On Optimal Connectivity Restoration in Segmented Sensor Networks

Myounggyu Won, Radu Stoleru, Harsha Chenji, and Wei Zhang

Department of Computer Science and Engineering, Texas A&M University, USA
{mgwon,stoleru,cjh,wzhang}@cse.tamu.edu

Abstract. This paper investigates the optimal connectivity restoration in segmented sensor networks, where mobile/relay nodes are optimally placed to form "bridges" among segments, such that both the average path length from nodes to the sink and the number of mobile nodes used are minimized. We formulate the optimal connectivity restoration as a multi-objective optimization problem and develop centralized and distributed algorithms for solving it. Given global network topology information, our centralized algorithm (i.e., Cut Restoration Genetic Algorithm or CR-GA) produces a Pareto Optimal set consisting of multiple non-dominated solutions. For scenarios where the global network topology is unknown (e.g., due to unexpected network segmentation) we develop a Distributed Connectivity Restoration algorithm (i.e., DCR). DCR restores network connectivity with lower overhead (when compared with CR-GA), at the cost of a suboptimal solution (i.e., the average path length and/or number of mobiles used). Through theoretical analysis, we prove that the worst case performance of DCR is bounded. We also show the effectiveness of our solutions through extensive simulations and a proof-of-concept system implementation and evaluation.

1 Introduction

As the cost and form factor of wireless sensor nodes shrink, we envision significant growth in the demand for *enterprise-scale* wireless sensor networks (WSNs). An enterprise-scale WSN consists of disconnected subnetworks called segments, each serving its own purpose. One example application is an enterprise-scale WSN for disaster management [1], in which one sensor subnetwork identifies victims under a rubble pile, while another subnetwork monitors the stability of a damaged building. An enterprise-scale WSN may also appear in typical WSN applications. An example is a volcano monitoring application. Since it is difficult to cover the entire area of a target mountain with nodes, a plausible design option is to deploy a number of disconnected sub-sensor networks in only critical regions. To enable a system-wide analysis, *data generated in each subnetwork must be efficiently transmitted* to a remote base station. Consequently, mechanisms for optimally connecting segments are of paramount importance for enterprise-scale WSNs.

Besides segmentation in enterprise-scale WSNs because of sparse deployments, networks can often be unexpectedly segmented if many sensors become disabled.

P. Demeester, I. Moerman, and A. Terzis (Eds.): EWSN 2013, LNCS 7772, pp. 131–148, 2013.

For example, unexpected network segmentation may occur when hostile users destroy sensors; when parts of the network are destroyed after a disaster, or even when environmental factors, such as wind, may arbitrarily relocate/disable sensors. It is important that the connectivity of these segmented networks must be immediately restored for correct operation.

Proactive protocols for connectivity restoration of a segmented sensor network have recently received attention [2][3]. These protocols use more powerful nodes, called mobile/relay nodes, to build "bridges" among segments, so that the network becomes connected. These mobile nodes can be of various forms – simple wifi switches, or devices that can even fly [4]. Paying attention to the cost of mobile nodes, these schemes have focused on minimizing the number of mobile nodes. However, building bridges with the minimum number of mobile nodes may lead to suboptimal routing paths between nodes and the sink (i.e., the path length). In fact, *bridges must be carefully placed by considering several aspects of a segmented network – the sizes and shapes of segments, and even possible holes in segments.*

Two examples depicted in Figures 1(a) and 1(b) show how the geometric information of segments, and holes in segments affect the solution of connectivity restoration obtained based on the minimum number of mobile nodes, respectively. For ease of presentation, we denote static nodes by SNs, and mobile nodes by MNs hereafter. If we are to minimize the number of MNs, a single MN (denoted by triangle p) can be deployed, as shown in Figure 1(a). In this case, the average hop count for all SNs in segment s_1 to reach the sink is 9.5. However, if we connect segments s_1 and s_2 through a bridge consisting of two MNs, denoted by triangles q and r, the average hop count to reach the sink is reduced to 3.5, at the cost of one more MN. Furthermore, existing connectivity restoration schemes do not consider possible holes in a network, which may negatively influence the average hop count. Figure 1(b) illustrates an example. A connectivity restoration scheme based on the minimum number of MNs will place a single MN denoted by triangle p. A notable fact is that some packets may have to unnecessarily travel along the perimeter the hole. However, by deploying two more MNs, denoted by triangles q and r, the average hop count can be reduced (i.e., packets can now be routed over the shortcut, to reach MN p).

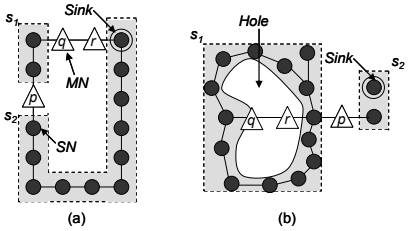

Fig. 1. The effects of (a) segment shape, and (b) holes on connectivity restoration

Additionally, protocols for connectivity restoration must be able to cope with unexpected network segmentation. More precisely, such protocols must provide mechanisms to autonomously identify network segmentation, abstract the information about segments and utilize it for optimal connectivity restoration. State-of-art protocols [2][3] do not offer such mechanisms.

In this paper, we propose algorithms and protocols designed to address the above issues. First, we define a problem called the Optimal Connectivity Restoration

Problem (OCRP) for a segmented WSN. OCRP minimizes both the number of deployed mobiles and the average path length from nodes to the sink such that the connectivity of the segmented network is restored. This problem is formulated as a multi-objective optimization problem. Based on the observation that the problem is NP-Hard, for solving it, we propose a centralized heuristic algorithm called the Connectivity Restoration Genetic Algorithm (CR-GA). The algorithm is designed for fast convergence towards the Pareto Optimal set by using a novel scheme for efficiently generating initial solutions, fast evaluation of solution validity (based on the concept of *virtual sensor*) and a reduction of the solution search space. Furthermore, in order to handle scenarios when the global network topology, i.e., the locations of nodes and their neighbors, is not known (e.g., when a network is unexpectedly segmented) we propose a Distributed Connectivity Restoration (DCR) algorithm. DCR autonomously detects network segmentation and establishes bridges to an adjacent segment without relying on the global topology. The distributed algorithm has lower computation overhead than CR-GA, at the cost of a suboptimal solution, i.e., longer average path length from nodes to the sink and/or more mobile nodes used – through a theoretical analysis, we demonstrate that DCR has a bounded worst case performance, when compared with the globally optimal solution. Lastly, we demonstrate the efficiency and feasibility of proposed solutions through extensive simulations and a proof-of-concept system implementation, respectively.

