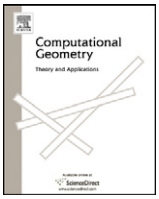
Computational Geometry 44 (2011) 206–215

Contents lists available at ScienceDirect

Computational Geometry: Theory and Applications

www.elsevier.com/locate/comgeo



Relay placement for fault tolerance in wireless networks in higher dimensions ☆

Abhishek Kashyap [a,*], Samir Khuller [b], Mark Shayman [a]

[a] Electrical and Computer Engineering, University of Maryland, College Park, United States
[b] Computer Science, University of Maryland, College Park, United States

ARTICLE INFO

Article history:
Received 4 August 2008
Accepted 17 November 2010
Available online 20 November 2010
Communicated by J. Mitchell

Keywords:
Fault tolerance
Network connectivity

ABSTRACT

In this paper we consider the problem of adding the smallest number of additional (relay) nodes to a network of static nodes with limited communication range so that the induced communication graph is 2-connected (we consider both edge and vertex connectivity). The problem is *NP*-hard. We develop algorithms that find close to optimal solutions for both edge and vertex connectivity. We prove the algorithms have an approximation ratio of $2M$ for nodes distributed in a d-dimensional Euclidean space, where M is the maximum node degree of a 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).

1. Introduction

We consider a wireless network of static nodes with limited communication range, where the induced communication graph may not be connected. We address the problem of providing connectivity and fault tolerance to the network.

We define fault tolerance as the existence of multiple internally vertex-disjoint (or edge-disjoint) paths between each pair of the static wireless nodes (terminal nodes). If k-vertex (edge) disjoint paths exist between each pair of nodes, the network is said to be k-vertex (edge) connected. A k-vertex (edge) connected graph has the property that the failure of any set of $(k-1)$ nodes (edges) cannot disconnect the network.

We use additional relay nodes, whose position we can control, to achieve the desired level of connectivity (number of vertex or edge disjoint paths) among the terminal nodes. We consider the problem to provide 2-(edge, vertex) connectivity among the terminal nodes using minimum number of relay nodes. The relay nodes are assumed to have same communication capabilities as terminal nodes, which have a fixed transmission range. We prove the algorithms have an approximation ratio of $2M$, when nodes are distributed in a d-dimensional Euclidean space, where M is the maximum node degree of a Minimum Spanning Tree in that space. We also extend the algorithms to provide k-connectivity, for $k \geqslant 2$.

An application of our work is to wireless sensor networks. A wireless sensor network is a group of sensor nodes with sensing, processing and communication capabilities, deployed to achieve a certain objective [19]. Typical applications of sensor networks are habitat monitoring, environmental monitoring, object tracking, etc. Sensor networks may exist in harsh network conditions, thus the network must be designed so that failure of some sensor nodes or some communication links between them does not disrupt the network.

☆ This research was partially supported by AFOSR under grant F496200210217, NSF under grant CNS-0435206, NSF CCF-0430650, NSF CNS-0519554. A preliminary version of this work (containing approximation analysis for networks in Euclidean plane) was published in Kashyap et al. (2006) [15]. The work was also published in first author's Ph.D. thesis (Kashyap, 2006) [14].

* Corresponding author.
E-mail addresses: abhi.kashyap@gmail.com (A. Kashyap), samir@cs.umd.edu (S. Khuller), shayman@ece.umd.edu (M. Shayman).

0925-7721/$ – see front matter
doi:10.1016/j.comgeo.2010.11.002

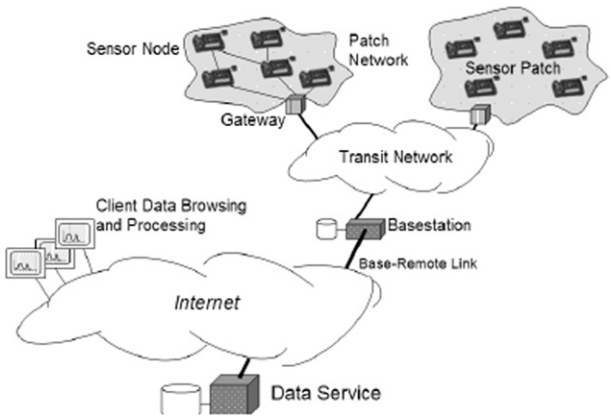


Fig. 1. Example application — Storm Petrel monitoring.

Fig. 1 shows an example sensor network [24] our work targets. The network was deployed for monitoring of birds, with patches of sensor nodes deployed around and inside burrows, each patch having a gateway (or multiple so it is not a single point of failure). The gateways are connected to a base station through a transit network. Sensor and gateway nodes have very limited energy. Thus, they transmit at low power levels, and have a limited transmission range. It may not be feasible to construct even a connected topology among the nodes due to their short transmission range and deployment in far-away regions. This example demonstrates the typical application of our work: the design of fault-tolerant transit network topology, where sensors are deployed in distant areas of interest.

