International Journal of Grid and Distributed Computing
Vol. 11, No. 5 (2018), pp.11-28
http://dx.doi.org/10.14257/ijgdc.2018.11.5.02

Whale Optimizer to Repair Partitioned Heterogeneous Wireless Sensor Networks

Gaurav Kumar Verma[1] and Virender Ranga[2]

[1,2]*Department of Computer Engineering National Institute of Kurukshetra, Kurukshetra, Haryana 136119*
[1]*gaurav.verma2329@gmail.com*, [2]*virender.ranga@nitkkr.ac.in*

Abstract

The problem of network partition in ad-hoc networks received attention in the recent years. Many solutions have been proposed such as algorithms based, heuristics based, approximations based and meta-heuristic based to place additional relay nodes in partitioned heterogeneous wireless sensor networks to resume its operation. However, placing additional relay nodes in the partitioned network is shown an NP-Hard problem, because locations for relay node placements are not known in advance. Meta-heuristics are proven best-suited solutions to solve such kind of NP-Hard problem as well as optimization problem due to their problem independent and stochastic nature. In this research paper, we have introduced a network partition problem and developed a new nature inspired solution called Whale Optimizer to Repair Partitioned Heterogeneous wireless sensor networks (WORPH) based on the social behaviour of whales in the nature. In the proposed solution, a whale tries to find the optimal locations for attacking its prey. We have mimicked the said behaviour of whales in our proposed solution while considering the initial locations of deployed RNs inside disjoint partitions. The observed optimal positions are being used to find the optimal locations for deploying new RNs in such a way that partitioned network is restored in an optimal way. The simulation results are observed and compared with state-of-the-art approaches to prove the effectiveness of our proposed solution.

Keyword: *Wireless sensor networks, Meta-Heuristics, Connectivity restoration, Relay node Placement, Optimization*

1. Introduction

In today's scenario, Wireless Sensor Networks (WSN) is shifting from homogenous networks (all nodes have same capabilities) to heterogeneous networks (different kinds of nodes have different capabilities). This network is called as Heterogeneous WSNs (HWSNs). The main feature of HWSNs is that sensor nodes (SN) have different communication range, ability to sense different parameters presents in the environment (*e.g.*, temperature, pressure, humidity, proximity, the density of poisonous gases *etc.*,) *etc.*, SNs are cheaper, small and have small batteries which decay with the period of time. Hence, HWSNs are considered widely in fields like defense, medical, monitoring environmental condition [1] where it is difficult for human beings to survive because it makes an easy, cost-efficient and safe way to set up a network. Due to harsh conditions in that environment, it is difficult to maintain proper coverage and connectivity at the same time [1], [2]. At the outset HWSNs, basically a collection of SNs which collects necessary information such as temperature, the density of poisonous gases, pressure, and humidity from the surrounding environment. The sink node or also called collector collects generated information (data) from the network and pass it to the base station (BS). SNs

Received (November 22, 2017), Review Result (March 1, 2018), Accepted (March 10, 2018)

ISSN: 2005-4262 IJGDC

are also called as source nodes as they collect all the required information from environment and process it and then forward it to collector node. In order to do this task nodes usually, a wireless network is established, generally radio communication, between other nearby nodes within range of SN. As SNs are cheaper and work in such harsh environment which may cause their battery power to drain fast. Once a SN fails, the data flowing through it towards the BS stops and that data will know flow through other nodes cause extra burden on these nodes. This lead to further failure of SNs due to extra load put on these sensors. When a large number of SNs failed in the network, it may lead to the creation of multiple partitions inside the network. In a partitioned network, SN can only communicate with those SNs that are present in the respective partition and hence they are cut off from rest of network. To restore lost connectivity again we require placing of special devices known as Relay Nodes (RNs). RNs are special devices having more battery power (some time have other means to charge the battery), have movement capability and they can collect all information (data) form the SNs and forward it to other neighbouring RNs and ultimately data reaches towards the BS. Due to deployment of RNs in the network, it distributes the load of SNs, hence it enhances the lifetime of the restored network. Deploying RNs inside partitioned network to re-join the disconnected partitions in HWSN is a well-known problem. As RNs are costly and deploying a large number of RNs increases the overall budget of the network. Hence, we have to place a minimum number of RNs at optimal positions. Optimal placement of RNs is proved to be an NP-Hard problem [2], [3]. Many techniques have been proposed such as algorithms based, heuristics based and approximations based to solve Relay Node Placement Problem (RNPP) in HWSNs. However, meta-heuristics have proven to be a best-suited technique to solve such NP-Hard problem (RNPP) as well as optimization problem due to their problem independent and stochastic nature as said above. Some popular meta-heuristics are already proposed which have solved RNPP are Grey Wolf Optimizer (GWO) [4], Particle Swarm Optimization (PSO) [5], Firefly Algorithm [6], Ant Colony Optimization (ACO) [7], and Genetic Algorithm (GA) [4]. Meta-heuristics have been classified into three main categories: evolutionary, physics-based, and Swarm Intelligence (SI) algorithms. All meta-heuristics accomplish their search process in two different stages: exploration and exploitation. Exploration covers whole solution space by a random walk of their search agents and exploitation is a local search and covers only a certain part of solution space by focusing towards candidate position [8]. Keeping in mind the previously proposed solutions to obtain better results from meta-heuristic algorithms, we have proposed a solution to solve partitioned HWSN with the optimal deployment of RNs. We have proposed a meta-heuristic optimization algorithm based solution depends on the social behaviour of whales. In the proposed solution, a whale tries to find the optimal location to attack its prey. We have mimicked this behaviour of whales to consider the initial locations of deployed RNs inside partitions. The calculated positions are used to find the optimal locations for deploying new RNs in such a way that partitioned network lost connectivity is restored in an optimal way. Our work is organised as follow: section 2 discusses all the previous work done in RNPP. Section 3 and 4 describes our system model and problem description. In Section 5 mapping of whale optimizer with our problem is shown. Section 6 and 7 describe our proposed solution with an example. The working of proposed work and pseudo code is in Section 7. In Section 8, we have shown the comparison of our proposed solution with state-of-the-art algorithms. Finally, we have discussed a precise conclusion in Section 9.

2. Related Works

Many solutions have been discussed in last few years to solve network partition problem that happens due to the failure of a large scale of deployed nodes in WSNs. All proposed solutions, like placing RNs or changing positions of existing RNs, can be

categorised into four types: Algorithm based, Meta-heuristics based, Approximation based and Heuristics based. To best our knowledge, a little work is being carried out in this domain by using meta-heuristics based approach.

2.1. Meta-Heuristics Based Techniques

Xu Yi-Han *et al.,* [9] solve the network partitioning problem by placing minimum additional RNs to form an interconnected network. The authors have formulated the problem as STP with minimum SPs and a BEL problem. Kumar G. *et al.,* [4] proposed swarm intelligences based solution for RNPP in WSNs. The author has proposed Grey wolf optimizer Algorithm for repairing segmented heterogeneous wireless sensor Network (GAIN) to restore lost connectivity to the partitioned network. Sharma R. *et al.,* [5] suggested, Federating Network using Particle Swarm Optimization (FN-JPSO) which is applied to restoring the lost connectivity in WSN. The author's approach first finds an appropriate node for each disconnected partition, and then SPs are created for next step. These SPs then used to create random spanning trees, then further as particles in FN-JPSO to provide an optimal interconnected network solution. In this way, FN-JPSO finds a local solution for each partition which provides the global best solution. Azharuddin M. *et al.,* [10] main aim is to reduce the number of RNs to be placed at the potential position and maximise the connectivity between the SNs and RNs. The authors have proposed a Genetic Algorithm for RNPP which is castoff to get a fast and less expensive solution. Lanza-Gutierrez J. M. *et al.,* [6] focus on three conflicting objectives: Average Sensitivity Area, Average Energy and Network Reliability in WSN through RNP. Author have considered different MO meta-heuristics to solve RNPP five of them to Evolutionary Computation, two standard GAs and two swarm intelligence based algorithms MO-Firefly algorithm and MO-Artificial Bee Colony which are based on the behaviour of fireflies and honey bees.

