Relay Placement for Higher Order Connectivity in Wireless Sensor Networks*

Abhishek Kashyap†, Samir Khuller‡ and Mark Shayman†
†Department of Electrical and Computer Engineering
University of Maryland, College Park, USA
Email: {kashyap, shayman}@glue.umd.edu
‡Department of Computer Science
University of Maryland, College Park, USA
Email: samir@cs.umd.edu

Abstract— **Sensors typically use wireless transmitters to communicate with each other. However, sensors may be located in a way that they cannot even form a connected network (e.g, due to failures of some sensors, or loss of battery power). In this paper we consider the problem of adding the smallest number of additional (relay) nodes so that the induced communication graph is 2-connected[1]. The problem is NP-hard. In this paper we develop $O(1)$-approximation algorithms that find close to optimal solutions in time $O((kn)^2)$ for achieving k-edge connectivity of n nodes. The worst case approximation guarantee is 10, but the algorithm produces solutions that are far better than this bound suggests. We also consider extensions to higher dimensions, and the scheme that we develop for points in the plane, yields a bound of $2d_{MST}$ where d_{MST} is the maximum degree of a minimum-degree Minimum Spanning Tree in d dimensions using Euclidean metrics. In addition, our methods extend with the same approximation guarantees to a generalization when the locations of relays are required to avoid certain polygonal regions (obstacles).**

We also prove that if the sensors are uniformly and identically distributed in a unit square, the expected number of relay nodes required goes to zero as the number of sensors goes to infinity.

Keywords: Sensor networks, Fault-tolerant topology design, Approximation algorithms, Relay placement.

I. Introduction

Wireless communication is central to the area of sensor networks. In a sensor network, sensor nodes collect data and forward it to sink nodes. Energy consumption is the dominating constraint in sensor nodes, thus multi-hop data forwarding through sensor nodes using long paths decreases the energy levels of sensor nodes at a very fast rate. Thus, a scalable solution is to cluster the sensor nodes and have a cluster-head for each cluster [1], [2], [3]. This can be called a two-tiered wireless sensor network [3]. We will call these cluster-heads backbone nodes. All sensor nodes forward data to the backbone node in their cluster. The backbone nodes form a backbone network among themselves using a communication channel different from the channel used by sensor nodes, and forward the data to the sink nodes through the backbone network. The backbone nodes typically have higher energy levels than sensor nodes and also have recharging capabilities, so energy constraints of backbone nodes are not a dominating issue.

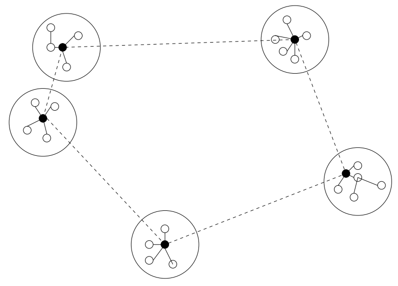


Fig. 1. Example hierarchical network

Figure 1 shows an example network with clusters of sensor nodes enclosed in circles and backbone nodes shown as solid nodes. The dotted lines between backbone nodes show the backbone network links. As wireless backbone networks carry aggregate traffic of the underlying network, the backbone network becomes more critical than the underlying sensor network. Failure of a backbone node or a link in the backbone network can impact the performance of a network. Thus, the backbone network topology needs to be fault-tolerant, i.e., it should have multiple internally vertex-disjoint (or edge-disjoint) routes between each pair of backbone nodes. If k vertex (edge) disjoint paths exist between each pair of nodes, the network is said to be k-vertex (edge) connected.

There has been recent work in topology control of (sensor) backbone networks. Hao et. al. [4] consider the problem of placing the minimum number of backbone nodes among a set of candidate locations such that each sensor node has paths to at least two backbone nodes, and the backbone nodes have at least two vertex-disjoint paths between them. They provide an approximation algorithm having an $O(D \log n)$ approximation ratio, where D depends on the diameter of the network and n is the number of sensor nodes in the network. Liu et. al. [5] consider the problem of placing backbone nodes in a network

[1]We consider both edge and vertex connectivity in this work.

*This research was partially supported by AFOSR under grant F496200210217, NSF under grant CNS-0435206, NSF CCF-0430650, NSF CNS-0519554.

of sensor nodes so that the network is 2-connected. They provide a $(6+\epsilon)$-approximation algorithm for connectivity and two approximation algorithms for 2-connectivity with ratios $(24+\epsilon)$ and $(6/T+12+\epsilon)$, where T is the ratio of backbone nodes needed for connectivity and sensor nodes in the network. Their problem is different from ours as they want the set of backbone nodes to be a dominating set among the sensor nodes, i.e., each sensor node should be directly connected to at least one backbone node.

In a sensor network, the sensor nodes may be deployed in clusters, with some clusters being far from each other. In this scenario, the locations of cluster-heads are not controllable, and we still need to construct a backbone network among the backbone nodes. The backbone nodes can be quite far from each other, thus power control cannot be used to connect them. We assume a fixed transmission range for each backbone node. We propose the use of additional relay nodes in the backbone network, whose position we can control, to achieve the desired level of connectivity among the backbone nodes. We do not restrict the backbone network to have any particular type of links; they can be omnidirectional/directional RF or free space optical point-to-to-point links.

The problem of constructing a connected network on backbone nodes using minimum number of relay nodes has been considered in [6], [7], [8]. Lin et. al. [6] prove the problem to be NP-Hard and propose an approximation algorithm for constructing a tree using relay nodes, and prove the algorithm to be a 5-approximation. The algorithm restricts the placement of relay nodes on lines joining pairs of backbone nodes. It then assigns a weight function to each pair of backbone nodes according to the number of relay nodes needed to connect them directly. They find a minimum spanning tree (MST) on this graph. Proofs of 4-approximation ratio for the algorithm are provided in [7] and [8], and the bound is proved to be tight. Chen et. al. [8] also provide a 3-approximation algorithm for the problem. Cheng et. al. [9] provide a 2.5-approximation randomized algorithm for placement of relay nodes to connect a given set of sensor nodes. Our algorithms work on any set of nodes distributed in a space, so they can be applied to flat sensor networks as well.

There has been work on probabilistic analysis of number of nodes required and their transmission range required for achieving connectivity and k-connectivity among randomly distributed nodes. Gupta and Kumar [10] consider the problem of finding the critical power so that the network is connected almost surely (with probability 1) as the number of nodes approach infinity, the nodes being distributed uniformly and independently in a unit disk. Bettstetter [11] studies the node degree properties of nodes distributed randomly in a network, and the connectivity and k-connectivity properties of the network. The author gives probabilities of having isolated nodes and approximates the probability of having a connected network to be equal to the probability of having no isolated node. Similar approximations are done for k-connectivity as well. Li et. al. [12] prove that for sufficiently large number of nodes in the network, there is a critical power level after which the network is k-connected with a certain non-zero probability. None of the frameworks seems to extend to the case of adding additional nodes in the network for achieving connectivity, as achieving connectivity by increasing power level of all nodes is not similar in spirit to achieving connectivity by adding relay nodes. Also, if backbone nodes are not located close to each other, addition of relay nodes is a more practical solution to achieving desired connectivity levels.

There has not been any work, other than [4] and [5], that considers achieving higher levels of connectivity using minimum number of relay nodes. We consider the problem of providing k-connectivity (both edge and vertex) for $k \geq 2$ among backbone nodes using minimum number of relay nodes. The contributions of this paper are as follows: (1) we provide algorithms for using relays to achieve k-(edge, vertex) connectivity among backbone nodes; (2) we prove the algorithm to be an O(1)-approximation with respect to the optimal for achieving 2-edge and 2-vertex connectivity; (3) we consider extensions to spaces of higher dimensions, and give O(1) bounds for higher dimension metric spaces; (4) we do worst case analysis of the algorithm of [6] as a function of number of backbone nodes in the network and transmission range of the nodes: we show that the upper bound on number of relays becomes a constant as number of backbone nodes increase, and it decreases exponentially with increasing transmission range; (5) we prove that the number of relay nodes required for achieving (2^d-1)-vertex (and edge) connectivity go to zero in the mean as number of backbone nodes (distributed uniformly and independently in d-dimensional Euclidian space) go to infinity; (6) we extend our algorithms to the generalization where the relays cannot be placed in certain polygonal regions (obstacles) and show the same approximation ratios hold for this generalization as well.