2 Related Work

Relay node placement: The *relay node placement problem* (RNP) determines where to deploy relay nodes, RNs in short, in order to achieve various objectives. These objectives include providing *connectivity* [5][6], *fault tolerance* [7][8][9], and *network lifetime* [10][11][12].

Lin and Xue [13] proved the hardness of the relay node placement problem *for connectivity* and proposed 5-approximation algorithm. Cheng et al. [6] proposed a faster randomized 2.5-approximation algorithm. Lloyd and Xue [5] then studied a more general problem with $R \geq r$, where R is the communication radius of RNs, and r is the communication radius of sensors. These algorithms, however, focus only on minimizing the number of relay nodes.

Some prior work pursued *fault tolerance* by ensuring that a given network is k-connected after deploying RNs [7][8][9]. Bredin et al. [8] presented an $\mathcal{O}(1)$-approximation algorithm for $k \geq 2$. Kashyap et al. [7] studied the fault tolerance with $k = 2$ and proposed 10-approximation algorithm. For more general case with $R \geq r$, Zhang et al. [9] proposed 14-approximation algorithm when $k = 2$.

Some researchers [10][11][12] focused on improving the *network lifetime* by deploying RNs. Hou et al. [10] jointly considered the energy provisioning and relay node placement with the objective of prolonging network lifetime. Wang et al. [11] studied the performance of dense WSNs when RNs are mobile. They showed that, with one mobile RN, the network lifetime can be increased by up to a factor of four. Wang et al. [12] considered the case with varying traffic, and

provided an algorithm to deploy RNs such that the network lifetime is maximized with traffic considerations. These algorithms, however, do not consider a disconnected (segmented) network.

Segmented WSNs: Abbasi et al. [14] proposed two decentralized algorithms for solving the connectivity restoration problem caused by *single node failure*. The algorithm coordinates the movement of mobile nodes in a cascading manner with the objective of minimizing the distance moved. Several work proposed to restore the connectivity of a segmented network caused by *multiple nodes' failure*. Almasaeid and Kamal [15] designed a scheme that models the movement of a mobile agent to make a segmented network connected over time. However, it is infeasible to assume that mobile nodes continuously move, because mobility consumes significant energy. Lee and Younis [2][3] considered the problem of federating disjoint segments. Especially, they focused on minimizing the number of relay nodes required to restore the connectivity. Noting that the connectivity-restoration problem is NP-hard, they provided a heuristic algorithm. Senel et al. [16] tackled the same problem by establishing a bio-inspired spider-web topology. However, these schemes focus only on minimizing the number of relay nodes.

3 System Model and Problem Formulation

We consider a disconnected wireless sensor network consisting of a set of segments denoted by $\mathcal{S} = \{s_1, s_2, ..., s_n\}$. Each segment may have holes (defined as regions without deployed nodes). In a segmented network, there are two types of deployed nodes: static nodes (SNs) denoted by the set $\mathcal{X} = \{SN_1, SN_2, ..., SN_N\}$, and mobile nodes (MNs) denoted by the set $\mathcal{Y} = \{MN_1, MN_2, ..., MN_M\}$. We assume that each node knows its location. The MNs are uniformly distributed in all segments. As we will clarify in Section 5, *we uniformly distribute MNs in segments, so that the WSN can autonomously cope with unexpected network segmentation.* Considering deployed SNs and MNs, we represent our segmented network as a HCG (Hybrid Communication Graph), formally defined as follows:

Definition 1. *A hybrid communication graph* $\mathsf{HCG}(r, \mathcal{X}, \mathcal{Y})$ *is an undirected graph with vertices* $\mathcal{X} \bigcup \mathcal{Y}$*, and edges defined as follows. Edge* e_{xy}*, where* $x, y \in \mathcal{X} \bigcup \mathcal{Y}$*, exists if and only if* $d(x, y) < \mathcal{R}$*, where* $d(x, y)$ *is the Euclidean distance between nodes* x *and* y*, and* $\mathcal{R}$ *is the communication range of a node.* ■

Each SN_i periodically senses the area of interest. Sensed data from each sensor is transmitted to the sink through the shortest path. We denote by P_i the path from SN_i to the sink and by $|P_i|$ the length of path P_i. We assume that MNs have significantly higher energy in comparison with SNs. Having defined our system model, we now formally describe the Optimal Connectivity Restoration Problem (OCRP), as follows:

Definition 2. *Given a set of SNs $\mathcal{X}$ and a set of segments $\mathcal{S}$ with holes,* OCRP *places a set of mobiles $\overline{\mathcal{Y}}$ ($\overline{\mathcal{Y}} \subseteq \mathcal{Y}$) satisfying the following three conditions: 1) $\frac{\sum_{i \in \mathcal{X}} |P_i|}{|\mathcal{X}|}$ (i.e., the average path length of all SNs) is minimized; 2) $|\overline{\mathcal{Y}}|$ is minimized; and 3) induced* HCG *is connected.* ■

In particular, the second condition of OCRP ensures the robustness against unexpected network segmentation; more specifically, by keeping more spare MNs ($i.e., \mathcal{Y} - \overline{\mathcal{Y}}$) uniformly distributed in segments, we improve the chance of autonomous network connectivity restoration (as it will be described in Section 5).

