Optimized Relay Node Placement for Establishing Connectivity in Sensor Networks

Fatih Senel and Mohamed Younis
Dept. of Computer Science and Electrical Engineering
University of Maryland Baltimore County
Baltimore, MD 21250
{fsenel1, younis}@cs.umbc.edu

Abstract—**Relay node placement in wireless sensor networks has gained importance due to its potential use in prolonging network life time, reducing data latency, and establishing connected topologies. In this paper we studied the relay node placement problem to establish multi-hop communication paths between every pair terminals (i.e., sensors) where each hop in the path is less than a common communication range. Such a problem is defined as Steiner Tree problem with minimum Steiner points and Bounded Edge-Length problem which is known to be NP-Hard. This paper presents a novel relay node placement heuristics called Incremental Optimization based on Delaunay Triangulation (IO-DT). The algorithm takes advantage of feasibility of finding optimal solution for the case of three terminals. IO-DT calculates the Delaunay triangulation (DT) of terminals and iterates over the formed triangles. In each iteration the algorithm steinerizes a triangle as part of the final topology if selecting such a triangle provides a reduction in total number of relay node required as compared to the *minimum spanning tree (mst)* based approach. The time complexity of IO-DT is quadratic in the number of terminals, which is superior to competing schemes. The performance of the algorithm is validated through simulation.**

Keywords: **Node placement, Network connectivity, Topology management, Wireless Sensor Networks**

I. INTRODUCTION

Recent years have witnessed growing interest in the applications of wireless sensor networks (WSN) [1]. Some of these applications serve in inhospitable environments and makes random node placement the only option for network deployment [2]. On the other hand, many applications of WSNs serve in controlled setups in which deterministic positioning of node is feasible and desirable [3]. Examples of these applications include security surveillance, habitat monitoring, safety assessment of factory floor, hazard detection in urban area, etc. The controlled nature of the application allows pre-planning and careful selection of where to place nodes and how they may communicate. Usually area or landmark coverage is the main criterion for determining where sensor nodes should be. For the inter-sensor connectivity, relay nodes (RNs) are deployed to ensure that data can be shared among the sensors, and between them and the network access points.

Relay nodes may also be used to restore connectivity in a WSN after the failure of multiple nodes that leaves the network partitioned into multiple disjoint segments. Another scenario is when multiple standalone WSNs are to be federated in order to combine their services and implement new tasks for certain emerging events. Determining the number and the position of RNs in all application scenarios is usually subject to optimization. The application designer often likes to employ the least RN count for achieving the desired connectivity goal. In addition, the positioning of these RNs should establish, possibly a multi-hop path between the terminals, e.g., sensors or WSN segments, subject to communication range constraints. This problem is equivalent to the Steiner Minimum Tree with Minimal Steiner Points and Bounded Edge-Length (SMT-MSPBEL), which is shown to be NP-Hard by Lin and Xue [6].

A number of heuristics have been proposed in the literature for solving the SMT-MSPBEL problem. Some of these approaches focuses on reducing the number of RNs and form a minimum spanning tree (*mst*), where the node degree of the individual RNs is two [4][5][6]. On the other hand, the focus of other work has been on the degree of connectivity of the formed WSN topology and less emphasis on the RN count [8][9]. In this paper, we present a novel algorithm that minimizes the required RN count for establishing connectivity. The proposed Incremental Optimization based on Delaunay Triangulation (IO-DT) optimally solves the SMT-MSPBEL problem for the case of three terminals and tries to find the minimal RN count for larger networks incrementally using subsets of three terminals**.** Basically, IO-DT calculates the Delaunay triangulation (DT) of terminals and iterates over the formed triangles. In each iteration the algorithm steinerizes a triangle as part of the final topology if selecting such a triangle provides a reduction in the total number of required relays as compared to an *mst* -based solution. IO-DT is validated through simulation is shown to outperform contemporary heuristics in the literature.

This paper is organized as follows. The next section discusses related work. Section III formally defines the problem and introduces some notations. Section IV describes the IO-DT approach in detail. In section V presents the simulation results. Finally section VI concludes the paper.

II. RELATED WORK

In addition to form connected topologies, relay node placement in WSNs has been pursued to improve the performance of WSNs, e.g., prolong network lifetime. A survey can be found in [3]. Given the scope of the contribution, we focus on work that targets network connectivity. Most of the published studies in this category opt to place minimum number of RNs in such a way that there exists a path between every pair of terminals and

978-1-4673-0921-9/12/$31.00 ©2012 IEEE

each hop in the path is less than or equal to the transmission range of the nodes. As we mentioned, this problem is shown to be NP-Hard by Lin and Xue [6]. They presented a polynomial time approximation algorithm that deploys RNs along the *mst* edges of terminals. The algorithm first constructs the complete graph of terminals and calculates *mst* of the using Kruskal's algorithm. RNs are populated along the tree edges at a distance at most *R* apart, where *R* is the communication range of a RN. In [7], Chen et al. proved that the approximation ratio of the algorithm Lin and Xue [6] is equal to 4.

