# Deploying Sensor Networks with Guaranteed Capacity and Fault Tolerance

Jonathan L. Bredin[†] Erik D. Demaine[‡] MohammadTaghi Hajiaghayi[‡**] Daniela Rus[‡]

[†]Department of Mathematics
Colorado College
14 East Cache la Poudre Street
Colorado Springs, CO 80903, USA

[‡]Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
32 Vassar Street
Cambridge, MA 02139, USA

## ABSTRACT

We consider the problem of deploying or repairing a sensor network to guarantee a specified level of multi-path connectivity ($k$-connectivity) between all nodes. Such a guarantee simultaneously provides fault tolerance against node failures and high capacity through multi-path routing. We design and analyze the first algorithms that place an almost-minimum number of additional sensors to augment an existing network into a $k$-connected network, for any desired parameter $k$. Our algorithms have provable guarantees on the quality of the solution. Specifically, we prove that the number of additional sensors is within a constant factor of the absolute minimum, for any fixed $k$. We have implemented greedy and distributed versions of this algorithm, and demonstrate in simulation that they produce high-quality placements for the additional sensors. We are also in the process of using our algorithms to deploy nodes in a physical sensor network using a mobile robot.

## Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems; F.2.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity—*Nonnumerical Algorithms and Problems*

## General Terms

Algorithms

## Keywords

Sensor networks, Graph theory, Simulations

## 1. INTRODUCTION

Sensor-network applications owe much of their popularity to broad and rapid deployment, frequently into hazardous environments. We use a robotic helicopter to deploy sensors to monitor outdoor environments and provide network connectivity for emergency response scenarios [1, 2]. Such rapid deployment, especially under extreme circumstances, exposes sensors to additional chance of failure and placement errors. Sensors may not be placed in exactly their desired locations because of wind or inaccurate localization. Sensors may fail from impact of deployment, fire or extreme heat, animal or vehicular accidents, malicious activity, or simply from extended use. These failures may occur upon deployment or over time after deployment: extensive operation may drain some of the nodes' power, and external factors may physically damage part of the nodes. Additionally, hazards may change devices' positions over time, possibly disconnecting the network. If any of these initial deployment errors, sensor failures, or change in sensor positions cause the network to be disconnected or lack other desired properties, we need to deploy additional sensors to repair the network.

In an example application, a network of cameras monitors the safety of a building compound. Each sensor does local image analysis to detect events such as motion within its field of view. Upon such events, sensors send images to a base station for more complex analysis such as tracking. This application relies on the network's ability to support a given amount of information flow. The application also illustrates that not all the nodes in the network require the same amount of communication. For example, the nodes along the trajectory of the tracked object will transmit more images and thus use more power to communicate. This means that their communication ability will be diminished and they may become depleted of power. In such a case, the network will have to be extended with new nodes to sustain the desired information flow.

Given the dynamic environment, it is desirable to have procedures to establish network properties, such as connectedness, in the event that multiple devices fail. We are interested in developing an algorithm that can run regularly in the background, to suggest repairs to a deployment mechanism once connectivity properties disappear. We wish for such an algorithm to minimize the number of nodes required to reestablish network properties. Such an algorithm could be used in a practical sensor network deployment in one of two modes. If the sensor network has significant redundancy,

** This work was partially done while the author was at Microsoft Research.


*MobiHoc'05,* May 25–27, 2005, Urbana-Champaign, Illinois, USA.

our algorithms can be used to select which nodes should be turned on at any point in time. This approach supports node failure; upon failures that damage the connectivity properties of the network our algorithm provides a method for selecting the wakeup and inclusion of sleeping nodes. If the sensor network does not have redundancy, our algorithm computes the locations where additional nodes need to be deployed in the network using a ground or flying robot. This results in a goal-directed approach for automated maintenance and repair of a network which optimizes the use of the powerful mobile node (e.g., the robot helicopter) tasked to do this operation by deploying additional sensors.

More specifically, to support both fault tolerance and capacity, we focus on the vertex-connectivity of the network. In the worst case, a $k$-connected network requires $k$ node failures to disconnect the network. Additionally, $k$-connectivity ensures that there exist at least $k$ node-disjoint paths between every pair of nodes in the network, providing additional bandwidth.

Given a desired value of $k$, we present and analyze a generic algorithm that determines how to establish $k$-connectivity by placing additional sensors geographically between existing pairs of sensors. Solving this problem with a minimum number of additional sensors is NP-hard [3]. Our approach is to transform the network repair problem to one of selecting the minimum-weight $k$-vertex connected subgraph of the complete graph underlying the sensor network and then applying existing graph-theoretic approximation methods. Our proposed algorithm provides a bound on the solution quality that is within a constant factor of the optimal solution, for any fixed $k$.

Due to limited communication or computational resources available to nodes, our proposed provable approximation algorithm for optimal $k$-connectivity repair may be difficult to implement on a physical sensor network. This limitation may not be significant because the algorithm can usually be run occasionally and in the background, so there is little need for a "real time" algorithm. Nonetheless, our characterization of the problem complexity shows how to further approximate the problem. We develop alternative methods for determining a low-cost $k$-connected graph that are simpler, faster, and able to run in the distributed sensor network. The modularity of our base algorithm allows us to trade computational speed for solution accuracy. In an experimental study, we implement in simulation the use of greedy and distributed algorithms and show that, in practice, the solution quality produced by these fast methods is not far from optimal.

Attaining $k$-connectivity has recently been studied in the context of power assignment, where instead of adding sensors the goal is to assign the sensors' communication power to ensure $k$-connectivity and minimize overall power consumption. This problem is also NP-hard. Ramanathan and Rosales-Hain [4] consider the special case of 2-connectivity and provide a centralized spanning-tree heuristic for minimizing the maximum transmit power in this case. Bahramgiri et al. [5] generalize the cone-based local heuristic of Wattenhofer et al. [6, 7] in order to solve the general $k$-connectivity setting. However, both of these works are heuristics and do not have provable bounds on the solution cost [8]. Lloyd et al. [9] present an 8-approximation algorithm for 2-connectivity, but they do not consider general $k$-connectivity. Hajiaghayi et al. [8] present a constant-factor approximation algorithm for $k$-connectivity for any fixed $k$. Recently, different sets of authors (see e.g. [10, 11, 12, 13]) used the notion of $k$-connectivity and the results of [5, 8] to deal with the fault-tolerance issues for static and dynamic settings.

Our problem has been considered before only in the special case $k = 1$, where the problem has applications in VLSI design and evolutionary/phylogenetic tree constructions in computational biology. See [14] for a description of these and other applications. The best approximation algorithm to our knowledge is a 5/2-approximation algorithm by Du et al. [15], again only for the case $k = 1$.

We proceed by introducing some definitions, notation, and models we will use for our algorithm. Section 3 presents an algorithm that takes the subgraph-repair problem as a modular black box to establish $k$-vertex connectivity in a network by adding new nodes geographically between existing ones. Whereas the algorithm is natural, the analysis in Section 4 providing an approximation bound is complicated. We present in Section 5 practical distributed modifications to our algorithm that work in the context of limited computational resources. Section 6 supports the heuristics through experiments comparing the performance of our simplified algorithms with our algorithm using optimal subgraph repair, and showing that our methods are superior to deployment according to additional random sampling. Finally, we conclude in Section 7 with discussion of improvements to our algorithm relying on tighter analysis to derive a polynomial-time approximation scheme.