We discretize the problem by dividing the network into grid regions, where each grid is a square with side $\frac{\mathcal{R}}{2\sqrt{2}}$ ensuring that a MN in a grid can reach MNs in neighboring grids. Grids can be created by pre-computing a rectangular region that wraps a target area and dividing the rectangular region. Each node then easily determines in which grid it is located based on its location. Now the OCRP problem is to decide the grid regions where MNs will be deployed. This decision is represented by a binary variable y_{ij}, where $y_{ij} = 1$ means a MN is placed and $y_{ij} = 0$ means no MN is placed, on the grid located at (i, j). OCRP is then formulated as a multi-objective optimization problem as shown in Figure 2. The first constraint ensures network connectivity, and second and third constraints specify the ranges of variables. Finding the minimum number of MNs for restoring network connectivity is NP-Hard [2]. Hence, OCRP is NP-Hard.

$$\text{Minimize} \quad \left[\frac{\sum_{i \in \mathcal{X}} |P_i|}{|\mathcal{X}|}, \sum_{i,j} y_{ij} \right]. \tag{1}$$

$$\text{HCG is connected.} \tag{1}$$

$$y_{ij} \in \{0, 1\}. \tag{2}$$

$$\sum_{i,j} y_{ij} \leq M. \tag{3}$$

Fig. 2. OCRP problem

4 Centralized Connectivity Restoration

In this section we present a centralized algorithm, called Connectivity Restoration Genetic Algorithm (CR-GA), for solving OCRP. Given global topology information, CR-GA finds a near-optimal set of locations for MNs. Genetic algorithms are well suited for solving multi-objective optimization problems, because they can find a set of non-dominated solutions in parallel by maintaining a population of solutions [17] and they can efficiently solve NP-Hard problems [18]. Since our problem is an NP-Hard multi-objective optimization problem, we propose a genetic algorithm called CR-GA. CR-GA is designed for fast convergence to a close-to-optimal solution and uses a novel initial solution generation scheme, a virtual sensor-based solution evaluation scheme, and solution search space limitation.

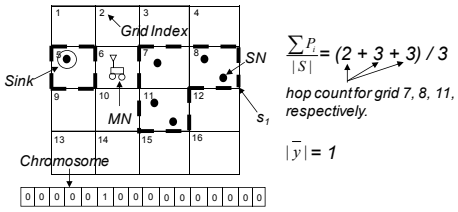

Fig. 3. A representation of a chromosome

Initial Population: Genetic algorithms represent solutions to given problems as chromosomes. A chromosome is encoded as a bit string. In our problem, each bit represents a grid in the network. A bit is set to 1 when a MN is placed in the corresponding grid; otherwise, the bit is set to 0. Given global topology information, CR-GA computes the average path length and the number of used MNs for each chromosome (i.e., chromosome's fitness or solution's optimality). However, computing the shortest paths for all SNs for each chromosome to obtain the average path length is computationally intensive. CR-GA thus uses an optional scheme for reducing the computation overhead, when nodes are relatively uniformly distributed. Consider Figure 3, which shows two segments (one containing the sink, and the other one containing five SNs) and a MN connecting the two segments. *CR-GA represents the SNs in each grid as a virtual SN at the center of the grid.* CR-GA then calculates the average path length by considering the shortest paths only for the virtual SNs in the grid network.

Having explained how the chromosome is constructed and how it's fitness is evaluated, we introduce a scheme for generating initial population of k chromosomes, where k is a system parameter. Producing *high-quality, yet diverse*, initial population is critical for fast convergence. We propose a scheme which consists of two steps. In the first step, we randomly choose a point from each segment. In the second step, we select $k_1 \in \mathbb{N}$, a parameter, and divide the 2π angle around the sink into $2\pi/k_1$ sets. Figure 4(a) shows an example with $k_1 = 4$, where different polygons represent segments. We then apply a heuristic Minimum Steiner Tree algorithm for each subregion. The first step of the scheme ensures diversity, i.e., diverse bridge locations are considered; the second step of the scheme aims to obtain high-quality initial population, i.e., the average path length from the randomly selected points to the sink are locally minimized in subregions. Figure 4(b) shows the results as a tree. We then set the bits of a chromosome corresponding to the grids intersecting with the resulting tree, if the grids are either outside segments, or inside holes in segments. We repeat the above process k times, obtaining k chromosomes – our initial population.

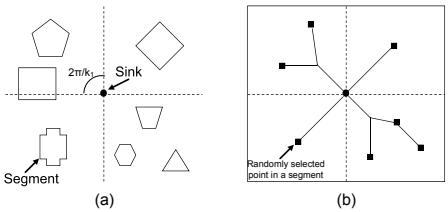

Fig. 4. (a) Initial population generation; and (b) Generated bridges

Evolution and Correction: A sequence of evolutionary processes – selection, crossover, and mutation – are applied to the initial population to produce a higher quality population. We apply the

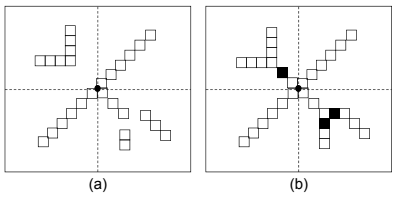

Fig. 5. Correction of a chromosome

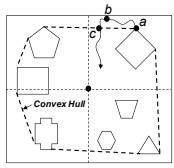

Fig. 6. Search space limitation

well-known rank-based selection algorithm [17] to implement the selection; more specifically, we rank each chromosome based on the number of dominations, e.g., if a chromosome is dominated by three chromosomes (i.e., both the number of used mobile nodes and the average path length are smaller than the three chromosomes), its rank is 3, and chromosomes with rank 0 are called the *non-dominated* chromosomes. We sort all k chromosomes in increasing rank order and select the first half. After the selection process, we randomly choose two chromosomes, say p_1 and p_2, from the selected chromosomes to perform a crossover operation. We select a position uniformly at random in a chromosome, say r. We then build a new chromosome by taking the first r bits from p_1, and the remaining bits from p_2. We repeat this operation $\frac{k}{2}$ times, creating a new set of k chromosomes. We then perform the mutation for the generated chromosomes, where we randomly select k_2 bits and switch them. After evolutionary processes are applied, some chromosomes might not satisfy our constraints. As shown in Figure 5(a), some segments are not connected to the sink. To address this problem, we first identify disconnected grids. For each such grid, we find the closest disconnected grid and connect them. Figure 5(b) shows the chromosome after the patching process. This evolution and correction process iterates until the set of non-dominated solutions converges, e.g., the algorithm stops when the set does not change for k_3 consecutive iterations.