2.2. Heuristics Based Techniques

Chouikhi S. *et al.,* [11] focus on the connectivity re-establishment in multichannel WSNs. For these two solutions based on graph coloring and the Steiner tree problem are proposed using centralized heuristics to solve the problem. Lalouani, W. *et al.,* [12] proposed a new geo-Steiner technique based on the straight skeleton approach and named it as Boundary-aware optimized Interconnection of Disjoint segments (BIND). BIND is proposed for static segment topology. BIND restores network connectivity by creating the shortest length topology in a Euclidian plane which interconnects a subset of SNs on partition boundaries by placing extra SPs which leads to the creation of a path between every pair of partition. In this approach, authors have used hyper graphs formalization to overcome the drawbacks of existing recovery solutions. Yuan B. *et al.,* [13] modelled and framed Wireless Underground Sensor Network (WUSN) with a goal to place a minimum number of aboveground RNs to relay traffic. In this, authors have formulated the problem in two phases (i) Load-Unrestricted RN Selection (LURNS) problem, (ii) Load-Balanced SN Assignment (LBSNA) problem. In the LURNS lifetime of the network is maximized by selecting a subset of SNs to placed RN and LBSNA problem of uniformly assigning the workload is taken into consideration by reassigning SNs to RNs. To solve LURNS two heuristics are proposed, the first algorithm works greedily and used to select a position to place RN where the second algorithm is used reduce the number of RNs which are placed before. Ma C. *et al.,* [14] propose a Pruning and Substitution based Heuristics (PSH) to solve Delay constrained RNPP. Proposed algorithm consists of two phases the covering phase which uses the shortest path based algorithm to place RN at a subset of predefined placement location ensuring coverage and delay constraints and the connecting phases uses a tree-based connecting (TCA) algorithm which establish a high tier network connectivity to ensure this TCA create the shortest path tree which connects the RNs

previously placed in the previous phase to the sink and then saves the placed RNs by pruning or substituting them one by one with RNs deployed at location where numerous possible paths passes through. Bhattacharya A. *et al.*, [15] studied on an optimal relay and sink node placement problem and proposes a polynomial time approximation algorithm (Smart Select) which is a greedy algorithm for selection of RNs and sink nodes. This solution gives an optimal solution if a single sink node is used otherwise diminishes to a greedy algorithm for weighted set cover. To improve the proposed work, authors have proposed a polynomial time heuristics a Destroy and Repair Heuristics which is a placement strategy for RNs and sink node. Kimençe Ş. *et al.*, [16] propose RNPP on weighted terrain structure to fulfill WSN connectivity with an aim to minimise the total weight of the positions on which RNs are placed. For this author proposed two solutions one is a mathematical formulation to find the optimal solution and a polynomial time heuristics to find a near-optimal solution in a certain sensible time limit.

2.3. Algorithm Based Techniques

Nitesh K. *et al.*, [17] use Jarvis March approach to create a spiral traversal for SNs, a traditional technique for creation of convex hull named as Minimum Assured Coverage and Connectivity (MACC). To ensure every SNs will be covered by a spiral algorithm, MACC incorporates them as vertices of spiral, to do so a spiral sequence is generated by Jarvis March in which a sequence number is given to every SNs. Problem with this approach is that it requires more computational time and high cost. Chang H. Y. *et al.*, [18] propose a new Jigsaw-based relay placement algorithm (JRPI) for indoor WSNs. Senturk I. F. *et al.*, [19] propose two distributed RNP algorithms to ensure network recovery for segmented WSN. The first algorithm virtual force-based movement approach gradually places new RNs to stretch the network. In the second one the game-theoretic approach the leader RNs based on probability distribution function (pdf) of partitions to be connected is determined. A priority is given for higher pdf partition and recovery proceeds in incremental pdf values of partitions until system-wide unique Nash equilibrium is achieved.

2.4. Approximation Algorithm Based Techniques

Wang X. *et al.*, [20] proposed Obstacle–Avoid connectivity restoration strategy based on Straight Skeletons (OASS). OASS uses both polygon based selection strategy in presence of obstacles in the environment and a straight skeleton based SMT is built. OASS is a three-phase technique; the first phase focuses on designing an adapted MST construction algorithm Ad-Prim for selection process. The second phase focuses on an obstacle-avoid algorithm OA to create shortest paths around obstacles. The third phase focuses on a straight skeleton based algorithm SSIN for shortest inter-components connection and obstacle avoidance. Ma C. *et al.*, [21] propose a Two-phase Set-Covering-based Algorithm (TSCA). First, a Connectivity-aware Covering Algorithm (CCA) for covering phase for SNs, helps in reducing RNP time in next phase. Second a Set-Covering-based Algorithm (SCA) for the connecting phase. Third an algorithm-TSCA based on CCA and SCA. Gao Z. *et al.*, [22] propose a solution for fault tolerant RNP, Diff-BS node and relay node hybrid communication graph, (DBY-HCG) which distinguishes RNs form BSs. Hence, decreases the cost caused by keeping k-connectivity between RNs and BSs. In [22] author proposed another algorithm, k-extended constrained independent RNP with BS, denoted by keCi- RNPB which can be used in both constrained and unconstrained version. Finally, an approximate algorithm focusing on finding the minimum length of k-vertex disjoint paths in connected sub-graph is proposed to extend previously proposed algorithm.

3. System Model

In this paper, a uniform structure of HWSN is taken in which SNs are placed randomly in a predefined area for this any node placement strategy which can use random deployment technique. SNs gather useful information by observing the environment and then forward this toward Sink Node/ Base Station. BS is situated at some predefined position to collect all useful information in the network. SNs are battery powered and their battery discharges gradually with time which leads to failure of SNs if this discharging is fast or WSNs are deployed in harsh environments where environmental factors like a forest fire, blast in battlefield, earthquake, landslide and volcanic eruption also affect their working leading to failure of large no. SNs. This failure of SNs on large scale creates disjoint partitions. Failure detection is done by sensing the traffic flow coming from the network at the base station. If the amount of traffic flow reduces than some certain limit, BS announces the failure of sensor nodes and start collecting information of every SNs which are alive. This can be done with SNs identification number.

4. Problem Description

Our proposed solution is to deploy minimum number of RNs inside the damaged area. At the outset, it heals the lost connectivity of disjoint networks. Initially, for deployment of SNs in the predefined area, any localization algorithm can be used (centralized or distributed [23]). We prefer to use distributed approach as centralized may case latency in the network. After this, BS receives information about the complete network. Due to the large number of SNs failure inside the network, partitions are created, which are detected by BS via observing a sudden decrease in traffic flow in the network. Initially, one RN is needed to be placed for every partition Par_i. Hence at least N_{par} RNs are required to reconnect every disjoint partition. The following assumptions are taken in our proposed solution:

i) All SNs and BS are considered static.

ii) All RNs have an equal transmission range of RN_r units.

iii) HWSN has its underlying routing protocol for a source to destination communication.

5. The Whale Optimizer

5.1. Motivation

The whale optimization algorithm (WOA) proposed by S. Mirjalili *et al.*, [8] is based on the social behaviour of whales (humpback whales (HW)). Whales are social animals, lives in groups or alone. Whales have special hunting technique called as Bubble-Net Feeding (BNF) technique. HW prefers to hunt a school of krill/ small fishes near to surface. BNF is done by producing bubbles along a circle or '9'-shape path. There are two forms of BNF upward spirals and double loops. HW recognize the location of their prey and encircle them by diving 12 m down and then start producing bubbles in a spiral shape around their prey and swim up toward the surface to attack. In our proposed solution, RNs, which are placed initially one for each disjoint partition, are considered as whales. Position of best prey is considered as new position to deploy new RNs. As whales know the positions of prey. In the similar way, we also considers the positions (like whales know the positions of prey) where RN are to be placed. However, we need to find the required positions in minimum time. Here whale and prey both are moving in space. Prey is trying to save himself from whales and whales are changing their position with respect to prey position to find the best attacking position to attack on prey (a position where prey is not predicting that attack will happen or where attacking leads to more damage (more

amount of food)). In this way, Whale Optimizer (WO) returns position of leader whale node which is considered the best position to place new RNs. The complete solution is shown in Section 6.