## 2. PRELIMINARIES AND MODELS

In this paper we consider static symmetric multi-hop ad-hoc wireless networks with omni-directional fixed-power transmitters that typically arise in the context of sensor networks. This model is considered by Blough et al. [16], Calinescu et al. [17], Kirousis et al. [18], and others in their works on connectivity. The model has many practical consequences; for example, many existing routing protocols can easily be accommodated by this model, in particular because links are bidirectional. Furthermore, many of the restrictions imposed by this model can be relaxed at the cost of additional communication. We summarize the main features of the model here.

An ad-hoc wireless network consists of a set of mobile devices (e.g., sensors) equipped with radio transceivers. Each radio transmitter is assigned a power setting and an orientation that define the reception area of its transmissions. We assume that all transmitters have a common, fixed maximum power setting, and refer to that as the *communication radius*. We also assume that the transceivers are *omni-directional* in the sense that they transmit in all directions equally. Both of these assumptions are satisfied by most wireless networks.

We make the further assumptions that our networks are *static* and that all communication links are *bidirectional* or *symmetric*. In a static network, the devices are stationary. If a device moves, the topology of the network can change in ways out of our control. In the symmetric link model, if a device $u$ can receive transmissions from a device $v$, then $u$ also has enough maximum power to transit to device $v$. In practice, most wireless networks experience problems from asymmetries, but the symmetric restriction simplifies routing protocols. We assume that the nodes in our networks tune down their effective ranges to the lowest common range.

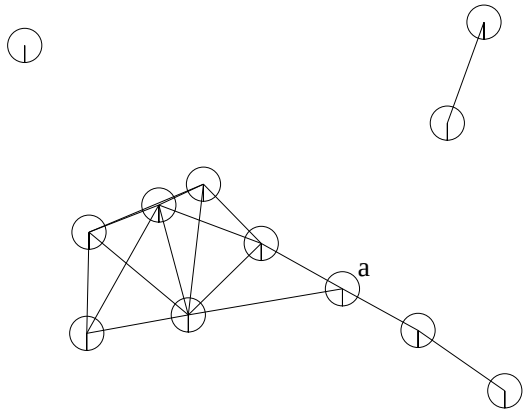


**Figure 1: An example unit-disk graph.**

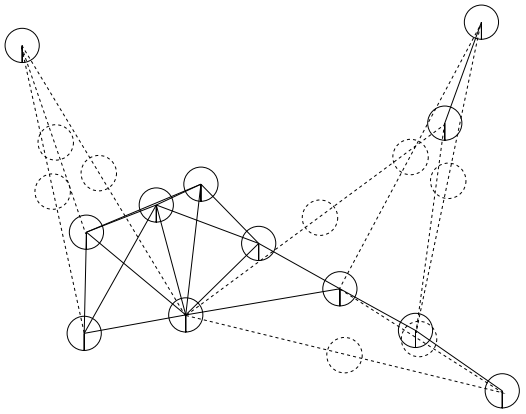

**Figure 2: The example unit-disk graph from Fig. 1 with added vertices, marked with dashed circles, to establish 3-connectivity among the original (solid) vertices. The new vertices lie closely enough to each other to ensure 3-connectedness for the entire graph. We omit drawing edges between the added vertices for clarity.**

Given these assumptions, we can model our wireless network as a *unit-disk graph*, $G = (V, E)$, where each vertex represents a device and is assigned two-dimensional coordinates. Two vertices are connected by an edge if and only if their distance is at most the communication radius. For simplicity of exposition, we normalize the coordinate assignment so that the communication radius is 1.

In this paper we suppose that the network is *multi-hop*, meaning that the mobile devices cooperate to route each others' messages. Thus we are interested in multi-node communication paths between the source and destination of a message. In anticipation of node failures resulting e.g. from power failure or power depletion of a mobile node, we are also interested in finding multiple disjoint communication paths between any source/destination pair.

In this paper we consider the following problem, given a sensor network represented as a unit-disk graph, we wish to compute and deploy the minimum number of additional devices to ensure that the resulting unit-disk graph satisfies the fault-tolerance constraint called vertex $k$-connectivity. A graph is *vertex $k$-connected* if there are at least $k$ vertex-disjoint paths connecting every pair of vertices, or equivalently, the graph remains connected when any set of at most $k-1$ vertices is removed. Hence our goal is to make the network resilient to $k$ node failures.

Fig. 1 shows an example unit-disk graph that is not 3-connected. The largest component of the graph is 1-connected as it can be separated with the removal of the vertex marked a, for example. The graph in Fig. 2 shows the same graph with additional vertices to establish 3-connectivity among vertices from the original graph.

Our problem has been considered before only in the special case $k = 1$, where the problem has also found application in VLSI design and evolutionary/phylogenetic tree constructions in computational biology. See [14] for a description of these and other applications. The problem is NP-hard even for $k = 1$ [3]. The best approximation algorithm to our knowledge is a 5/2-approximation algorithm by Du et al. [15], again only for the case $k = 1$.

An important related problem is, given a weighted complete graph $K$ and a number $k \geq 1$, to find minimum-weight $k$-vertex-connected subgraph of $K$. This problem can be viewed as analogous to our problem of $k$-fault tolerance but for wired networks. Frank and Tardos [19] and Khuller and Raghavachari [20] were among the first authors who worked on this problem. The problem is NP-hard, and there are by now several polynomial-time approximation algorithms with guaranteed performance ratios. An *$\alpha$-approximation algorithm* is a polynomial-time algorithm whose solution cost is at most $\alpha$ times the optimal solution cost. Kortsarz and Nutov [21] developed a $k$-approximation algorithm. At the heart of this algorithm is a combinatorial algorithm of Gabow [22] whose running time is $O(k^2|V|^2|E|)$ (an improvement to an algorithm of Frank and Tardos [19]). They also develop better approximation algorithms for small $k$, specifically, an approximation ratio of $\lceil (k+1)/2 \rceil$ for $k \leq 7$. Cheriyan et al. [23] improved the $\Theta(k)$ approximation ratio with an $O(\lg k)$-approximation algorithm, provided that the number of vertices in the graph is at least $6k^2$. This algorithm is based on an iterative rounding method and the ellipsoid algorithm applied to a linear-programming relaxation of exponential size, and hence is not very practical.

Our algorithms will use as a subroutine any one of these approximation algorithms for minimum-weight $k$-connected subgraph. We suppose that the subroutine we use has an approximation ratio of $\alpha$, and state our own approximation ratio in terms of $\alpha$. This generality allows us to choose an algorithm according to practicality, or to chose a future improvement to the state-of-the-art for this problem, and understand the impact on the final approximation ratio. Note that a better approximation algorithm, with performance ratio $2 + (k-1)/n$ [21], is known if the graph weights satisfy the triangle inequality, but the weighted complete graphs we consider do not satisfy this property. Algorithm 1 presents the formal operation.

## 3. ALGORITHM FOR CONNECTIVITY REPAIR

In this section we describe our algorithm for minimizing the number of additional sensors to guarantee $k$-connectivity. The algorithm is relatively simple, building on approximation algorithms for minimum-weight $k$-vertex-connected subgraph. This modular design allows us to use several can-

didate algorithms for finding $k$-connected subgraphs and achieve a range of trade-offs between quality and performance. In particular, we can use a constant-factor approximation algorithm for $k$-connected subgraphs and obtain a constant-factor approximation algorithm for $k$-connectivity repair, for any fixed $k$. The analysis of this algorithm is complicated because we need to prove that the simplicity of the algorithm does not prevent it from finding more intricate, better solutions; this topic is addressed in the next section.