In [4], Cheng et al., employ a three-step heuristic to form Steiner Minimum Tree with minimum Steiner Points. In the first step it connects the nodes where the distance between the nodes is less than or equal to *R*. In the second step, it forms 3-stars as follows; for each subset of three nodes $\{u, v, w\}$, if there exists a point s such that s is at most R units away from u, v, and w. In the last step the algorithm populates RNs along *mst* edges connecting two different connected components. In [5], Lloyd et al. studied the same problem assuming that relays and sensors have different transmission ranges. In this paper, we focus on the same problem studied in [4][6][7]. Unlike these published work, the proposed IO-DT algorithm finds a subset of triangles and forms triangular Steiner trees by connecting the vertices of the triangles at a point inside the triangle which will be called as Discrete Fermat Point. IO-DT algorithm significantly performs better than the algorithms presented in [4][6][7], as will be confirmed by the simulation experiments. In [10], we have studied the problem in the context of federating disjoint network segments and presented an algorithm called FeSTA. The algorithm iterates over all possible triangles and greedily selects triangles to apply Discrete Fermat Point optimization. IO-DT has significantly less time complexity than FeSTA (i.e., $O(n^2)\ vs.\ O(n^4)$) by considering only Delaunay triangles rather than all possible triangles.

The focus of [8] and [9] is also on a variant of the SMT-MSPBEL problem where additional metric is to be optimized. In [8], Lee and Younis strive to connect the terminals inward so that the inter-terminal topology has a high node degree. They proposed an algorithm called CORP that models the deployment area as an equal-sized grid cells. CORP has two phases. In the first phase, the algorithm iteratively identifies border terminals and identifies the best cell to deploy an RN. The cells connecting multiple terminals are called junction cells. In the second phase, after all terminals are connected, the algorithm prunes redundant RNs. In [9] a bio-inspired heuristic called SpiderWeb has been proposed to form topologies that not only exhibit stronger connectivity but also achieve better sensor coverage and enable balanced distribution of traffic load on the employed relays. Unlike these algorithms, is reducing the number of RNs is the main objective for IO-DT.

III. Fundamental Definitions

The problem that we study can be formally defined as follows: "Given *n* disjoint terminals, determine the least number and position of relay nodes required for establishing inter-terminal connectivity." Optimal solution to this problem can be achieved by forming Steiner Minimum Tree with Steiner Points and Bounded Edge Length (SMT-MSPBEL). SMT-MSPBEL finds the minimum cost sub-tree that spans all terminals with upper-bounded edge lengths. Since the SMT-MSPBEL problem is NP-Hard, we pursue heuristics. The following are important definitions that cover the basis of the IO-DT algorithm.

Definition 1: The minimum number of SPs required for forming SMT-MSPBEL for two terminals u and v is called the *sp-weight* of the edge (u, v) and is denoted by $W_{sp}(u, v) = \left\lceil \frac{d(u,v)}{R} \right\rceil - 1$, where $d(u, v)$ is the Euclidean distance between u and v, and R is the maximum edge length which is in essence the transmission range of the nodes.

Definition 2: Let T_i be a triangle with vertices (u, v, w). The Fermat Point (FP) f_i (also called Torricelli Point) for T_i is the point that minimizes the sum of Euclidean distances from the vertices of T_i to f_i, i.e., minimize $d(u, f_i) + d(v, f_i) + d(w, f_i)$.

Definition 3: Let T_i be the triangle (u, v, w) and p be a point inside T_i. The number of relays required for connecting the vertices of T_i at an arbitrary point x is called the fp-weight of T_i at x and denoted as:

$$W_{fp}(T_i, x) = W_{sp}(u, x) + W_{sp}(v, x) + W_{sp}(w, x) + 1$$
$$= \left(\left\lceil \frac{d(u,x)}{R} \right\rceil - 1\right) + \left(\left\lceil \frac{d(v,x)}{R} \right\rceil - 1\right) + \left(\left\lceil \frac{d(w,x)}{R} \right\rceil - 1\right) + 1$$
$$= \left\lceil \frac{d(u,x)}{R} \right\rceil + \left\lceil \frac{d(v,x)}{R} \right\rceil + \left\lceil \frac{d(w,x)}{R} \right\rceil - 2$$

The point φ_i that minimizes the fp-weight of triangle T_i is called Discrete Fermat Point (DFP) and the weight at DFP is called the *dfp-weight* of the triangles and denoted as $W_{fp}(T_i, \varphi_i)$, or simply $W_{dfp}(T_i)$. If we remove the ceiling brackets and multiply by *R*, the minimization becomes linear in the distance and the point φ_i would be the Fermat Point [14]. The DFP is the discretized version of Fermat Point, which is the basis for the name.