There has been recent work in topology control of sensor networks. For example, Bredin et al. [3] present an $O(1)$-approximation algorithms for achieving k-vertex connectivity among both terminal nodes and the added relay nodes using minimum number of relays. Their analysis is for nodes distributed in a Euclidean plane. The approximation ratio they prove is $O(k^4)$, which turns out to be 1152 for $k=2$. Their algorithms use a similar approach as ours and their analysis for the component that achieves the same objective as ours gives an approximation ratio of 48 for $k=2$, while our analysis proves the approximation ratio to be 10 for nodes distributed in a Euclidean plane. We analyze our algorithm for $k=2$ for nodes distributed in Euclidean spaces of fixed dimension and prove a ratio of $2M$, where M is the maximum node degree in a minimum-degree Minimum Spanning Tree on nodes distributed in the Euclidean space. Note that M is five for a Euclidean plane, and thus the analysis gives a ratio of 10 for Euclidean plane, which is the same as proven by the analysis for the Euclidean plane in [15]. This is the first approximation bound for higher-dimensional Euclidean spaces for $k>1$. Hao et al. [13] consider the problem of placing the minimum number of backbone nodes (relays) 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. Misra et al. [26] provide $O(1)$-approximations for one and two-connectivity problems, where relay node placement is constrained to certain candidate locations. Liu et al. [22] consider the problem of placing relays in a network 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 relays needed for connectivity to the number of sensor nodes. Their problem restricts the set of relays to be a dominating set among the sensor nodes, i.e., each sensor node should be directly connected to at least one relay node. Srinivas et al. [30] consider a generalization of the same problem, where relay nodes have a communication radius much larger than the sensor nodes. The problem is to use minimum relay nodes such that they form a dominating set, and the graph on relay nodes is connected.

The problem of constructing a connected network on terminal nodes using a minimum number of relay nodes has been considered in [4,21,25]. In [21], the problem is shown to be *NP*-Hard and an approximation algorithm for constructing a tree using relay nodes is given, and the algorithm is shown to be a 5-approximation. The algorithm restricts the placement of relay nodes on lines joining pairs of terminal nodes. It then assigns a weight function to each pair of terminal 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 [25] and [4], and the bound is proved to be tight. Măndoiu and Zelikovsky [25] prove the approximation ratio to be $M-1$ for nodes distributed in higher dimension metric spaces. Chen et al. [4] also provide a 3-approximation algorithm for the problem. Cheng et al. [5] provide a 2.5-approximation randomized algorithm for placement of relay nodes to connect a given set of terminal nodes.

Authors in [7,12,23,31] consider the problems of one and two-connectivity where relay nodes have higher communication range (equal to R) than terminals (r), in Euclidean plane. They define one-tier problem where a path between each pair of terminals can go through both terminals and relays, and two-tier problem where only relays can be used in each path between terminals. Efrat et al. [7] provide a PTAS for the two-tier connectivity problem, and a 3.11-approximation for the one-tier problem. They also prove that no PTAS exists for the one-tier problem. Zhang et al. [31] provide a 14-approximation

for the one-tier two-connectivity problem and $(10+\epsilon)$-approximation for the two-tier two-connectivity problem. Lloyd and Xue [23] present a 7-approximation algorithm for the one-tier and a $(5+\epsilon)$-approximation algorithm for the two-tier one-connectivity problem. Han et al. [12] study several versions of this problem, where sensor nodes have different communication radii.

The contributions of this paper are as follows: (1) we provide algorithms for using relays to achieve k-(edge, vertex) connectivity among terminals; (2) we prove the algorithms to be O(1)-approximation with respect to the optimal for $k=2$, for nodes distributed in a Euclidean space of fixed dimension; (3) we extend our algorithms to the generalization where the relays cannot be placed in certain polygonal regions and show the same approximation ratios hold for this generalization as well.

The paper is organized as follows: Section 2 gives the network model and problem statement. Section 3 describes the proposed algorithm for achieving k-edge connectivity and gives the proof of approximation ratio for 2-edge connectivity. Section 4 describes the k-vertex connectivity approximation algorithm, and proves the approximation ratio for $k=2$. Section 5 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 6 concludes the paper.

2. Network model and problem statement

We model the network as a graph $G=(V,E)$, where V is the set of sensor 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. It is assumed that a node can connect to all nodes within its transmission range. The network can be in any d-dimensional Euclidean space (with fixed d), i.e., the distance between two nodes is considered to be the Euclidean distance between them. A link $e=(x,y)$ belongs to E if nodes x and y are within unit distance of each other.