**Algorithm 1** K-CONNECTIVITY-REPAIR

```
1: input: k, set V of vertices and their coordinates
2: if n ≥ k then
3:    E ← {(v, w) | v, w ∈ V, v ≠ w}
4:    K ← (V, E)
5:    ω ← new V × V array
6:    for vertices v, w ∈ V do
7:       ω[v, w] ← ⌈‖v − w‖⌉ − 1
8:    end for
9:    call K-CONNECTED-SUBGRAPH (k, K, ω)
      to compute α-approximate minimum-weight
      k-connected spanning subgraph S of (K, ω)
10:   for edge (v, w) ∈ E(S) do
11:      for i = 1, 2, ..., ω[v, w] do
12:         t ← i/(ω[v, w] + 1)
13:         place k sensors at position (1 − t) · v + t · w
14:         place k − 1 sensors at position v
15:         place k − 1 sensors at position w
16:      end for
17:   end for
18: else
19:   call K-CONNECTIVITY-REPAIR (1, V)
20:   N ← {newly placed sensors}
21:   for x ∈ V ∪ N do
22:      place k − 1 sensors at position x
23:   end for
24: end if
```

The algorithm divides into two cases. Our description concentrates on the main case that the number $n$ of original sensors is at least the desired connectivity $k$. The second case that $n < k$ is simpler and we consider it later.

First we compute a weighted complete graph $K$ on the same set of vertices as the given graph $G$. The weight on an edge $\{v, w\}$ is one less than the ceiling of the Euclidean distance between the two points $v$ and $w$: $\omega(v, w) = \lceil \|v - w\| \rceil - 1$. This weight is zero if $v$ and $w$ are already connected by an edge in the unit-disk graph $G$, and otherwise it is the number of additional sensors required to connect $v$ and $w$ by a straight path.

Second we run an $\alpha$-approximation algorithm to find an approximately minimum-weight $k$-vertex-connected subgraph of this weighted complete graph $K$. See Section 2 for a summary of known theoretical approximation algorithms for this problem; see Section 5 for more practical implementations, including a greedy approach and a fast distributed algorithm. Note that our constructed graph $K$ likely does not satisfy the triangle inequality: if $G$ is connected, then there is a zero-weight path between every pair $v, w$ of vertices, so the triangle inequality would require that all edges $\{v, w\}$ have weight 0, which is rarely the case in $K$.

Third we translate the chosen edges in the $k$-vertex-connected subgraph of $K$ into a placement of new sensors. We place $\omega$ clusters of $k$ collocated sensors along the line segment connecting the endpoints of each edge of weight $\omega$, spacing the clusters a unit distance apart. In addition, for each edge of weight $\omega > 0$, we place $k - 1$ additional sensors at each endpoint of the edge.

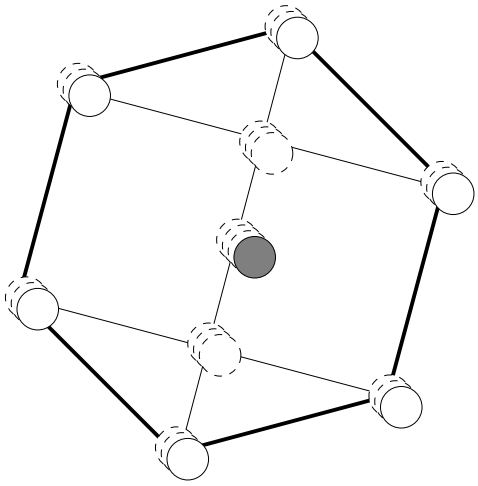

**Figure 3: An illustration of the approximation algorithm's performance in establishing 3-connectivity. Solid vertices exist in the input at the peripheral and establish the heavy edges for zero cost. The algorithm chooses to add the thinner edges and places additional sensors, marked with dashed circles, along the new edges. The optimal repair adds only the shaded vertex. We omit the optimal solution's edges to avoid clutter.**

In the case that $n < k$, the graph $K$ has fewer than $k$ vertices and therefore has no $k$-connected subgraph. We run the algorithm for $k = 1$ to compute an approximately minimum number of additional sensors that connect the sensors. Then we replicate each original and additional sensor $k$ times by adding $k - 1$ more copies.

Fig. 3 demonstrates an example of how the approximation algorithm establishes 3-connectivity to the network formed by the six peripheral vertices. The dark edges have no cost as the original graph already supports them. The K-CONNECTED-SUBGRAPH routine chooses the light edges to establish 3-connectivity and the approximation places new vertices, marked with dashed circles, along the chosen edges. The optimal solution places a single point, drawn as gray, in the graph center.

## 4. ANALYSIS

The main difficulty in the analysis of our algorithm's performance ratio is that the *Steiner points*—i.e., additional points other beyond the input points—can be placed in infinitely many possible locations. In particular, there may be some locations to place a Steiner point that simultaneously interconnect several pairs of original points. Our algorithm does not search for such "hub locations", nor will it notice that it found one if it happens to use one. Another, more subtle problem along the same lines is that it is not always optimal to connect pairs of original points by straight sequences of Steiner points. Rather, it may be beneficial to

connect several original points by straight lines to common Steiner points. A third challenge is that our objective is to (approximately) minimize the number of added sensors, not the total number of sensors. In particular, if the graph is already $k$-connected, any approximation algorithm must not add any sensors, because otherwise the ratio to the optimal cost of 0 would be infinite. Thus we need to exploit the existing connectivity among the original sensors, because we cannot afford to pay for it again. This more difficult objective prevents us from using structures whose cost depends on the number of original sensors. (Otherwise we could repeat a minimum spanning tree on the original sensors $k$ times, which would be a trivial $O(k)$ approximation on the total number of sensors.) These issues prevent us from using standard approximation algorithms and analysis.

Again we first consider the main case that $n \geq k$.

LEMMA 1. *For any set of original and Steiner sensors, there is a subgraph $G'$ of the induced unit-disk graph $G$ such that (1) for each edge of $G'$ incident to at least one Steiner sensor, we can assign it to one of its Steiner endpoints such that each Steiner sensor is assigned at most $6k$ edges, and (2) for any set $S$ of less than $k$ vertices, two vertices are connected in $G-S$ if and only if they are connected in $G'-S$.*

PROOF. We construct $G'$ by considering the Steiner sensors in an arbitrary order, and showing by induction that it suffices to connect each Steiner sensor to at most $6k$ original sensors and/or Steiner sensors that come earlier in the order. Let $s_1, s_2, \ldots, s_m$ denote the Steiner sensors, ordered arbitrarily. For each $0 \leq i \leq m$, let $G_i$ denote the graph on all original sensors and just the first $i$ Steiner sensors $\{s_1, s_2, \ldots, s_i\}$. Thus $G_m = G$. We will define a subgraph $G'_i$ of $G_i$ for each $0 \leq i \leq m$, such that $G'_i$ satisfies the two properties of the lemma with respect to $G_i$ (i.e., substituting $G_i$ and $G'_i$ for $G$ and $G'$). Thus $G'_m$ will serve as the desired $G'$.