Definition 4: Let T_i be a triangle with vertices (u, v, w). Given a minimum spanning tree G_{mst} that includes u, v, and w, assume that Υ_1 and Υ_2 are two paths in G_{mst} from u to v and from u to w, respectively, i.e., $\Upsilon_1 \nsubseteq \Upsilon_2$ and $\Upsilon_2 \nsubseteq \Upsilon_1$. Let e_1 and e_2 be the edges having maximum *sp-weight* in Υ_1 and Υ_2, respectively. Also let e_3 and e_4 be the edges having the maximum *sp-weight* in $\Upsilon_1 - \Upsilon_2$ and $\Upsilon_2 - \Upsilon_1$, respectively. The *mst-weight* of T_i is denoted and computed as follows:

$$W_{mst}(T_i) = \begin{cases} W_{sp}(e_1) + W_{sp}(e_2) & if\ e_1 \neq e_2 \\ W_{sp}(e_1) + \max\{W_{sp}(e_3), W_{sp}(e_4)\} & ow \end{cases}$$

The two *mst* edges e_1 and e_2 (or e_1 and the longest of e_3 and e_4, if $e_1 = e_2$) are called the associated *mst* edges with T_i. Note that if a triangle T_i contains two *mst* edges $e_1 = (u, v)$ and $e_2 = (u, w)$ then the *mst*-weight of T_i will be:

$$W_{mst}(T_i) = W_{sp}(e_1) + W_{sp}(e_2)$$

Definition 5: The min-weight of a triangle T_i is the minimum of *mst-weight* and *dfp-weight* of T_i

$$W(T_i) = \min\{W_{mst}(T_i), W_{dfp}(T_i)\}$$

The gain of T_i is the difference between *mst-* and *dfp-weights* of T_i, i.e., $Gain(T_i) = W_{mst}(T_i) - W_{dfp}(T_i)$

IV. Establishing Inter-Terminal Connetivity

In this section we present the details of our IO-DT algorithm. The intuition behind IO-DT is based on feasibility of finding an optimal solution for three terminals, which we refer to as DFP optimization in the balance of the paper

A. DFP Optimization

Per definition 3, the DFP, φ_i, of a triangle $T_i(u,v,w)$ is a point which minimizes $W_{sp}(u,\varphi_i) + W_{sp}(v,\varphi_i) + W_{sp}(w,\varphi_i)$. In order to find a DFP, we first compute candidate points by drawing circles with incremental radii (*R, 2R, 3R, …, kR*) from each corner, as illustrated in Figure 1. The intersection points of the circles are identified as candidate locations for DFP. We calculate the *fp-weights* at each candidate point and pick the point having minimum *fp-weight* as the DFP. The motivation of drawing the circles to find candidate locations comes from the definition of *sp-weight*. Let the *sp-weight* of an edge (u,φ_i) be $k-1$. Then, $W_{sp}(u,\varphi_i) = \left\lceil \frac{d(u,\varphi_i)}{R} \right\rceil - 1 = k-1$, and $(k-1)R < d(u,\varphi_i) \leq kR$. Hence the value of $d(u,v)$ can be at most kR. In other words if the *sp-weight* of the edge (u,φ_i) is $k-1$, then φ_i cannot be located outside the circle with radius kR and centered at u. Based on this observation we used intersection points of circles as the candidate locations of DFP. We have proven that this procedure yields optimal solution for the SMT-MSPBEL problem for the case of 3 terminals. The proof is not included in this paper due to space constraints. Figure 2 shows topologies based on steinerizing *mst* edges and based on the DFP for a sample triangle. In this example, the *dfp-weight* is less than the *mst-weight*.

B. The IO-DT Heuristic

As discussed earlier one of the intuitive approaches for establishing inter-terminal connectivity is to form steinerized *mst* where RNs are placed with a distance of *R* units apart from each other along the *mst* edges of the terminals. However, steinerizing a triangle via DFP optimization may reduce the required number of RNs to connect the vertices of a triangle, as illustrated in Figure 2. The main idea behind the IO-DT algorithm is to leverage the DFP optimization by identifying subsets of three terminals (triangles), placing RNs at the DFP of these triangles and modifying the *mst* of terminals on-the-fly to form steinerized *mst* on the modified tree where the required number of RNs for restoring connectivity is minimized. The key question for the algorithm is how to identify a subset of triangles which minimizes total number of relay nodes. Since there are $\binom{n}{3} = O(n^3)$ possible triangles, where n is the number of terminals, the complexity of the solution increases significantly if all triangles are to be considered. Most of these triangles, however, are not needed since considering big triangles having multiple terminals inside, may require many redundant RNs. For example in Figure 3(a), the triangle $T(u,v,w)$ contains the terminal z. The DFP optimization of the triangle $T(u,v,w)$ yields 7 RNs one of which is redundant as shown in Figure 3(b). Therefore the algorithm should avoid those redundancies. In other words we need a triangulation such that no terminal is located inside any triangle. Delaunay Triangulation (DT) meets this requirement and thus enables us to determine the set of triangles that ought to be considered. Hereafter, we use Γ_{dt} to denote the DT graph of terminals.