We assume we have relay nodes that 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) such that the set of nodes V is k-edge (vertex) connected ($k\geqslant 2$) in the resulting graph $G'=(V',E')$, $V\subseteq V'$, $E\subseteq E'$. The objective is to construct a graph such that $\forall u,v\in V, \lambda(u,v)\geqslant k$; where $\lambda(u,v)$ is the number of edge-disjoint (or internally vertex-disjoint) paths between u and v in G'.

3. Algorithm for k-edge connectivity

We follow the relay placement framework of the connectivity algorithm of [21]. 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 multi-graph G_c on the terminal nodes. There are k edges between each pair of terminal nodes in G_c. The weight function for the edges is defined as $c_e=\lceil |e|\rceil-1$, 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 (G'_c) of the multi-graph G_c.

The problem of finding the minimum cost spanning k-edge connected subgraph of a graph is *NP*-Hard [11]. Thus, we use an approximation algorithm for the problem, proposed in [17]. 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 [10], which finds k edge-disjoint spanning trees from a root vertex in a directed graph. It is worth noting that the weight function $\lceil |e|\rceil-1$ is not a metric as it does not satisfy triangle inequality. Thus, the approximation algorithm of [17] is the best known for the problem. In the resulting subgraph from the approximation algorithm of [17], 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 $2M$ 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 complete algorithm takes $O((kn)^2+n'm(n+n'))$ time. Algorithm 1 describes the algorithm. For a network in a cuboid of length L, the maximum number of relays on any edge in G_c is $O(L)$, and the number of edges in the graph at the output of Step 3 of Algorithm 1 (G'_c) is $k(n-1)$, thus, $n'=O(knL)$ and $m=O((knL)^2)$. Therefore, the algorithm takes $O((knL)^4)$ time.

We briefly explain the algorithm for computing the approximate minimum-weight (cost) 2-edge connected subgraph [17] of a multi-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 [10] 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 [20] based algorithm, we use Frank's weighted matroid intersection algorithm [8] and Roskind and Tarjan's algorithm for computing edge-disjoint spanning trees [29], which is based on greedy

Algorithm 1 Relay placement for k-edge connectivity

1: Make a multi-graph $G_c = (V, E_c)$ by adding k edges between each pair of vertices of graph G.
2: Weight the edges of the graph as $c_e = \lceil |e| \rceil - 1$. $|e|$ represents the length of edge e.
3: Compute an approximate minimum cost spanning k-edge connected subgraph from this graph G_c using the approximation algorithm proposed in [17]. 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.

matroid algorithm [20]. Due to the complexity of the algorithms, we do not describe them here. Details of these can be found in [8,10] and [29].

3.1. Proof of approximation ratio for 2-edge connectivity

In this section, we consider the case of achieving 2-edge connectivity between terminal nodes. Let the terminal nodes be placed in a Euclidean space of fixed dimension with MST number M [28]. MST number is defined as the maximum node degree in a minimum-degree Minimum Spanning Tree (MST) spanning points from the space. The approximation ratio for the MST based algorithm of [21] for connecting terminals using minimum relays has been shown to be $M - 1$ in [25]. The MST number for the Euclidean plane is 5 [27], three-dimensional Euclidean space is 12, and rectilinear plane (two-dimensional space with metric defined by L_1 norm) is 4. We analyze the worst-case performance of our algorithm and prove a performance bound of the algorithm with respect to the optimal solution. We prove that the algorithm proposed for 2-edge connectivity is a $2M$-approximation.

We start with some notation. 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 (as in our algorithm) beads and the optimally placed relay nodes 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 3.1 states the main result of this section.

Theorem 3.1. *If the optimal network uses s Steiner nodes so that terminals distributed in Euclidean space of MST number M are* 2*-edge connected, Algorithm* 1 *forms a network with maximum of* $2Ms$ *beads and zero Steiner nodes, in which the terminal nodes are* 2*-edge connected.*

Proof. To prove Theorem 3.1, we prove the following lemma, and Theorem 3.1 follows directly.

Lemma 3.2. *A* 2*-edge connected network on terminal nodes using minimum number of beads contains at most* Ms *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 on terminals (having the minimum number of Steiner nodes).

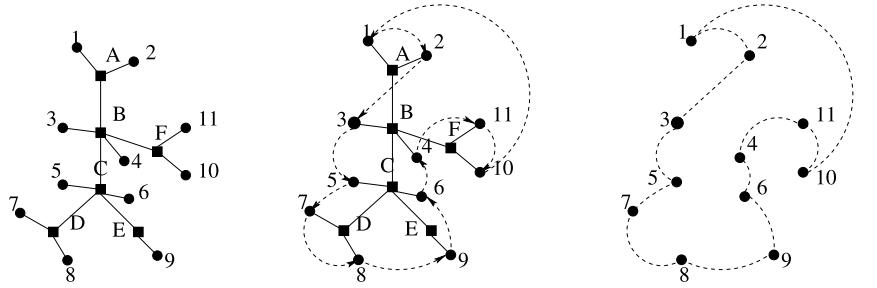
We follow the procedure of Algorithm 2 to construct a 2-edge connected network that has beads and no Steiner nodes. We will prove that this network does not contain more than Ms beads.