Search Space Limitation: In order to reduce the convergence time of our algorithm, we propose to limit the search space. More precisely, we consider the placement of MNs only within the convex hull of all segments (see Figure 6 for an example) based on the theorem (we omit the proof due to space limit):

Theorem 1. *The optimal solution does not place MNs outside the convex hull of network segments.*

As described, if information about global topology is given, CR-GA obtains a set of non-dominated solutions for OCRP. However, such information may not be available, especially when a network is unexpectedly segmented due to, for example, a large number of disabled sensors by hostile users. The following section describes distributed heuristic algorithms that allow for autonomous connectivity restoration.

5 Distributed Connectivity Restoration

This section presents a distributed heuristic algorithm called the Distributed Connectivity Restoration (DCR) algorithm. The DCR algorithm establishes locally optimal bridge(s) between two adjacent segments without considering all segments in a network; thus, DCR has lower overhead (when compared with CR-GA), at the cost of a suboptimal solution (possibly longer paths from nodes to the sink and/or more MNs used), allowing any MN to compute the solution for OCRP. We first describe an overview of the algorithm.

The DCR algorithm consists of mainly three phases. In the first phase, nodes autonomously detect network segmentation and find the *boundary information* of the segment they belong to. The second phase delivers the boundary information to an adjacent segment. Since this information can not be delivered via packet transmissions because the network is segmented, our protocol uses the concept of *ferrying* – one MN in a disconnected segment stores the boundary information and moves towards the sink until it meets an adjacent segment. It is important to observe that since mobility involves very high energy consumption, it may be difficult to move all the way to the sink, especially for large scale networks. Upon reaching an adjacent segment, the ferry performs the third phase, where it finds the locally optimal (i.e., between two adjacent segments) solution for OCRP. The following sections describe the details of each phase.

Detection and Abstraction of Segments: Nodes can detect segmentation through various methods, such as distributed network cut detection algorithms [19]. Once a node detects network segmentation, it broadcasts a control packet to nodes in the disconnected segment it belongs to. Upon receiving this control packet, nodes in the disconnected segment execute a boundary detection algorithm, e.g., [20] to find the boundary nodes of the segment. When the boundary node detection phase is finished, the boundary node with the largest ID becomes the leader. This leader node stores the locations of the boundary nodes and then broadcasts its ID to the MNs in the segment.

Movement of a Ferry: Upon receiving the ID of the leader, MNs inform the leader of their remaining energy. The leader then selects a MN with the largest remaining energy and sends the locations of the boundary nodes to the selected MN. The selected MN, after receiving this information, starts the ferrying process, by traveling towards the sink until it meets an adjacent segment.

When a ferry reaches an adjacent segment, it checks *the state of the segment* – A segment's state is *disconnected* when all nodes in the segment are disconnected from the sink; otherwise, *connected*. If the state is *connected*, then the ferry executes the third phase of the DCR algorithm, which finds a locally optimal set of locations for MNs, that connects the two adjacent segments. If the state is *disconnected*, the ferry waits until the state changes to *connected*.

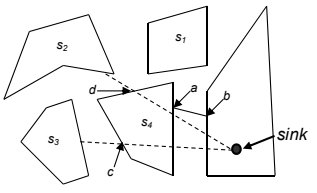

Fig. 7. An example of ferry movement in DCR

Consider Figure 7 for an example. Assume that network segmentation resulted in four segments denoted by $\{s_1, s_2, s_3, s_4\}$. Assume that s_2 first sends a ferry along the dotted line towards the sink. This ferry meets a node at point d and checks the state of segment s_4, which is *disconnected*, because it is not yet connected to the segment containing the sink. Thus, this ferry waits until the state changes to *connected*. Next, assume that segment s_4 sends a ferry.

This ferry reaches the segment containing the sink, and decides the location of bridge $\overline{ab}$. The state of segment s_4 changes to *connected*; and the waiting ferry sent from segment s_2 now builds a locally optimal bridge by running the third phase of the DCR algorithm, described in the following section.

Computation of Locally Optimal Solution: This section explains the details of the third phase, summarized in Algorithm 1. The computation of a locally optimal solution involves three major steps: 1) candidate grids selection; 2) bridge placement on holes; and 3) bridge selection.

Algorithm 1 DCR: code for ferry f

```
1: if s_c reached then
2:     // Step 1; VisEdge(s_c, s_d): Return grids on
   visible edges of s_c and s_d.
3:     {V_c, V_d} ← VisEdge(s_c, s_d).
4:     // Step 2
5:     for each g_d ∈ V_d, compute h(g_d).
6:     if s_c is connected then
7:         // Step 3
8:         for each (g_d, g_c) pair, g_d ∈ V_d, g_c ∈ V_c,
9:           compute (nm, pl).
10:        for each nm, find pl_min.
11:        for each nm, compute μ.
12:        find (g_d, g_c, h(g_d)) s.t. μ is maximized.
13:    else
14:        wait until s_2 is connected.
```

We are given two adjacent segments: one in a *disconnected* state denoted by s_d, and the other one in a *connected* state denoted by s_c. A ferry sees a network as a set of grid regions, as explained in Section 3 (See Figure 8). Define a set of grids that are contained in segments s_d and s_c by G_d and G_s, respectively. In particular, one grid in s_c is called the *destination grid* and denoted by t. The destination grid is either a grid containing the sink when the sink is in s_c, or a grid containing the entry point of a bridge that connects to other connected segment when the sink is not in s_c.

The first step of the algorithm is to find a set of *target grids* for adjacent segments s_c and s_d. Given s_d and s_c, we first find edges visible to each other. Consider Figure 8 for an example. The two segments are represented by triangles $\triangle abc$ and $\triangle def$. In this example, the visible edges are $\{\overline{bc}, \overline{ca}\}$ for segment s_c, and $\{\overline{de}\}$ for segment s_d. Target grids are the grids that are located on the visible edges of the two segments. We denote the set of target grids for s_d by V_d, and for s_c by V_c.

