# Relay Placement Algorithms for Communication in a Heterogeneous Propagation Environment

Nihesh Rathod$^{\alpha,\beta}$, and Rajesh Sundaresan$^{\alpha,\beta}$
$^{\alpha}$Department of Electrical Communication Engineering,
$^{\beta}$Robert Bosch Centre for Cyber-Physical Systems,
Indian Institute of Science, Bangalore, India. 560012

***Abstract*—A vast majority of the IoT devices will be connected in a topology where edge-devices push data to a local gateway which in turn forward the data to the cloud for analytics and archiving. In large networks, many edge-nodes will be located far away from their nearest gateways. Due to their locations and limited transmission power, these edge-nodes might experience poor network coverage. To provide reliable connectivity to these edge-nodes, relays/repeaters may have to be placed at a few locations until a gateway is reached. Network deployment engineers often do this based on field visits, surveys, measurements, initial deployments, followed by fine-tuning. For the scalability of such deployments, automated network planning tools are essential. Such tools should be able to predict coverage based on the edge-node locations using perhaps GIS data, identify the need for relays/repeater and, if needed, suggest the number and locations of relays, in order to provide reliable connectivity. This entire process should be as fast and automated as possible for rapid network deployment. In this paper, using a black box RSSI estimator between any candidate pair of transceiver locations, we propose a network deployment framework that uses either Ant Colony Optimisation (ACO) or Differential Evolution (DE) to identify the number and location of relays for meeting specified quality of service constraints. We also show how to handle multiple gateways.**

***Index Terms*—GIS, Heterogeneous, Internet of Things, RF propagation tool, RSSI, Sub-GHz.**

## I. Introduction

Automation of network deployment is a necessity to enable IoT expansion at the currently projected scales. Instead of relying on surveying for getting an estimation of network coverage, prior knowledge of the terrain can lead to better coverage prediction at an early stage of deployment. The ability to predict coverage without actual deployment saves valuable engineering resources and can lead to rapid network deployments. This paper is an embodiment of this idea and showcases many interesting problems towards its implementation. In this paper, we will use the Indian Institute of Science campus (IISc campus) in Bangalore, India, as a testbed to explain our ideas, algorithms, and results. One reason is that, despite its small size (roughly 2km-by-2km), it already offers a very diverse propagation environment that is representative of a large city. Given various data sources, i.e., edge devices in an IoT network, located on a map and a destination (gateway to the cloud) also located on a map, given a Geographical Information System (GIS), the problem is to identify locations of relays to get data from the IoT edge devices to the gateway, meeting specific Quality of Service constraints. This problem can be divided into two subproblems.

- The first subproblem is to identify the link quality between any pair of points, a potential transmitter-receiver pair. Outdoor environments are typically heterogeneous with carrier pathways traversing different propagation environments (with different path loss exponents, fading parameters, shadowing parameters, etc.).
- The second subproblem is to use the solution of the first subproblem (as a black-box) and focus on network deployment. This includes the determination of the required number of relays, their locations, powers, etc., for meeting specific QoS requirements.

The focus of this paper is the second subproblem.

In earlier work, Rathod et al. [1] provided a data-driven coverage tool that could handle heterogeneity in propagation environments. Extensive measurements in example environments provided a library of propagation models. They then used GIS data processing on the region to be deployed to identify locally homogeneous regions and map each to a library environment. They then used this to predict coverage between a potential transmitter and a receiver in the region under consideration. We use their tool as a black-box to find the Received Signal Strength Indicator (RSSI) between any two points on the map.

In this paper, we formulate and address the relay placement problem. Specifically, given a number of IoT edge node locations, a gateway location and a minimum RSSI threshold $R$, find the minimum number of relays and their location so that there is a spanning tree all of those links have RSSI exceeding $R$. This relay placement problem to meet QoS requirements is likely to be a hard problem because it is a version of a Steiner tree problem [2]. Moreover, **it has a complicated geometric structure for link costs between a potential pair of nodes, with the structure coming from the packet error rate for wireless transmission in a *heterogeneous* environment**. In particular, **the geometric structure is not associated with the Euclidean distance**. We propose two heuristics to solve this problem: an Ant Colony Optimisation (ACO) algorithm and a popular genetic algorithm called Differential Evolution (DE). Both algorithms need to be adapted to handle our objective. Besides the complicated geometric structure of link costs (coming from the heterogeneous propagation environment)

ISBN 978-3-903176-18-8 © 2019 IFIP

does not seem to be easily exploitable. We also study a variant of the above problem where there may be multiple gateways and the goal is for each IoT edge node to connect to one of the gateways. Again we adopt the ACO and DE algorithm to handle this modified objective.

The rest of the paper is organised as follows. In Section II we briefly introduce the relay placement problem, provide a short introduction to the Ant Colony Optimisation (ACO) algorithm, explain how ACO is adapted for finding acceptable solutions to the well known NP-hard Steiner tree problem [2], and show some example outcomes. We then formally define the relay placement problem in the heterogeneous region with one gateway and show how ACO can be adapted to solve it. We end this section by showing the results of our adapted ACO algorithm on the IISc campus. Section III deals with Differential Evolution (DE) algorithm and has roughly the same structure as Section II. In Section IV, we address the relay placement problem with multiple gateways. We also show the effectiveness of modified ACO and DE algorithms along with a divide-and-conquer strategy towards solving the multiple gateway problem. We end the paper with some concluding remarks and indications of ongoing works. Unlike earlier relay placement works (e.g., [3], [4]) the main feature of this work is the capability of handling heterogeneous environments in conjunction with the coverage tool of [1] and thus making it ineffective to compare it with the state of the art results for a homogeneous propagation region [5].