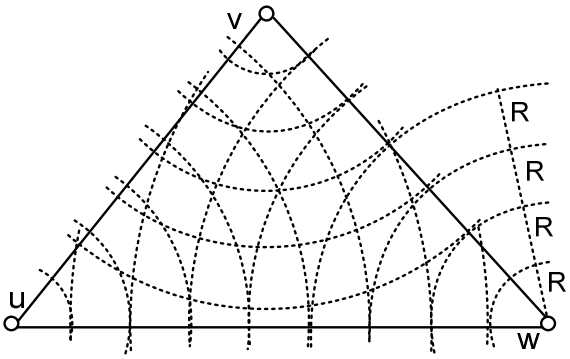

Figure 1: The intersection points of the circles are DFP candidates. Radius of each circle is multiple of R

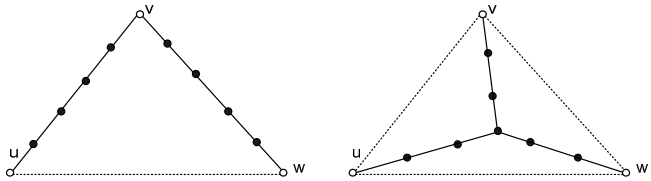

Figure 2: Illustration of DFP optimization of a triangle (a) steinerizing *mst* edges (b) DFP optimization.

The IO-DT algorithm looks for the best option for connecting the terminals of each triangle such that the required RN count for the entire network is minimized. It, however, tries to do so by incrementally considering groups of 3 terminals. The first step for IO-DT is to form G_{mst}, i.e., the *mst* of all terminals, and determine Γ_{dt}. IO-DT then considers each triangle in Γ_{dt} to decide on whether *mst* edges or DFP based steinerization is better. By definition 4, in order to calculate the *mst-weight* of a triangle, we need to find two *mst* paths Υ_1 and Υ_2. The sum of the *sp-weight* of the two largest edges e_1 and e_2 in Υ_1 and Υ_2, respectively, will be the *mst-weight* of the triangle. Since the *mst* changes dynamically, i.e., after each RN deployment at a DFP of one of the triangles, the *mst-weight* and the gain of a triangle may change depending on the previously deployed RNs. To tackle these challenges, we pursue a greedy heuristic.

Let $\Phi^{\Gamma_{dt}}$ be the sorted list of triangles in the Γ_{dt} in an ascending order according to the *dfp-weight* of the individual triangles. The rationale for pursing such an order is that we like to introduce edges with the smallest possible *sp-weight* to the emerging network topology, i.e., involving the least RN count, so that we can remove existing edges with large *sp-weight* in subsequent iterations. The algorithm iteratively processes the

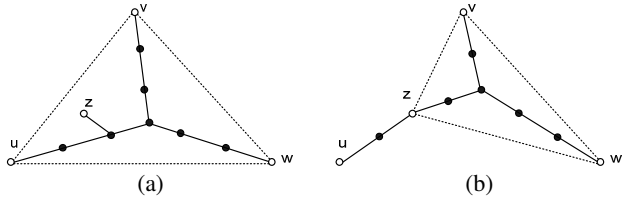

Figure 3: Illustration of redundant node deployment as a result of DFP optimization of a triangle having another terminal inside. (a) 7 RNs are required (b) 6 RNs are required for connecting 4 terminals.

triangles in $\Phi^{\Gamma_{dt}}$. In each iteration the *mst-weight* and the gain of the selected triangle $T_i(u,v,w)$ are calculated. If T_i has positive gain, the algorithm deploys an RN at the DFP of T_i, and updates the *mst* by removing the two *mst* edges that are associated with T_i and adding the three edges (u,φ_i), (v,φ_i) and (w,φ_i) instead, where φ_i is the DFP of T_i.The pseudo code for IO-DT algorithm is provided in Algorithm 1.

Theorem 1: The run time complexity of Algorithm 1 is $O(n^2)$ where n is the number of terminals.

Proof: The IO-DT heuristic consists of a series of algorithms. We will analyze each operation individually. Before we continue the proof we'll give the followings lemmas:

Lemma 1: Both the number of edges and the number of triangles in $DT(P)$ can be bounded by $O(n)$ where n is the number of points in P [13].

Lemma 2: the *mst* is a sub-graph of Γ_{dt} [11].

Lemma 3: The cost of finding a path between two vertices in a tree is $O(|V|+|E|)$, where $|V|$ and $|E|$ are the number vertices and edges, respectively. In *mst*, since $|E| = |V| - 1$, we have $O(|V|)$ as the overall time complexity [12].

Back to the proof, in line 2 the algorithm calculates DT of terminals where the cost is $(n \log n)$ [13]. The cost of forming a

Procedure IO − DT

Input: *set of terminal points P*
Output: *set of relay nodes S and modified* G_{mst}