Algorithm 2 starts by finding the connected components (SC_i) of Steiner nodes in the graph constructed on the Steiner nodes. It constructs a Breadth First Search (BFS) spanning tree on Steiner nodes for each connected component, starting with any Steiner node in that component as the root. Let the trees be $ST_1, \ldots, ST_m$. The algorithm then removes Steiner nodes of a connected component SC_j 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, and the resulting graph has zero Steiner nodes and is 2-edge connected on the terminals.

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 the 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 that were

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 (SC_i) in G_S.
3: Construct a BFS spanning tree in each connected component, and call the trees $ST_1, \ldots, ST_m$.
4: (1) Repeat the following from $j = 1$ to m.
(2) 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 for adding beads and removing Steiner nodes is explained in Algorithm 3.
5: Output the resulting graph G_m.



(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.

2-edge connected because of the Steiner nodes in ST_j. As we do this for all trees $ST_1, \ldots, ST_m$,[1] and do not create any cut-edge in any step, the resulting network is 2-edge connected on the terminals.[2]

Algorithm 3 describes the algorithm for construction of the cycle between terminal nodes connected to the Steiner nodes of ST_j. Fig. 2(a) shows an example, where $A, B, \ldots, F$ are Steiner nodes and $1, 2, \ldots, 11$ are terminals. The algorithm works as follows: Start at the root of ST_j (call the root st_1, dropping subscript j for simplicity), node A in the example. Connect to st_1 all terminal nodes within its transmission range, and mark them. Let the set of marked terminal nodes be $\{t_1, \ldots, t_k\}$, nodes $1, 2$ in the example. Start a Depth First Search (DFS) traversal of the tree formed by $ST_j \cup \{t_1, \ldots, t_k\}$ (rooted at st_1), starting with any child of st_1. The children of a node are traversed in the order of their distance from each other, i.e., the next child to traverse is the one closest to the current child being traversed and the first to traverse is the one closest to the parent node (in the example, traversal order is $1, 2, B$). Whenever a new Steiner node st_i is encountered in the traversal, all unmarked terminal nodes in its transmission range are connected to it, and added to the set of marked terminal nodes (thus k increases at this step). In the example, the tree becomes $ST_j \cup \{1, 2, 3, 4\}$ when B is encountered. 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. Complete the cycle by connecting the last added terminal to the first terminal encountered in the DFS traversal.[3] The edges longer than unit length are added using the required number of beads. Fig. 2(b) shows the cycle created between the terminal nodes in the example. Finally, remove the Steiner nodes. Fig. 2(c) shows the final topology on these terminals nodes.

We first state the bound on number of beads required in Algorithm 3, followed by proof of the bound. The number of beads (b_j) added while constructing G_j from G_{j-1} is bounded as $b_j \leqslant Ms_j$, where s_j is the number of Steiner nodes in ST_j. The total number of beads required (b) can then be bounded as $b = \sum_{j=1}^{m} b_j \leqslant \sum_{j=1}^{m} Ms_j = Ms$, which proves Lemma 3.2.

We now prove the bound on b_j. We first prove a property of the nearest-neighbor traversal of neighbors around a node, the property is stated in Lemma 3.3.

Lemma 3.3. *Let there be an arbitrary set of points $P = \{x_1, \ldots, x_l\}, l > 0$ distributed in a unit ball centered at a point x_0 in d-dimensional space. We form a cycle C consisting of the points in P: Add x_1 to C. From the last point added to C, add an edge to the closest point in $P \backslash C$, add the new point to C, and repeat. Finally, add an edge to x_1 when $P = C$. The maximum number of edges of length greater than one is bounded by M, the MST number of the space.*

[1] If two terminal nodes are adjacent in multiple cycles formed while removing the Steiner components, we form maximum two beaded links between them. This suffices for maintaining 2-edge connectivity as no cycle is broken.

[2] We 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.

[3] Note that there will be at least two terminals connected to the Steiner nodes of ST_j. If there were only one terminal node, the Steiner nodes of ST_j could be deleted from the optimal Steiner graph without affecting the connectivity. In case of two terminal nodes, the cycle formed has two edges between the same pair of terminals.