5.2. About Whales

The Whale Optimization Algorithm (WOA) proposed by S. Mirjalili *et al.*, [8] is based on the social behaviour of Whales. In this proposed work, they have considered Humpback Whales (HWs) out of all spices for evaluation.

5.2.1. Encircling the Prey

HWs know the positions of their prey. Since the optimal position of prey in the search space is not known a priori. Hence it is considered that the current best candidate solution is the target prey or it is close to optimum. In our proposed solution we deploy a RN per partition. Now as we do not know the location of new deployed RN a priori. Therefore, our proposed algorithm assumes the current best candidate location is our target prey or it is the location near to optimal. Following equations are used to find the optimal location of RN:

$$\overrightarrow{D_{leader}} = \left|\vec{C}.\overrightarrow{leader_{pos}}(t) - \overrightarrow{Position}(t)\right| \quad (1)$$

$$\overrightarrow{Position}(t+1) = \overrightarrow{leader_{pos}} * (t) - \vec{A}.\vec{D}_{leader} \quad (2)$$

Where $\vec{A}$ and $\vec{C}$ indicates a coefficient vector. $\overrightarrow{leader_{pos}}$ indicates the best solution at that time. $\overrightarrow{Position}$ is the position vector. | | indicates absolute value and . indicates an element-by-element multiplication.

$$\vec{A} = \overrightarrow{2a}.\overrightarrow{r_1} - \vec{a} \quad (3)$$

$$\vec{C} = 2.\overrightarrow{r_2} \quad (4)$$

Where $\overrightarrow{r_1}$ and $\overrightarrow{r_2}$ are two random vectors in [0,1] and $\vec{a}$ is decreasing linearly from 2 to 0.

5.2.2. Bubble-Net Attacking Technique (Exploitation Phase)

It is designed as follow:

1. ***Shrinking Encircling Mechanism*:**

   Since calculated position from equation 1 depends upon $\vec{A}$ which is a random value in the interval $[-a, a]$ and $\vec{a}$ is linearly decreasing from 2 to 0 with the execution of while loop. Due to this behaviour of $\vec{a}$, $\vec{A}$ is also changing its random values from [-1, 1]. The result of current search agent is changing w.r.t. its position. Hence, its value varies between a current location and current selected best agent.

2. ***Spiral Updating Position*:**

   HW first calculates how far HW is from target prey (current best solution). By using equation (5). The helix-shaped movement of HW can be mimic by using equation (6).

$$\overrightarrow{Dis} = \left|\overrightarrow{leader_{pos}}(t) - \overrightarrow{Position}(t)\right| \quad (5)$$

$$\overrightarrow{Position}(t+1) = \overrightarrow{Dis}\,.\, e^{bl}.\cos(2\pi l) + \overrightarrow{leader_{pos}}(t) \quad (6)$$

Where b is a constants defined for logarithmic spiral shape. l is a random number from $[1, 2]$ and . indicates an element-by-element multiplication.

As HW moves around the target prey within shrinking circle and along a spiral shaped path simultaneously. Hence to mimic this approach we are using $'p'$ a random number between [0,1] used to define probability. Therefore, WO uses p to choose either to go for (i) or (ii).

$$\overrightarrow{Position}(t+1) = \begin{cases} Equation\ (2) & if\ p < 0.5 \\ Equation\ (6) & if\ p \geq 0.5 \end{cases} \quad (7)$$

Along with these two attacking techniques, HW also searches for prey in random space.

Algorithm 1 Pseudo code of Whale Optimizer algorithm

```
Procedure WO(initial Population):
  Initialize population of HW search agents as Position_i (i = 1,2, ..., n)
  Select a random leader_pos from Position for initial best search agent.
  while (t < Max number of iterations) do
    for each HW search agent do
      Calculate r1, r2, A, C, l, p
      if (p < 0.5) do
        if (|A| ≥ 1) do
          Select a random leader as Rand_leader
          Update position of current HW search agent by using equation (9)
        end if
        else if (|A < 1) do
          Update current HW search agent position by using equation (2)
        end else if
      else if (p ≥ 0.5) do
        Update current HW search agent position by using equation (6)
    end for
    if check any HW search agent is gone beyond search space do
      Take it back in search space.
    Select a new random leader_pos from current solution space
    Increment t
  end while
return leader_pos
end procedure
```

5.2.3. Search for Prey (Exploration Phase)

Because this is an exploration phase, thus, we have performed a global search here. Therefore, we have used the value of $\vec{A}$ for given HW to do an exploration of prey if the value of $|\vec{A}| > 1$ by using equation (9). In this phase, the current search agent updates its position w. r. t. randomly chosen HW agent (which may not be the best search agent). It's mathematical modeling is described as follow:

$$\overrightarrow{D_{leader}} = \left|\vec{C}.\overrightarrow{Rand_{leader}} - \overrightarrow{Position}(t)\right| \quad (8)$$

$$\overrightarrow{Position}(t+1) = \overrightarrow{Rand_{leader}} - \vec{A}.\overrightarrow{D_{leader}} \quad (9)$$

$\overrightarrow{Rand_{leader}}$ is a new leader selected randomly from given HW search agents for current execution. For more details about WO refer [8].

6. Our Proposed Solution

Our proposed solution solves RNPP for the restoration of disjoint partitioned networks by using WO. The mapping of WO with our proposed algorithm has already been discussed in section V. All disconnected partitions are considered as HW and an optimal location for attacking prey is found by using WO which are considered as RN positions for connecting the disjoint partitioned network. Initially, we have a fully connected and functional network but due to failure of large number of SNs multiple partitions are created inside the networks. Further, 2D vector positions of disconnected partitions are used to fetch the initial population of HW in WO and WO returns the best location of HW for attacking prey. This is an iterative process. For each iteration, the population of HW changes due to change in the number of partitions inside the disjoint network. Our proposed algorithm calculates RNs position inside the damaged area to restore lost connectivity with the help of WO. Our proposed solution consists of four phases:

- Placing initial RNs
- Discovering the Neighbours
- Using WO to place RNs
- Termination Phase

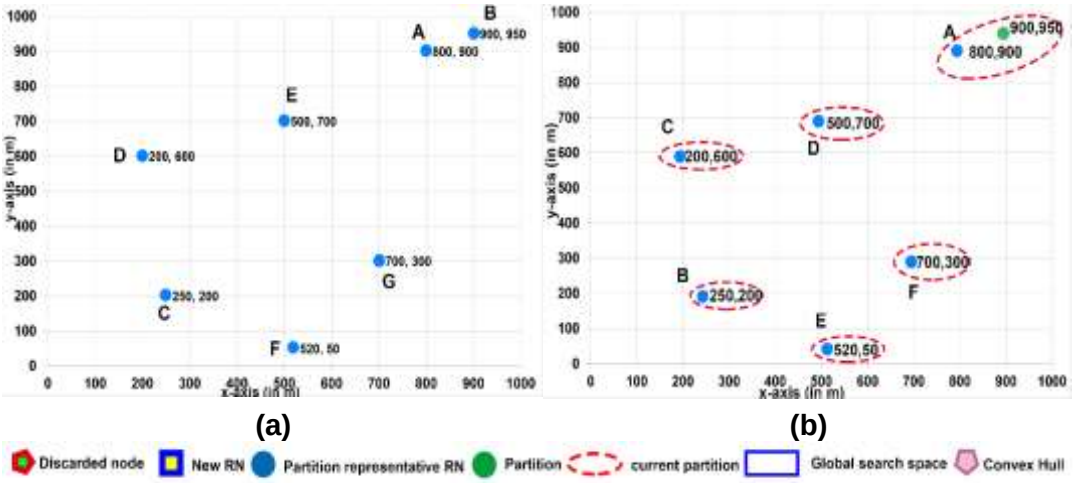

Figure 1. (a) Showing Initial Positions of Each Partitions A-G (b) Showing Neighbouring Partitions Where $RN_r = 150$ Meter

6.1. Placing Initial RNs

This is our initial phase, as we have said initial network is well connected. The traffic flow coming towards BS from the network is stable. But after a large number of SNs fails then the traffic flow inside the network decreases which is detected by BS. BS declares failure and starts collecting data of live SNs to find the number of partitions. Now, our proposed solution is used to find the optimal locations of RN to be placed to restore the disjoint partitions. Initially, our proposed solution is placing one RN for each partition as shown in Figure 1(a). Each RN acts as a gateway for that partition and any communication with the SNs in that partition would be possible only through this RN and it will act as a cluster head. Figure 1(a) shows a disjoint network with 7 partition named as A (800,900), B (900,950), C (250,200), D (200,600), E (500, 700), F (520, 50), G (700,300) with their XY coordinate values. Our proposed solution takes the XY coordinates as input and produces a number of RNs with their XY coordinates as output.