1 $S \leftarrow \emptyset$
2 *Let* $\Gamma_{\mathrm{dt}}(P, E_{dt})$ *be the DT of P where* E_{dt} is the edges in DT
3 *Let* $G_{mst}(V_{mst}, E_{mst})$ *be the mst of P where* $V_{mst} = P$
4 *Let* $\Phi^{\Gamma_{dt}}(P)$ *be the sorted list of triangles in* Γ_{dt}
5 ***for each*** $T_i(u,v,w) \in \Phi^{\Gamma_{dt}}(P)$ ***do***
6 *Let* Υ_1 be a path in G_{mst} from u to v and Υ_2 be a path in G_{mst} from u to w where $\Upsilon_1 \nsubseteq \Upsilon_2$ and $\Upsilon_2 \nsubseteq \Upsilon_1$
7 *Let* e_1*and* e_2 *be the edges having maximum sp − weight in* Υ_1 *and* Υ_2*, respectively.*
8 *Also let* e_3 *and* e_4 *be the edges having the maximum sp − weight in* $\Upsilon_1 - \Upsilon_2$ *and* $\Upsilon_2 - \Upsilon_1$*, respectively.*
9 $e_x \leftarrow e_1$ and $e_y \leftarrow null$
10 ***if*** $e_1 \neq e_2$ ***then***
11 $e_y \leftarrow e_2$
12 ***else if*** $W_{sp}(e_3) > W_{sp}(e_4)$ ***then***
13 $e_y \leftarrow e_3$
14 ***else***
15 $e_y \leftarrow e_4$
16 ***endif***
17 $W_{mst}(T_i) = W_{sp}(e_x) + W_{sp}(e_y)$
18 ***if*** $W_{mst}(T_i) > W_{dfp}(T_i)$ ***then***
19 *put* s_i *to the DFP of* T_i
20 $S \leftarrow S \cup \{s_i\}$
21 $V_{mst} \leftarrow V_{mst} \cup S$
22 $E_{mst} \leftarrow \left(E_{mst} - \{e_x, e_y\}\right) \cup \{(u,s_i),(v,s_i),(w,s_i)\}$
23 ***endif***
24 ***endfor***
25 ***return*** *S and* G_{mst}

Algorithm 1: Pseudo code for IO-DT Algorithm

mst using Kruskal's Algorithm is $O(|E| \log |V|)$, where $|E| = O(n)$ and $|V| = n$ (by lemma 1, and lemma 2). Thus the cost of forming the *mst* is (line 3) is $O(n \log n)$. By Lemma 1, since the number of triangles in $DT(P)$ is $O(n)$, the cost of sorting the triangles in line 4 becomes $O(n \log n)$. The "for" loop (lines 5 to 24) iterates $O(n)$ times. Inside the loop, the algorithm calculates two paths Υ_1 and Υ_2 (line 6), and finds the two edges e_x and e_y having the largest *sp-weights*. By Lemma 3, the cost of finding a path in a tree is $O(n)$. Thus the complexity of the for loop is $O(n^2)$. Hence the overall time complexity is $O(n^2)$. ■

C. Illustrative Example

Figure 4 illustrates how IO-DT algorithm works. First the algorithm calculates Γ_{dt} of the terminals. A subset of three terminals forms a triangle. The algorithm sorts the list of triangles which appear in Γ_{dt} according to their *dfp-weights*. Table 1 shows the sorted list of triangles. Figure 4(a) shows the DT of terminals. In Figure 4(a) the solid lines represent the *mst* edges and dashed lines are edges that belong only to Γ_{dt}; recall that per Lemma 2 the *mst* is a sub-graph of Γ_{dt}. IO-DT is a greedy algorithm and at each iteration the algorithm picks next the triangle in the sorted list having the least *dfp-weight*. In the first iteration, the algorithm picks $T(t_7, t_9, t_{10})$ and calculates the two paths Υ_1 and Υ_2 from t_7 to t_9, and t_7 to t_{10}, respectively. For this triangle, $\Upsilon_1 = \{(t_7, t_9)\}$ and $\Upsilon_2 = \{(t_7, t_{10})\}$. Thus,

$$W_{mst}\big(T(t_7, t_9, t_{10})\big) = W_{sp}(t_7, t_9) + W_{sp}(t_7, t_{10}) = 2.$$

Since the *dfp* and *mst* weights of the triangle $T(t_7, t_9, t_{10})$ are equal, the algorithm skips this triangle. Similarly in the second iteration the triangle $T(t_1, t_7, t_{10})$ is skipped due to the same reason. In the third iteration the triangle $T(t_2, t_8, t_{12})$ is picked, where $\Upsilon_1 = \{(t_2, t_8)\}$ and $\Upsilon_2 = \{(t_8, t_{12})\}$. Since the *dfp-weight* of the triangle is less than the *mst-weight*, the algorithm places an RN, s_1, at the DFP of $T(t_2, t_8, t_{12})$ and updates the *mst* as illustrated in Figure 4(b). In the next iteration the triangle $T(t_1, t_{10}, t_{12})$ is considered, where $\Upsilon_1 = \{(t_1, t_{10})\}$ and $\Upsilon_2 = \{(t_{10}, t_7), (t_7, t_9), (t_9, t_{14}), (t_{14}, t_6), (t_6, t_2), (t_2, s_1), (s_1, t_{12})\}$. Since (t_6, t_2) is the edge with the largest weight in Υ_2, where $W_{mst}\big(T(t_1, t_{10}, t_{12})\big) = W_{sp}(t_1, t_{10}) + W_{sp}(t_6, t_2) = 3$, the algorithm places an RN, s_2, to the DFP of $T(t_1, t_{10}, t_{12})$ and updates the *mst* by removing the edges (t_1, t_{10}) and (t_6, t_2), and adding (t_1, s_2), (t_{10}, s_2) and (t_{12}, s_2) as illustrated in Figure 4(c). The rest of the execution of the algorithm is depicted in Table 1 and Figures 4 (d) and (e). Finally the algorithm steinerizes the *mst* edges to form SMT-MSPBEL in Figure 4(f).

