallel and Distrib. Syst.*, vol. 20, no. 9, pp. 1366–1379, 2009 (doi: 10.1109/TPDS.2008.246).

[17] K. Akkaya, F. Senel, A. Thimmapuram, and S. Uludag, "Distributed recovery from network partitioning in movable sensor/actor networks via controlled mobility", *IEEE Trans. on Computers*, vol. 59, no. 2, pp. 258–271, 2010 (doi: 10.1109/TC.2009.120).

[18] W. Wang, V. Srinivasan, and K.-C. Chua, "Using mobile relays to prolong the lifetime of wireless sensor networks", in *Proc. 11th Ann. Int. Conf. on Mob. Comput. and Network. MobiCom 2005*, Cologne, Germany, 2005, pp. 270–283 (doi: 10.1145/1080829.1080858).

[19] M. Y. Sir, I. F. Senturk, E. Sisikoglu, and K. Akkaya, "An optimization-based approach for connecting partitioned mobile sensor/actuator networks", in *Proc. IEEE Conf. on Comp. Commun. Worksh. INFOCOM WKSHPS 2011*, Shanghai, China, 2011, pp. 525–530 (doi: 10.1109/INFCOMW.2011.5928869).

[20] R. C. Shah, S. Roy, S. Jain, and W. Brunette, "Data MULEs: Modeling a three-tier architecture for sparse sensor networks", in *Proc. of 1st IEEE Int. Worksh. on Sensor Netw. Protocols and Appl.*, Anchorage, AK, USA, 2003 (doi: 10.1109/SNPA.2003.1203354).

[21] G. Wang, G. Cao, T. La Porta, and W. Zhang, "Sensor relocation in mobile sensor networks", in *Proc. IEEE 24th Ann. Joint Conf. of the IEEE Comp. and Commun. Soc. INFOCOM 2005*, Miami, FL, USA, 2005, vol. 4, pp. 2302–2312 (doi: 10.1109/INFCOM.2005.1498517).

[22] C. Muro, R. Escobedo, L. Spector, and R. Coppinger, "Wolf-pack (Canis lupus) hunting strategies emerge from simple rules in computational simulations", *Behavioural Processes*, vol. 88, no. 3, pp. 192–197, 2011 (doi: 10.1016/j.beproc.2011.09.006).

[23] R. L. Graham, "An efficient algorithm for determining the convex hull of a finite planar set", *Inform. Process. Lett.*, vol 1, no. 4, pp. 132–133, 1972 (doi: 10.1016/0020-0190(72)90045-2).

[24] X. Cheng, D.-Z. Du, L. Wang, and B. Xu, "Relay sensor placement in wireless sensor networks", *Wireless Networks*, vol. 14, no. 3, pp. 347–355, 2008 (doi: 10.1007/s11276-006-0724-8).

[25] D. Yang, S. Misra, X. Fang, G. Xue, and J. Zhang, "Two-tiered constrained relay node placement in wireless sensor networks: Efficient approximations", in *Proc. 7th Ann. IEEE Commun. Soc. Conf. on Sensor Mesh and Ad Hoc Commun. and Netw. SECON 2010*, Boston, MA, USA, 2010, pp. 1–9 (doi: 10.1109/SECON.2010.5508241).

[26] G. Robins and A. Zelikovsky, "Tighter bounds for graph Steiner tree approximation", *SIAM J. on Discrete Mathem.*, vol. 19, no. 1, pp. 122–134, 2005 (doi: 10.1137/S0895480101393155).

Gaurav Kumar received his B.Tech. degree in Computer Science and Engineering from Gautam Buddh Technical University Lunckow in 2012 and M.Tech. degree in Computer Engineering from NIT Kurukshetra Haryana, India in 2017. Currently, he is working as Assistant Professor in University Departments, Rajasthan Technical University, Kota Rajasthan. His interest areas are fault tolerance in Wireless sensor networks, Internet of Things, and Big Data.
E-mail: gngauravnain@gmail.com
Department of Computer Engineering
National Institute of Technology Kurukshetra
Haryana, India

Virender Ranga received his Ph.D. degree from Computer Engineering Department of National Institute of Technology, Kurukshetra, Haryana, India in 2016 followed by M.Tech. degree in 2004, and B.Tech. degree in 2001. He has published more than 50 research papers in various International SCI Journals in the area of computer communications as well as reputed international conferences. Presently, he is an Assistant Professor in the Computer Engineering Department since 2008. He has been conferred by Young Faculty Award in 2016 for his contributions in the field of computer communications. He is an active reviewer of many reputed journals of IEEE, Springer, Elsevier, Taylor & Francis, Wiley and Inder-Science. His research areas include wireless sensor and ad-hoc networks, data science, IoT security and FANET security.

E-mail: virender.ranga@nitkkr.ac.in
Department of Computer Engineering
National Institute of Technology Kurukshetra
Haryana, India