In the base case, $G'_0 = G_0$ consists of just the original sensors, and the desired properties hold trivially: the first property because there are no Steiner sensors, and the second property because $G'_0 = G_0$. For the induction step, suppose we have already constructed $G'_{i-1}$ and we wish to construct $G'_i$. Each $G_i$ differs from the previous $G_{i-1}$ only in that it includes an additional vertex $s_i$ and some incident edges, and we will construct $G'_i$ similarly to differ from $G'_{i-1}$ only around $s_i$. Thus for $G'_i$ to satisfy the first property we need only that the degree of $s_i$ in $G'_i$ is at most $6k$; then we can assign all of these edges to $s_i$. We divide the neighbors of $s_i$ in $G_i$ into six groups by dividing the unit disk centered at $s_i$ into six equal pie wedges of angle $60°$. (The neighbors of $s_i$ in $G_i$ are precisely those sensors in the unit disk centered at $s_i$.) Then we select $k$ arbitrary neighbors from each of the six groups (or we select the entire group if it has size less than $k$), and make those $6k$ or fewer vertices the neighbors of $s_i$ in $G'_i$. Certainly $s_i$ has degree at most $6k$ in $G'_i$, so the first property holds. The key property of this construction is that all vertices in the same group are connected by edges in $G_i$, because each pie wedge has diameter 1.

Finally we must show that $G'_i$ satisfies the second property that, for any set $S$ of less than $k$ vertices, two vertices are connected in $G_i - S$ if and only if they are connected in $G'_i - S$. Consider some set $S$ of less than $k$ vertices. Because $G'_i$ is a subgraph of $G_i$, we need to show only that two vertices $v$ and $w$ connected in $G_i - S$ are also connected in $G'_i - S$. (Thus, in particular, the vertices $v$ and $w$ under consideration are not in $S$.) If $v$ and $w$ are connected by a path in $G_i - S$ that does not visit $s_i$, then that path also exists in $G_{i-1} - S$, so by induction the vertices are connected in $G'_{i-1} - S$ and thus in the supergraph $G'_i - S$. (In particular, if $S$ contains $s_i$, then this case applies.) Otherwise, we know that any path connecting $v$ and $w$ in $G_i - S$ visits $s_i$, and thus in particular any such path visits a vertex in the unit disk centered at $s_i$.

Let $v'$ be the first vertex along a path connecting $v$ and $w$ in $G_i - S$ that is inside the unit disk centered at $s_i$, and let $w'$ be the last vertex along the same path that is inside the unit disk centered at $s_i$. (Note that $v'$ might be $v$, and $w'$ might be $w$.) Because $v$ and $v'$ are connected by a path that does not visit $s_i$, by the previous case they are connected in $G'_{i-1} - S$, and similarly $w$ and $w'$ are connected in $G'_{i-1} - S$. We cannot have $v'$ and $w'$ in the same pie wedge of the unit disk, because then they would be connected via an edge in $G_{i-1}$ and thus connected in $G_{i-1} - S$ and by induction connected in $G'_{i-1} - S$, so there would have been a path connecting $v$ and $w$ that does not use $s_i$.

Now we argue that $v'$ and $s_i$ are connected in $G'_i - S$; by a symmetric argument $w'$ and $s_i$ are connected in $G'_i - S$, and thus $v$ and $w$ are connected in $G'_i - S$. If $v' = s_i$ (which happens precisely when $v = s_i$), then they are trivially connected. Otherwise, $v'$ is in one of the six pie wedges surrounding $s_i$. If there are at most $k$ vertices in the pie wedge containing $v'$, then $s_i$ has edges to all of them in $G'_i$, and thus in particular there is an edge between $s_i$ and $v'$. Otherwise, among the $k$ neighbors of $s_i$ in $G'_i$ in the pie wedge containing $v'$, at least one neighbor $v''$ must be in $G'_i - S$ because $S$ has size less than $k$. Because $v'$ and $v''$ are in the same pie wedge, they are connected by an edge in $G_{i-1}$ so by induction connected in $G'_{i-1}$. Adding the edge between $v''$ and $s_i$ to this path, we find that $v'$ and $s_i$ are connected in $G'_i$. Therefore $v$ and $w$ are connected in $G'_i$. □

This lemma shows that $G$ and the constructed subgraph $G'$ are essentially the same in terms of connectivity. The next lemmas show how to remove Steiner points from $G'$ again without losing any connectivity. We consider separately each "Steiner component" defined as follows. The *Steiner component* rooted at a Steiner sensor $s$ is formed by growing a set of vertices and edges in $G'$ starting with $\{s\}$ and stopping after we reach any original sensors. The Steiner component includes the edges connecting pairs of sensors in the component provided at least one of the endpoints is a Steiner sensor. (Equivalently, a Steiner component is a connected component of the induced subgraph of $G'$ on the Steiner sensors, together with the edges connecting these Steiner sensors to original sensors and these original sensors.)

LEMMA 2. *If $G'$ has at least $k$ original vertices and is vertex $k$-connected, then the number of original vertices in each Steiner component is at least $k$.*

PROOF. The set of original vertices of a Steiner component forms a cut in $G'$ unless that Steiner component is all of $G'$. In either case we must have at least $k$ original vertices in the Steiner component. □

Every Steiner component $C$ has a spanning tree $T(C)$ in which the original sensors are leaves of $T(C)$.

LEMMA 3. *The number of edges in an Eulerian tour of the spanning tree $T(C)$ of a Steiner component $C$ in $G'$ is at most $12k$ times the number of Steiner sensors in $C$.*

PROOF. The number of edges in the Eulerian tour is exactly twice the number of edges of $T(C)$. Each of these edges can be assigned to one of the Steiner sensors in $C$, and by Lemma 1, the number of assignments is at most $6k$ times the number of Steiner sensors in $C$. Therefore the number of edges in the Eulerian tour is at most $12k$ times the number of Steiner sensors in $C$. □

For any integer $n \geq 3$ and any positive integer $k \leq n$, the *Harary graph*[1] $H_{k,n}$ is the $k$-connected graph on $n$ vertices $v_1, v_2, \ldots, v_n$ where each $v_i$ is connected via an edge to the preceding $\lceil k/2 \rceil$ vertices $v_{i-1}, v_{i-2}, \ldots, v_{i-\lceil k/2 \rceil}$ and the succeeding $\lceil k/2 \rceil$ vertices $v_{i+1}, v_{i+2}, \ldots, v_{i+\lceil k/2 \rceil}$.

We consider the following procedure for *replacement of a Steiner component $C$*. First we remove the Steiner sensors in $C$. Second we take an Eulerian tour of the spanning tree $T(C)$. Third we construct a Harary graph $H_{k,m}$ on the $m$ original sensors in $C$ ordered by the order in which the Eulerian tour visits these leaves of $T(C)$. (By Lemma 2, $m \geq k$, so the Harary graph exists.) We view the edges of this graph as edges in the weighted complete graph $K$, and add these edges to our graph wherever they do not already exist. (We avoid the addition of multiple edges, but in fact single edges and multiple edges are equivalent for our purposes of vertex $k$-connectivity.) The resulting graph is no longer a unit disk graph: some edges come from the original unit-distance constraint, and others edges come from $K$. Once we replace all Steiner components in $G'$, we obtain a subgraph of $K$ with no Steiner sensors.

LEMMA 4. *The total weight of edges of $K$ that replace a Steiner component $C$ is at most $3k^3 + 12(k^2 + k)$ times the number of Steiner sensors in $C$.*

PROOF. Each edge in the Harary graph connects two original sensors that are within distance at most $\lceil k/2 \rceil$ in the order defined by the Eulerian tour. We charge the weight of this edge to the path of the Eulerian tour that connects these two original sensors. The weight of the edge is at most the number of edges in the path of the Eulerian tour because the edge in $K$ represents a shortcutting of the path taken by the Eulerian tour. We distribute the charge on the path of the Eulerian tour to the subpaths connecting consecutive original sensors in the Eulerian tour. Each subpath is charged at most $\lceil k/2 \rceil^2$ times, one for each edge of the Harary graph that spans that subpath. By Lemma 3, the number of edges in the Eulerian tour is at most $12k$ times the number of Steiner sensors in $C$. Therefore the total weight of the replacement is at most $12k\lceil k/2 \rceil^2 \leq 12k(k/2+1)^2 = 3k^3 + 12(k^2 + k)$ times the number of Steiner sensors in $C$. □

LEMMA 5. *If $G'$ is vertex $k$-connected, then replacement of all Steiner components in $G'$ results in a vertex $k$-connected subgraph of $K$.*

PROOF. We claim that replacement of a Steiner component preserves vertex $k$-connectivity. Let $C_1, C_2, \ldots, C_r$ denote the Steiner components in $G'$. For each $0 \leq i \leq r$, let $G'_i$ denote $G'$ after replacement of the first $i$ Steiner components. Thus $G'_0 = G'$ is $k$-connected.