## II. Ant Colony Optimisation for relay placement with single gateway

Suppose that we are given a particular deployment scenario. This includes the GIS data for that region, gateway location, and IoT edge device locations. The RSSI computing algorithm in [1] predicts the average RSSI between any pair of nodes. If each end node has connectivity to the gateway either via a direct link (RSSI $> R$) or via other edge nodes as repeaters (RSSI $> R$ in every link in the path), no extra relay nodes are needed and we can go ahead to deploy the network.

Relays are needed if one or more nodes are unable to reach the gateway. We would like to minimise the number of these new relays because they result in extra hardware costs, more maintenance costs, etc. This problem of minimising the number of relays is related to the well-known Steiner tree problem. For a homogeneous propagation environment, this is simply the Euclidean Steiner tree problem (STP) which is defined as follows.

Given $n$ points in the plane $\{x_1, x_2, \ldots, x_n\}$, connect all the points by line segments of minimum total length in such a way that any two points in the set may be interconnected by line segments either directly, or via other points in the set or via other new points. For a general $n$, STP is an NP-hard problem [2]. So, instead of trying to solve STP optimally, we now describe an ant colony optimisation based meta-heuristic algorithm to find a good solution in a reasonable time with very practical computational resources.

### A. Ant Colony Optimisation (ACO)

Some species of ants start their search for food by wandering in random directions. When an ant finds a food source, it leaves a substance called “pheromone” forming a trail while returning to the ant colony. When other ants come across this pheromone trail, they follow the trail to the food source instead of wandering in random directions. If they too find the food source, they reinforce the pheromone trails while returning to the ant colony.

Pheromone trails have an interesting property of evaporation with time, which decreases the attractiveness of a particular path to a food source with time. Amongst multiple possible paths to the food source, the more the time required for an ant to travel on a particular path and back again, the more the evaporation of pheromone on that path. On the other hand, a shorter path will be travelled more frequently and thus the pheromone density on the shorter paths becomes stronger than on the longer paths. Pheromone evaporation also serves as a way to avoid paths that are only locally optimal. If pheromone will not evaporate with time then all ants will follow the path taken by the first ant to find the food source, thereby constraining the discovery of alternate shorter and better paths than the first one.

The end result is that when an ant finds a good short path from the colony to the food source, other ants are likely to follow the path and positive feedback will eventually allow ants to reinforce this good short path from the food source to the colony.

Ant colony optimisation (ACO) algorithm is a meta-heuristic algorithm based on this behaviour of the ants. ACO is known to produce acceptable solutions in a reasonable time for some instances of NP-hard problems like the famous Travelling Salesman Problem. ACO also has the advantage of dynamically adapting to changes in the graph structure. ACO can be run continuously to adapt to changes in real time. Due to the above-mentioned properties, ACO is an attractive choice for solving our problem. For the remainder of this section, we describe how we have adapted ACO for first solving the Euclidean STP problem and then for solving the relay placement problem in a heterogeneous environment.

### B. ACO for solving Euclidean Steiner tree problem

Readers familiar with the ACO for Euclidean TSP may skip this section. The iterative ACO algorithm involves many variables over which the objective function is minimised or maximised. It also needs the range of each of these variables. Further, it requires a way to evaluate the value of the objective function given the variables. Apart from these, ACO also has a few algorithmic parameters like the number of ants, the maximum number of iterations allowed, error values, etc. We will not discuss the effect of these parameters on the output of the algorithm as we have kept them fixed throughout our experiment.

We first create a function needed to assess the benefit of one extra node ($k = 1$) at location $(d_{ix}, d_{iy})$, as a function of this location. This function forms a minimum spanning tree (MST) on the given vertices $(d_{ix}, d_{iy}) \cup \{x_1, x_2, \ldots, x_n\}$

by considering the Euclidean distance between two points as edge weight. Then we add all the edge weights of this MST and call the total weight $w$. We treat this $w$ as the function evaluation at the point $(d_{ix}, d_{iy})$. We pass this function and a random initial point to ACO and ask ACO to minimise this $w$ across $(d_{ix}, d_{iy})$. After completing its iterations, ACO will output a possibly better location $(d_{ix}^{'}, d_{iy}^{'})$. In doing so, either convergence to within the tolerance $\epsilon$ is attained or we crossed the maximum number of iterations. We then restart the algorithm but this time with two ($k = 2$) extra nodes placed at random. We continue to increase number of extra nodes $k$ until we reach $n-2$. We have chosen the number $n-2$ as an upper limit on the extra nodes for Euclidean STP because it is known that the maximum number of extra nodes, also known as Steiner nodes, is $n-2$ where $n$ is the number of points [6]. We then pick the best $k$ with the lowest MST weight. This approach may need multiple random restarts for each $k$ since the iterative algorithm may settle down at a local minimum.

Figure 1 shows the results of this approach to solve STP for edge devices forming regular polygons. The $n$ points were kept at vertices of different polygons. The choice of regular polygons allows us to validate the solutions with those in the literature [6].

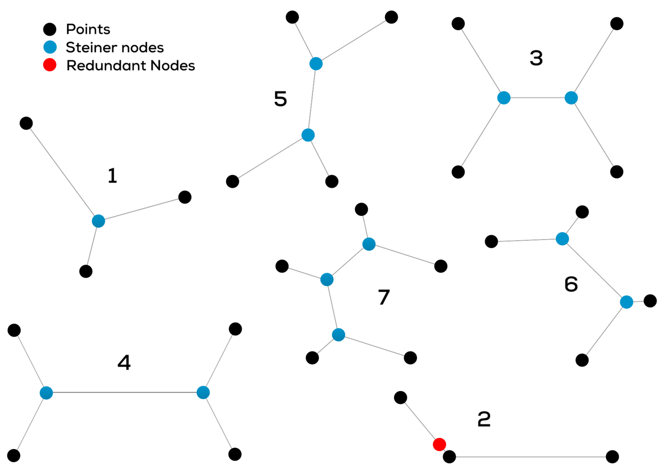

Fig. 1: Solutions to Steiner Tree Problems for regular polygons. Case numbers printed next to the solutions are referenced in the discussion.