V. Perfromance Evaluation

A. Experiment Setup and Baseline for Comparison

The performance of IO-DT is validated through simulation. We have created varying number terminals (10 to 25) located randomly in area of interest (1500m × 1500m). We have fixed the communication ranges of RNs (R) to 100m. We compare the performance IO-DT with three baseline algorithms, namely SMST [6][7], SMT-MSP 3-app [4], and FeSTA [10]. We consider the following metrics for assessing the performance:

- *Number of RNs*: Minimizing the number of RNs for restoring the connectivity is the primary objective of IO-DT.

Table 1: List of triangles in Γ_{dt} as being considered during the execution of the IO-DT algorithm.

Triangle T_i	$W_{dfp}(T_i)$	$W_{mst}(T_i)$	Action
(t_7, t_9, t_{10})	2	2	skip
(t_1, t_7, t_{10})	2	2	skip
(t_2, t_8, t_{12})	2	3	1) Place s_1 to the DFP 2) Remove (t_2, t_8) and (t_8, t_{12}) from *mst* 3) Add (t_2, s_1), (t_8, s_1) and (t_{12}, s_1) to *mst*
(t_1, t_{10}, t_{12})	2	3	1) Place s_2 to the DFP 2) Remove (t_1, t_{10}) and (t_2, t_6) from *mst* 3) Add (t_1, s_2), (t_{10}, s_2) and (t_{12}, s_2) to *mst*
(t_6, t_{11}, t_{15})	3	4	1) Place s_3 to the DFP 2) Remove (t_6, t_{15}) and (t_{11}, t_{15}) from *mst* 3) Add (t_6, s_3), (t_{11}, s_3) and (t_{15}, s_3) to *mst*
(t_3, t_8, t_{15})	3	4	1) Place s_4 to the DFP 2) Remove (t_3, t_8) and (t_6, t_{14}) from *mst* 3) Add (t_3, s_4), (t_8, s_4) and (t_{15}, s_4) to *mst*
(t_2, t_6, t_{15})	3	2	skip
(t_2, t_9, t_{14})	3	3	skip
(t_2, t_6, t_{14})	3	3	skip
(t_4, t_5, t_{13})	3	4	1) Place s_5 to the DFP 2) Remove (t_4, t_{13}) and (t_5, t_{13}) from mst 3) Add (t_4, s_5), (t_5, s_5) and (t_{13}, s_5) to *mst*
(t_2, t_9, t_{10})	3	2	skip
(t_2, t_{10}, t_{12})	3	2	skip
(t_3, t_8, t_{12})	4	1	skip
(t_3, t_{11}, t_{15})	4	2	skip
(t_2, t_8, t_{15})	4	2	skip
(t_5, t_6, t_{14})	4	4	skip
(t_4, t_5, t_{14})	4	3	skip
(t_4, t_9, t_{14})	4	4	skip
(t_4, t_7, t_9)	5	3	skip
(t_5, t_6, t_{11})	5	3	skip

- *Average Node Degree*: This is the average number of neighbors of each RN in the resulting topology. A higher node degree indicates stronger connectivity and helps in balancing the traffic load and reducing the data latency.
- *Expected Path Length*: This is the expected path length in terms of number of hop between two terminals. Shorter paths are desired since it reduces the data delivery delay.

B. Performance Results

This section provides the performance results. The result for each configuration is the average of 100 different random topologies. We observed that with a %95 confidence interval, our results stayed within %6-%10 of the sample mean.

Figure 5(a) shows that IO-DT and FeSTA performs significantly better than SMST and SMT-MSP 3-app. The performance gap increases as the number of terminals increases, since, the number of triangles which IO-DT and FeSTA consider, increases with the growth in terminal count. On the other hand, the performance of IO-DT is slightly better than FeSTA. The performance advantage of IO-DT over FeSTA is further illustrated in Figure 5(b). The figure shows the number of RNs gained in each algorithm with respect to SMST. Clearly the gain for IO-DT is more than for FeSTA, which indicates that IO-DT requires fewer RNs than FeSTA. The reason for such performance will be clear when we discuss Figure 6.