The paper is organized as follows: Section II gives the network model and problem statement. Section III describes the proposed algorithm for achieving k-edge connectivity. Section IV gives the proof of approximation ratio for 2-edge connectivity. Section V describes the k-vertex connectivity approximation algorithm, and proves the approximation ratio for $k=2$. Section VI gives the approximation bounds for the algorithms for nodes distributed in spaces of higher dimensions. Section VII extends the algorithms to work with the same approximation ratio for the generalization where relays cannot be placed in certain polygonal regions of the network. Section VIII discusses the worst case bounds on the number of relays as a function of number of nodes and their transmission range. It also gives the proof of expected value of number of relays going to zero for uniform distribution of backbone nodes. Section IX discusses the simulation results. Section X concludes the paper and discusses the future work.

II. Network Model and Problem Statement

We model the network as a graph $G=(V,E)$, where V is the set of backbone nodes, which we call terminal nodes, and E is the set of links between them. We assume each node has a limited transmission range, which we normalize to one by normalizing the length of the network. It is assumed that a node can connect to all nodes within its transmission range. The network can be in any d-dimensional Euclidian space, i.e., the distance between two nodes is considered to be the Euclidian distance between them. A link $e = (x, y)$ belongs to E if nodes x and y are within unit distance of each other. The links can be either omnidirectional RF, directional RF or Free Space Optical links (without obscuration).

We assume we have relay nodes, which are identical to the terminal nodes in terms of their transmission range and type of links. We assume we have control over the location of relay nodes. Thus, we place the relay nodes in the network so that the desired level of connectivity between terminal nodes is achieved. The problem can be formally stated as follows:

Given a graph $G = (V, E)$, find the minimum number of relay nodes needed (and their locations) so that the set of nodes V is k-edge (vertex) connected ($k \geq 2$) in the resulting graph $G' = (V', E'), V \subseteq V', E \subseteq E'$. The objective is to construct a graph s.t. $\lambda(u, v) \geq k \, \forall \, u, v \in V$ where $\lambda(u, v)$ is the number of edge-disjoint (or internally vertex-disjoint) paths between u and v in G'.

III. Algorithm for k-edge connectivity

We follow the relay placement framework of the connectivity algorithm of [6]. To connect two terminal nodes outside each other's transmission range, the relay nodes are placed on the straight line connecting the two nodes. The algorithm for achieving k-edge connectivity is described as follows. The algorithm proceeds by forming a complete graph on the terminal nodes. Equation 1 gives the weight function used for the edges, where $|e|$ is the length of an edge. The weight represents the number of relay nodes required to form an edge. We do not allow the relay nodes to have edges other than the ones required to form the edge they are placed on. Then it computes an approximate minimum cost spanning k-edge connected subgraph of the complete graph.

$$c_e = \lceil |e| \rceil - 1 \tag{1}$$

The problem of finding the minimum cost spanning k-edge connected subgraph of a graph is NP-Hard [13]. Thus, we use an approximation algorithm for the problem, proposed by Khuller and Vishkin in [14]. The algorithm achieves an approximation ratio of 2 for the problem, and takes $O((kn)^2)$ time for a graph with n nodes. The algorithm uses the matroid intersection based algorithm of Gabow [15], which finds k edge-disjoint spanning trees from a root vertex in a directed graph. It is worth noting that the weight function of Equation 1 is not a metric as it does not satisfy triangle inequality. Thus, the approximation algorithm of [14] is the best known for the problem. In the resulting subgraph from the approximation algorithm of [14], the relay nodes are placed to form the links (of length greater than one) of the subgraph. In the next section, we prove that this algorithm has an approximation ratio of 10 for 2-edge connectivity. The solution is then improved by removing some relays. The relays are allowed to form edges with all nodes in their transmission range and sequentially removed if k-edge connectivity is preserved. We call this step the *sequential removal step*, and it takes $O(n'((n + n')m))$ time, where n' is the number of relays before the sequential removal step, and m is the number of edges in the network formed by the terminals and relays. Thus, the first part of the algorithm takes $O((kn)^2)$ time, while the enhanced algorithm takes $O((kn)^2 + n'm(n + n'))$ time. Algorithm 1 describes the algorithm.

Algorithm 1 Relay placement for k-edge connectivity

```
1: Make a complete graph $G_c = (V, E_c)$ by adding edges
   between all the vertices of graph $G$ (if an edge already
   exists, a new edge is not added between that pair of
   vertices).
2: Weight the edges of the graph as follows. $|e|$ represents
   the length of edge $e$.
       $c_e = \lceil |e| \rceil - 1$
3: Compute an approximate minimum cost spanning $k$-edge
   connected subgraph from this graph $G_c$ using the approxi-
   mation algorithm proposed in [14]. Let the resulting graph
   be $G'_c$.
4: Place relay nodes (number equal to the weight of the edge)
   on the edges in $G'_c$ with link costs greater than zero.
5: For all pairs of nodes (including the relay nodes) in $G'_c$
   within each other's transmission range, form an edge.
6: For the relay nodes sorted arbitrarily, do the following
   (starting at $i = 1$):
     • Remove node $i$ (and all adjacent edges).
     • Check for $k$-edge connectivity between the terminals.
     • If the graph is $k$-edge connected, repeat for $i = i + 1$,
       else put back the node $i$ and corresponding edges, and
       repeat for $i = i + 1$.
     • Stop when all relay nodes have been considered.
7: Output the resulting graph.
```

We briefly explain the algorithm for computing the approximate minimum-weight (cost) 2-edge connected subgraph [14] of a graph G: Create a directed graph D having anti-parallel directed edges for each undirected edge in G, each having the same weight as the corresponding undirected edge. Pick any vertex as the root vertex. Run Gabow's algorithm [15] to get k edge-disjoint spanning trees. Construct a directed graph $G'_D = (V, E')$ containing the edges of all trees. Construct an undirected graph $G'_c = (V, E'')$, where an edge $(u, v) \in E''$ if $(u, v) \in E'$ and/or $(v, u) \in E'$. The algorithm outputs G'_c. For implementing Gabow's matroid intersection ([16]) based algorithm, we use Frank's weighted matroid intersection algorithm [17] and Roskind and Tarjan's algorithm for computing edge-disjoint spanning trees [18], which is based on greedy matroid algorithm [16]. Due to the complexity of the algorithms, we do not describe them here. Details of these can be found in [15], [17] and [18].

IV. Proof of Approximation Ratio for 2-edge connectivity

In this section, we consider the case of achieving 2-edge connectivity between terminal nodes distributed in a Euclidian plane. We analyze the worst-case performance of our algorithm and prove a performance bound of the algorithm with respect to the optimal solution. We assume the existence of an optimal solution, and use its properties to prove the worst case bound of the solution given by our algorithm.

We start with some notations. Let T be the set of terminals, and S be the set of optimally placed Steiner nodes (relay nodes) needed to achieve 2-edge connectivity among the terminal nodes. Let s be the number of Steiner nodes needed when we place them optimally, i.e, $s = |S|$. In the proof, we will call the relay nodes placed on straight lines between terminals just to form the edge they are placed on (as in our algorithm) as beads and the optimally placed relay nodes as Steiner nodes.

As a recap of our algorithm, it forms a 2-edge connected network among the terminal nodes by placing additional links between them, and if two terminal nodes are more than unit distance apart, it adds beads (relay nodes) to form that link. When we add such a link of length l, it consists of $\lceil l \rceil - 1$ beads. Theorem 4.1 states the main result of this section.

Theorem 4.1: If the optimal network uses s Steiner nodes so that terminals distributed in a Euclidian plane are 2-edge connected, Algorithm 1 forms a network with maximum of $10s$ beads and zero Steiner nodes, in which the terminal nodes are 2-edge connected.

To prove Theorem 4.1, we prove the following lemma, and Theorem 4.1 follows directly.

Lemma 4.2: A 2-edge connected network on terminal nodes using minimum number of beads contains at most $5s$ beads, where s is the minimum number of Steiner nodes needed.

Proof: Let $G_0 = (V_0, E_0)$ be the optimal 2-edge connected network, with the minimum number of Steiner nodes.