As shown in Figure 1, for a triangle with all the angles $< 120°$ ($n = 3$, case 1), the ACO algorithm solves the STP with $k = n - 2 = 1$ Steiner node. Further, the location of the Steiner node is at the centroid of the triangle, which is intuitive. For a square ($n = 4$, case 3), a rectangle ($n = 4$, case 4), a parallelogram ($n = 4$, case 5) and a trapezoid ($n = 4$, case 6), the ACO algorithm matches with the best possible solution described in the literature [6]. For $n = 4$, case 6, not only does it stop at $k = 2$, but it also gives the correct locations of the Steiner nodes. For a pentagon ($n = 5$, case 7), the solution was obtained with $k = 3$. The solution to Case 2, a triangle with one of the angles $> 120°$ ($n = 3$), has a very interesting interpretation. The algorithm always outputs a Steiner node on the line connecting the triangle corners, which is marked with the red circle as a redundant node. It means that the solution to STP is to connect three points directly. There is no other better way to form MST in this case. Interestingly, we did not configure our algorithm to treat the triangles in cases 1 and 2 differently based on the angles. The ACO algorithm identified these correct solutions without special considerations. This highlights the robustness of the algorithm for different configurations of $n$ points. Note that the objective function involves Euclidean distances with exploitable properties such as the triangle inequality. In the next subsection, we will show our approach to solve the relay placement problem in a heterogeneous region by adopting this algorithm.

### *C. ACO for solving the relay placement problem in a heterogeneous region*

While solving the STP, we used the Euclidean distance for constructing the minimum spanning tree and for finding the minimum edge weight $w$. For using the same ACO algorithm in a similar way in our problem of relay placement in a heterogeneous region, we must adopt this objective function on which the ACO algorithm operates. Our adaptation is as follows. Note that the objective function operates on a set of points (transmitters, relays, and one destination). First, the heterogeneity of the propagation environment must be captured in the link quality. Rathod et. al in [1] have already shown how to estimate the RSSI between any pair of points on the map in a heterogeneous environment. We, therefore, use their RSSI computing engine as a black-box. Next, we use the estimated RSSIs between any pair of points to come up with link costs. Higher the RSSI, lower the cost, so the link cost should be a monotone decreasing function of the RSSI. We took $f(\text{RSSI}) = -\text{RSSI}$ merely for simplicity. Third, we computed the MST with these as link costs. Finally, the value output by the function is the minimum RSSI among the links on the obtained MST.

**Problem:** Given $n$ number of transmitter locations $\{x_1, x_2, \ldots, x_n\}$ and an aggregating location $\{y\}$ in a heterogeneous region, an RSSI threshold $R$, find the minimum number of relays $k$ and their locations $\{d_1, d_2, \ldots, d_k\}$ so that each edge $e$ of the resulting Minimum Spanning Tree on $S$ = $\{y\} \cup \{x_1, x_2, \ldots, x_n\} \cup \{d_1, d_2, \ldots, d_k\}$, with link costs coming from the function $f$ that takes link RSSIs to link costs, has RSSI $\geq R$ on each link:

$$\min_{\substack{(i,j)\in E(MST(S))\\ i\neq j}} \text{RSSI}(i,j) \geq R. \tag{1}$$

Here $E(MST(S))$ is the edge set of the graph $MST(S)$.

Some remarks on the objective function are in order. First, we try to ensure that the minimum RSSI is at least as large as the target RSSI of $R$. If not, as we will see below, we will increase the number of relays and continue our search. Second, in order to find the MST, we have attributed a cost $f(\text{RSSI}(i,j))$ for the link $(i,j)$. While we have used $f(r) = -r$, other link cost functions of RSSI can be easily handled. Third, we have given equal weight to every link of a candidate tree in arriving at the MST. It is easy to adapt this to a search for a minimum weight spanning tree, as would arise

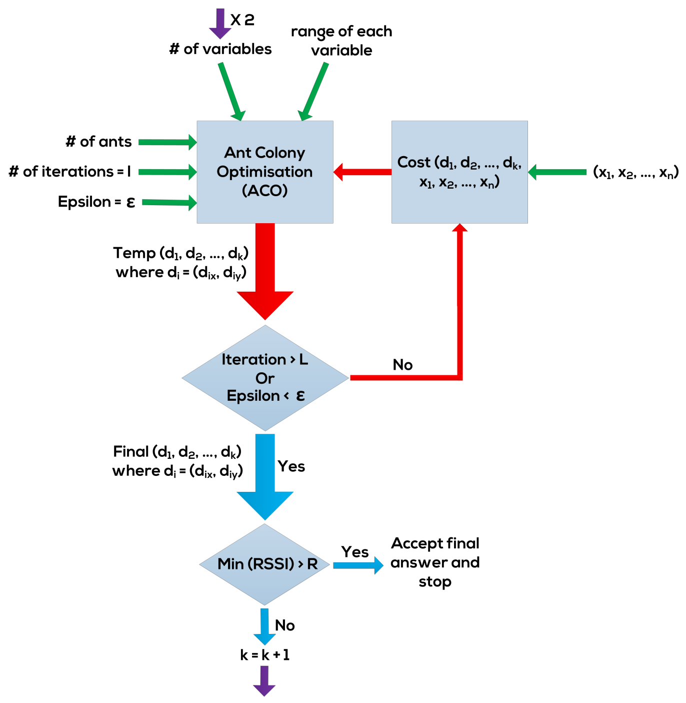
X 2
# of variables
range of each variable
# of ants
# of iterations = I
Epsilon = ε
Ant Colony Optimisation (ACO)
Cost (d₁, d₂, ..., dₖ, x₁, x₂, ..., xₙ)
(x₁, x₂, ..., xₙ)
Temp (d₁, d₂, ..., dₖ) where dᵢ = (dᵢₓ, dᵢᵧ)
Iteration > L Or Epsilon < ε
No
Final (d₁, d₂, ..., dₖ) where dᵢ = (dᵢₓ, dᵢᵧ)
Yes
Min (RSSI) > R
Yes
Accept final answer and stop
No
k = k + 1