The average node degree comparison of the algorithms is illustrated in Figure 6. Again, IO-DT and FeSTA outperform SMST and SMT-MSP 3-app algorithms in terms of average node degree. In SMST and SMT-MSP 3-app algorithms almost all of the RNs have two neighbors since they are populated along *mst* edges. In contrast, IO-DT and FeSTA place some of the RNs to the DFPs of the triangles, which yield a node degree of three. Thus if the algorithm finds more triangles to optimize, the average node degree gets closer to three, since more RNs will have a node degree of three. Interestingly, FeSTA yield higher average node degree than IO-DT, although IO-DT requires fewer RNs as illustrated in Figure 2(b). The results in Figure 6 indicate that, FeSTA finds more triangles to optimize than IO-DT, with a less overall gain. Therefore, it can be said that some of the triangles selected by FeSTA do not contribute in minimizing the number of RNs, as was evident in Figure 5(b). FeSTA lists all possible triangles (i.e. 3-combinations of terminals) and sorts the triangles according their weights. The weight of a triangle in FeSTA, equals the minimum of *mst-weight* and *dfp-weight*. In addition, the *mst-weight* of a triangle in FeSTA is equal to the sum of sp-weights of the two shortest edges. These two edges, however, may not be on the *mst* of the terminals. In other words local *mst* of the three terminals of a triangle may not be a subset of the global *mst* of all terminals. Due to this fact, selecting this triangle does not minimize the total RN count, but increases average node degree. In IO-DT, this issue is avoided since at each iteration the *mst-weight* of a triangle calculated using global *mst* of terminals.

Figure 7 compares the performance of the algorithms in terms of the expected path length. To calculate expected path length, the shortest RN-based paths between all pairs of terminals are computed using Floyd-Warshall algorithm, and the average of these path is reported. The results in Figure 7 indicate that IO-DT and FeSTA yields shorter expected path lengths than SMST and SMT-MSP 3-app. This because deploying RNs at the DFPs of triangles reduces path length within the triangle, which reduces overall expected length.

VI. Conclusion

In this paper, we have presented IO-DT, a novel relay node (RN) placement algorithm which aims at minimizing the required number of RNs for establishing connected topologies in wireless sensor networks. The idea behind the algorithm is to take advantage of the Discrete Fermat Point (DFP) optimization by considering groups of three non-collinear terminals (i.e. triangles). IO-DT computes the Delaunay Triangulation of terminals and iteratively checks if the DFP optimization of each triangle provides reduction in total number of RNs as compared

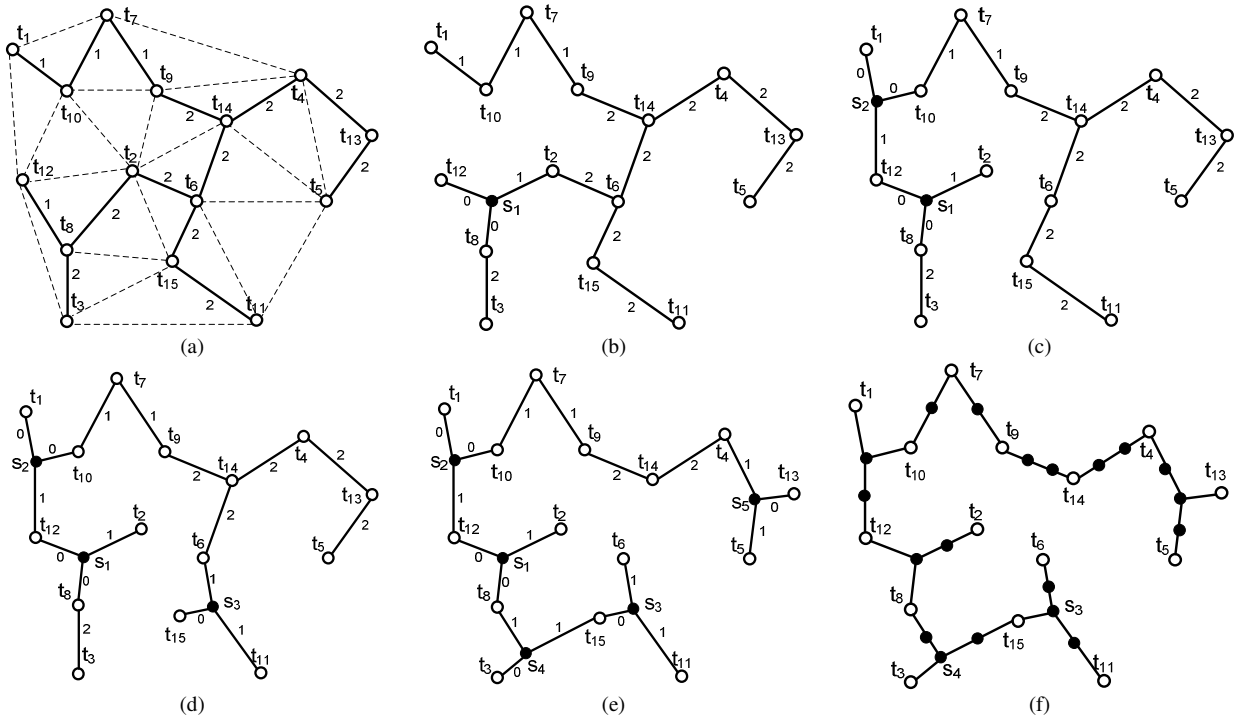

Figure 4. A Step-by-step illustration of how IO-DT algorithm works. Terminals and RNs are represented by hallow and solid nodes, respectively.

minimum spanning tree (*mst*) based solution. In case of such a gain, IO-DT places a RN at the DFP of the triangle and updates the *mst* on-the-fly. After processing all triangles, the algorithm steinerizes the *mst* edges to form a connected topology. IO-DT has been shown to have $O(n^2)$ time complexity, which is significantly better than comparable approaches. The simulation results have demonstrated that IO-DT significantly outperforms contemporary heuristics in the literature, not only by employing fewer relays but also in terms of the average node degree and expected path length, which makes formed topology more robust against traffic overload and node failure. In the future we plan to extend IO-DT to establish connectivity in 3D settings.