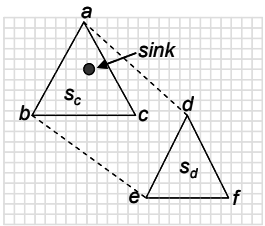

Fig. 8. An illustration of visible edges

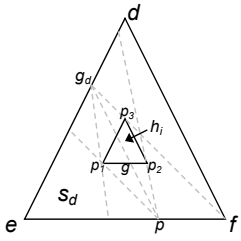

Fig. 9. Bridge placement on holes

In the second step, the algorithm places bridges over holes in the segment. For each $g_d \in V_d$ and each hole h_i, invisible edges of hole h_i from g_d are identified. See Figure 9 for an example. By drawing two tangent lines from g_d to hole h_i, we can find that line segments $\overline{p_1p_2}$ and $\overline{p_2p_3}$ are the invisible edges. Define the set of grids on the invisible edges for hole h_i by V_{h_i}. Now for each $g \in V_{h_i}$, we consider a line starting from g_d, passing through g. We denote the farthest intersection with the edges of segment s_d by p as shown in Figure 9 (there may exist multiple such intersections). If line segment $\overline{gp}$ intersects other holes, h_i is not considered. We then consider two tangent lines from p to hole h_i. These two tangent lines, with the edges of hole h_i and possibly with the edges of segment s_d, create a region A_g, which represents the number of grids that will contribute to the reduction of the average path length by placing the bridge on that hole. For example, the two tangent lines from p (i.e., $\overrightarrow{pp_1}$ and $\overrightarrow{pp_2}$) create a region $A_g = \{p, p_1, p_2\}$. We then select g' from V_{h_i} such that A_g is maximized. We denote such grid g' for each $g_d (\in V_d)$ by $h(g_d)$.

nm	pl
4	6
4	7
5	4
6	4
6	3

⇨

nm	pl_{min}	μ	
4	6	-	
5	4	2	★
6	3	1.5	

Fig. 10. An example for marginal utility computation

In the third step, we consider line segment $\overline{g_dg_c}$ for each $g_d (\in V_d)$ and $g_c (\in V_c)$ as a bridge connecting two adjacent segments s_d and s_c. If line segment $\overline{g_dg_c}$ intersects any of the visible edges, the line segment is not considered. Now for each pair (g_d, g_c), representing a bridge, we compute the average path length denoted by pl and the number of used MNs denoted by nm as follows: $pl = \frac{\sum_{g \in G_d}(d(g,g_d)+d(g_d,g_c)+d(g_c,t))}{|G_d|}$, where nm represents the number of grids on $\overline{g_dg_c}$. Here the term $d(p, q)$ refers to the length of the shortest path connecting p and q. In particular, for computing $d(g, g_d)$, we consider two cases: (1) placing bridges on holes according to pre-computed $h(g_d)$ (i.e., placing MNs on line segment $\overline{g_dh(g_d)}$ that is within hole(s)); (2) not placing bridges on holes. After computing pl and nm for all (g_d, g_c) pairs, we have a set of (nm, pl) pairs (the table on the left-hand side of Figure 10 gives an example). Different from CR-GA (which produces a Pareto Frontier), due to the lack of computational capabilities, the DCR algorithm chooses one pair that maximizes the marginal utility. Marginal utility shows the incremental contribution of each added MN to the average path length. Choosing the solution with maximum marginal utility thus leads to the most economic decision. For example, for each nm, we first find the minimum pl, denoted by pl_{min}. The table on the right-hand side shows pairs (mn, pl_{min}). For each pair (mn, pl_{min}), we then compute the marginal utility, denoted by μ, as follows: $\mu = \frac{pl - pl_{min}}{nm - nm_{min}}$. In our example, from all the pairs (mn, pl_{min}), our DCR algorithm selects $(5, 4)$, the most economic decision.

6 Algorithms Analysis

As presented in Section 5, the DCR algorithm finds a locally optimal solution for *two adjacent* segments. If we consider *all* segments in a network, however,

a simple combination of locally optimal solutions may not guarantee optimal performance. Thus, in this section, we address the following research question: *how much worse is the performance of the DCR algorithm, when compared with the centralized CR-GA?*

For answering the question, we consider a network with a circular shape centered at the sink. The radius of the network is r, where $r \gg 1$. As mentioned in Section 3, there are n segments in the network. Considering a very large network (i.e., $r \gg 1$) for deriving worst-case bounds, segments and MNs are represented as points in the network.

We first identify the worst case scenario for the DCR algorithm and analyze how much worse it is, when compared with the globally optimal solution. The following lemma proves the worst-case average path length and number of used MNs for the DCR algorithm.

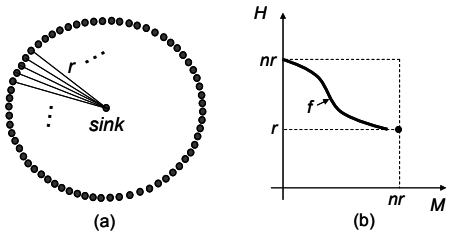

Fig. 11. (a) Worst case scenario; (b) The domain and codomain of Pareto Frontier

Lemma 1. *The DCR algorithm shows the worst performance when n segments are uniformly positioned on the circumference of the network.*

Proof. Figure 11(a) shows the worst case scenario. It is easy to note that, for this scenario, the average path length is r, and the number of used MNs is nr. Assume by contradiction that there is a scenario with either the average path length greater than r, or the number of used MNs greater than Nr. In order to have the average path length greater than r, there must be at least one segment with its path length greater than r. Since, for this scenario, the DCR algorithm places a bridge as a straight line towards the sink, the path length cannot be greater than r, i.e., a contradiction. Similarly, in order to have the number of used MNs greater than nr, we must have at least one bridge with more than r MNs; a bridge with more than r MNs is no longer a straight line. ■

We define a two dimensional Cartesian coordinate system with the domain (i.e., X-axis, and denoted by M) representing the number of used MNs and the codomain (i.e., Y-axis, and denoted by H) representing the average path length. Then, the Pareto frontier, i.e., *the solution of CR-GA*, is a curve represented by a function $f : M \rightarrow H$. We are interested in the maximum distance between any point on the curve (a CR-GA solution) and the point that represents the worst-case DCR solution (i.e., as obtained by Lemma 1). We call this distance *performance gap*. The main idea for obtaining the maximum performance gap is to bound the domain and codomain of function f. The following two lemmas find the bounds for the domain and codomain of function f, respectively.