We follow the procedure of Algorithm 2 to construct a 2-edge connected network using only beads and no Steiner nodes. We will prove that this network does not contain more than $5s$ beads.

Algorithm 2 starts by finding the connected components of Steiner nodes in the graph G_S constructed on the Steiner nodes. It constructs a minimum-degree minimum spanning tree (MST) for each connected component. Let the trees be $ST_1, .., ST_m$. It then removes Steiner nodes of the jth connected component of G_S from G_{j-1} and adds beads between the terminals connected to those Steiner nodes to get G_j which is also 2-edge connected between terminal nodes (Step 4). The process is repeated for all connected components, until the resulting graph has no Steiner nodes and is 2-edge connected on the terminals. We first mention two useful properties that hold for each of the trees $ST_1, .., ST_m$, which will be used in proving the approximation bound:

Property 4.3: The maximum degree of any Steiner node in the trees is bounded by five [19]. This property comes from the fact that the maximum degree of a node in a minimum-degree MST on nodes distributed in a Euclidian plane is bounded by five, and is called the MST number of the Euclidian plane.

Algorithm 2 Construction of 2-edge connected network with beads

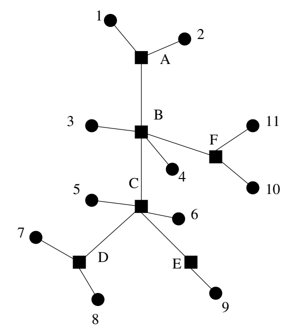
```
1: Define a graph $G_S = (S, E_S)$ on the Steiner nodes, where an edge $(u, v)$ is in $E_S$ if it is an edge between the Steiner nodes $u, v$ in $G_0$.
2: Find all the connected components in $G_S$.
3: Form a minimum-degree minimum spanning tree in each connected component, and call the trees $ST_1, .., ST_m$.
4: Repeat the following for $j = 1$ to $m$:
     • Remove the Steiner nodes contained in $ST_j$ from $G_{j-1}$ and add beads between terminals to get the graph $G_j$, which is also 2-edge connected on the terminals. The procedure of adding beads and removing Steiner nodes is explained later.
5: Output the resulting graph $G_m$.
```

Property 4.4: The angle between any pair of neighbors of a Steiner node in its tree ST_i is at least 60 degrees [8]. This can be seen from the fact that if the angle between two neighbors (x, y) at a Steiner node j were less than 60 degrees, the MST could be shortened by deleting an edge ($j - x$ or $j - y$) and forming the edge $x - y$.

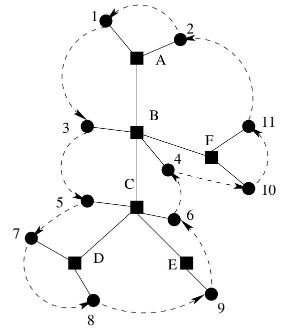
Let us now explain the procedure to construct G_j from G_{j-1} by adding beads between the terminal nodes and removing Steiner nodes. Consider a graph formed by the Steiner nodes in ST_j and the terminal nodes within the transmission range of these Steiner nodes. Call this graph H_j. We add a cycle using beaded (and direct) links between the terminals contained in H_j in G_{j-1} and delete the Steiner nodes of ST_j to get G_j. Thus, all the terminals in H_j are 2-edge connected to each other. This procedure does not create a cut-edge and maintains 2-edge connectivity between the terminal nodes which were 2-edge connected because of the Steiner nodes in ST_j. As we do this for all trees $ST_1, .., ST_m$, and do not create any cut-edge in any step, the resulting network G_m is 2-edge connected on the terminals[2]. Algorithm 3 describes the procedure of constructing such a cycle, which is also explained below.

For constructing a cycle among the terminal nodes connected to the Steiner nodes in ST_j, we start from a Steiner node st_1. We connect all the terminal nodes within its transmission range to st_1, and mark them. Let the set of marked terminal nodes be $\{t_1, .., t_k\}$. We start a Depth First Search (DFS) traversal of the tree (T_j) formed by $ST_j \cup \{t_1, .., t_k\}$, rooted at st_1, traversing the children of each node in an anti-clockwise manner. We start the DFS traversal with any terminal neighbor (say t_1) of st_1. Whenever a new Steiner node st_j is encountered in the traversal, connect all unmarked terminal nodes in its transmission range to it in T_j, and add them to the set of marked terminal nodes (thus k increases

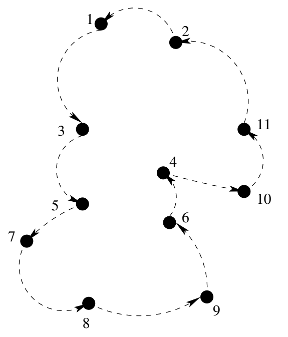
[2]We will show in the next section that this procedure does not even create a cut-vertex if the Steiner network is 2-vertex connected on terminals.



(a) Tree on Steiner nodes and terminals



(b) Depth first traversal and cycle creation



(c) Graph after removal of Steiner nodes

Fig. 2. Example for removal of Steiner nodes and addition of beads

Algorithm 3 Removal of Steiner nodes and addition of beads

```
1: Start at a Steiner node $st_1$.
2: Connect to it all terminals within its transmission range, and mark them.
3: Construct a tree $T_j$ rooted at $st_1$, with the vertex set as the Steiner nodes in $ST_j$ and a leaf vertex corresponding to each marked terminal node. The edges are the edges of $ST_j$ and the edges between each Steiner node and its terminal neighbors.
4: Do a Depth First Search (DFS) traversal of $T_j$, starting from any terminal neighbor of $st_1$. For each node, traverse its children in an anticlockwise manner.
5: Each time a new Steiner node $st_i$ is encountered, connect with it all unmarked terminal nodes in its range, and mark them. Update $T_j$ by adding these terminal nodes, and continue DFS traversal by going anticlockwise around $st_i$ from the edge between $st_i$ and its parent.
6: Remove the branches of $T_j$ which do not have any terminal nodes.
7: Connect all the terminal nodes in order of their DFS traversal and complete the cycle between them.
8: Add beads to all added edges of length greater than one.
9: Add the newly added edges to $G_{j-1}$, and remove the Steiner nodes of $ST_j$ and all incident edges from $G_{j-1}$. The resulting graph is $G_j$.
```

at this step). Then, remove the branches of T_j which do not have any terminal nodes. Example in Figure 2(a) shows such a tree at the end of the DFS traversal, with Steiner nodes $\{A, .., F\}$ and terminals $\{1, .., 11\}$. While doing the DFS traversal, add required number of beads to form a link between each terminal with the next terminal encountered in the DFS traversal. Figure 2(b) shows the construction of the cycle. The edges longer than unit length are added using the required number of beads. Then, remove the Steiner nodes. The resulting graph for the example is shown in Figure 2(c).

We now give a bound on the number of beads added while constructing G_j from G_{j-1}. The condition in Equation 2 states the bound, where b_j is the number of beads added and s_j is the number of Steiner nodes in ST_j. Then, Equation 3 bounds the total number of beads required, which proves Lemma 1.

$$b_j \leq 5s_j \tag{2}$$

$$b = \sum_{j=1}^{m} b_j \leq \sum_{j=1}^{m} 5s_j = 5s \tag{3}$$

We now prove the bound. We first define our charging scheme, i.e., how we charge the beads to the Steiner nodes in ST_j. We charge one bead to a Steiner node st_i each time one of the following ordered pair of edges is traversed:

- **Type I**: Steiner-st_i-Steiner.
- **Type II**: Steiner-st_i-Terminal, with the Euclidian distance between the end-nodes being more than one.
- **Type III**: Terminal-st_i-Steiner, with the Euclidian distance between the end-nodes being more than one.
- **Type IV**: Terminal-st_i-Terminal, with the Euclidian distance between the end-nodes being more than one.

In a DFS traversal, each ordered pair of neighboring edges around a node is traversed once. Notice that for the beads charged by the pair of edges of Type II,III,IV, the angle between the edges at the Steiner node st_i is *greater* than 60 degrees. For the pair of Type I, the angle is *at least* 60 degrees (by Property 4.4). The pairs of Type I around a Steiner node is bounded by five (by Property 4.4). Also, as the traversal is anti-clockwise always, the angles subtended by all these pairs of edges at st_i are non-overlapping. Thus, it can be easily shown that the total pairs of such edges around the Steiner node st_i in the tree T_j is within five as the total angle traversed by these edges is bounded by 360 degrees.