6.2. Discovering the Neighbours

This is our second phase. In this phase, each RN finds whether there is a neighbour present or not. For this, we have used the Euclidean distance between the two RNs (such as $\left(A\ (\ x_1, y_1)\ and\ \ B(x_2, y_2)\right)$ by using Equation (10)

$$ED_{A,B} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{10}$$

As shown in Figure 1(b) each partition finds its neighbour as explained in lines (2-9) of pseudo code 2. As shown in the Figure 1(a) we have 7 partitions with 7 RNs for each. Because the range of RNs is higher than SNs as a result, there may be a chance of communication between two partitions without deploying additional RN in between them. Hence by checking the condition as explained in line 5 two partitions are neighbours. If that happens then proposed solution combines these partitions as one. This is shown in figure 1(b) that A (800, 900) and B (900, 950) are neighbours. Therefore, they are considered as one partition. Hence new partitions will be A (800, 900), B (250, 200), C (200, 600), D (500,700), E (520, 50), F (700, 300). This phase is helpful in reducing the number of partitions further hence RNs. Also, this is observed in the simulations results that $Count_{RNs} \geq N_{par}$ because we are placing one RNs for each partition in the initial phase.

6.3. Using Whale Optimizer to Place RNs

In this phase, we call WO algorithm which gives a location to deploy *New_RN*. Here we check whether to deploy *New_RN* at that position or not. Thus, this phase is shown in rounds. This procedure is explained in lines (10-30) of pseudo code. For connecting the disjoint network we are placing RNs in an inward direction towards the centre of the affected area. Hence we have passed the coordinates of current partitions obtained from the second phase to WO algorithm. Now as explained earlier WO is a meta-heuristic algorithm hence it has both exploration phase and exploitation phase means it will search in global as well as the local area. Therefore we reduce the global search space of WO by finding the minimum value of XY coordinates and the maximum value of XY coordinate of given partitions. Now WO will not give value beyond this. Hence less chance of inappropriate values (means less discarded RNs). Now a *New_RN* will be considered to deploy if it is inside the convex hull of partitions. After placing the *New_RN* we again find the neighbours of *New_RN* means how many partitions are joined by it. If no partition is in range of *New_RN* than that node will be considered as a new partition for next round as explained in line 22 else all the partitions are joined together as one as in lines (23-27). Figure 2(a) shows 6 partitions. On passing these partitions to WO a point (90, 678) is returned when we check it whether it is inside convex hull or not we find it is outside of convex hull hence we have discarded it. The rectangle is shown in Figure 2(a) with blue line represents the space in which WO has hunted. On next round, WO returns a point (607, 557) which is inside the convex hull as shown in Figure 2(a). Hence we have considered it and we have to find its neighbours. Figure 2(b) shows that point (607, 557) have two neighbour points (500, 700) of partition D and point (700, 300) of partition F thus consider them as one partition and point (500,700) represents this partition. Hence, 5 partitions left as A (800, 900), B (250, 200), C (200, 600), D (500, 700) and E (520, 50) for the next round. Similarly, WO returns a new point (442, 610) which is inside the convex hull as shown in Figure 2(c). Hence we have to find its neighbours according to the line (17, 21). Point (200, 600) of partition C and point (500, 700) of partition D are the neighbours of point (442, 610) hence consider these points one partition as shown in

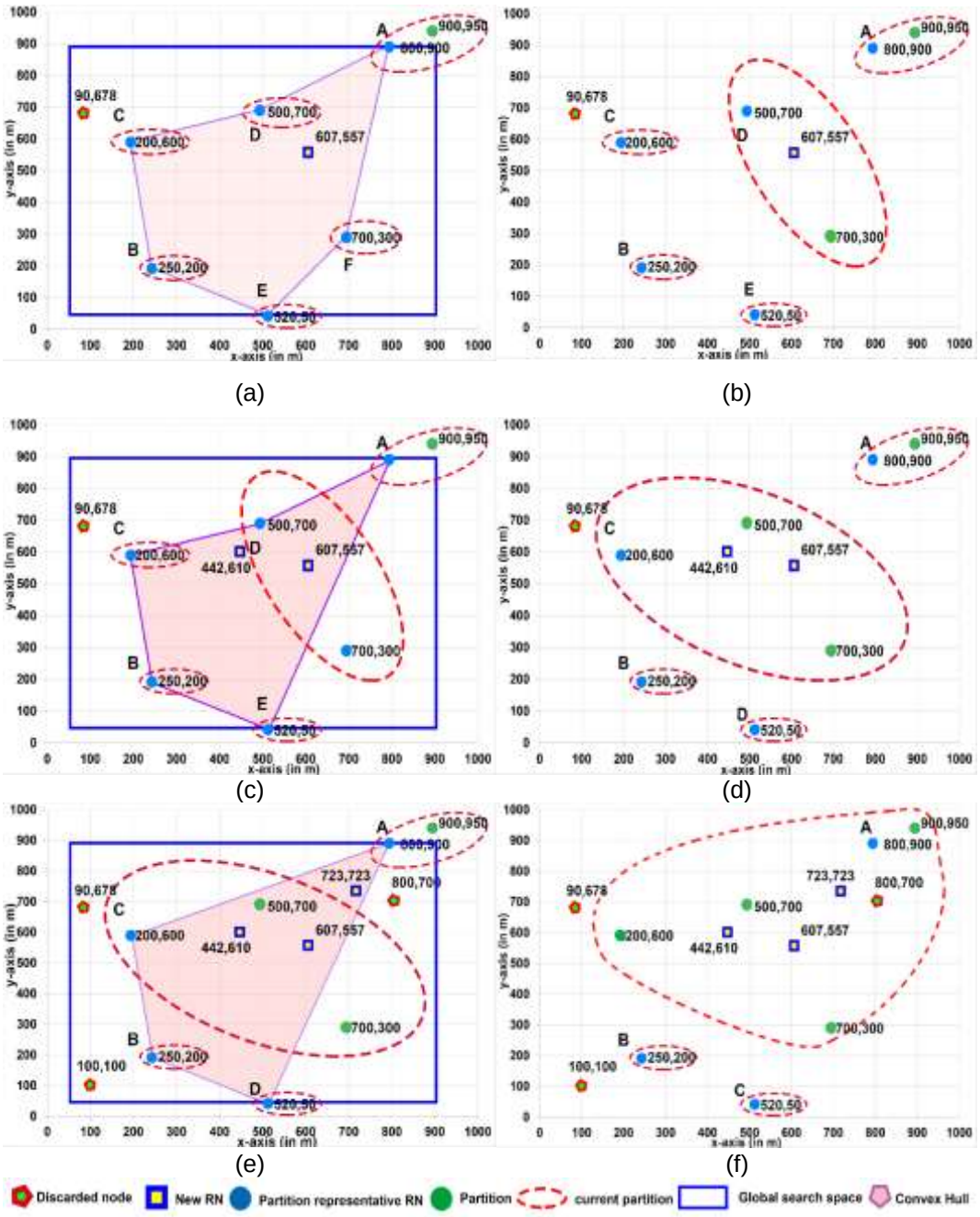

Figure 2. Network Scenarios (a-f)