Lemma 2. *The domain M of f is bounded by $0 < M \leq nr$.*

Proof. Since the path length from any segment to the sink for CR-GA is greater or equal to r, the average path length for CR-GA is greater or equal to r, i.e., $H \geq r$. Assume by contradiction that $M > nr$. Then, we have $M > nr$ and $H \geq r$, which means that any solution for CR-GA (i.e., points on the curve f) is worse than the solution obtained by the DCR algorithm (i.e., both the number of used MNs and average path length are greater than the DCR algorithm). ■

Lemma 3. *The codomain M of f is bounded by $r \leq H \leq nr$.*

Proof. By Lemma 2, we know that $H \geq r$. Since the domain of f is bounded by nr, the average path length is maximized when all paths from segments are aggregated into a single path of length nr. ■

Theorem 2. *The performance gap is bounded by* $nr\sqrt{1 + (\frac{n-1}{n})^2}$

Proof. Based on Lemma 2 and Lemma 3, the Pareto optimal curve f can be one of any possible curves defined in $0 < M \leq nr$ and $r < H \leq nr$, as shown in Figure 11(b). Thus, the maximum distance from point (nr, r) to curve f is the distance from point (nr, r) to point $(0, nr)$, which is $nr\sqrt{1 + (\frac{n-1}{n})^2}$. ■

Theorem 2 shows that the performance of DCR degrades asymptotically linearly with the number of segments n and the network diameter r. The interpretation of this result is that, since DCR builds bridges based on adjacent segments without taking into account all segments in the network, the overall performance degrades when there are more segments. Besides, if the diameter of a network is large, the distances between segments and the sink are more likely to be longer; thus, the performance degrades, because a better solution may be found by aggregating such long paths. However, this result also proves that the performance does not degrade arbitrarily, only linearly with the number of segments and the network diameter.

7 Simulation Results

For performance evaluation, we consider a 2,000m × 2,000m area with randomly generated segments of different sizes and shapes. Sensor nodes are uniformly deployed in each segment. To account for more realistic wireless communication, we adopt the radio model [21], which defines the *degree of irregularity* (DOI) as the maximum radio range variation in the direction of radio propagation. In our experiments, the radio range of a node is 40m with DOI=0.4, resulting in a network density of approximately 8 nodes/radio range.

We implemented DCR, CR-GA, and the state-of-art cut restoration scheme called Cell-based Optimized Relay node Placement (CORP) [2] in C++. CORP is a state-of-the-art centralized heuristic algorithm for restoring network connectivity by using the fewest MNs possible. For fair performance comparison between DCR and CR-GA, we select a CR-GA solution on the Pareto Optimal set (i.e., an average path length and the corresponding number of used

MNs) with the largest marginal utility. We used the following values for CR-GA: $k_1 = 4, k_2 = 10$ % of total bits, and $k_3 = 15$. CR-GA was executed on a PC with 64bit Ubuntu, Intel Core i7 CPU, and 8 GByte memory.

For our evaluation, we measured the average path length in hops and the number of used MNs by varying several properties related to a segmented sensor network: Segment Size (SS), Number of Segments (NS), Location of Sink (LS), and Hole Size (HS). A segment was represented as a polygon. The vertices of the polygon were selected within a randomly located circle with radius SS. SS is thus used to control the size of a segment. We ensure that the area covered by a segment is at least 20% of that of a circle with radius SS. The parameter LS represents the distance between the sink and the center of the network. The default values for our parameters were: SS=200, NS=4, LS=0, HS=0.

Evaluation of CR-GA: In each iteration of CR-GA, a set of chromosomes are generated. CR-GA computes the rank of each chromosome based on the average path lengths and numbers of MNs of the set of chromosomes. Our proposed virtual sensor (VS) can significantly reduce the time to compute chromosome's rank, when nodes are uniformly distributed. To verify this, we compared the time taken by CR-GA for computing ranks when VS is used, with the time taken when VS is not used. Figure 12 presents the results. As shown, the average rank-computation time increased linearly with the number of nodes. In contrast, CR-GA that uses VS showed a constantly small average rank computation time.

Figure 13 shows the Pareto Optimal Set obtained for a network with the default setting. Each point of the graph represents a feasible solution. As the graph shows, when we can afford a large number of MNs, we can achieve a better average path length by placing more MNs; in contrast, a solution that uses a small number of MNs has a longer average path length, but the spare MNs can be used for detecting unexpected network separation, increasing robustness.

Effect of Number of Segments: In this section we investigate how the Number of Segments (NS) affects the performance of the three protocols. We varied NS from 4 to 12, while having all other parameters set to default values. Figure 14 and Figure 15 show the average path length and the number of used MNs for the three protocols, respectively. Comparing CR-GA and CORP, we observed that both used a similar number of MNs regardless of NS. However, CR-GA produced much smaller average path length up to 20%. The reason is that, while CORP tries to minimize only the number of used MNs, CR-GA minimizes both the average path length and the number of used MNs. It should be noted that CR-GA involves higher computation than CORP, because it is based on a genetic algorithm. However, what really important is higher network performance achieved by optimizing both the number of MNs and average path length, because the computation of CR-GA is performed only once in a powerful device. Comparing CR-GA and DCR, we observe that DCR produced a slightly smaller average path length, i.e., about 4%. The reason is that DCR builds multiple bridges towards the sink without merging them. The smaller average path length, however, required a significantly higher (about 35%) number