Acknowledgement: This work is supported by the National Science Foundation, award # CNS 1018171.

REFERENCES

[1] I. F. Akyildiz W. Su, Y. Sankarasubramaniam, E. Cayirci, "Wireless sensor networks: a survey", *Computer Networks*, Vol. 38, pp. 393-422, 2002.

[2] D. Cerpa and Estrin, "ASCENT: adaptive self-configuring sensor networks topologies," *Proc. of the INFOCOM'02*, New York, NY, June 2002

[3] M. Younis and K. Akkaya, "Strategies and Techniques for Node Placement in Wireless Sensor Networks: A Survey," *Journal* of *Ad-Hoc Network*, 6(4), pp. 621-655, 2008.

[4] X. Cheng, D.-z. Du and L. Wang and B. Xu, "Relay Sensor Placement in Wireless Sensor Networks," Wireless Networks, 14(3), pp. 347-355, 2008.

[5] E. L. Lloyd, G. Xue, "Relay Node Placement in Wireless Sensor Networks," *IEEE Trans. on Comp.*, 56(1), pp. 134-138, Jan 2007

[6] G. Lin, G. Xue, "Steiner Tree Problem with Minimum Number of Steiner Points and Bounded Edge-length," *Information Processing Letters*. 69(2), pp. 53-57, 1999.

[7] D. Chen, et al., "Approximations for Steiner Trees with Minimum Number of Steiner Points," *J. of Global Opt.*, Vol. 18, No. 1, pp. 17-33, 2000.

[8] S. Lee, M. Younis, "Optimized Relay Placement to Federate Segments in Wireless Sensor Networks" *IEEE J. on Selected Area in Comm., special issue on Mission Critical Networking*, 28(5), pp. 742 – 752. June 2010.

[9] F. Senel, M. Younis and K. Akkaya, "Bio-inspired Relay Node Placement Heuristics for Repairing Damaged Wireless Sensor Networks," in *IEEE Trans. on Vehicular Tech.*, 60(4), pp. 1835-1848, 2011.

[10] F. Senel, M. Younis, "Relay node placement in structurally damaged wireless sensor networks via triangular steiner tree approximation," *Elsevier Computer Communications*, 34(16), pp. 1932-1941, 2011.

[11] F. Aurenhammer, "Voronoi diagrams - a study of a fundamental geometric data structure," *ACM Computing Surveys*, 23(3), pp. 345-405, 1991.

[12] T. Cormen, C. Leiserson, R. Rivest and C. Stein; Introduction to Algorithms (2nd Ed); *MIT Press and McGraw-Hill*, 2001.

[13] M. de Berg, O. Cheong, M. van Kreveld, M. Overmars, "Computational Geometry: Algorithms and Applications (3rd Ed.)", *Springer*, 2008

[14] N. Kazarinoff, Geometric Inequalities, New York: Random House, 1961

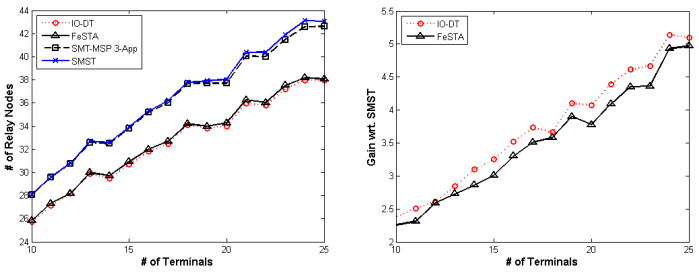

Figure 5:(a) Comparison of RN count under varying # of terminals (b) is the gain with respect to SMST.

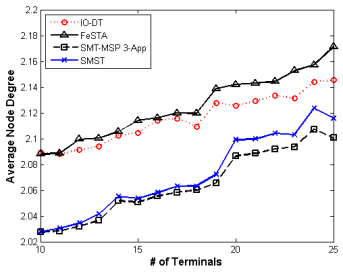

Figure 6: Comparing average node degree for varying # of terminals

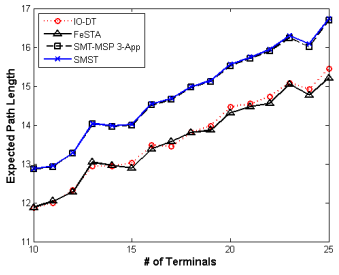

Figure 7: Comparing expected path length for varying # of terminals