Every time we add an edge to connect two terminal nodes in T_j, we claim that the number of beads required (given by Equation 1) can be charged to the Steiner nodes encountered in the DFS traversal between the two terminal nodes using our charging scheme. Let the two terminal nodes to be connected be t_x and t_y, and let there be $l > 0$ Steiner nodes on the DFS path between them. Renumber the Steiner nodes on the DFS path as $st_1, .., st_l$. We consider the following cases and prove that the charging scheme charges the required number of beads to the Steiner nodes:

- **Case 1**: $l = 1$: This case is depicted in Figure 3(a). If both the terminals are connected to the same Steiner node

TABLE I

BEAD CHARGING FOR EXAMPLE OF FIGURE 2

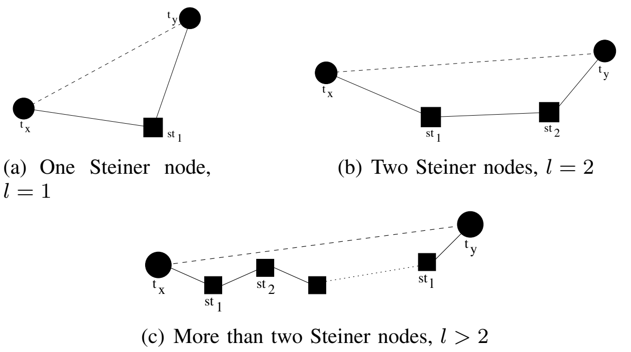
Edge	(1,3)	(3,5)	(5,7)	(7,8)	(8,9)	(9,6)	(6,4)	(4,10)	(10,11)	(11,2)	(2,1)
Charged To	A,B	B,C	C,D	D	D,C,E	E,C	C,B	B,F	F	F,B,A	A
No. of Beads	1	1	1	1	2	1	1	1	0	2	1

st_1, a bead is needed only if they are more than distance one apart. In that case, the pair of edges $t_x - st_1 - t_y$ is of Type IV and thus the Steiner node st_1 can be charged for the bead required.

- **Case 2**: $l = 2$: This case is depicted in Figure 3(b). Let the two Steiner nodes in the DFS path be st_1 and st_2, with st_1 being the parent of st_2 in the tree, i.e., the first encounter of st_1 in the DFS traversal is before st_2. Let t_x be connected to st_1 and t_y to st_2. Since st_1 is the parent of st_2, we connected all unmarked neighboring terminal nodes to st_1 first. Thus, t_y is more than distance one apart from st_1. If two beads are needed between t_x and t_y, the distance between t_x and st_2 is more than one and between st_1 and t_y is more than one. Thus the pairs of edges $t_x - st_1 - st_2$ and $st_1 - st_2 - t_y$ are Type III and II for Steiner nodes st_1 and st_2 respectively. Thus, one bead can be charged to each Steiner node. If we need one bead between t_x and t_y, that can be charged to st_2 as the pair of edges $st_1 - st_2 - t_y$ is always Type II for st_2. The explanation for the case where st_2 is the parent of st_1 is similar.
- **Case 3**: $l > 2$: This case is depicted in Figure 3(c). The total number of beads required is upper bounded by the number of Steiner nodes on any path between t_x and t_y. One bead can be charged to each of the nodes $st_2, .., st_{l-1}$ as the pair of edges involving them in the path are of Type I. If the distance between t_x and st_2 is less than one, the path can be modified by connecting t_x directly to st_2, and there is a path with $l-1$ Steiner nodes, and thus st_1 can be removed. If it is greater than one, then a bead can be charged to st_1 as the pair of edges $t_x - st_1 - st_2$ is of Type III. Similarly, st_l can either be removed from the path between t_x and t_y, or it can be charged because of the pair of edges $st_{l-1} - st_l - t_y$ being Type II. Thus, there is a path of $l-2$, $l-1$ or l Steiner nodes between t_x and t_y (which is the upper bound for the number of beads required), and there are enough Steiner nodes that can be charged once.

We have shown that the charging scheme charges the required number of beads to the Steiner nodes, and each Steiner node is charged maximum of five times. Adding over all connected components of Steiner nodes in the network, the total number of beads required is within five times the number of Steiner nodes. Thus, the relation of Equation 3 holds for the beaded network we constructed. Hence, the relation holds for the optimal 2-edge connected beaded network as well. □

For the example of Figure 2, Table I gives the Steiner nodes that can be charged for each added edge according to the charging scheme.



(a) One Steiner node, $l = 1$ (b) Two Steiner nodes, $l = 2$

(c) More than two Steiner nodes, $l > 2$

Fig. 3. DFS paths of different lengths

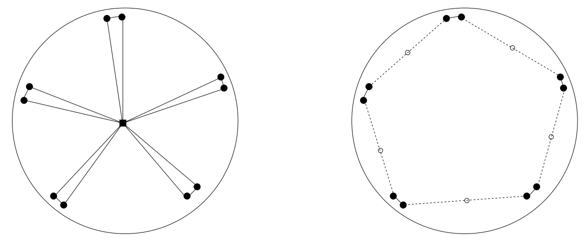

(a) Optimal Steiner node network (b) Optimal beaded network

Fig. 4. Approximation ratio tightness example

Now, we give an example to prove the bound of Equation 3 is tight. Consider the network in Figure 4(a). The circular nodes are terminal nodes, which are 2-edge connected using a single Steiner node in the middle of the circle in the optimal Steiner node solution. If we remove the Steiner node, the optimal network with beads will have a beaded link between every alternate pair of terminal nodes to have a cycle, and that would require five beads. The resulting network is shown in Figure 4(b).

Proof of Theorem 4.1: The algorithm by Khuller and Vishkin [14] is a 2-approximation for finding the minimum cost (cost of each edge being number of beads required to form it) k-edge connected subgraph. Thus, the number of beads required is at most $2 * 5s = 10s$. The last step of Algorithm 1 (sequential removal step) removes some relays from the network by allowing the relays to connect to all nodes within the transmission range, so the resulting network after sequential removal also has maximum of $10s$ relay nodes. □

V. ALGORITHM FOR k-VERTEX CONNECTIVITY

We propose an algorithm for achieving k-vertex connectivity among terminal nodes using relays. We follow the same

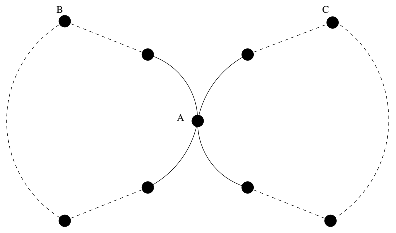


Fig. 5. Example cut-vertex generated while adding beads and removing Steiner nodes

framework as for edge-connectivity, i.e., add relays only on the line joining two terminal nodes. We follow the same algorithm as Algorithm 1, with some components changed. To find a minimum cost k-vertex connected spanning subgraph of a complete graph on terminal nodes, we use the 2-approximation algorithm by Khuller and Raghavachari [20] for $k = 2$, and the k-approximation algorithm by Kortsarz and Nutov [21] for $k > 2$. Also, in the sequential relay removal step (last step of the algorithm), we check for k-vertex connectivity rather than k-edge connectivity.

We briefly explain the algorithm for computing the approximate minimum-weight 2-vertex connected subgraph [20] of a graph G: Create a directed graph D having anti-parallel directed edges for each undirected edge in G, each having the same weight as the corresponding undirected edge. Pick any two vertices x, y in D. Augment D by adding a new vertex r and adding two new directed edges of weight 0 from r to x and y. Use the algorithm of Frank and Tardos [22] on this graph with r as the root to find a minimum weight subgraph H with two openly disjoint paths between r and every vertex of D. The algorithm of Frank and Tardos is based on submodular flows. Let $S \subseteq E$ be the set of edges in G such that at least one of the copies of each edge is in H. Since S was obtained from H, for any vertex v in G, there are two openly disjoint paths between v and x, y in S. The algorithm returns $S \cup e$ as the solution, where e is the edge (x, y). If repeated on all possible pairs (x, y), the algorithm is a 2-approximation of the optimal.