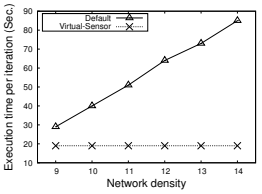

Fig. 12. Computation speed w/ and w/o VS

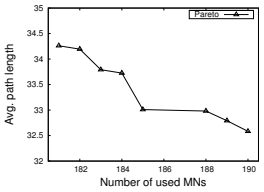

Fig. 13. Pareto Optimal set for default settings

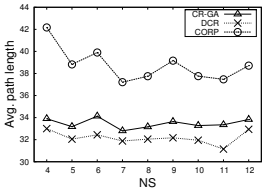

Fig. 14. Effect of NS on average path length

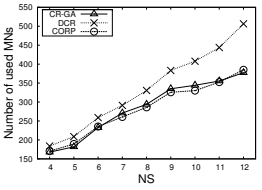

Fig. 15. Effect of NS on number of mobiles

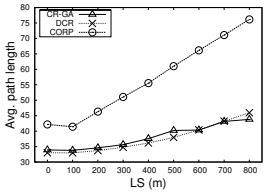

Fig. 16. Effect of LS on average path length

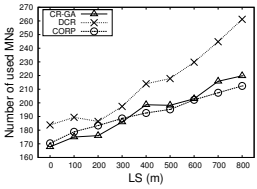

Fig. 17. Effect of LS on number of mobiles

of used MNs. An interesting observation was that the difference in the number of MNs used by the two protocols increased as NS increased. We believe this result confirms our theoretical analysis which shows that the performance gap between CR-GA and DCR increases with the number of segments.

Effect of Sink Location: In this section we investigate how the Location of the Sink (LS) affects the performance of the three protocols. We select sink locations to be LS meters away from the center of the network, towards one corner of the network. We set all other parameters to their default values. Figure 16 and Figure 17 show the average path length and the number of used MNs for the three protocols, respectively. One immediate observation was that the average path length and the number of used MNs for all three protocols increased as we increased LS. This is simply because when the sink is located far from the center of the network, a packet must travel longer distance to reach it. Comparing CR-GA and CORP, we found that the two protocols used a similar number of MNs. However, CR-GA produced much smaller average path lengths up to 75%. An interesting observation was that CORP's performance was more significantly affected by LS, than CR-GA; more precisely, while the average path length of CR-GA gradually increased with increasing LS, the average path length of CORP increased more steeply. The reason is that CORP is designed to build bridges towards the center of the network. Next, we compared CR-GA with DCR. While they achieved similar average path lengths, DCR used more MNs up to 20%. The reason is the same as above, namely that DCR builds bridges towards the sink without merging them. In fact, CR-GA finds a balance between the number of MNs and the average path length by appropriately merging bridges. An interesting observation was that the difference between the number of MNs used by DCR and CR-GA increased with increasing LS. The reason is that,

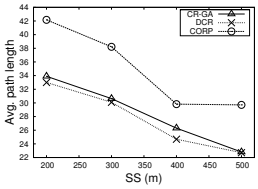

Fig. 18. Effect of SS on average path length

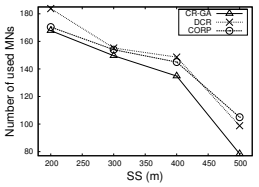

Fig. 19. Effect of SS on number of mobiles

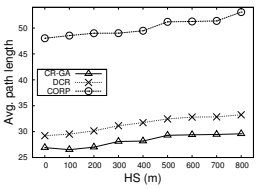

Fig. 20. Effect of HS on average path length

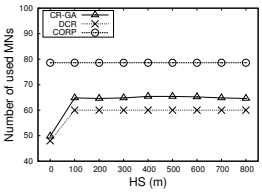

Fig. 21. Effect of HS on number of mobiles

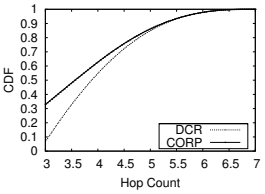

Fig. 22. CDF of hop count

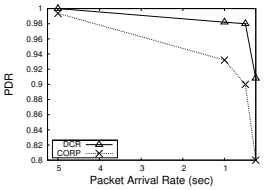

Fig. 23. Packet delivery ratio

since DCR favors building a direct bridge towards the sink, if the sink is far, it requires more MNs to connect the segment to the sink.

Effect of Segment Size: In this section we investigate the effect of Segment Size (SS). Figure 18 and Figure 19 depict the average path length and the number of used MNs for the three protocols, respectively. An immediate observation was that as we increased SS, both the average path length and the number of used MNs decreased for all protocols. The reason is simply that larger segments take more space in the network, leaving fewer empty spaces. Thus fewer MNs are required for connectivity restoration. Comparing CR-GA and DCR, we observed that both showed a similar performance in terms of the average path length. The difference comes from the number of used MNs, as CR-GA used about 10% fewer MNs on average, a relatively small improvement. This result is not surprising since the number of segments was the default value of 4. With few segments, DCR builds bridges quite well. Second, we compared CR-GA and CORP. We observed that the average path length of CORP was worse than CR-GA by up to 25%. The reason is that CORP does not consider the average path length. An interesting observation was that the difference in the number of MNs between CORP and CR-GA became larger as SS increased. We believe the reason is that CORP chooses a *representative node* (i.e., the starting point of bridges, selected for each segment) without considering the size and shape of segments in a network. Therefore, the impact of sub-optimally selected representative nodes becomes greater as the segment size becomes larger (i.e., more possible locations for selecting representative nodes).

Effect of Hole Size: To evaluate the impact of holes on protocols' performance, we consider a scenario with two large rectangular-shaped segments (200m $\times$ 1,000m), and place holes of varying sizes in them. In each segment we create a bar-shaped hole, of varying heights (i.e., bars with different sizes of

100m × [0, 800]m). Figure 20 and Figure 21 depict the average path length and the number of used MNs for the three protocols, respectively. As shown in Figure 20, for all three protocols, the average path length increased with an increasing hole size. The reason is that larger holes result in longer, detoured routing paths. It should be noted that, although DCR and CR-GA place bridges on holes, depending on the locations of bridges, there are still nodes that use detoured paths, thereby showing small increases. Comparing CR-GA and DCR, we observed that CR-GA produced about 9% smaller path length by using more MNs. The reason is that, while DCR places only a single, straight-line bridge over a hole, CR-GA places multiple bridges, or even merge bridges. Comparing CR-GA and CORP, we observed that although CR-GA places MNs on a hole, CORP used more MNs for restoring the connectivity. The reason is the large sizes of the two segments used in this experiment. As we mentioned previously, CORP more likely to choose representative nodes far from the sink, when the size of a segment is large. This suboptimal selection of representative nodes results in a large number of MNs. Furthermore, since CORP does not handle holes, it has longer average path length.