Algorithm 3 Removal of Steiner nodes and addition of beads in ST_j

1: Start at root st_1 of ST_j.
2: Connect to it all terminals within its transmission range, and mark them.
3: Construct a tree T_j, with the vertex set as the Steiner nodes in ST_j and a leaf vertex corresponding to each marked terminal vertex. The edges are the edges of ST_j and the edges between each Steiner node and the marked terminal vertices connected to it.
4: Do a Depth First Search (DFS) traversal of T_j rooted at st_1, starting with any child of st_1. For each node, traverse its children according to their distance from each other, i.e., the next child traversed is the child closest to the current child being traversed. The first child to be traversed at a Steiner node is the one closest to the parent node.
5: Each time a new Steiner node st_i is encountered, connect it to all unmarked terminal vertices in its range, and mark them. Update T_j by adding these terminal vertices, and continue DFS traversal around st_i from the edge between st_i and its parent.
6: Connect all the terminal vertices in order of their DFS traversal and complete the cycle between them.
7: Add beads to all added edges of length greater than one.
8: 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.

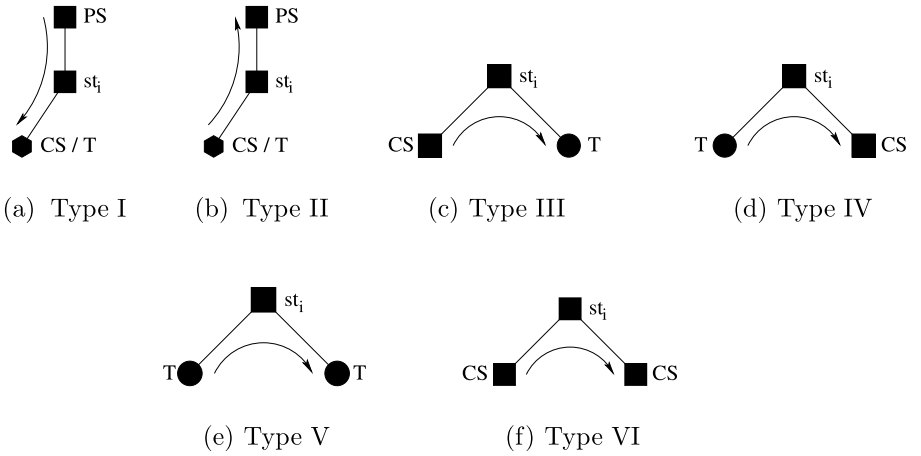


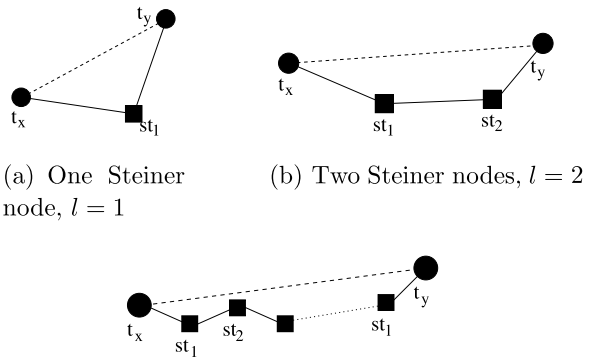
Fig. 3. Types of edge pairs charged to Steiner node st_i.

Proof. Rename the points in P such that x_i is the ith point being added to C. Start constructing C, and each time an edge of length greater than one is encountered, mark its end-point that was already in C, call it y_i ($i = 1$ for first such edge) and increase i by one. Let the set of marked points be P'.

Remove the unmarked points of P ($x_i \notin P'$) from the unit ball. The ball now contains x_0 and the marked points $y_i \in P'$. When a point in P was added to P', it was more than unit distance from all the points not in C, thus the distance of $y_i, i \geqslant 1$ is greater than one from all points $y_j, j > i$. Thus, the distance of each y_i is greater than one from all points in $P'\backslash\{y_i\}$. Thus, the only possible spanning tree on the set of points in $P' \cup x_0$ is a star with x_0 directly connected to all points $y_i \in P'$. The degree of x_0 in this MST is equal to the number of points in P', which is equal to the number of edges of length greater than one in C. As the degree of a minimum degree MST in the space is bounded by M, there cannot be more than M edges of length greater than one in C. □

We now define our charging scheme, i.e., how we charge the beads to the Steiner nodes in ST_j. We call the parent Steiner node of a Steiner node as PS, the child Steiner nodes as CS, and terminal node as T. When a node in consideration can be any one of CS or T, we refer to it as CS/T. We classify the ordered pairs of edges traversed around a Steiner node st_i during a DFS traversal of the tree into six types. Fig. 3 shows the types of ordered pair of edges, where the arrows represent the order of traversal among the edges. We charge one bead to a Steiner node st_i each time one of the six types of ordered pair of edges is traversed. The six types of edge pairs of interest are listed below:

