On Improving Connectivity of Static Ad-Hoc Networks by Adding Nodes

Henri Koskinen, Jouni Karvo and Olli Apilo
Helsinki University of Technology (TKK), Networking Laboratory, P.O. Box 3000, FIN-02015 TKK, Finland
{Henri.Koskinen, Jouni.Karvo, Olli.Apilo}@tkk.fi

***Abstract*— The connectivity of a given static, disconnected ad-hoc network can be improved by placing additional nodes in the network, forming connections between separate clusters of connected nodes. Finding optimal locations to place the additional nodes is a difficult problem. We give definitions of two problems: connecting the network with a minimal number of additional nodes, and maximizing utility from a given number of additional nodes in case complete connectivity cannot be established. The former problem reduces in a limit case to that of the Euclidean Steiner minimal tree. We present heuristic algorithms that can be used to efficiently attack these problems: a minimum spanning tree algorithm and two greedy algorithms, all applicable to both problems. The algorithms are feasible in their computation effort. We study the performance of these algorithms by simulations. We also consider the generalization of the problem to k-connectivity and recognize its relation to another NP-hard problem, namely, that of graph augmentation.**

I. Introduction

Ad hoc networks are by nature constructed "automatically", by the nodes adapting to the neighboring nodes and building up a network. In this context, the network topology is random, and in particular, no connectivity is guaranteed: the nodes may be so sparsely located that they are unable to make up a connected network.

This has motivated a wide range of research, with a primary interest in the connectivity of random networks. The most popular and simple topology model has been the Boolean one, which we also use in this paper. In this model, two nodes are considered directly connected if the distance between them does not exceed the transmission range, a network parameter. Under this model, the main object of study becomes geometric graphs. Percolation properties of such graphs when the positions of the network nodes are distributed according to a homogeneous Poisson point process in the infinite plane have been studied in [1]. In the case of a finite domain, the probability of a random network being connected depends only on the probability distribution of the critical transmission range for connectivity: for a given set of nodes, this critical range is equal to the greatest edge length in the Euclidean minimum spanning tree of the nodes, as pointed out in [2]. Asymptotic scaling laws for the critical range have been derived in [3]. The notion of the critical range can be generalized to k-connectivity, which guarantees connectivity to prevail at the failure of any $k-1$ nodes. The distributions of the critical ranges for k-connectivity with any k are not known when the number of nodes is finite. Attempts to determine analytically the probability of k-connectivity of finite random networks have been made in [4], based on using as an approximation the probability of every node having at least k neighbors. Efficient algorithms for determining the critical ranges for the purposes of simulation and empirical modelling are developed in [5], and empirical models describing the convergence of the distributions of the critical ranges to the known asymptotic ones [6] are presented in [7].

As an example of the findings in these studies, one may look at Figure 1 which shows the transmission range that, for nodes distributed uniformly at random in a 1 km $\times$ 1 km area, provides different degrees of k-connectivity with 99% confidence, according to the empirical models in [7]. One may conclude from the figure that the required transmission ranges are too long to achieve high bit rate connections efficiently with current radio technology. Recognizing that the scenario chosen for the figure could well describe an ad hoc network applied in a practical emergency or disaster recovery situation, we are concerned with what can be done when an ad hoc network needs to be formed but the users are too far apart to form a network with a desired level of connectivity. More precisely, in this paper we study the option of improving the connectivity of a static ad-hoc network by carrying extraneous nodes to the scene. The problem is where to put these nodes so as to minimize the number of nodes required for a connected network, or to maximize the utility