Assume by induction that $G'_{i-1}$ is $k$-connected. Consider any set $S$ of less than $k$ vertices in $G'_i$ whose removal disconnects two vertices $v$ and $w$ in $G'_i - S$. By the induction hypothesis, there is a path connecting $v$ and $w$ in $G'_{i-1} - S$. Let $v'$ and $w'$ be the first and last vertex along that path that belong to Steiner component $C_i$. Thus $v'$ and $w'$ are both original vertices and therefore also present in $G'_i$. Also, $v$ and $v'$ are connected by the same subpath in $G'_i - S$, as are $w$ and $w'$. Because the replacement Harary graph is $k$-connected, removal of $S$ cannot disconnect it, so $v'$ and $w'$ are connected in the Harary graph and thus in $G'_i - S$. Therefore $v$ and $w$ are connected in $G'_i - S$. The result follows by induction. □

THEOREM 6. *In the case $n \geq k$, the algorithm is an $O(k^4\alpha)$-approximation on the minimum number of added sensors to attain vertex $k$-connectivity of the entire unit-disk graph.*

PROOF. Consider the optimal set of added sensors that results in a vertex $k$-connected unit-disk graph $G$. We construct $G'$ as in Lemma 1, and then perform a replacement of each Steiner component. By Lemmas 1 and 5, the resulting subgraph of $K$ is vertex $k$-connected. By Lemma 4, the total weight of the subgraph of $K$ is at most $3k^3 + 12(k^2 + k)$ times the number of Steiner sensors in $G$. The minimum-weight $k$-connected subgraph of $K$ can have only smaller weight, so is also at most $3k^3 + 12(k^2 + k)$ times the number of Steiner sensors in $G$. Our algorithm finds an $\alpha$-approximation to the minimum-weight $k$-connected subgraph, and then for every edge of weight $w \geq 1$, adds $kw + 2k - 2 < 3kw$ sensors. (This replication guarantees vertex $k$-connectivity of the entire graph because the removal of less than $k$ vertices cannot disconnect the subgraph of $K$, nor can it disconnect the sensors that constitute an edge of $K$.) Therefore the number of sensors added by the algorithm is less than $(9k^4\alpha + 36(k^3 + k^2)\alpha)$ times the optimal number of added sensors used in $G$. □

In the second case that $n < k$, the replication guarantees vertex $k$-connectivity of the entire graph because the removal of less than $k$ vertices still leaves at least one copy of the original spanning tree which is connected. We apply the $k = 1$ analysis to show that the number of added sensors for 1-connectivity is $O(1)$ times the optimal. The optimal number of added sensors for 1-connectivity is certainly a lower bound on the optimal number of added sensors for $k$-connectivity. The replication of these sensors costs an additional factor of $k$. The replication of the original sensors uses $k(k-1)$ additional sensors, which can be charged to the optimal number of added sensors for $k$-connectivity, which is at least 1 because the original graph cannot be $k$-connected. Therefore we obtain an approximation ratio of at most $k^2 + O(k)$.

COROLLARY 7. *For any fixed $k \geq 1$, our algorithm is an $O(1)$-approximation on the minimum number of added sensors to attain vertex $k$-connectivity of the entire unit-disk graph.*

The analysis bounds the number of vertices used to establish $k$-connectivity in the network through considering only points lying on edges in the complete weighted graph

[1] In fact, Harary graphs are defined differently when $k$ is odd. However, this definition allows us to obtain the desired approximation bound in this paper.

representing the network. Thus it leads us to simpler algorithms to achieve $k$-connectivity that do not have to consider Steiner points. We discuss such an algorithm next.

# 5. PRACTICAL IMPLEMENTATIONS

In this section we consider a practical implementation of the $k$-connectivity repair algorithm analyzed in the previous section. All algorithms that we consider are based on the K-CONNECTIVITY-REPAIR outline given in Algorithm 1, but use different subroutines for K-CONNECTED-SUBGRAPH (line 9). The guaranteed $\alpha$-approximation algorithms for $k$-connected subgraphs [21, 23] are complicated and may be difficult to implement on computationally and communication-constrained sensor network nodes. We implement simple greedy and distributed algorithms for K-CONNECTED-SUBGRAPH.

Algorithm 2 illustrates a greedy solution to K-CONNECTED-SUBGRAPH. The algorithm consists of two phases. We begin with an empty subgraph of the specified graph $G$. In the first phase, the algorithm repeatedly adds edges $(v, w)$ from the graph $G$ in increasing order by weight $\omega(v, w)$. The first phase ends once the subgraph is $k$-connected. In the second phase, the algorithm repeatedly attempts to remove edges $(v, w)$ from the subgraph, in decreasing order by weight $\omega(v, w)$, but putting the edge back if it was necessary for $k$-connectivity. Experimentally, this pruning stage can remove 58–85% of the added edges. The resulting subgraph is therefore a $k$-connected subgraph of $G$, and we expect that the weight of the chosen edges is reasonably small.

**Algorithm 2** Greedy K-CONNECTED-SUBGRAPH

```
1: input: k, G = (V, E), ω
2: G' ← (V, ∅)
3: E' ← {(v, w) | v, w ∈ V}
4: for (v, w) ∈ E' in increasing order of ω[v, w] do
5:    E(G') ← E(G) ∪ {(v, w)}
6:    if G' is k-connected then
7:       break
8:    end if
9: end for
10: for (v, w) ∈ E(G') in decreasing order of ω[v, w] do
11:    G'' ← (V, E(G') \ {(v, w)})
12:    if G'' is k-connected then
13:       G' ← G''
14:    end if
15: end for
16: return G'
```

This greedy algorithm produces the same subgraph as the following less-efficient algorithm: start from the complete graph $G$ as the subgraph, and repeatedly remove edges $(v, w)$ that do not destroy $k$-connectivity, in decreasing order by weight $\omega(v, w)$. For $k = 1$, this algorithm (and hence the original greedy algorithm) finds the optimal minimum spanning tree; it is essentially dual to Prim's algorithm. Thus the greedy algorithm generalizes a minimum-spanning-tree construction to $k \geq 1$, so we expect that it does well.

The greedy algorithm, however, can have arbitrarily poor performance, as depicted in Figure 4 when the graph forms a long narrow U-shaped chain to be repaired to 2-connectivity. The greedy algorithm chooses to reinforce the existing links by placing new nodes on top of existing ones. The optimal repair places the two gray nodes to create a loop. Since the chain can be arbitrarily long, but still require only a fixed number of nodes to repair, the ratio of the worst-case greedy cost to optimal is unbounded.