Fig. 2: Flow diagram of the algorithm.

if every edge node generated traffic at a uniform rate and the links closer to gateway carried more traffic.

Figure 2 shows the flow diagram of our approach. As mentioned in section II-B, ACO is an iterative algorithm. This loop is highlighted in red color in Figure 2. The function indicated "Cost($\cdots$)" constructs the Minimum Spanning Tree on $\{y\} \cup \{x_1, x_2, \ldots, x_n\} \cup \{d_1, d_2, \ldots, d_k\}$ in each iteration and outputs the value of the minimum RSSI (across links) on the MST. ACO tries to maximise this value. When the algorithm converges or the number of iterations is exhausted, we perform an additional test. We check each link for the minimum required RSSI $R$. If all the links pass this minimum RSSI threshold check then we accept the solution. Otherwise, we increase the number of allowed relays by one and restart the algorithm.

We stress-tested our algorithm over many adversarial instances. As an example, in Figure 3, we kept our transmitters and the gateway at the corners of the image, shown as +s, in Figure 3. The transmitters and gateway were kept so far apart that direct communication between any pair was impossible. Then we let our algorithm suggest the relay locations, after taking heterogeneity in the account. Figure 3 shows two results of this experiment. Suggested relay locations are marked with red circles. In both results, the number of relays required to connect the network is six. The locations suggested are different due to convergence to local minima.

## III. Differential Evolution for Relay Placement with Single Gateway

In this section, we highlight another bio-inspired heuristic algorithm called Differential Evolution (DE) for solving the relay placement problem explained in Section II. Differential Evolution is a stochastic, parallel, direct search global optimisation method. It is quite robust and is often fast. DE tries to mimic the Darwinian theory of evolution based on the notion of "survival of the fittest".

### *A. Differential Evolution (DE)*

Just like ACO, DE is also an iterative algorithm. Inputs to the DE algorithm is similar to the ACO algorithm: variables over which the objective function is minimised or maximised, the range for each of these variables, a procedure to put out the value of an objective function given the variables. Apart from these, the DE algorithm also has a few parameters which are different from those of the ACO algorithm. These include the number of solutions generated in each iteration (also known as population), mutation coefficient, crossover probability, the maximum number of iterations allowed, etc. Just as with the ACO algorithm, we will not discuss the effect of these parameters on the output of the algorithm since we have kept them fixed throughout our experiments.

The DE algorithm works as follows. In the very first iteration, it generates a number of solutions for the given problem, as specified by the population parameter. It then evaluates each of the generated solutions and calculates a "fitness" based on the output of the objective function and stores it. In the next iteration, it evolves the current population using crossover among themselves and generates a new population. It then compares the fitness of the new population against the fitness of the old population. Any new individual showcasing better fitness replaces an old individual. All the other new individuals whose fitness are worse than those of the old individuals are dropped from the list. (See the pseudo-code provided for details.) At the end of this iteration, a few individuals in the old list get replaced with new and better individuals, making the solution quality of the new population better than that of the old population. We then repeat this process for a certain number of generations while constantly measuring the solution quality across the population. The algorithm terminates early if all the individuals converge to a common solution.

A few important points to note about this algorithm are as follows. As is evident from the procedure described above, this algorithm does not need to compute the gradient for finding next iterate. This means that DE does not require the objective function to be differentiable unlike traditional optimisation algorithms, and so DE can be used when the objective functions are not continuous or are noisy. Another key thing to note here is that DE is a black-box optimisation toolbox, which can be used when the function to be optimised is very complex.

### *B. DE for Euclidean Steiner tree problem*

We use the above mentioned DE algorithm for our problem as follows. We start by generating population for just one ($k = 1$) relay location $(d_{ix}, d_{iy})$. We then generate a minimum spanning tree (MST) on vertices $(d_{ix}, d_{iy}) \cup \{x_1, x_2, \ldots, x_n\}$ using Euclidean distance between two points as cost.
Next, we calculate the fitness of the candidate location $(d_{ix}, d_{iy})$ as relay. As before, we take the fitness of the