- **Type I**: PS-st_i-CS/T (Fig. 3(a)). The distance between the parent Steiner node and the node at the other end is always greater than one. If the node at the other end is a Steiner node, the distance is greater than one as the tree ST_j is a BFS tree (nodes two levels away in the tree cannot have distance less than one). If the other end is a terminal node, the distance is greater than one due to the construction procedure of T_j (if the distance were less than one, the terminal would be a child of the PS node).
- **Type II**: CS/T-st_i-PS (Fig. 3(b)). The distance between the parent Steiner node and the node at the other end is always greater than one, reason being the same as explained for Type I edge pair.
- **Type III**: CS-st_i-T (Fig. 3(c)), if the Euclidean distance between the end-nodes is more than one.
- **Type IV**: T-st_i-CS (Fig. 3(d)), if the Euclidean distance between the end-nodes is more than one.
- **Type V**: T-st_i-T (Fig. 3(e)), if the Euclidean distance between the end-nodes is more than one.
- **Type VI**: CS-st_i-CS (Fig. 3(f)), if the Euclidean distance between the end-nodes is more than one.



(a) One Steiner node, $l = 1$

(b) Two Steiner nodes, $l = 2$

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

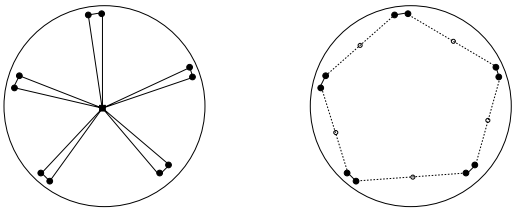
Fig. 4. DFS paths of different lengths.

As all ordered pairs of edges are traversed once during the traversal, and the ones for which a bead is charged have the end-points more than distance one apart, the maximum number of times a Steiner node is charged for a bead is M (by Lemma 3.3).

We need to prove that this charging scheme charges the required number of beads in the construction of the cycle among terminals. Every time we add an edge to connect two terminal nodes in T_j, we claim that the number of beads required ($\lceil |e| \rceil - 1$) can be charged to the Steiner nodes encountered in the DFS traversal between the two terminal nodes using our charging scheme. Let two terminal vertices 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 to $st_1, \ldots, st_l$. 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 Fig. 4(a). If both the terminals are connected to the same Steiner node 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 V and thus the Steiner node st_1 can be charged for the bead required.
- **Case 2**: $l = 2$: This case is depicted in Fig. 4(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. 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 both the pairs of edges, $t_x - st_1 - st_2$ and $st_1 - st_2 - t_y$ are Type IV and I respectively 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 I 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 Fig. 4(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. There are two cases to be considered:
  1. The DFS path stays on one branch of the DFS tree. Let st_i be the parent of st_{i+1} for $i \in \{1, \ldots, l-1\}$. One bead can be charged to each of the nodes $st_2, \ldots, st_l$ 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 IV. Thus, there is a path of $l-1$ 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. The case of going up a DFS branch, i.e., when st_i is the parent of st_{i-1} for $i \in \{2, \ldots, l\}$ is similar.
  2. The DFS path moves between two branches of the DFS tree. Let st_k be the Steiner node at which the two branches start. In this DFS path, st_i is the parent node of st_{i-1}, $i \in \{2, \ldots, k\}$, and st_i is the parent node of st_{i+1}, $i \in \{k, \ldots, l-1\}$. One bead can be charged to each of the nodes $st_1, \ldots, st_{k-1}$ as the pair of edges involving them in the path are of Type II. Similarly, one bead can be charged to each of the nodes $st_{k+1}, \ldots, st_l$ as the pair of edges involving them in the path are of Type I. If the distance between st_{k-1} and st_{k+1} is less than one, the path can be modified by connecting st_{k-1} directly to st_{k+1}, and there is a path with $l-1$ Steiner nodes, and thus st_k can be removed. If it is greater than one, then a bead can be charged to st_k as the pair of edges $st_{k-1} - st_k - st_{k+1}$ is of Type VI. Thus, there is a path of $l-1$ 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 M times. Adding over all connected components of Steiner nodes in the network, the total



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

Fig. 5. Approximation ratio tightness example.

number of beads required is within M times the number of Steiner nodes. Thus, the bound $b \leqslant Ms$ holds for the beaded network we constructed. Hence, the relation holds for the optimal 2-edge connected beaded network as well. □

The algorithm of [17] 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 $2Ms$. The last step of Algorithm 1 (sequential removal step) removes beads from the network by allowing them to connect to all nodes within the transmission range, so the resulting network after sequential removal also has maximum of $2Ms$ relay nodes. □

The bound of M for an optimal beaded network is tight for 2-edge connectivity. Let there be M pairs of nodes (placed right next to each other) placed more than unit distance apart around a Steiner node in an optimal Steiner solution. The optimal beaded network will connect them in a cycle using M beads in place of one Steiner node. Fig. 5(a) shows the example for terminal nodes in a Euclidean plane. 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 ($M = 5$ for Euclidean plane). The resulting network is shown in Fig. 5(b).