Figure 2(d). Now, four partitions are left as A (800, 900), B (250, 200), C (200, 600), D (520, 50) for the next round. Figure 2(e) shows the next round here WO returns point (100,100) which is outside convex hull formed from current partition list hence discard it and go for next round. Here WO returns point (800, 700) which is also outside of convex hull; therefore, we discard it also and go for next round. In this round, WO returns point (723, 723) which is inside convex hull hence find its neighbouring partitions. Point (800, 900) of partition A and point (500, 700) of partition C are the neighbours of point (723, 723). Thus we consider them as one partition and point (800, 900) is the representative point of this partition as shown in Figure 2(f). Hence 3 partitions left as A (800, 900), B (250,200), C (520, 50) for next round. Figure 3(a) shows next round in this round WO returns a point (523, 523) which is inside the convex hull; therefore, we go for finding its

neighbours. Point (500, 700) of partition A is the neighbour of point (523, 523). Hence add point (523, 523) in partition A and go for next round as shown in Figure 3(b). In next round, WO returns point (296, 218) as this point is inside convex hull as shown in figure 3(c) we find its neighbouring partitions. Point (250, 200) of partition B and point (520, 50) of partition C are the neighbours of point (296, 218) as shown in Figure 3(d). Hence, consider them as one partition and now point (250, 200) will be the representative for this partition as shown in Figure 3(d). Hence two partitions left A (800, 900) and B (250, 200) which leads to the end of this phase and going towards termination phase.

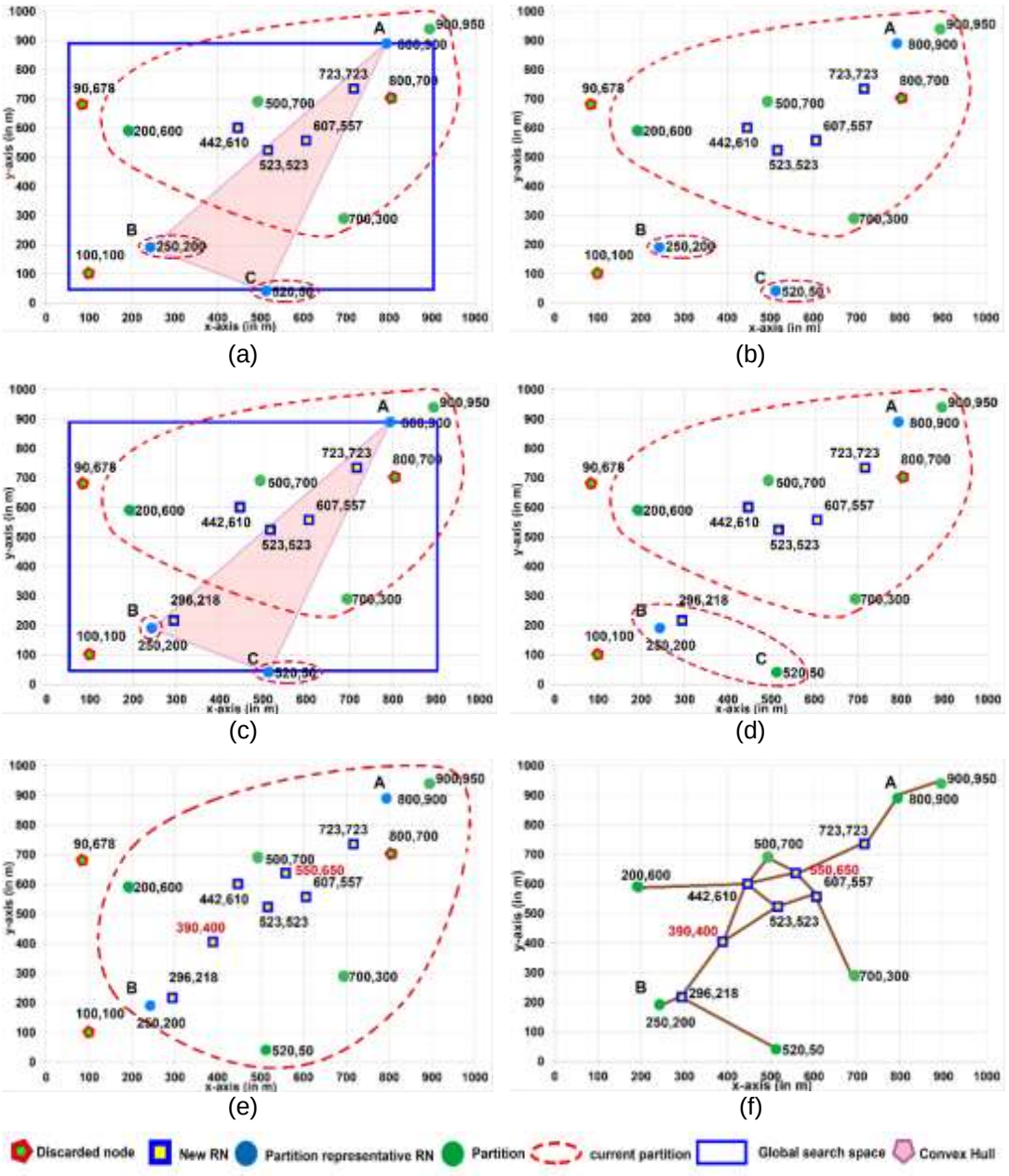

Figure 3. Network Scenarios (a-d)

6.4. Termination Phase

This is our last phase called termination phase. This phase starts when the number of partitions is less than or equal to 2. There are two cases while connecting the partitions in

the third phase. The first case when only one partition is left in that case proposed solution simply returns RN count and positions. While in the second case, when two partitions are left as explained in lines (31-33) of pseudo code, the proposed solution finds the required number of RNs to restore the lost connectivity between these two partitions by using the

Pseudocode of Our Proposed WORPH Solution

1. **Procedure** WORPH($Area, RN_r, N_{partition}$)
2. Initialize $Partition[N_{par}][2]$ array, $Partition\_list[N_{par}][2]$ list and $Count_{RN} \leftarrow 0$
3. **for** $i \leftarrow 0$ *to* N_{par} **do**
4. **for** $j \leftarrow i + 1$ *to* N_{par} **do**
5. **if** Euclidean_Distance($Partition[i], Partition[j]$) $< RN_r * 2$ **then** //**Algorithm 2**
6. Add $Partition\_list[j]$ in $Partition\_list[i]$ and remove it from $Partition$ array and $Partition\_list$ list
7. **end if**
8. **end for**
9. **end for**
10. **while** size of $Partition > 2$ **do**
11. Initialize $New\_RN[2]$
12. $PCH \leftarrow$ compute a convex hull of all coordinates in $Partition$ array
13. $New_{RN} =$ WO($Partition$)
14. **if** check New_RN lie inside convex hull *PCH* **then**
15. $Count_{RN}$ ++
16. Initialize $Partition\_in\_Range$ a 2D array and $C\_Partition\_in\_Range \leftarrow 0$
17. **for** $i \leftarrow 0$ *to* N_{par} **do**
18. **if** Euclidean_Distance($Partition[i], New_{RN}$) $< 2 * RN_r$ **then**
19. $Partition\_in\_Range \leftarrow Partition[i]$ and $C\_Partition\_in\_Range$ ++
20. **end if**
21. **end for**
22. **if** $C\_Partition\_in\_Range == 0$ **then** add New_RN as new partition in $Partition$ array and $Partition\_list$ list
23. **else if** $C\_Partition\_in\_Range == 1$ **then** add New_RN to respective Partition in $Partition$ array and $Partition\_list$ list
24. **else**
25. **for** $i \leftarrow 1$ to size of $Partition\_in\_Range$ **do**
26. Join all partition within range together in $Partition\_list$ and remove them from both $Partition$ array and $Partition\_list$ list instead of first
27. **end for**
28. **end if**
29. **end if**
30. **end while**
31. **if** size of *Partition* array is 2 **then**
32. $Count\_RN \leftarrow Count\_RN + (Euclidean\_Distance(Partition[0], Partition[1]) - 2 * RN_r)/(2 * RN_r)$
33. **end if**
34. **return** $Count\_RN$

end procedure

formula given in line 32 of pseudocode. Figure 3(e) shows this phase which returns two points (390, 400) and (550, 650) which connects point (800, 900) of partition A with point (250, 200) of partition B. Hence it returns the total RNs count ($Count\_RN$).