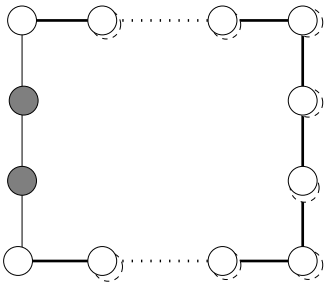

**Figure 4: An example of poor performance by the greedy algorithm to establish 2-connectivity. The greedy algorithm places the dashed nodes at the same position as the existing nodes, whereas the optimal repair places the gray nodes to construct a circuit.**

Algorithm 2 repeatedly tests for $k$-connectivity– a time consuming operation.[2] The algorithm also requires global knowledge of the candidate edges. To address both problems, we distribute a locally greedy algorithm that grows and merges $k$-connected components. Each component elects a leader to compute the cost of joining neighboring components. Two disjoint $k$-connected components form a larger $k$-connected component if bridged with $k$ vertex-disjoint edges.

Members of a $k$-connected component report the cost of joining some number of the closest other components. The leader uses an auction-based assignment implementation [24] to choose $k$ vertex-disjoint paths used to merge with each component. Each leader then proposes merging with its lowest-cost counterpart. A propositioned leader accepts the first offer that matches its lowest computed cost.

When components are smaller then $k$ nodes, leaders merge components by greedily creating $k$-cliques.

The distributed algorithm relaxes the need for synchronization, complete centralized knowledge of node distances, and $k$-connectivity testing. Each node, however, still requires localization and the locations of the components to which it proposes joining. Additionally, the algorithm has no pruning stage similar to the second stage of Algorithm 2, so we expect it to generate heavier subgraphs.

# 6. EXPERIMENTAL RESULTS

We implement in simulation both the centralized greedy-repair and the distributed solutions from the previous section to compare their performance with a sampling-based and the optimal subgraph repair solutions. The sampling-based algorithm scatters new vertices uniformly through the environment until the original nodes are $k$-connected, similar to the starting point of [25], except that we do not require $k$-connectivity to the additional nodes. For small graphs, we compare each of the three previous algorithms against the optimal placement, restricted by Algorithm 1, that we compute through combinatorial branch-and-bound search using Algorithm 2 as an initial bound.

[2]The runtime improves if the connectivity test first checks that the minimum degree for each node is at least $k$.

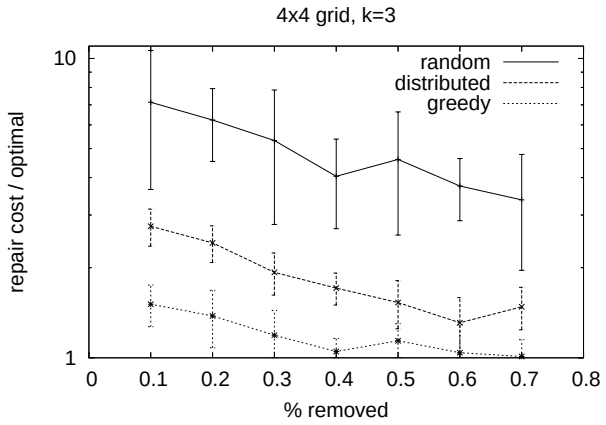


**Figure 5: The cost, relative to optimal, to repair 4×4 grid graphs to 3-connectivity. The figure plots mean repair cost of groups of ten experiments with error bars denoting the observed (biased) standard deviation.**

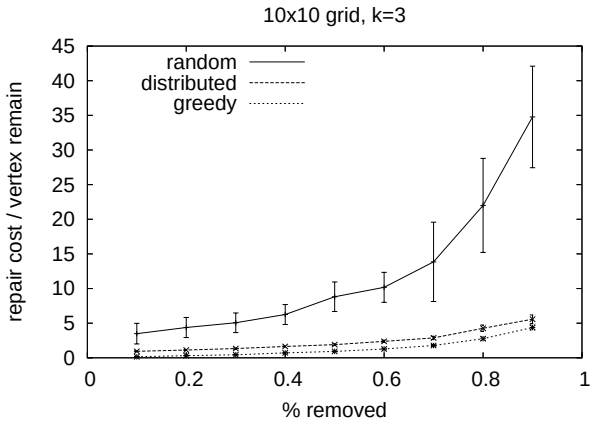


**Figure 6: The cost, relative to the number of vertices remaining, to repair damaged 10×10 grid graphs to 3-connectivity. The figure plots mean repair cost of groups of ten experiments with error bars denoting the observed (biased) standard deviation.**

We test each algorithm on uniformly-generated and grid graphs that have had some portion removed to destroy 3-connectivity. Surprisingly, greedy and distributed solutions perform close to optimal subgraph repair and far better than resampling the environment. We consider two graph structures to repair. The first is a two-dimensional rectangular grid with vertices evenly spaced one unit apart. With the exceptions of the corners, these graphs are 3-connected. For a chosen size parameter, $p$, vertices are removed from a graph with $n$ vertices until there are fewer than $pn$ vertices remaining and the graph is no longer 3-connected. The second class of graphs are generated through a Las-Vegas method. The vertices are uniformly placed inside a fixed-size rectangular area until the resulting graph is $k$-connected. The graphs to repair have vertices removed in the same fashion as the grid graphs.

We measure the number of vertices required to repair a damaged graph to $k$-connectivity, forgetting the vertices removed from the original graph. For these experiments, we only guarantee that the original vertices are $k$-connected, not the additional ones. Multiplying the cost by $k$ (for replication) provides an upper bound on the cost of repairs guaranteeing that the additional vertices are also $k$-connected. Empirically, however, additional vertices almost always lie closely enough to $k$-connected without extra duplication.

Fig. 5 plots the mean number of vertices added to the graph to reestablish 3-connectivity to 4×4 grids for the greedy, distributed, and randomized implementations. Each of the plotted values are normalized first by the number of vertices in the damaged graph and then by the optimal cost to repair the graph. The greedy algorithm frequently finds the optimal repair for graphs with more than 30% of their vertices removed.

It is difficult to measure the optimal cost of larger graphs, but we observe that the repair costs for $10 \times 10$ grid graphs plotted in Fig. 6 scale similarly to the costs of $4 \times 4$ before normalization by the optimal.

Figs. 7 and 8 plot results for the same experiments on denser, uniformly-generated graphs. For comparison to optimal, we are able only to look at uniformly distributed graphs

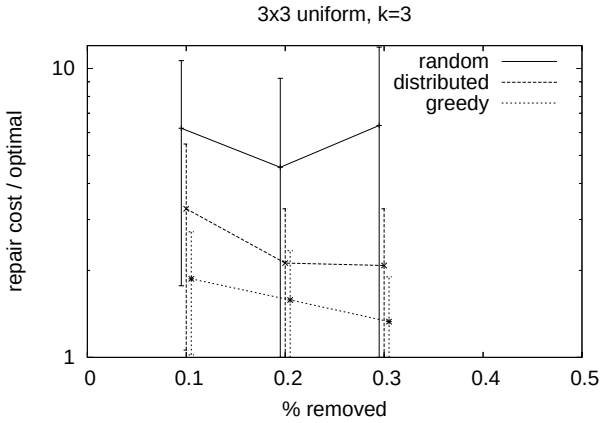


**Figure 7: The cost of reestablishing 3-connectivity to graphs uniformly-generated in a 3×3 area. We plot the mean repair cost relative to optimal and (biased) standard deviation for groups of ten experiments.**

in a $3 \times 3$ area, where the graphs have on average 48 nodes, but some have as many as twice that. Additionally, we were only able to compute optimal repairs for graphs sustaining at most 30% removal. We also plot experiments on $7 \times 7$ uniformly-generated graphs that average 250 vertices. Again the repair costs increase similarly in both of the two scales.