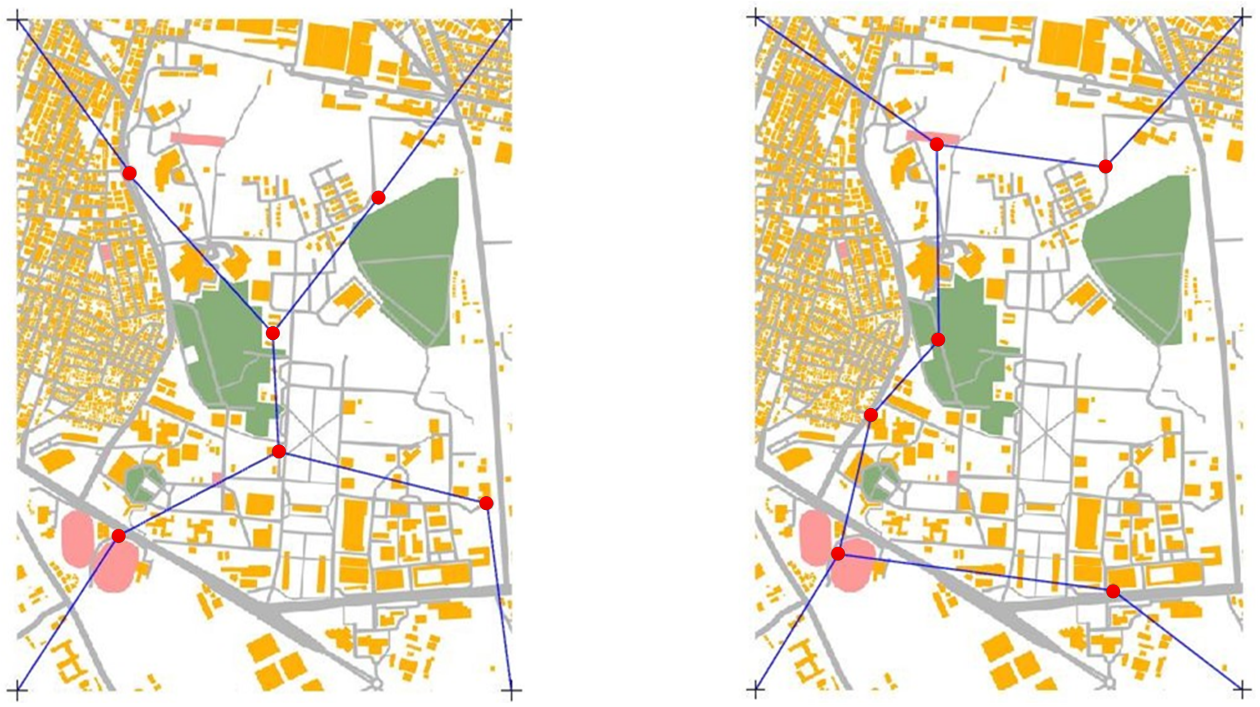
Fig. 3: Results for single gateway point and three transmitters in a heterogeneous region.

**DE algorithm for relay placement problem in a heterogeneous region**

---

**Data:** A set of $n$ transmitters $T = \{x_1, x_2, \ldots, x_n\}$ where each $x_i, 1 \leq i \leq n$ represents a transmitter location in the Euclidean plane and RSSI Threshold $R$.

**Result:** A set of $k$ relays $D = \{d_1, d_2, \ldots, d_k\}$ where each $d_i, 1 \leq i \leq k$ represents a relay location in the Euclidean plane.

---

location to be the negative of the maximum edge weight in the generated MST. The lower the maximum edge weight the better the fitness. The objective of the DE algorithm is to maximise the fitness of the population. When the fitness stabilises across the generations, we stop the execution of the algorithm. We then increase the number of relay location by one ($k = 2$) and rerun the DE algorithm. We compare the result of $k = 2$ with $k = 1$. If the solution improves for $k = 2$ then we continue the algorithm with $k = 3$. This process continues till $k = n - 2$ as it is the upper limit of the number of extra nodes needed for solving the Euclidean Steiner tree problem. If we do not get better results by increasing the number of relays, then we retain the solution with the previous value of $k$. The pseudo-code above specifies the DE algorithm.

All the solutions shown in Figure 1 were also achieved by this approach, which is reassuring.

### *C. DE for solving the relay placement problem in a heterogeneous region*

The approach we take to modify the above-mentioned algorithm for a heterogeneous propagation environment is the same as described in Section II-C. Again, we set RSSI between a pair of transmitter and receiver to be the RSSI estimate put out by the algorithm in [1]. While explaining DE, we did not mention the exact steps to generate a new population from the old population. There are many ways to do this, namely, rand/1/bin, rand/2/bin, best/1/bin, best/2/bin, rand-best/1/bin, etc. These different approaches are suited to different problems. For our relay placement problem, we used the rand/1/bin scheme to generate the new population. This scheme is explained in more detail in the pseudo-code above. To stress-test our algorithm, as before, the transmitters and gateway were kept at the corners of the map. Then we let the DE algorithm find the best relay locations. Figure 4 shows the results of this algorithm when we ran it independently. DE algorithm was able to connect the transmitters with five relays. In the next section, we pose the problem with multiple gateways and show how we can use the same algorithm with some extra work to solve the multiple gateways problem.

## IV. Relay placement problem with multiple gateways in a heterogeneous region

Often large-scale networks are deployed with multiple gateways to cover larger areas and distribute traffic across multiple aggregators. Despite the presence of multiple gateways, end sensor nodes in large deployment areas (say city-wide deployment) may still need relays to connect to the nearest gateway. In this section, we explain a simple heuristic approach to solve