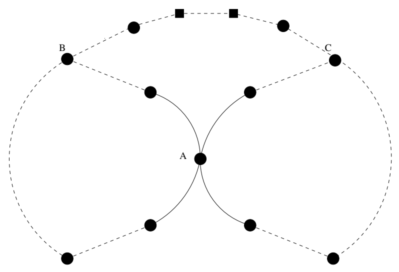
A. Proof of Approximation Ratio for 2-Vertex Connectivity

In this section, we prove that the vertex-connectivity algorithm is an O(1)-approximation for $k = 2$. Theorem 5.1 states the desired result. We follow the same terminology as the last section.

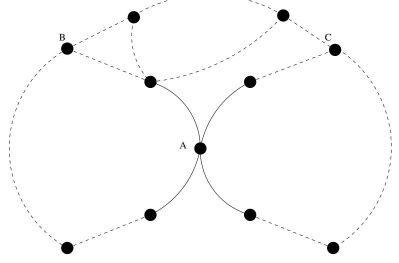
Theorem 5.1: If the optimal network uses s Steiner nodes so that terminals are 2-vertex connected, our algorithm forms a network with maximum of $10s$ beads and zero Steiner nodes, in which the terminal nodes are 2-vertex connected.

To prove Theorem 5.1, we prove the following lemma, and Theorem 5.1 follows directly as we use a 2-approximation for finding the minimum-cost (beads) beaded network.

Lemma 5.2: A network 2-vertex connected on terminal nodes using minimum number of beads contains at most $5s$ beads, where s is the minimum number of Steiner nodes needed.



(a) Path through terminals and Steiner nodes



(b) Path through terminals and beaded cycles

Fig. 6. Existence of alternate path

Proof: We follow exactly the same constructions and proof as the proof for 2-edge connectivity, so the maximum number of beads required is $5s$. Thus, we only need to prove that the beaded graph constructed by Algorithm 2 is 2-vertex connected on terminals. Thus, we need to prove that we do not create any cut-vertex while removing the connected components of Steiner nodes $ST_1, .., ST_m$ and adding beads between terminal nodes at each step using Algorithm 3. As the optimal network using Steiner nodes is 2-vertex connected on terminals, if we do not create any cut-vertex while removing Steiner nodes and adding beads, the resulting network will be 2-vertex connected.

We prove the 2-vertex connectivity of the beaded network by contradiction. Assume the procedure of Algorithm 2 creates a cut-vertex while using Algorithm 3 for adding beads. Figure 5 shows such a cut-vertex, and two cycles it belongs to. Dotted lines indicate the path through multiple terminal nodes (with beaded or non-beaded links). The cut-vertex A is a part of two cycles created by removing two connected components of Steiner nodes. As A is a cut-vertex, all paths between B and C are through A, as else A would not be a cut-vertex. If the network was 2-vertex connected on the terminal nodes when all the Steiner nodes were present, then B and C had two vertex-disjoint paths between them. Let the connected component of Steiner nodes corresponding to left cycle containing B be ST_i and for the right cycle containing C be ST_j, $i \neq j$. Of the two vertex disjoint paths between B and C before the removal of these Steiner nodes, one must have been through A. As the two components did not have any direct edge between them, so the second path between B and C must have been through some Steiner nodes not belonging to ST_i or ST_j and terminal nodes other than A, as shown in Figure 6(a) (square nodes are Steiner nodes). If those Steiner nodes have not been removed yet, the path still exists.

If they have been removed, all the terminal nodes connected to them have been connected through a cycle, and thus internally vertex-disjoint (disjoint from first path) path between B and C not passing through A still exists, as shown by the example in Figure 6(b). Thus, B and C still have two vertex disjoint paths after removal of ST_i and ST_j, and thus A is not a cut-vertex. So, the procedure of adding beads and removing Steiner nodes (Algorithm 3) does not create any cut-vertex, and the beaded network constructed by Algorithm 2 is 2-vertex connected on terminals. □

VI. Extension to Higher Dimensions

We consider the extension to other metric spaces in this section. Let the terminal nodes be placed in any metric space with MST number d_{MST} [23]. MST number is defined as the maximum possible degree of a minimum-degree minimum spanning tree (MST) spanning points from the space. The approximation ratio for the MST based algorithm of [6] for connecting terminals using minimum relays has been shown to be $d_{MST} - 1$ in [7]. The MST number for a Euclidian plane is 5, a three-dimensional Euclidian space is 13, and a rectilinear plane is 4.

We can prove that the algorithms for 2-edge connectivity and 2-vertex connectivity are $2d_{MST}$-approximation. We omit the proof here due to lack of space.

VII. Generalization to restricted relay placement

We extend our results for terminals distributed on a Euclidian plane to the scenario where relays cannot be placed in certain polygonal regions of the network . We call these regions as forbidden regions. We assume that two nodes can communicate if they are within each other's transmission range even when there is a forbidden region between them. We modify the edge and vertex connectivity algorithms to work with the same approximation guarantees for this generalization.

We follow the same algorithms as before for both edge connectivity and vertex connectivity. It may not be possible to connect two terminals by placing relay nodes on the straight line between them due to the forbidden regions. Thus, Equation 1, which represents the number of relays needed to connect two terminals by placing relays on the line between them, cannot be used to weight the edges of the network formed on terminal nodes in our algorithms. Recently, a polynomial time algorithm has been proposed for placing the minimum number of relay nodes needed to form a link between two nodes with the presence of polygonal forbidden regions between them [24] . The problem is called the puddle-jumper problem. We modify our edge weights by running the algorithm given in [24] on each pair of terminals in the network to find the minimum number of relay nodes needed for each link, and using that as the weight of each edge. We then run our edge connectivity and vertex connectivity algorithms on a network with these edge weights. Then, for the selected links, we place the relays according to the algorithm given in [24].

A. *Proof of approximation ratio*

We now prove that the approximation ratio for the 2-edge and 2-vertex connectivity algorithms is 10. We follow the same construction as before, the only change being that beads (relay nodes) are not placed on straight lines between terminal nodes now; instead they are placed optimally taking forbidden regions into account. The only part of the proof that needs reconsideration to take forbidden regions into account is when Steiner nodes on a tree (ST_j) are removed from the optimal Steiner solution and beads are placed to make the cycle between terminal nodes connected to tree ST_j (see Algorithm 3). We argue that the number of relays needed to form a beaded link between two terminals is still upper bounded by the number of Steiner nodes encountered in the depth first traversal between the two terminals: Take any two terminals being connected using beads, and let a be the number of Steiner nodes on the DFS path between them. Thus, there is a placement of Steiner nodes to connect the two terminal nodes using a Steiner nodes. As even Steiner nodes could not be placed in forbidden regions, and we connect the terminals using beads placed according to the optimal algorithm of [24], the number of beads required is upper bounded by a. Thus, each bead can still be charged to a different Steiner node on the DFS path between the terminals. We showed in Section IV that each Steiner node is charged at most 5 times, so the total number of beads required for replacing the Steiner node tree ST_j is still $5s_j$, s_j being the number of Steiner nodes in the ST_j. Thus the total number of beads required in the network is at most $5s$ for the beaded network using minimum number of beads, s being the number of optimal Steiner nodes. As our algorithms use 2-approximations for finding the optimal beaded network, the algorithms are 10-approximations.

VIII. Worst case and expected value bounds

In this section, we provide some bounds on the number of relays required to connect a set of terminals. We provide the bounds for the algorithm of [6]. The algorithm weights each edge among the terminal nodes by the number of relay nodes needed to form the edge, and finds a minimum spanning tree (MST) on the terminal nodes. The weight function is given in Equation 1, where $|e|$ is the edge length. We first provide a set of worst case bounds as a function of the transmission range of the nodes and independent of the number of terminal nodes. We then use another approach to show that the upper bound on number of relays required increases and then becomes constant as the number of nodes in the network increases. We then show that the expected number of relays required for achieving $(2^d - 1)$-vertex connectivity among terminal nodes distributed in a d-dimensional space goes to zero as the number of terminals goes to infinity.