6.5. Generated Topology

Our proposed solution (WORPH) has the ability to generate well-organized and well-connected network topology. Our proposed solution deploys RNs inside the damaged area in an inward manner. This produces an efficient topology. Final produced topology is shown in Figure 3(f). By analysing the topology in Figure 3(f) we have concluded some features of it:

- *Connectivity* is one of the important factors for any network topology as it shows the strength of network when some fault came. If connectivity of the network is high then it can tolerate a large number of failures in the network. Figure 3(f) shows that resulting topology which has less connectivity at the edges but it becomes higher as we observe the internal nodes. Therefore whenever there is some failure in the network then it does not partition the network again due to high connectivity of the resultant topology. It shows that our proposed solution is also fault tolerant.
- *Energy Efficient:* we are deploying one RN for each partition and all the communication with SNs will be happening through this RN only so it will act as a gateway (cluster head) hence lead to increase the lifetime of the network. It exhibits that the resulting topology is energy efficient.
- *Distributed traffic load:* We have seen that centralized traffic load may result in continuous failure of the central point due to heavy load but not in case of distributed traffic load. Figure 3(f) shows a distributed traffic load between all nodes in the network. Therefore, it also helps in increasing overall network lifetime.

Algorithm 2: Euclidean Distance between two coordinates in X-Y plain

1. **procedure** Euclidean_Distance(P_1, P_2)
2. $a \leftarrow | P_1[0] - P_2[0] |^2$
3. $b \leftarrow | P_1[1] - P_2[1] |^2$
4. **return** $\sqrt{a+b}$
5. **end procedure**

7. Explanation of Our Proposed Solution (WORPH)

In this section, we explain the pseudo- code of WORPH algorithm which is shown in the pseudo code. This pseudo code can be sliced into four phases. Our proposed algorithm maintains a 2D array *Partition* [ ][ ] for the coordinates of representing point of particular partition, a 2D list $Partition\_list$ [ ][ ] which is used to handle all the XY coordinate values of points which are added to join the partition and $Count_{RN}$ to hold total number of RNs required to restore lost connectivity. The integer variable N_{par} shows the number of disconnected partitions in the network. Lines (3-9) show the neighbour discovery phase of our proposed algorithm. If Par_i and Par_j are in range according to line 5 then they will be joined by using line 6. Lines (10-30) show the iterative process for deploying RNs by calling WO for each iteration. In this iterative process, line 12 computes convex hull by using quick hull algorithm and the points inside $Partition$ array are taken as input for convex hull. Line 13 returns the optimal position of *New_RN* by calling WO algorithm. Our proposed algorithm checks whether *New_RN* lies inside convex hull or not in line 14. If yes then it will be added else it will be discarded. Lines (17-21) find all the neighbouring partitions of *New_RN*. Line 22 increase the number of partitions by adding *New_RN* as new partition in both *partition* array and $Partition\_list$ list if no neighbour is found. Line 23 adds *New_RN* to respective neighbour found before in lines (17-21). When *New_RN* has more than one neighbour means it is joining two or more partition is explained in lines (24-29). This ends the third phase and began the final phase which is

the termination phase and in this proposed solution returns RNs count. In this lines (31-33) find the total number of RNs needed to join last two partitions by using the formula given in line 32.

8. Performance Evaluation and Comparison

Table 1. Simulation Parameter

Parameter	Value
Area	1000m x 1000m
Nodes	100-500
Total number of partition	5-9
Communication range of RNs	50m-325m
Number of deployed RNs	4-73

In this section, the performance and effectiveness of our proposed solution is discussed. Also, we have given the comparison with state-of-the-art algorithms. Our proposed solution is implemented in python language. By observing the experimental results, it is seen that minimum RN count increases as a number of partitions increases in our proposed solution. Here, our aim is to observe a minimum number of RNs count. Table 1 shows the parameters used in simulation to observe the performance analysis of our proposed WORPH algorithm.

8.1. Performance Metrics

For the experimental purpose, we have considered a variable range of RNs with fixed number of partitions and fixed range of RNs while the number of partitions is varying. The following listed metrics are used to validate the performance of a proposed algorithm.

- *Number of Partitions* (N_{par}): When the number of partitions (N_{par}) increases inside the network it leads to deploy more RNs for connecting them. As discussed in the previous section that number of RNs is prepositional to the number of segments. Hence performance matric is directly impacted by minimum RN counts.
- *Communication range of RN*(RN_r): As RN communication range increases it will be able to cover a large area. Hence we require less number of RNs to cover the same area which can be seen in our experimental results. As we increase the range of RNs then count to connect the same number of partitions decreases. Hence better results are achieved.
- *Total number of RNs*($Count\_RN$): RNPP in HWSN mainly focuses on this parameter. We can observe from the results that our proposed solution gives better results in comparison with other algorithms.

8.2. Comparison of State-of-the-art Approaches

In this section we have shown some recent published and well-known algorithms in this research domain. In view of the above metric, our aim is to solve RNPP and propose an efficient solution with that. In this section, we have shown a brief introduction of five other RNPP algorithms. The first algorithm GAIN [4] uses meta-heuristics approach (the grey wolf optimization) for finding the SP to deploy RNs for joining partitions and author is using convex hull approach to deploy RNs inside damaged area. The second algorithm produces convex hull and computes SP for every 3 neighbouring nodes repetitively until only two partitions are left (ORC [24]). The third algorithm uses SMT with a minimum number of SPs to discovery the positions of RNs to be placed such that they may lead to

restoring global network connectivity in WSN (STP-MST [25]). The fourth algorithm creates MST based on single-tiered RNP (MST-1tRN [26]). Finally, the last algorithm studies the problem of connected single cover in which each SN is covered by single RN (1CSCP [27]).

8.3. Simulation Results and Discussion

In this section, we have shown the performance of our proposed approach through experimental results. Our proposed algorithm is simulated in python with multiple configurations. Each configuration has a different number of partitions N_{par} and varying communication range of RNs(RN_r). All SNs are randomly deployed in the area of interest (*i.e.*, 1000m x 1000m). The communication range of RNs varies from 50m to 325m and value of N_{par} have been taken as 5, 6, 7 and 8. All experiments results are taken into attention with an average of 30 individual results.