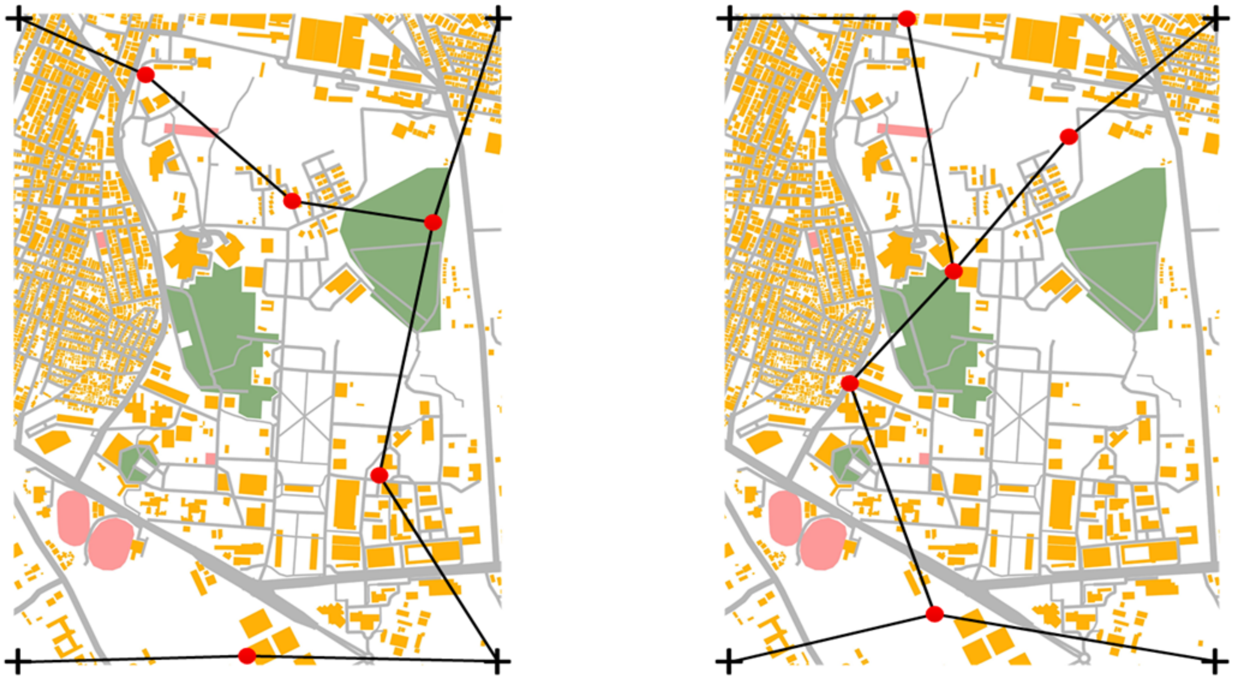
Fig. 4: Results for single gateway point and three transmitters in a heterogeneous region.

the connectivity problem when there are multiple gateways. The formal description is as follows.

**Problem:** Given $n$ number of transmitter locations $\{x_1, x_2, \ldots, x_n\}$, aggregating locations $\{y_1, y_2, \ldots, y_m\}$ in a heterogeneous region and an RSSI threshold $R$, find the minimum number of relays $k$ and their locations $\{d_1, d_2, \ldots, d_k\}$ so that each edge $e$ of the resulting constrained Minimum Weight Forest on $S = \{y_1, y_2, \ldots, y_m\} \cup \{x_1, x_2, \ldots, x_n\} \cup \{d_1, d_2, \ldots, d_k\}$ has RSSI $\geq R$. Further, the forest must cover all the transmitter locations, and every component of the forest must contain at least one of the aggregators:

$$\min_{\substack{(i,j)\in E(cMWF(S))\\ i\neq j}} RSSI(i,j) \geq R. \tag{2}$$

Here $cMWF(S)$ is a minimum weight spanning forest that meets the spanning constraint and the constraint that every component of the forest contains at least one aggregator. Remarks similar to those following equation (1) apply here as well.

### *A. Solution using ACO for multiple gateways*

We now propose one heuristic approach. The idea to ensure that the constraint is satisfied is to break the above-mentioned problem down to smaller problems each of which is similar to the problem which we solved in section II-C, and solve the smaller problems individually. We do that as follows. First, calculate the RSSI between each end node and gateway. Associate each end node to the gateway with the largest RSSI is the highest even if it is < -100 dBm. Create a cluster for each gateway by collecting all the end nodes associated with it. By doing this, we divide the entire network into smaller sub-networks corresponding to each cluster having precisely one gateway. Moreover, the use of computed RSSI based on the propagation environment ensures that the clustering takes both heterogeneity and geographical locations into account. We can now use the same algorithm described in section II-C to find relay locations for each cluster. Figure 5 shows the solution obtained using this approach.

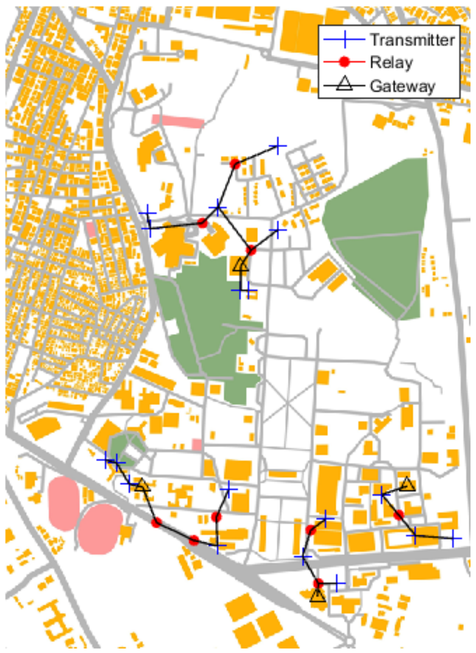

Fig. 5: Network design for four gateways and multiple transmitters in a heterogeneous region.