8 System Evaluation

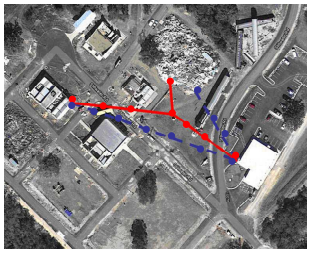

Fig. 24. A deployment area at a Disaster Training Facility

As a proof-of-concept system, we implemented our DCR algorithm in TinyOS 2.1.1 for the TelosB platform and compared it with CORP. We deployed 10 TelosB motes in each of two segments in a disaster training facility of approximately 150m by 150m, as shown in Figure 24. Routing paths from motes to the sink were obtained using CTP [22]. Motes reported events, with varying reporting rates, i.e., 250msec, 500msec, 1sec, and 5sec, by sending a packet. We manually placed TelosB motes on the computed bridges based on DCR and CORP, using TelosB motes as MNs. As Figure 24 shows, the solid line represents the bridges for CORP, and the dotted line the bridges for DCR; filled circles represent MNs. We measured the path length in hops and computed the packet delivery ratio at the sink.

Figure 22 shows the cumulative distribution function for path length (in hops) for DCR and CORP. As shown, DCR has a smaller hop count. The reason is that CORP, to reduce the number of MNs, merged two bridges, resulting in longer paths. We then compared the Packet Delivery Ratio (PDR) of DCR and CORP. Figure 23 shows the results. For both protocols, the PDR decreased as the packet arrival interval decreased. A notable observation was that the PDR of CORP more rapidly decreased. We believe that the reason is because the merged paths increased the chance of collisions and possible congestion.

9 Conclusions

This paper investigates optimal connectivity restoration in segmented WSNs, an important, largely unexplored problem. Given the global network topology, our centralized algorithm is used to restore the network connectivity such that both the average path length and number of used mobile nodes are minimized. A distributed scheme is also developed for enabling nodes to autonomously cope with unexpected network segmentation at reduced computational costs.

Acknowledgements. We thank Dr. Marco Zuniga for shepherding this paper. This work was funded in part by NSF awards 1127449, 1145858, and 0923203.

References

1. George, S.M., Zhou, W., Chenji, H., Won, M., Lee, Y., Pazarloglou, A., Stoleru, R., Barooah, P.: DistressNet: a wireless AdHoc and sensor network architecture for situation management in disaster response. IEEE Communications (2010)
2. Lee, S., Younis, M.: Optimized relay placement to federate segments in wireless sensor networks. IEEE JSAC (2010)
3. Lee, S., Younis, M.: Recovery from multiple simultaneous failures in wireless sensor networks using minimum steiner tree. JPDC (2010)
4. Pruhit, A., Sun, Z., Mokaya, F., Zhang, P.: Sensorfly: Controlled-mobile sensing platform for indoor emergency response applications. In: IPSN (2011)
5. Lloyd, E., Xue, G.: Relay node placement in wireless sensor networks. IEEE TOC (2007)
6. Cheng, X., Du, D.-Z., Wang, L., Xu, B.: Relay sensor placement in wireless sensor networks. WINET (2008)
7. Kashyap, A., Khuller, S., Shayman, M.: Relay placement for higher order connectivity in wireless sensor networks. In: INFOCOM (2006)
8. Bredin, J.L., Demaine, E.D., Hajiaghayi, M., Rus, D.: Deploying sensor networks with guaranteed capacity and fault tolerance. In: MobiHoc (2005)
9. Zhang, S.W., Xue, G., Misra: Fault-tolerant relay node placement in wireless sensor networks: Problems and algorithms. In: INFOCOM (2007)
10. Hou, Y., Shi, Y., Sherali, H., Midkiff, S.: Prolonging sensor network lifetime with energy provisioning and relay node placement. In: SECON (2005)
11. Wang, W., Srinivasan, V., Chua, K.-C.: Extending the lifetime of wireless sensor networks through mobile relays. IEEE/ACM ToN (2008)
12. Wang, F., Wang, D., Liu, J.: Traffic-aware relay node deployment: Maximizing lifetime for data collection wireless sensor networks. IEEE TPDS (2011)
13. Lin, G.-H., Xue, G.: Steiner tree problem with minimum number of steiner points and bounded edge-length. Inf. Process. Lett. (1999)
14. Abbasi, A., Younis, M., Akkaya, K.: Movement-assisted connectivity restoration in wireless sensor and actor networks. IEEE TPDS (2009)
15. Almasaeid, H.M., Kamal, A.E.: Data delivery in fragmented wireless sensor networks using mobile agents. In: MSWiM (2007)
16. Senel, F., Younis, M., Akkaya, K.: Bio-inspired relay node placement heuristics for repairing damaged wireless sensor networks. IEEE TVT (2011)

17. Fonseca, C.M., Fleming, P.J.: Genetic algorithms for multiobjective optimization: Formulation discussion and generalization. In: Proceedings of the 5th International Conference on Genetic Algorithms (1993)
18. Kim, H., Shin, K.: Asymmetry-aware real-time distributed joint resource allocation in ieee 802.22 wrans. In: INFOCOM (2010)
19. Won, M., George, M., Stoleru, R.: Towards robustness and energy efficiency of cut detection in wireless sensor networks. Elsevier Ad Hoc Networks (2011)
20. Li, F., Luo, J., Zhang, C., Xin, S., He, Y.: Unfold: Uniform fast on-line boundary detection for dynamic 3d wireless sensor networks. In: MobiHoc (2011)
21. He, T., Huang, C., Blum, B.M., Stankovic, J.A., Abdelzaher, T.: Range-free localization schemes for large scale sensor networks. In: MobiCom (2003)
22. Gnawali, O., Fonseca, R., Jamieson, K., Moss, D., Levis, P.: Collection tree protocol. In: SenSys (2009)