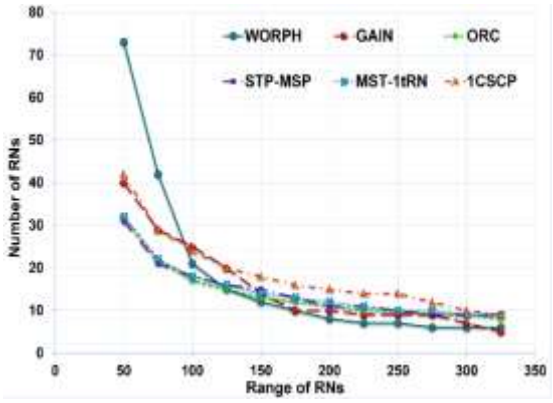

Figure 4. Comparison of WORPH with Other Algorithms and $N_{par} = 5$

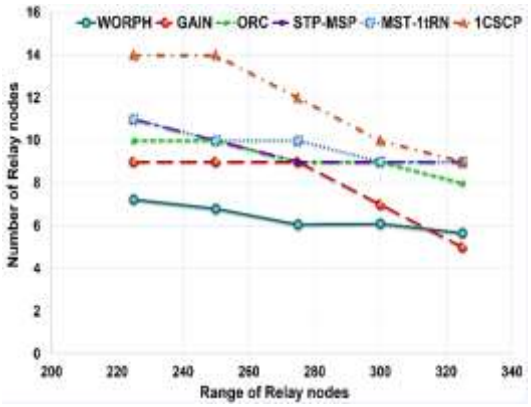

Figure 5. Magnified Version of Figure 4 and $N_{par} = 5$

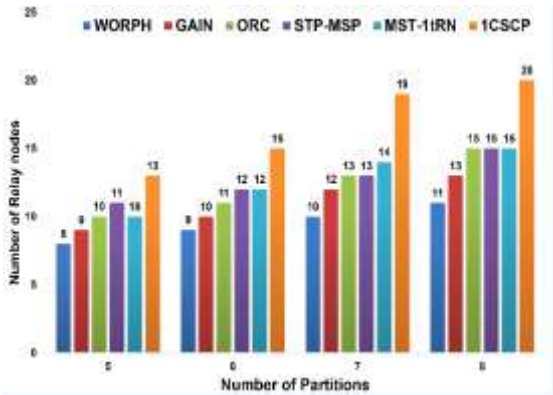

Figure 6. Comparison of WORPH with Other Algorithms and $RN_r = 200$

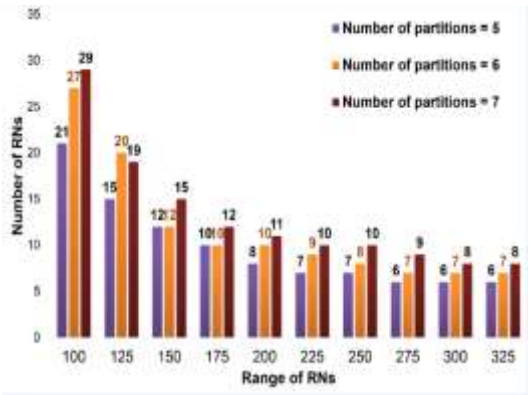

Figure 7. Comparing Results of WORPH for Varying Number of Partitions

8.3.1. Number of RNs Deployed

In Figure 4 the performance of WORPH is compared with other five comparative algorithms in terms of total number of RNs deployed to restore the partitioned WSN connectivity. For all the results shown in Figure 4 number of partitions are fixed as $N_{par} = 5$ and communication range of RNs is varying from 50m to 325m in intervals of 25m. By observing the results we can say that RNs count decrease as the range of RNs

(RN_r) increase. Also, we can see that number of RNs increase exponentially with the smaller value of RN_r. Hence it proves that communication range and number of RNs is inversely proportional to each other. The performance of WO between ranges 50-100m is poor as compared to others algorithms. In this range, SP based algorithms are giving good results as they consider less number of nodes to find the optimal location of RN. While GAIN and WO considers all nodes as search agents. Also GAIN is better in this range than WO reason behind this, GAIN is using more iteration for finding optimal location while WO is only using 5 iterations for this which make it fast experimental results also shows that better results can be obtained if we increase the number of iterations for WO which leads an increase in execution time. But WO shows better results than any other algorithm for ranges between 100-325. In real time application range of relay node is taken as 200m or greater than 200m. Figure 5 is the magnified version of Figure 4 ranges from 225 to 325 here we can easily observe that WO is performing better than all other algorithms. In Figure 6 we have compared WORPH with other five algorithms by keeping the range of RN constant ($RN_r = 200$) while varying number of partitions as 5, 6, 7, 8.

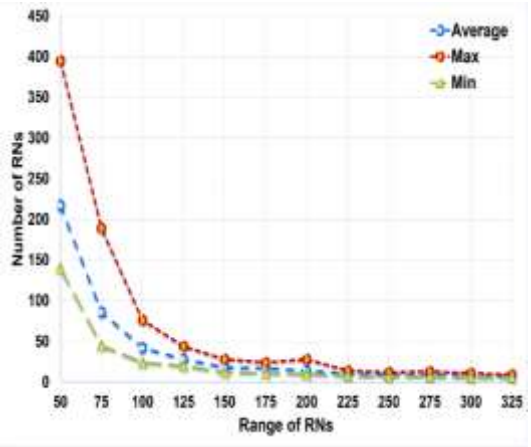

Figure 8. Number of RNs vs. Range of RNs

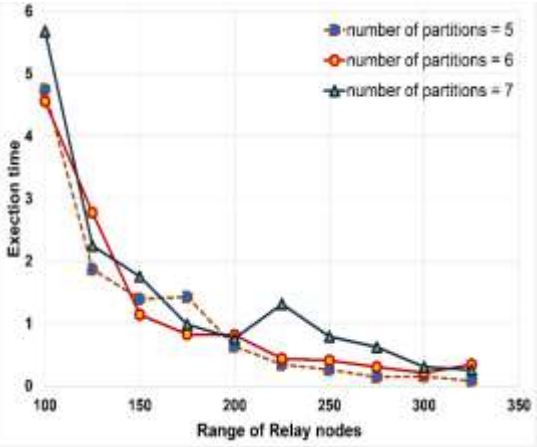

Figure 9. Execution Time vs. Range of RNs with Varying N_{par}

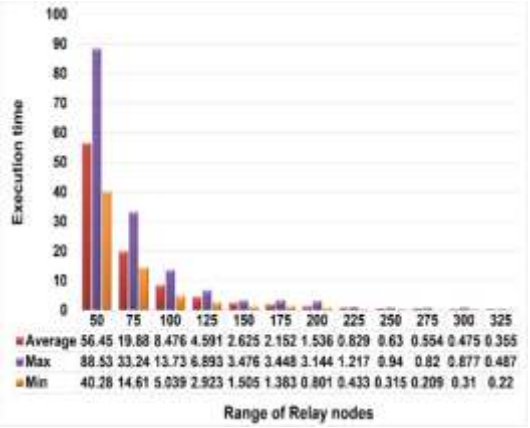

Figure 10. Execution Time vs. Range of RNs

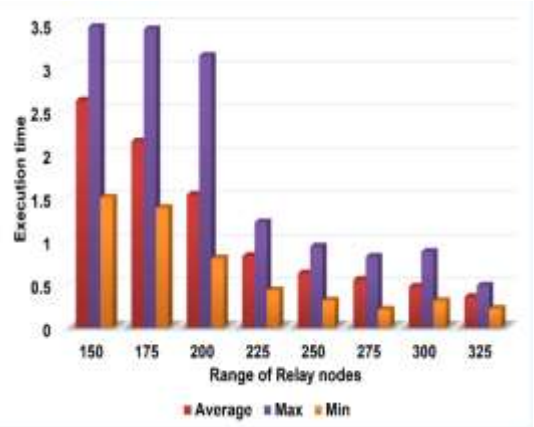

Figure 11. Magnified Version of Figure 10 and $N_{par} = 5$

By observing the results we can see that 1CSCF is showing worst results in all cases. All the SMT based algorithms are showing similar results in terms of RN count. We observe GAIN is better than all of these previously discussed algorithms but WO has performed better by giving the least RN count as compared to all five algorithms. In

Figure 7 we have compared WORPH with varying range of RNs as well as the number of partitions. As we have discussed previously that the number of RNs increases with increase in the number of partitions. This can be observed from Figure 7 less number of partitions results less in number of RNs and for ranges from 175-325 results are similar. In Figure 8 we have considered the worst case for our algorithm. In this, we have taken 5 partitions far away from each other due to which finding optimal result become difficult here we are varying the range of relay node. Figure 8 shows us average, maximum and minimum RN count given by WO on testing it 30 times.