All the points marked with (+) in Figure 5 are actual locations of our IoT deployment for a smart water distribution application with overhead tanks, ground level reservoirs and

```
Initialization;
$k \leftarrow 0$;
$n_{itr} \leftarrow$ number of iteration;
$p \leftarrow$ population;
$m \leftarrow$ mutation coefficient;
$c_p \leftarrow$ crossover probability, $0 \le c_p \le 1$;
$V_{best} \leftarrow 0$, minimum function value;
$I_{best} \leftarrow 0$, index of minimum function value;
$p_{best} \leftarrow \varnothing$, best relay locations;
```

```
Repeat
    for $i \leftarrow 1$ to $p$ do
        $D_i^0 = Random\{d_{i,1}^0, d_{i,2}^0, \ldots, d_{i,k}^0\}$;
        $S = \{x_1, x_2, \ldots, x_n\} \cup D_i^0$;
        $F_i^0 = \min_{\substack{(p_1,p_2) \in E(MST(S)) \\ p_1 \neq p_2}} RSSI(p_1, p_2)$;
        #[$F$ is negative of fitness];
    end
    $(V_{best}, I_{best}) = \min F_i^0$; $p_{best} = D_{I_{best}}^0$;
    $D^0 = (D_1^0, \ldots, D_p^0)$;
    for $j \leftarrow 1$ to $n_{itr}$ do
        for $i \leftarrow 1$ to $p$ do
            $H_{temp} = D^{j-1} \setminus \{D_i^{j-1}\}$;
            $a, b, c = rand \in H_{temp}$;
            $d_{temp} = a + m * (b - c)$;
            #[]Note: $d_{temp} = (d_{temp}(1), \ldots, d_{temp}(k))$;]
             $D_i^j \leftarrow \varnothing$;
            for $l \leftarrow 1$ to $k$ do
                $b_{temp} = Random(0, 1)$;
                if $b_{temp} > c_p$ then
                    $D_i^j \leftarrow D_i^j \cup d_{temp}(l)$ #[replace];
                else
                    $D_i^j \leftarrow D_i^j \cup D_i^{j-1}(l)$ #[retain];
                end
            end
            $S_t = \{x_1, x_2, \ldots, x_n\} \cup D_i^j$;
            $F_{temp} = \min_{\substack{(p_1,p_2) \in E(MST(S_t)) \\ p_1 \neq p_2}} RSSI(p_1, p_2)$;
            if $F_{temp} < F_i^{j-1}$ then
                $D_i^j = D_i^j$; $F_i^j = F_{temp}$;
            else
                $D_i^j = D_i^{j-1}$; $F_i^j = F_i^{j-1}$;
            end
        end
        $(V_{best}, I_{best}) = \min F_i^j$; $p_{best} = D_{I_{best}}^j$;
    end
    if $V_{best} \ge R$ then
        print "Solution Reached";
        $Output \leftarrow p_{best}$;
        break;
    else
        $k \leftarrow k + 1$;
    end
end
```

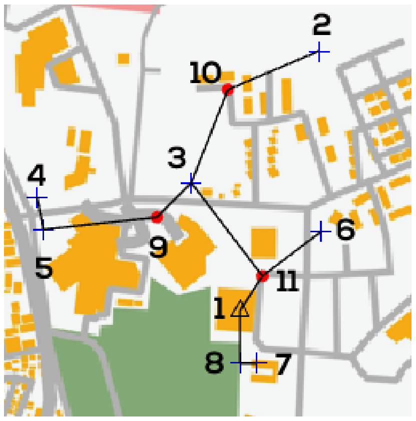

Fig. 6: Network design for North Side of IISc campus.

| Node Id (Tx) | Node Id (Rx) | RSSI(dBm) |
|---|---|---|
| 1 | 8 | -99.1163 |
| 1 | 11 | -95.7368 |
| 2 | 10 | -85.7739 |
| 3 | 9 | -74.9758 |
| 3 | 10 | -85.8552 |
| 3 | 11 | -87.8343 |
| 4 | 5 | -69.9692 |
| 5 | 9 | -88.9698 |
| 6 | 11 | -80.6244 |
| 7 | 8 | -77.2838 |

TABLE I: Network Connectivity.

flow meters. Gateway locations are marked with the triangle symbol. We ran our algorithm to predict the coverage and identify whether we need relays to ensure connectivity. Our proposed multiple-gateway algorithm suggested nine relays and their locations which are marked with the red circles. Figure 6 shows a zoomed-in version for the North cluster. As shown in the Table I, all the links are predicted to perform above minimum RSSI threshold $R$ of -100 dBm. Note that one should not compare this solution with the solutions of Figure 3 or 4 which is for a different set of transmitter and gateway locations.

### B. Solution using DE for multiple gateways

We use the same divide-and-conquer approach in Section IV-A but now with the DE algorithm. The results of this simulation are shown in Figure 7. The number of relays needed in both cases remain the same. But, there was a change in the location of the relays. Figure 8 shows a zoomed-in version for the North cluster. Table II shows the observed RSSI between each pair of transmitter and receiver. Notice that all the links in the network have RSSI greater than RSSI threshold $R$ of -100 dBm. The link RSSIs are also a little more balanced than in Table II.