The performance of the greedy repair algorithm is surprising and requires some explanation. Fig. 9 shows an example of a uniformly-generated 3-connected graph. Nodes A, B, and C are removed and the greedy repair algorithm takes the resulting graph as input. The algorithm adds node Z. Node A is superfluous to preserving connectivity in the original graph; it resides at the graph's fringe. Since it resides in a densely populated region, node B is redundant. Thus, these two node removals require no attention for repair. The greedy algorithm replaces node C with Z, reestablishing 3-

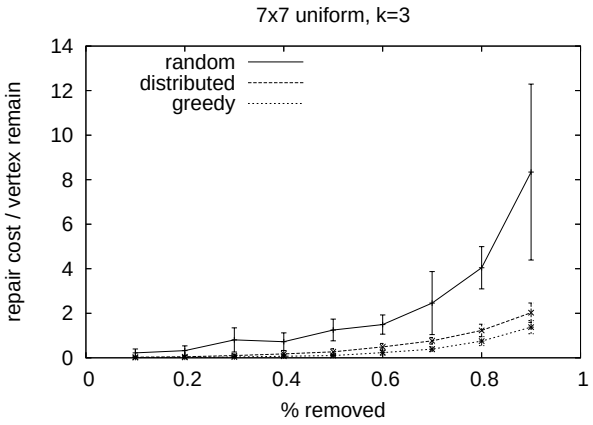


**Figure 8: The cost of reestablishing 3-connectivity to graphs uniformly-generated in a 7×7 area. We plot the mean repair cost relative to optimal and (biased) standard deviation for groups of ten experiments.**

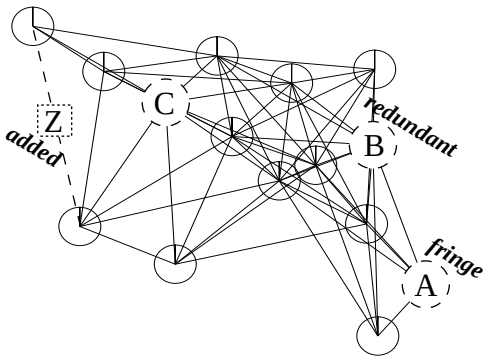


**Figure 9: A typical graph-repair example. The destruction process removes nodes A, B, and C. The greedy repair algorithm adds node Z.**

connectivity.

Since many of the vertices randomly selected to be removed from the input graphs do not contribute to the graph's $k$-connectedness, we also test the repair algorithms on more-selectively damaged graphs. To better ensure that removed vertices contribute to connectivity, we uniformly-generate 3-connected 10×10 unit-disk graphs, select two pairwise distant vertices, $s$ and $t$, and remove a vertex in the middle of some number $s$-$t$ paths. Fig. 10 shows an example 3-connected $10 \times 10$ unit-disk graph with two groups of vertices marked for removal. The smaller, circular grouping denotes a substantial disconnection that will affect connectivity, whereas the larger group will completely disconnect the graph.

Table 1 shows the repair costs of removing enough vertices to completely disconnect $s$-$t$ pairs and removing only enough edges to diminish the number vertex-disjoint $s$-$t$ paths. The table shows the increase in the number of vertices required to reestablish 3-connectivity, relative to the optimal repair

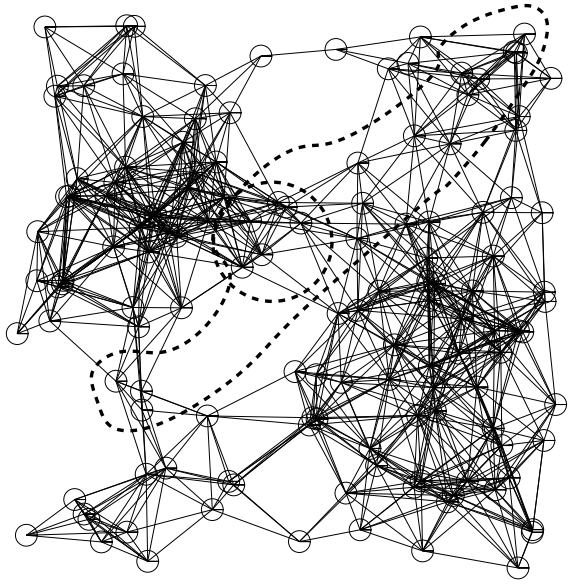

**Figure 10: An example of complete disconnection, represented by the larger dashed region, and substantial vertex removal, denoted by the dashed circle.**

| | greedy | | dist | | random | |
|---|---|---|---|---|---|---|
| | $\mu$ | $\sigma$ | $\mu$ | $\sigma$ | $\mu$ | $\sigma$ |
| complete | 1.35 | 0.72 | 1.59 | 0.76 | 3.94 | 3.27 |
| substantial | 2.13 | 1.68 | 2.14 | 2.77 | 7.65 | 9.36 |

**Table 1: The repair cost relative to optimal of repairing uniformly uniformly-generated 3-connected 3×3 unit-disk graphs. The table reports the mean and observed standard deviation for ten trials. The "complete" entry represents graphs that were completely disconnected, whereas the "substantial" entry shows the repair cost of removing only enough vertices to diminish the number of pairwise-disjoint paths from two distant fringe vertices.**

cost. As the greedy and distributed algorithms can better localize the area to be repaired, they perform about 3 times better than the random repair algorithm.

## 7. CONCLUSION

An interesting variation on our problem is to change the goal from vertex $k$-connectivity to what we call *partial $k$-connectivity*: there should be $k$ vertex-disjoint paths between every pair of original sensors, but not necessarily between pairs involving added sensors. This weaker goal is natural if only the original sensors serve a useful purpose (e.g., carrying information), and the added sensors only serve for additional connectivity between them. A variation of our approach may lead to efficient approximation algorithms for this case as well. In the complete graph $K$, we replicate each edge $k$ times, and assign a weight to each replicated edge equal to the original edge weight if it is positive, or 1 if the original weight is zero. The point of this modification is that, in the partial $k$-connectivity model, repeat-

ing additional sensors can increase connectivity between two vertices, unlike regular $k$-connectivity. For our approach to work, however, we need a stronger version of the subroutine for finding the approximate minimum-weight $k$-connected subgraph that supports a multigraph as input. Alternatively, we can subdivide each edge and distribute the weight arbitrarily between the two halves, and use a subroutine that finds the approximate minimum-weight subgraph that is partially $k$-connected on the specified set of original vertices. There is evidence that this problem is significantly more difficult than regular $k$-connectivity [26]. If we had either such subroutine, it would seem that the rest of our approach would lead to efficient approximation algorithms for partial $k$-connectivity.

Our analysis of the approximation ratio of our algorithm is likely not tight; we believe that the same algorithm has an asymptotically smaller approximation ratio than what we prove. Our experimentation confirms this intuition. An example is the case of 1-connectivity, where our algorithm simply short-cuts an Eulerian tour of a minimum spanning tree. Chen et al. [14] proved that this algorithm has a worst-case approximation ratio of precisely 4. In contrast, our analysis for general $k$ proves an upper bound of somewhat more than 4 in the case $k = 1$. The main advantage of our approach and analysis is its generality, applying for arbitrary $k$.

We conjecture that our approximation results can be further strengthened to find a polynomial-time approximation scheme (PTAS), i.e., an algorithm that attains an approximation ratio of $1 + \epsilon$ for any desired $\epsilon > 0$. We believe that such a PTAS can be obtained using the techniques of Arora [27] and Mitchell [28]. These techniques have been successfully applied to obtain a PTAS for the related problem of finding the minimum-Euclidean-length $k$-connected subgraph [29]. Our conjecture is wide open even for the case of $k = 1$. To the best of our knowledge, the only related problem known to have a PTAS is the easier problem of approximating the total number of sensors, instead of the number of added sensors, for $k = 1$ [14]. For our problem and $k = 1$, the best known approximation algorithm has an approximation ratio of 5/2 [15]. For our problem and $k > 1$, our algorithms are the only known approximation algorithms.