4. Algorithm for k-vertex connectivity

We propose an algorithm for achieving k-vertex connectivity among terminal nodes using relays. We follow the same framework as for edge-connectivity, i.e., add relays only on the line joining two terminal nodes. We construct a simple complete graph on terminal nodes, rather than a multi-graph. The graph has an edge between each pair of terminal nodes, with the same edge weight as defined before. 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 of [16] for $k = 2$, and the k-approximation algorithm of [18] for $k > 2$. The algorithm takes $O(k^2n^3m)$ time, where n is the number of terminals and m is the number of edges in the graph (which is $n(n-1)$ for a complete graph, as in our case). For $k \leqslant 7$, we can use the improved approximation algorithms given in [2,6]. 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 [16] 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 [9] 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.

4.1. 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 4.1 states the desired result. We follow the same terminology as the last section.

Theorem 4.1. *If the optimal network uses s Steiner nodes so that terminals are 2-vertex connected, our algorithm forms a network with maximum of $2Ms$ beads and zero Steiner nodes, in which the terminal nodes are 2-vertex connected.*

Proof. To prove Theorem 4.1, we prove the following lemma, and Theorem 4.1 follows directly as we use a 2-approximation for finding the minimum-cost beaded network.

Lemma 4.2. *A network that is* 2*-vertex connected on terminal nodes using the minimum number of beads contains at most* Ms *beads, where* s *is the minimum number of Steiner nodes needed.*

Proof. We follow similar constructions and proof as the proof for 2-edge connectivity. The only change in the construction of the complete graph is that when we encounter a Steiner component in the optimal Steiner graph, that is connected to only two terminals, we form a single edge between them rather than a cycle as for 2-edge connectivity. Also, if two terminals are adjacent in multiple cycles formed while removing Steiner components, we keep only one of the edges between them.[4] Thus, the maximum number of beads required is Ms, as for 2-edge connectivity. Therefore, we only need to prove that the beaded graph constructed is 2-vertex connected on terminals.

We prove the 2-vertex connectivity of the network using induction. The proof is similar to the proof of 2-vertex connectivity in [3]. Recall the notation: there are m Steiner components in the optimal solution, and the graph after removal of component SC_j from G_{j-1} is G_j. G_m denotes the beaded graph with zero Steiner nodes. We need to prove that G_m is 2-vertex connected on terminals.

G_0 is the optimal Steiner graph, that is 2-vertex connected between the terminals. Let G_{i-1} be 2-vertex connected on the terminals. Thus, removal of any vertex w does not disconnect the terminals in G_{i-1}. We prove by contradiction that all terminals are connected in $G_i - \{w\}$ as well. Let u and v be the two terminals which are disconnected in $G_i - \{w\}$. All terminal pairs (u, v) have a path in $G_{i-1} - \{w\}$. If the path does not use more than one terminal connected to component SC_i, u and v are connected in $G_i - \{w\}$ as well. If the path uses at least two terminal vertices connected to SC_i (u_1, v_1 being the first and last terminals connected to SC_i on the path), that path exists as well if there are at least three terminals connected to SC_i, since we form a cycle (that is 2-vertex connected) between all terminals connected to SC_i. If there are just two terminals (which will be u_1, v_1), a direct edge exists between them and thus a path exists between u and v in $G_i - \{w\}$. Thus, G_i is 2-vertex connected on terminals. Therefore, by induction, G_m is 2-vertex connected on terminals. □

5. Generalization to restricted relay placement

We extend our results for terminals distributed in a Euclidean plane to the scenario where relays cannot be placed in certain polygonal regions of the network. We call these regions 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, weight $c_e = \lceil |e| \rceil - 1$, 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 [1]. The problem is called the puddle-jumper problem. We modify our edge weights by running the algorithm given in [1] 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 [1].

5.1. Proof of approximation ratio

We now prove that the approximation ratio for the 2-edge and 2-vertex connectivity algorithms is 10 for terminals distributed in the Euclidean plane. 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 [1], 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 3.1 that each Steiner node is charged at most 5 times ($M = 5$ for Euclidean plane), 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 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.

[4] All the cycles still exist, thus it does not affect the 2-vertex connectivity requirements.

6. Conclusion

We consider the problem of constructing a fault-tolerant topology among static nodes distributed in a Euclidean space of fixed dimension 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 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.