### C. Discussion

The algorithm explained in subsections IV-A and IV-B might lead to a sub-optimal solution. This scenario is shown in Figure 9 for which a better solution is to connect all the end nodes (+) to gateway 1 and to let gateway 2 operate by itself. The association of green end nodes to gateway 1 and

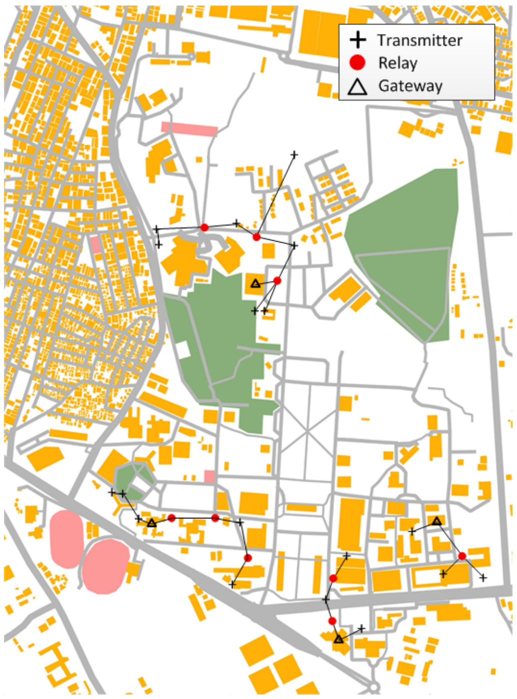

Fig. 7: Network design for four gateways and multiple transmitters in a heterogeneous region.

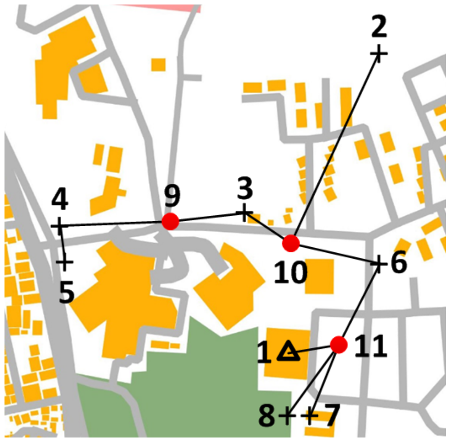

Fig. 8: Network design for North Side of IISc campus.

| Node Id (Tx) | Node Id (Rx) | RSSI(dBm) |
|---|---|---|
| 1 | 11 | -85.8028 |
| 2 | 10 | -85.3519 |
| 3 | 9 | -87.5490 |
| 4 | 9 | -81.2334 |
| 5 | 4 | -71.8457 |
| 8 | 11 | -88.5876 |
| 10 | 3 | -89.2899 |
| 10 | 6 | -87.2158 |
| 11 | 6 | -88.8446 |
| 11 | 7 | -86.2015 |

TABLE II: Network Connectivity.

red end node to gateway 2 in the clustering step led to this sub-optimal solution. The reason for this aberration is that our divide-and-conquer procedure that identifies the smaller sub-problems to apply the earlier single-gateway problem is not directly optimising equation (2).

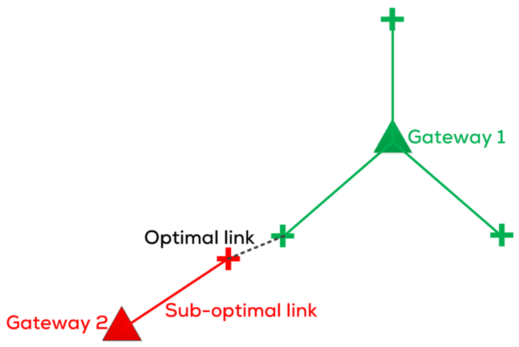

Fig. 9: Algorithm leading to a sub-optimal solution.

## V. Conclusions and Ongoing Work

In this paper, we used a function of RSSI as a distance measure between the transmitter and receiver when placed in a heterogeneous propagation environment. Two approaches, ACO-based approach, and DE algorithm, for relay placements, were first checked to give good solutions for finding the minimum number of relays and their locations in the Euclidean Steiner tree problem. We then showed how to adapt these approaches to solve the relay placement problem in both single and multiple gateways scenarios under heterogeneous propagation environments. We are not aware of other algorithms or benchmarks for the heterogeneous environment to compare our algorithms and outcomes.

We are working on the following improvements to our current implementation.

- Design of more sophisticated algorithms than the divide-and-conquer approach given in Section IV-A to handle the suboptimality highlighted in IV-C.
- Incorporation of location constraints on relay locations.

## VI. Acknowledgements

This work was supported in part by the Department of Electronics and Information Technology, Government of India, and in part by the Robert Bosch Centre for Cyber-Physical Systems, Indian Institute of Science, Bangalore. We would like to thank Renu Subramanian for helping us in simulations. N. Rathod was supported by a Cisco Research Scholarship.

## References

[1] N. Rathod, R. Subramanian, and R. Sundaresan, "Data-driven and gis-based coverage estimation in a heterogeneous propagation environment," in *IEEE Global Communications Conference*, IEEE, 2018.

[2] F. K. Hwang, D. S. Richards, P. Winter, and P. Widmayer, "The steiner tree problem, annals of discrete mathematics, volume 53," *ZOR-Methods and Models of Operations Research*, vol. 41, no. 3, p. 382, 1995.

[3] A. Bhattacharya and A. Kumar, "A shortest path tree based algorithm for relay placement in a wireless sensor network and its performance analysis," *Computer Networks*, vol. 71, pp. 48–62, 2014.

[4] A. Bhattacharya, S. M. Ladwa, R. Srivastava, A. Mallya, A. Rao, D. G. R. Sahib, S. Anand, and A. Kumar, "Smartconnect: A system for the design and deployment of wireless sensor networks," in *Communication Systems and Networks (COMSNETS), 2013 Fifth International Conference on*, pp. 1–10, IEEE, 2013.

[5] P. Crescenzi, V. Kann, and M. Halldórsson, "A compendium of np optimization problems," 1995.

[6] M. Brazil, R. L. Graham, D. A. Thomas, and M. Zachariasen, "On the history of the euclidean steiner tree problem," *Archive for history of exact sciences*, vol. 68, no. 3, pp. 327–354, 2014.