We work with terminal nodes distributed in a unit-length cuboid in a d-dimensional space. Let the transmission range of each node be Δ $(= 1/l$, where l is the length of the network when transmission range is assumed to be 1). We denote the terminal nodes by points $\{x_1, x_2, .., x_n\} \subset [0, 1]^d$, and take

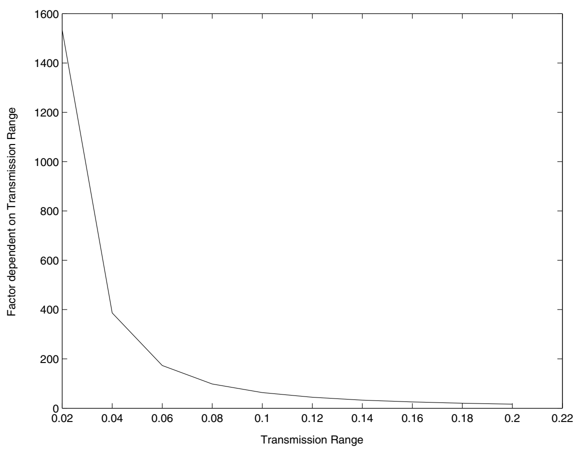


Fig. 7. Relationship between upper bound on number of relays and transmission range in MST-based algorithm

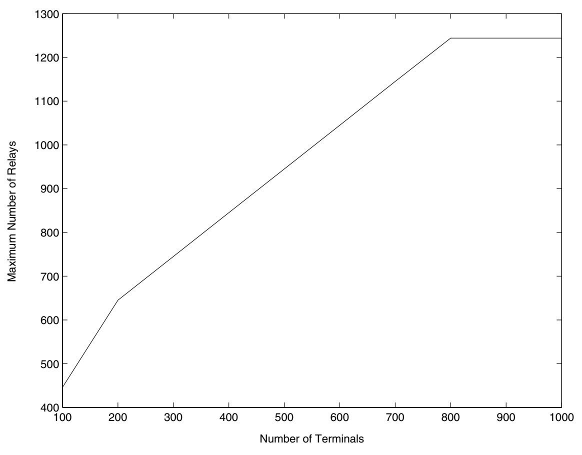


Fig. 8. Upper bound on number of relays for varying number of terminals in MST-based algorithm

the weight of each edge to be $\psi(e) = |e|$, i.e., the Euclidean length of the edge.

A. Worst case bound as function of transmission range

We give bounds on the number of relays required in the MST with respect to the transmission range of the nodes, and independent of the number of terminals. Let T_0 denote a minimum weight spanning tree on the terminals, according to the weight function $\psi(e) = |e|$. Let $\nu_d(x)$ denote the number of edges in T_0 having length greater than x. We start with the following Lemma from [25].

Lemma 8.1: There is a constant β_d depending only on the dimension $d \geq 1$ such that

$$\nu_d(x) \leq \beta_d x^{-d} \tag{4}$$

We use the result to bound the number of relays we require for forming a tree between the terminal nodes using the algorithm of [6]. We need a relay node for every Δ length of a link. We group the edges of the MST into p groups according to their lengths. Let the group C_i contain edges with length between $i\Delta$ and $(i+1)\Delta$. The number of groups, p, is $\lceil \sqrt{d}/\Delta \rceil$, where $\sqrt{d}$ is the maximum edge length in the network.

$$C_i = \{e | i\Delta < |e| \leq min\{(i+1)\Delta, \sqrt{d}\}\}, i = 0, .., p-1 \tag{5}$$

The number of relays needed to form an edge in group C_i is i. Let $b(\Delta)$ denote the total number of relays needed to form all edges in the MST. We use Lemma 8.1 and sum over the groups C_i to find the upper bound for the number of relays required. The number of elements in group C_{p-1} is bounded as $|C_{p-1}| \leq \beta_d((p-1)\Delta)^{-d}$. Equation 6 gives the bound for other sets. The total number of beads required is given by $\sum_{i=1}^{p-1} i|C_i|$, which is the sum of left hand side of the equations in Equation 6. By adding the equations from $j = 1$ to $p - 1$, we get the bound on the total number of beads in Equation 7, which we call $B(\Delta)$. Here, H^d_{p-1} is harmonic number of order $(p - 1)$ of power d. Figure 7 plots the variation of $B(\Delta)/\beta_d$ for $d = 2$. As can be seen, the upper bound on the number of relays required decreases exponentially with the increase in transmission range of the nodes.

$$\sum_{i=1}^{j} |C_{p-i}| \leq \beta_d((p-j)\Delta)^{-d}, j = 1, .., p-1 \tag{6}$$

$$\begin{aligned} b(\Delta) &\leq \sum_{i=1}^{p-1} i|C_i| \leq \sum_{i=1}^{p-1} \beta_d(i\Delta)^{-d} \\ &= \beta_d \Delta^{-d} \sum_{i=1}^{p-1} i^{-d} = \beta_d \Delta^{-d} H^d_{p-1} \end{aligned} \tag{7}$$

B. Worst case bound as function of number of nodes

We now provide the bounds with respect to the number of nodes in the network. We use some observations from [25] to provide the bounds. Let T_0 denote a minimum weight spanning tree on the terminals, and E denote the set of edges in T_0. We follow Kruskal's algorithm [26] for constructing an MST: Join two nearest points, then iteratively join two nearest connected components. Then, let $e_i, 1 \leq i < n$, denote the edges in the order in which they are added to T_0. We have the upper bound of Equation 8 on the edge lengths, where e_i is the edge added in ith step of Kruskal's algorithm.

$$|e_i| \leq \alpha_d(n - i + 1)^{-1/d} \tag{8}$$

To prove this, the following result is used: If there is a set of m points $\{x_1, x_2, .., x_m\}$ in $[0, 1]^d$, then one can select a pair of points x_i and x_j with $|x_i - x_j| \leq \alpha_d m^{-1/d}$, where α_d is a constant. A direct application of pigeonhole principle leads to this result. Also, $\alpha_d = 2\sqrt{d}$ suffices. At ith iteration of Prim's algorithm, there are $n - i + 1$ components in the network, and the result of Equation 8 follows.

The number of relays required (B) can then be upper bounded as in Equation 9. Here, Δ is the transmission range of the nodes when network length is one. As the equation shows, the bound on number of relays required becomes a constant as the number of terminals (n) increases beyond a threshold. The expected behavior would be to have the number of relays increase with number of terminals, and then start decreasing as n increases beyond a threshold. That is true for the average case, but in the worst case, it does not decrease as one can put new terminal nodes at the same positions as the existing nodes, and thus the number of relays required does not decrease.

$$\begin{aligned} B &\leq \sum_{i=1}^{n-1}(\lceil\frac{\alpha_d(n-i+1)^{-1/d}}{\Delta}\rceil - 1) \\ &= \sum_{i=2}^{n}(\lceil\frac{\alpha_d i^{-1/d}}{\Delta}\rceil - 1) \\ &= \begin{cases} \sum_{i=2}^{n}(\lceil\frac{\alpha_d(i)^{-1/d}}{\Delta}\rceil - 1) & \text{if } n \leq \lfloor(\alpha_d/\Delta)^d\rfloor \\ \sum_{i=2}^{\lfloor(\alpha_d/\Delta)^d\rfloor}(\lceil\frac{\alpha_d(i)^{-1/d}}{\Delta}\rceil - 1) & \text{otherwise} \end{cases} \end{aligned} \quad (9)$$

For $d = 2$ and $\alpha_2 = 2\sqrt{2}$, we plot the bound in Figure 8. The transmission range is fixed at 0.1 and n is varied. As we can see, the bound on the number of relays required increases in a piecewise linear manner and becomes constant after a certain threshold (which depends on the transmission range of the nodes). Using the same analysis the bound on the length of the MST has been shown to increase as $O(n^{(d-1)/d})$ [25].

C. Asymptotic bound on number of relays

In this section, we show that the mean number of relays required goes to zero as the number of terminals goes to infinity. The terminals are assumed to be uniformly and independently distributed in $[0,1]^d$. The analysis holds for number of relays required to achieve up to $(2^d - 1)$-vertex connectivity between terminals (which implies up to $(2^d - 1)$-edge connectivity as well). The result is stated in Lemma 8.2.