8.3.2. Execution Time

In Figure 9, it is shown that execution time of our proposed solution when range and number of partitions both are varied. By observing this graph, we can say that as range of RN decreases the execution time increases, because it requires additional number of RNs to be placed to restore the lost connectivity of the partitioned network. Also as the number of partitions increases, execution time increases because WORPH has considered additional number of partitions in the initial phase to find RN position (or SP for deployment of RN). Figure 10 shows the average, minimum and maximum time of execution for the Figure 8. We observe that it requires more time to find optimal location of RNs to restore lost connectivity when the range of RNs decreases from 125 to 75. Also, our proposed solution gives similar results between ranges 225-325 and 150-200. Figure 11 is a magnified view of Figure 10 between ranges 150-325.

9. Conclusion and Future Scope

Generally, WSNs are deployed in a harsh and unfriendly environment where normal person or equipment can't reach. In that environment, SNs are more prone to failure (means their failure rates are high). When a large number of SNs fail it may lead to the creation of multiple partitions inside the well-established network. Disconnected partitions are unable to communicate with each other or base Station. Hence, we proposed whale optimizer solution (WORPH) to heal the partitioned network. Our proposed solution has used wolf optimizer which executed in rounds and each round it has returned a position to deploy *New_RN*. Each round is maintained its own convex hull for coordinates/positions of the current set of partitions. If *New_RN* lies inside convex hull, than it is added in the list, otherwise it is discarded. Our proposed solution terminated when a number of partitions become less than three. In termination phase, RN's location is observed using Euclidean equality. We have compared the performance of WORPH with other well-known RNP algorithms. The simulation results proved that our proposed algorithm performed better than other proposed solutions and deployed a minimum number of relay nodes for large transmission range of RNs. It is also shown that generated topology has good connectivity with balance traffic load resulting in a fault tolerant and an energy efficient network is obtained. In future, we are planning to extend our work to provide 2-connectivity while further decreasing the execution time and relay nodes count in the deployment area. With this, our proposed solution will also provide a better load distribution and greater connectivity due to the availability of multiple different paths for communication. Further, we will try to reduce the global search space for WO which leads to decrease in execution time and less number of discarded RNs.

References

[1] V. Ranga, M. Dave and A. K. Verma, "Network partitioning recovery mechanisms in WSANs: a survey", Wireless personal communications, vol. 72, no. 2, **(2013)**, pp. 857-917.
[2] V. Ranga, M. Dave and A. K. Verma, "Relay Node Placement for Lost Connectivity Restoration in Partitioned Wireless Sensor Networks", Proceedings of International Conference on Electronics and Communication Systems (ECS 2015), **(2015)**, pp. 170-175.

[3] E. L. Lloyd and G. Xue, "Relay node placement in wireless sensor networks", IEEE Transactions on Computers, vol. 56, no. 1, **(2007)**, pp. 134-138.
[4] G. Kumar and V. Ranga, "Healing Partitioned Wireless Sensor Networks", International Conference on Ubiquitous Computing and Ambient Intelligence, Springer, Cham, Switzerland, **(2017)**, pp. 545-557.
[5] P. Sharma and V. Ranga, "Particle Swarm Optimization for Disconnected Wireless Sensor Networks", Computing and Network Sustainability, Springer, Singapore, **(2017**), pp. 413-421.
[6] 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", Applied Soft Computing, vol. 30, **(2015)**, pp. 675-687.
[7] K. R. Kamal, "Ant Colony Optimization for Jointly Solving Relay Node Placement and Trajectory Calculation in Hierarchical Wireless Sensor Networks", **(2014**).
[8] S. Mirjalili and A. Lewis, "The whale optimization algorithm", Advances in Engineering Software, vol. 95, **(2016)**, pp. 51-67.
[9] Y. H. Xu, W. G. Jiao, Y. Wu and J. Song, "Variable-dimension swarm meta-heuristic for the optimal placement of relay nodes in wireless sensor networks", International Journal of Distributed Sensor Networks, 1550147717700895, vol. 13, no. 3, **(2017)**.
[10] M. Azharuddin and P. K. Jana, "A GA-based approach for fault tolerant relay node placement in wireless sensor networks", Computer, Communication, Control and Information Technology (C3IT), 2015 Third International Conference on IEEE, **(2015)** February 1-6.
[11] H. A. Hashim, B. O. Ayinde and M. A. Abido, "Optimal placement of relay nodes in wireless sensor network using artificial bee colony algorithm", Journal of Network and Computer Applications, vol. 64, **(2016)**, pp. 239-248.
[12] W. Lalouani, M. Younis and N. Badache, "Optimized repair of a partitioned network topology", Computer Networks, vol. 128, **(2017)**, pp. 63-77.
[13] B. Yuan, H. Chen and X. Yao, "Optimal relay placement for lifetime maximization in wireless underground sensor networks", Information Sciences, vol. 418, **(2017)**, pp. 463-479.
[14] C. Ma, W. Liang and M. Zheng, "PSH: A Pruning and Substitution Based Heuristic Algorithm for Relay Node Placement in Two-Tiered Wireless Sensor Networks", Wireless Personal Communications, vol. 94, no. 3, **(2017)**, pp. 1491-1510.
[15] A. Bhattacharya, A. Rao, K. P. Naveen, P. P. Nishanth, S. V. R. Anand and A. Kumar, "QoS constrained optimal sink and relay placement in planned wireless sensor networks", Signal Processing and Communications (SPCOM), 2014 International Conference on IEEE, **(2014)** July 1-5.
[16] S. Kimençe and I. Bekmezci, "Weighted relay node placement for wireless sensor network connectivity", Wireless networks, vol. 20, no. 4, **(2014)**, pp. 553-562.
[17] K. Nitesh and P. K. Jana, "Relay Node Placement with Assured Coverage and Connectivity: A Jarvis March Approach", Wireless Personal Communications, vol. 98, no. 1, **(2018)**, pp. 1361-1381.
[18] H. Y. Chang, Y. H. Huang and T. L. Lin, "A novel relay placement algorithm based on puzzle games for indoor wireless sensor networks", Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), 2014 Tenth International Conference on IEEE, **(2014)**, pp. 682-685.
[19] I. F. Senturk, K. Akkaya and S. Yilmaz, "Relay placement for restoring connectivity in partitioned wireless sensor networks under limited information", Ad Hoc Networks, vol. 13, **(2014)**, pp. 487-503.
[20] X. Wang, L. Xu and S. Zhou, "A Straight Skeleton Based Connectivity Restoration Strategy in the Presence of Obstacles for WSNs", Sensors, vol. 17, no. 10, **(2017)**, pp. 2299.
[21] C. Ma, W. Liang and M. Zheng, "Delay constrained relay node placement in two-tiered wireless sensor networks: A set-covering-based algorithm", Journal of Network and Computer Applications, vol. 93, **(2017)**, pp. 76-90.
[22] Z. Gao, K. Chen, W. Cheng, Y. Hao and X. Li, "K-extended constrain independent relay node placement with base stations in two-tiered wireless sensor network", Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), 2014 Tenth International Conference on IEEE, **(2014)**, pp. 823-826.
[23] N. A. Alrajeh, M. Bashir and B. Shams, "Localization techniques in wireless sensor networks", International Journal of Distributed Sensor Networks, vol. 9, no.6, **(2013)**, pp. 304-628.
[24] S. Lee and M. Younis, "Optimized relay node placement for connecting disjoint wireless sensor networks", Computer Networks, vol. 56, no. 12, **(2012)**, pp. 2788-2804.
[25] X. Cheng, D. Z. Du, L. Wang and B. Xu, "Relay sensor placement in wireless sensor networks", Wireless Networks, vol. 14, no. 3, **(2008)**, pp. 347-355.
[26] E. L. Lloyd and G. Xue, "Relay node placement in wireless sensor networks", IEEE Transactions on Computers, vol. 56, no. 1, **(2007)**, pp. 134-138.
[27] D. Yang, S. Misra, X. Fang, G. Xue and J. Zhang, "Two-tiered constrained relay node placement in wireless sensor networks: Efficient approximations", Sensor Mesh and Ad Hoc Communications and Networks (SECON), 2010 7th Annual IEEE Communications Society Conference, **(2010)** June 1-9.