In our current work, we are implementing the distributed $k$-connectivity repair algorithm on a mote network that interacts with a mobile robot. We are also developing improved distributed versions of the $k$-connectivity algorithm.

## ACKNOWLEDGMENTS

Support for this work has been provided in part by NSF grants IIS-0426838, IIS-0225446 and ITR ANI-0205445, and from Intel, Boeing, MIT Project Oxygen, and by award Number 2000-DT-CX-K001 from the Office for Domestic Preparedness, US Department of Homeland Security. Points of view in this document are those of the authors and do not necessarily represent the official position of the US Department of Homeland Security.

## 8. REFERENCES

[1] P. Corke, S. Hrabar, R. Peterson, D. Rus, S. Saripalli, and G. Sukhatme, "Autonomous deployment of a sensor network using an unmanned aerial vehicle," in *Proceedings of the 2004 International Conference on Robotics and Automation*, New Orleans, USA, 2004.

[2] P. Corke, S. Hrabar, R. Peterson, D. Rus, S. Saripalli, and G. Sukhatme, "Deployment and connectivity repair of a sensor net with a flying robot," in *Proceedings of the 9th International Symposium on Experimental Robotics*, Singapore, 2004.

[3] G.H. Lin and G. Xue, "Steiner tree problem with minimum number of Steiner points and bounded edge-length," *Inform. Process. Lett.*, vol. 69, no. 2, pp. 53–57, 1999.

[4] R. Ramanathan and R. Rosales-Hain, "Topology control of multihop radio networks using transmit power adjustment," in *Proceedings of Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pp. 404–413. March 2000.

[5] M. Bahramgiri, M. Hajiaghayi, and V.S. Mirrokni, "Fault-tolerant and 3-dimensional distributed topology control algorithms wireless multi-hop networks," in *Proceedings of the 11th IEEE International Conference on Computer Communications and Networks (ICCCN)*, pp. 392–398. 2002.

[6] R. Wattenhofer, L. Li, V. Bahl, and Y. M. Wang, "Distributed topology control for power efficient operation in multihop wireless ad hoc networks," in *Proceedings of twentieth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pp. 1388–1397. 2001.

[7] L. Li, J. Halpern, V. Bahl, Y. M. Wang, and R. Wattenhofer, "Analysis of a cone-based distributed topology control algorithm for wireless multi-hop networks," in *Proceedings of ACM Symposium on Principle of Distributed Computing (PODC)*, pp. 264–273. 2001.

[8] M. Hajiaghayi, N. Immorlica, and V. S. Mirrokni, "Power optimization in fault-tolerant topology control algorithms for w ireless multi-hop networks," in *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking*. 2003, pp. 300–312, ACM Press.

[9] E. Lloyd, R. Liu, M. Marathe, R. Ramanathan, and S. Ravi, "Algorithmic aspects of topology control problems for ad hoc networks," *Proceedings of the Third ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2002.

[10] D. Blough, M. Leoncini, G. Resta, and P. Santi, "The K-neigh protocol for symmetric topology control in ad hoc networks," in *ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*. June 2003.

[11] X.Y. Li, W.Z. Song, and Y. Wang, "Efficient topology control for wireless ad hoc networks with non-uniform transmission ranges," *ACM Wireless Networks*, in press.

[12] X.Y. Li, Y. Wang, P.J. Wan, and C.W. Yi, "Robust deployment and fault tolerant topology control for wireless ad hoc networks," in *ACM International*

*Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*. June 2003.

[13] Ning Li and Jennifer C. Hou, "FLSS: a fault-tolerant topology control algorithm for wireless networks," in *Proceedings of the 10th Annual International Conference on Mobile Computing and Networking*. 2004, pp. 275–286, ACM Press.

[14] D. Chen, D.Z. Du, X.D. Hu, G.H. Lin, L. Wang, and G. Xue, "Approximations for Steiner trees with minimum number of Steiner points," *J. Global Optim.*, vol. 18, no. 1, pp. 17–33, 2000.

[15] D. Du, L. Wang, and B. Xu, "The Euclidean bottleneck Steiner tree and Steiner tree with minimum number of Steiner points," in *Computing and Combinatorics (Guilin, 2001)*, vol. 2108 of *Lecture Notes in Comput. Sci.*, pp. 509–518. Springer, Berlin, 2001.

[16] D.M. Blough, M. Leoncini, G. Resta, and P. Santi, "On the symmetric range assignment problem in wireless ad hoc networks.," *Proceedings of the 2nd IFIP International Conference on Theoretical Computre Science (TCS)*, pp. 71–82, 2002.

[17] G. Calinescu, I.L. Mandoiu, and A. Zelikovsky, "Symmetric connectivity with minimum power consumption in radio networks," in *Proceedings of 17th IFIP World Computer Congress*, pp. 119–130. 2002.

[18] Lefteris M. Kirousis, Evangelos Kranakis, Danny Krizanc, and Andrzej Pelc, "Power consumption in packet radio networks," *Theoret. Comput. Sci.*, vol. 243, no. 1-2, pp. 289–305, 2000.

[19] A. Frank and E. Tardos, "An application of submodular flows," *Linear Algebra Appl.*, vol. 114/115, pp. 329–348, 1989.

[20] S. Khuller and B. Raghavachari, "Improved approximation algorithms for uniform connectivity problems," *J. Algorithms*, vol. 21, no. 2, pp. 434–450, 1996.

[21] G. Kortsarz and Z. Nutov, "Approximating node connectivity problems via set covers," in *Proceedings of the Third International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX)*, pp. 194–205. 2000.

[22] Harold N. Gabow, "A representation for crossing set families with applications to submodular flow problems," in *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms (Austin, TX, 1993)*, New York, 1993, pp. 202–211, ACM.

[23] J. Cheriyan, S. Vempala, and A. Vetta, "An approximation algorithm for the minimum-cost $k$-vertex connected subgraph," *SIAM J. Comput.*, vol. 32, no. 4, pp. 1050–1055 (electronic), 2003.

[24] Dimitri Bertsekas, *Network Optimization: Continuous and Discrete Models*, Athena Scientific, Nashua, NH, 1998.

[25] Volkan Isler, Sampath Kannan, and Kostas Daniilidis, "Sampling based sensor-network deployment," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai, Japan, Sept. 2004, pp. 172–177.

[26] G. Kortsarz, R. Kraughgamer, and J. L. Lee, "Hardness of approximation for vertex-connectivity network-design problems," in *Proceedings of 5th International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX)*, pp. 185–199. 2002.

[27] S. Arora, "Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems," *J. ACM*, vol. 45, no. 5, pp. 753–782, 1998.

[28] J. S. B. Mitchell, "Guillotine subdivisions approximate polygonal subdivisions: a simple polynomial-time approximation scheme for geometric TSP, $k$-MST, and related problems," *SIAM J. Comput.*, vol. 28, no. 4, pp. 1298–1309 (electronic), 1999.

[29] A. Czumaj and A. Lingas, "A polynomial time approximation scheme for Euclidean minimum cost $k$-connectivity," in *Automata, languages and programming (Aalborg, 1998)*, vol. 1443 of *Lecture Notes in Comput. Sci.*, pp. 682–694. Springer, Berlin, 1998.