Acknowledgements

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

References

[1] E. Arkin, E. Demaine, J. Mitchell, The puddle-jumper problem, personal communication.
[2] V. Auletta, Y. Dinitz, Z. Nutov, D. Parente, A 2-approximation algorithm for finding an optimum 3-vertex-connected spanning subgraph, Journal of Algorithms 32 (1999) 21–30.
[3] J.L. Bredin, E.D. Demaine, M. Hajiaghayi, D. Rus, Deploying sensor networks with guaranteed capacity and fault tolerance, ACM MobiHoc (2005) 309–319.
[4] D. Chen, D.-Z. Du, X.-D. Hu, G.-H. Lin, L. Wang, G. Xue, Approximations for Steiner trees with minimum number of Steiner points, Theoretical Computer Science 262 (2001) 83–99.
[5] X. Cheng, D.-Z. Du, L. Wang, B. Xu, Relay sensor placement in wireless sensor networks, Wireless Networks 14 (3) (2008) 347–355.
[6] Y. Dinitz, Z. Nutov, A 3-approximation algorithm for finding optimum 4,5-vertex connected spanning subgraphs, Journal of Algorithms 32 (1999) 31–40.
[7] A. Efrat, S.P. Fekete, P.R. Gaddehosur, J.S.B. Mitchell, V. Polishchuk, J. Suomela, Improved approximation algorithms for relay placement, in: ESA, LNCS 5193 (2008) 356–367.
[8] A. Frank, A weighted matroid intersection algorithm, Journal of Algorithms 2 (1981) 328–336.
[9] A. Frank, E. Tardos, An application of submodular flows, Linear Algebra and its Applications 114/115 (1989) 329–348.
[10] H.N. Gabow, A matroid approach to finding edge connectivity and packing arborescences, Journal of Computer and System Sciences 50 (2) (1995) 259–273.
[11] M. Garey, D. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman and Company, 1979.
[12] X. Han, X. Cao, E. Lloyd, C.-C. Shen, Fault-tolerant relay node placement in heterogeneous wireless sensor networks, IEEE INFOCOM (2007) 1667–1675.
[13] B. Hao, J. Tang, G. Xue, Fault-tolerant relay node placement in wireless sensor networks: Formulation and approximation, IEEE HPSR (2004) 246–250.
[14] A. Kashyap, Robust design of wireless networks, Ph.D. thesis, Electrical and Computer Engineering, University of Maryland, 2006.
[15] A. Kashyap, S. Khuller, M. Shayman, Relay placement for higher order connectivity in wireless sensor networks, IEEE INFOCOM (2006) 2229–2240.
[16] S. Khuller, B. Raghavachari, Improved approximation algorithms for uniform connectivity problems, Journal of Algorithms 21 (2) (1996) 434–450.
[17] S. Khuller, U. Vishkin, Biconnectivity approximations and graph carvings, Journal of the ACM 41 (2) (1994) 214–235.
[18] G. Kortsarz, Z. Nutov, Approximating node connectivity problems via set covers, Algorithmica 37 (2003) 75–92.
[19] B. Krishnamachari, Networking Wireless Sensors, Cambridge University Press, 2005.
[20] E.L. Lawler, Combinatorial Optimization: Networks and Matroids, Holt, Rinehart and Winston, New York, 1976.
[21] G.-H. Lin, G.L. Xue, Steiner tree problem with minimum number of Steiner points and bounded edge-length, Information Processing Letters 69 (1999) 53–57.
[22] H. Liu, P.-J. Wan, X. Jia, Fault-tolerant relay node placement in wireless sensor networks, International Computing and Combinatorics Conference (COCOON) (2005) 230–239.
[23] E.L. Lloyd, G. Xue, Relay node placement in wireless sensor networks, IEEE Transactions on Computer 56 (1) (2007) 134–138.
[24] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, J. Anderson, Wireless sensor networks for habitat monitoring, ACM WSNA (2002) 88–97.
[25] I. Măndoiu, A. Zelikovsky, A note on the MST heuristic for bounded edge-length Steiner trees with minimum number of Steiner points, Information Processing Letters 75 (4) (2000) 165–167.
[26] S. Misra, D. Hong, G. Xue, J. Tang, Constrained relay node placement in wireless sensor networks to meet connectivity and survivability requirements, IEEE INFOCOM (2008) 281–285.
[27] C. Monma, S. Suri, Transitions in geometric minimum spanning tree, Discrete and Computational Geometry 8 (1992) 265–293.
[28] G. Robins, J.S. Salowe, Low-degree minimum spanning trees, Discrete Computational Geometry 14 (1995) 151–165.
[29] J. Roskind, R.E. Tarjan, A note on finding minimum-cost edge-disjoint spanning trees, Mathematics of Operations Research 10 (4) (1985) 701–708.
[30] A. Srinivas, G. Zussman, E. Modiano, Mobile backbone networks – construction and maintenance, IEEE INFOCOM (2007) 1649–1657.
[31] W. Zhang, G. Xue, S. Misra, Fault-tolerant relay node placement in wireless sensor networks: Problems and algorithms, IEEE INFOCOM (2007) 1649–1657.