Lemma 8.2: If n terminal nodes are i.i.d. uniformly distributed in $[0,1]^d$, the number of relays required to achieve $(2^d - 1)$-vertex ($(2^d - 1)$-edge) connectivity between terminals goes to zero in the mean value as n goes to infinity, i.e.,

$$\lim_{n\to\infty} E[Relays] \to 0 \quad (10)$$

Proof: Let the transmission range of each node be $\Delta < 1$. We divide the network volume into d-cuboids of edge-length $\Delta/(2\sqrt{d})$, such that the distance between the farthest points in two neighboring cuboids is Δ. Thus, if we have a terminal node in each cuboid, the network will be a grid, and in the case of a single terminal in a corner cuboid, that terminal will have minimum $(2^d - 1)$ neighbors, and all non-corner cuboids have number of neighbors greater than $(2^d - 1)$. Thus, the grid network is $(2^d - 1)$-vertex (and edge) connected. Let n_i denote the random variable for the number of terminals in cuboid i. The number of cuboids in the network is $(2\sqrt{d}/\Delta)^d$, which we denote by $N(\Delta, d)$. Equation 11 gives the probability that a cuboid i has zero terminal nodes when n terminal nodes are i.i.d. uniformly distributed in $[0,1]^d$. The number of relays required to form a grid network is equal to the number of empty cuboids in the network. Thus, the expected number of relays required can be written as in Equation 12.

$$P\{n_i = 0\} = (1 - 1/N(\Delta, d))^n \doteq p, i = 1, .., N(\Delta, d) \quad (11)$$

$$E[Relays] = N(\Delta, d)p = N(\Delta, d)(1 - 1/N(\Delta, d))^n \quad (12)$$

As n approaches infinity, $(1 - 1/N(\Delta, d))^n$ approaches zero, and thus the relation of Equation 10 holds. □

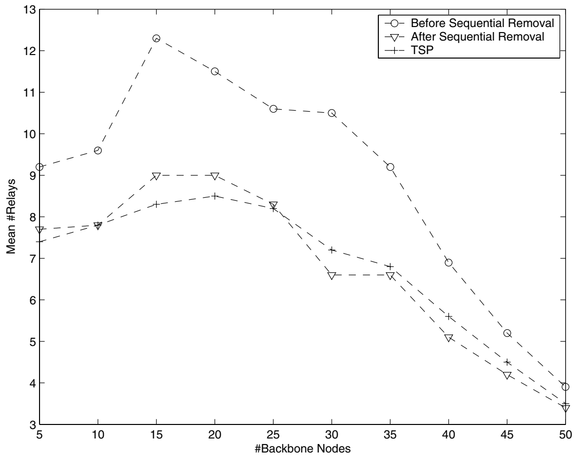


Fig. 9. Mean number of relays for 2-edge connectivity, $\Delta = 0.2$, $d = 2$

IX. Simulation Results and Discussion

We generate a random network in a unit length square in a two dimensional space. The nodes are located uniformly and randomly in the network. We simulate the algorithm for k-edge connectivity for $k = 2, 3$. We note the results for number of relays required both at the output of Khuller and Vishkin's approximation algorithm [14], and after sequential removal of relays ensuring k-edge connectivity (see Algorithm 1). For 2-edge connectivity, we propose another algorithm that formulates the problem as traveling salesperson problem (TSP), and compare the results with that. The algorithm constructs a complete graph with edge weights defined as in Equation 1, and find a TSP tour on the graph. The total weight of the tour represents the number of relays required. We use Concorde [27] to compute the optimal TSP tour.

We first fix the transmission range of the nodes at 0.2 and vary the number of backbone nodes in the network. We randomly generate backbone node locations 10 times. Figures 9 and 10 show the average and maximum number of relays required (over 10 simulations) for 2-edge connectivity for varying number of backbone nodes. The average number of relays required increases with the number of backbone nodes, and then decreases as the number of backbone nodes goes beyond a threshold. The output of Algorithm 1 is nearly the same as the TSP output. The performance of Algorithm 1 gets slightly better than the TSP performance as the number

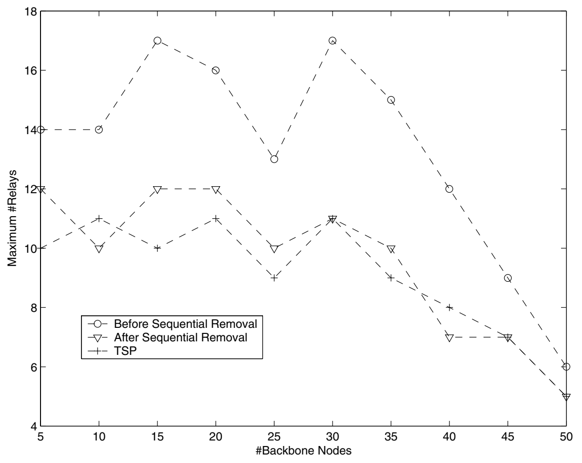


Fig. 10. Max. number of relays for 2-edge connectivity, $\Delta = 0.2$, $d = 2$

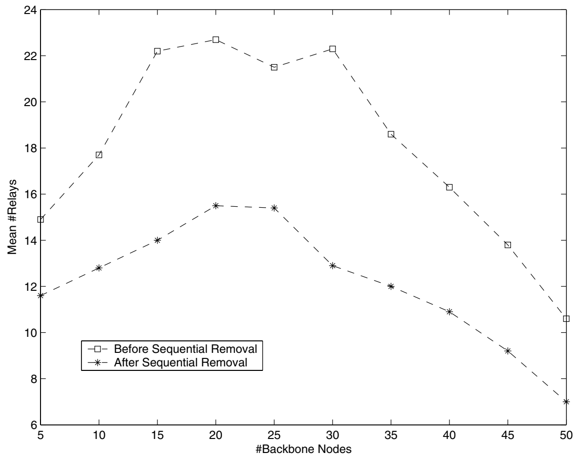


Fig. 11. Mean number of relays for 3-edge connectivity, $\Delta = 0.2$, $d = 2$

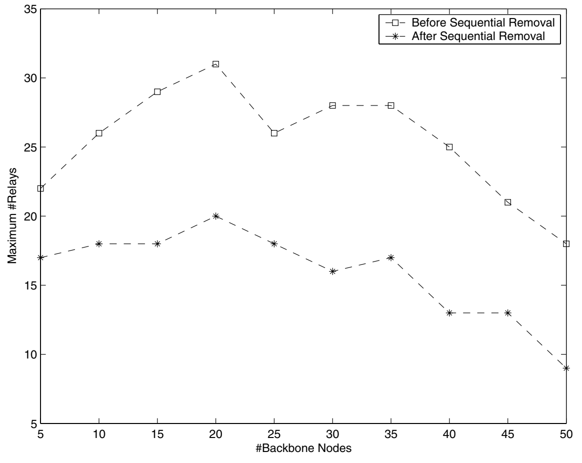


Fig. 12. Max. number of relays for 3-edge connectivity, $\Delta = 0.2$, $d = 2$

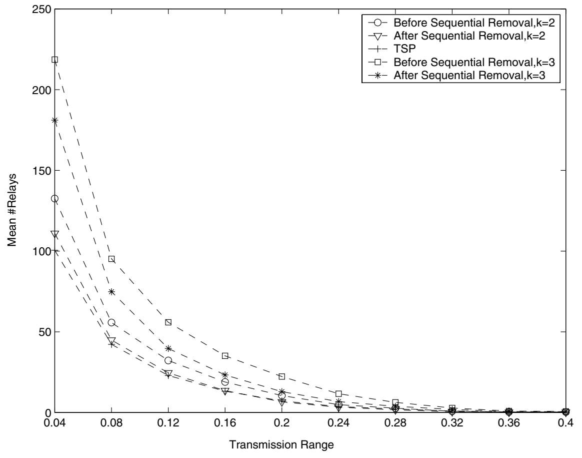


Fig. 13. Mean number of relays needed for varying transmission range, $N = 30$, $d = 2$

of terminals is increased. Although the TSP algorithm works almost as well as our algorithm, it can be shown that it has an approximation ratio of infinity. The reason is that while 2-edge connected components of terminals in the network may not need any relays, forming a TSP tour among them may need relays. Thus, the worst case performance of TSP may be very bad on certain networks. The other advantage of our algorithm is that it can be used to construct networks with connectivity levels higher than two as well, while the TSP based algorithm works only for 2-connectivity.

Figures 11 and 12 show the average and maximum number of relays required (over 10 simulations) for 3-edge connectivity for varying number of backbone nodes. The number of relays required follows the same pattern as for 2-edge connectivity, and the gains achieved by sequential relay removal step are more than in the 2-edge connectivity case. This is because there are many more relays in the vicinity of each other in the output before sequential removal in the case of 3-edge connectivity. Thus, when allowed to form links in the neighborhood, the fraction of redundant relays (which can be removed) is much more for $k = 3$ than $k = 2$.

We now fix the number of backbone nodes at 30, and vary the transmission range from 0.02 to 0.4. The set of locations of the backbone nodes is generated randomly 10 times, and simulations for each transmission range are run on each of those networks. Figures 13 and 14 show the mean and maximum number of relays required over the simulations for 2- and 3-edge connectivity. The number of relays required decreases exponentially with the transmission range of the nodes. Also, the performance of our algorithm is similar to the TSP output. It was shown in Section VIII that the maximum relays required for connectivity decreases exponentially with the transmission range of the nodes. The simulations show a similar behavior for 2-edge and 3-edge connectivity.

It is worthwhile to note that the results show similar behavior if we increase/decrease the network area or increase the number of backbone nodes. The number of backbone

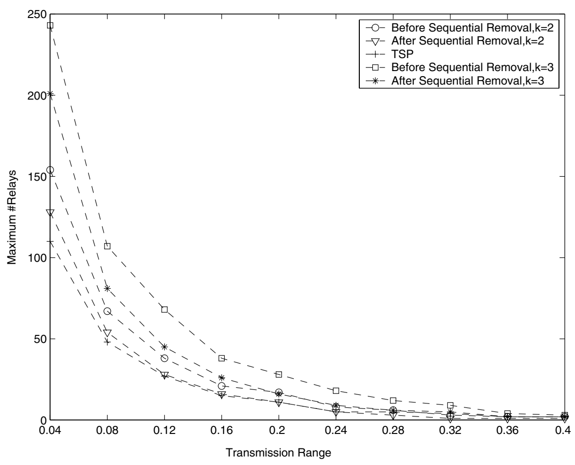


Fig. 14. Maximum number of relays needed for varying transmission range, $N = 30$, $d = 2$

nodes in the simulations represent actual sizes of backbone networks, and so the results represent the expected behavior of the algorithms.

X. Conclusion and Future work

We consider the problem of constructing a fault-tolerant backbone network using additional relay nodes. We give $O(1)$-approximation algorithms for 2-edge and 2-vertex connectivity in terms of the number of relay nodes required. The bound for 2-edge connectivity is proved to be tight. The algorithms also work for achieving k-connectivity for higher values of k. We analyze previously proposed algorithms for achieving connectivity in a network, both for deterministic worst case bounds and probabilistic expected value bounds. Simulations show our 2-edge connectivity algorithm gives results similar to an algorithm based on traveling salesperson problem, which may perform infinitely worse relative to the optimal and cannot be extended to work for connectivity higher than two. Simulations also show the behavior of the proposed algorithms for k-edge connectivity is same as the bounds shown for connectivity algorithms. We extend our algorithms to work with the same approximation guarantees for the generalization where the relay nodes cannot be placed in certain polygonal regions of the network.

There are some open problems to be considered in future work. The first problem is the derivation of an approximation ratio for the proposed algorithms for $k > 2$. An $O(1)$ approximation ratio may exist for $k > 2$. Another extension of the work is to place the relay nodes more intelligently, and be able to bound the performance of the algorithm relative to the optimal, i.e., provide an approximation ratio for the corresponding algorithm.

Acknowledgment

The authors would like to thank Amol Deshpande for useful discussions on sensor networks.

References

[1] G. Gupta and M. Younis, "Fault-tolerant clustering of wireless sensor networks," *IEEE WCNC*, pp. 1579–1584, 2003.

[2] ——, "Load-balanced clustering of wireless sensor networks," *IEEE Intl. conf. on Comm.*, pp. 1848–1852, 2003.

[3] J. Pan, Y. T. Hou, L. Cai, Y. Shi, and S. X. Shen, "Topology control for wireless sensor networks," *IEEE/ACM Mobicom*, pp. 286–299, 2003.

[4] B. Hao, J. Tang, and G. Xue, "Fault-tolerant relay node placement in wireless sensor networks: Formulation and approximation," *IEEE Workshop on High Performance Switching and Routing*, pp. 246–250, 2004.

[5] H. Liu, P. Wan, and X. Jia, "Fault-tolerant relay node placement in wireless sensor networks," *International Computing and Combinatorics Conference (COCOON)*, 2005.

[6] G.-H. Lin and L. Wang, "Steiner tree problem with minimum number of steiner points and bounded edge-length," *Information Processing Letters*, vol. 69, pp. 53–57, 1999.

[7] I. Măndoiu and A. Zelikovsky, "A note on the mst heuristic for bounded edge-length steiner trees with minimum number of steiner points," *Information Processing Letters*, vol. 75(4), pp. 165–167, 2000.

[8] 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," *Theoretical Computer Science*, vol. 262, pp. 83–99, 2001.

[9] X. Cheng, D.-Z. Du, L. Wang, and B. Xu, "Relay sensor placement in wireless sensor networks," *ACM Winet*, 2004.

[10] P. Gupta and P. R. Kumar, "Critical power for asymptotic connectivity in wireless networks," *Stochastic Analysis, Control, Optimization and Applications: A Volume in Honor of W.H. Fleming, W. M. McEneaney, G. Yin, and Q. Zhang (Eds.)*, 1998.

[11] C. Bettstetter, "On the connectivity of ad hoc networks," *The Computer Journal*, vol. 47(4), pp. 432–447, 2004.

[12] X.-Y. Li, P.-J. Wan, Y. Wang, and C.-W. Yi, "Fault tolerant deployment and topology control in wireless networks," *IEEE/ACM MobiHoc*, pp. 117–128, 2003.

[13] M. Garey and D. Johnson, *Computers and Intractability: A guide to the theory of NP-Completeness*. Freeman and Company, 1979.

[14] S. Khuller and U. Vishkin, "Biconnectivity approximations and graph carvings," *Journal of the ACM*, vol. 41(2), pp. 214–235, 1994.

[15] H. N. Gabow, "A matroid approach to finding edge connectivity and packing arborescences," *IEEE Annual Symposium on Foundations of Computer Science*, pp. 812–822, 1991.

[16] E. L. Lawler, *Combinatorial optimization: Networks and matroids*. Holt, Rinehart and Winston, New York, 1976.

[17] A. Frank, "A weighted matroid intersection algorithm," *Journal of Algorithms*, vol. 2, pp. 328–336, 1981.

[18] J. Roskind and R. E. Tarjan, "A note on finding minimum-cost edge-disjoint spanning trees," *Mathematics of Operations Research*, vol. 10(4), pp. 701–708, 1985.

[19] C. Monma and S. Suri, "Transitions in geometric minimum spanning tree," *Discrete and Computational Geometry*, vol. 8, pp. 265–293, 1992.

[20] S. Khuller and B. Raghavachari, "Improved approximation algorithms for uniform connectivity problems," *Journal of Algorithms*, vol. 21(2), pp. 434–450, 1996.

[21] G. Kortsarz and Z. Nutov, "Approximating node connectivity problems via set covers," *Algorithmica*, vol. 37, pp. 75–92, 2003.

[22] A. Frank and E. Tardos, "An application of submodular flows," *Linear Algebra and its Applications*, vol. 114/115, pp. 320–348, 1989.

[23] G. Robins and J. S. Salowe, "Low-degree minimum spanning trees," *Discrete Computational Geometry*, vol. 14, pp. 151–165, 1995.

[24] E. Arkin, E. Demaine, and J. Mitchell, "The puddle-jumper problem," *Personal Communication*, 2005.

[25] J. M. Steele, "Growth rates of euclidian minimal spanning trees with power weighted edges," *The Annals of Probability*, vol. 16(4), pp. 1767–1787, 1988.

[26] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*, 2nd ed. The MIT Press, 2001.

[27] Concorde, *http://www.tsp.gatech.edu/concorde.html*.