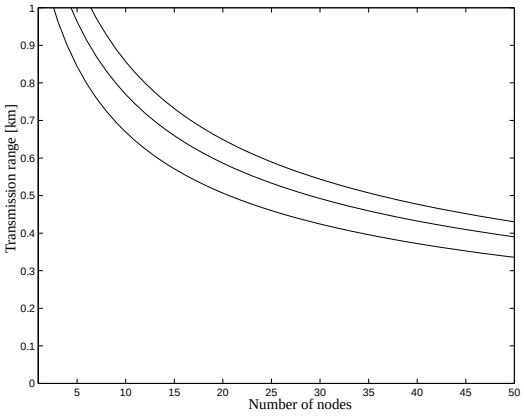


Fig. 1. The transmission range predicted to ensure k-connectivity with 99% confidence for randomly placed nodes in a 1 km $\times$ 1 km area. 3-, 2-, and 1-connectivity, from top to bottom.

of the network. We present algorithms that suggest locations for such additional nodes. Networks where adding extraneous nodes is feasible are some sensor networks — see [8] for a survey on sensor networks — and such ad-hoc networks that are used in a controlled situation where some central entity can organize the deployment of the nodes.

To our knowledge, the connectivity problem in ad hoc networks has not been addressed so far from this practical viewpoint. Placing additional nodes, "base stations", in the network for connectivity has been studied in [1] from a theoretical point of view. The paper suggests that the more dimensions the network has, the less the addition of base stations improves connectivity. However, these results relate to a stochastic model, and the locations of the base stations were not optimized to the actual node locations.

This paper is organized as follows: in the next section we give a thorough description of the problem setting, the underlying assumptions, and the network model. Section III defines essentially two optimization problems for the node addition. Section IV describes the first heuristic algorithm, the Minimum Spanning Tree algorithm. A more efficient algorithm, the Greedy Tessellation algorithm is presented in Section V, and the last and most evolved algorithm, the Greedy Triangle algorithm in Section VI. Some observations on the generalization of the problem to k-connectivity are made in Section VII. Section VIII contains performance analysis of the algorithms, aided by simulation results. Finally, Section IX concludes the paper with a summary and reflections of future work.

II. Problem Setting and Network Model

As hinted earlier, the motivation for our problem stems from an emergency scenario. We consider a group of agents, e.g. fire fighters, deployed in some region, who need to establish communications in the form of an ad hoc network. For this purpose, each agent is equipped with a terminal device; from now on, we will refer to these devices as *terminal nodes*. In support of forming the network, there is a team in possession of additional transceivers that can be used as relays in the network; we will call these transceivers *relay nodes*. We assume that both the terminal nodes and the relay nodes are based on same standard hardware and therefore have equal transmission and reception capabilities. The task of the support team is to place relay nodes in the region so that the terminal nodes and the relay nodes together can form a connected wireless multihop network where each link can provide a desired rate of communication to support the service required by the agents, say, a speech application. The problem we are interested in is to optimize the points where the support team should place relay nodes, given the locations of the terminal nodes.

The key assumption behind our problem statement is that the locations of the terminal nodes are known. The position of a node can be estimated using GPS or the future Galileo positioning systems, yielding accurate enough estimates (within 10–20 m from the true position), and being cost-effective enough for implementation. Another way of positioning nodes is using triangulation [9], but this requires that each part of the network has enough nodes with known positions. Thus, triangulation methods can be used to locate single nodes that are not on the coverage of other positioning systems.

Another important assumption is that the location information of the terminal nodes can be collected even though the network is not connected. The motivation behind this assumption is that depending on the solutions on the physical layer, it can be possible to be able to sustain low bitrate communications over much further distance than to provide quality of service. In this case, the network is able to convey control information even if efficient communications are not possible. In other words, in this problem we define connectivity using a linkwise throughput requirement.

We use the commonly studied Boolean model for the network. This implicitly means that we assume the dominating factor affecting communications to be the path loss of radio signals coupled with a constant-level background noise, rather than the interference between simultaneous transmissions. Within this model, the transmission power employed by the network nodes (which we assume to be the same for all nodes), the path-loss model, and the signal-to-noise ratio required at reception for a desired rate of communication are woven into a single parameter, the transmission range. Given the geographical positions of all nodes, the network topology is then represented by a geometric graph, where two nodes are assumed to have an undirected edge, representing a bidirectional link, between them if and only if the distance between them does not exceed the transmission range.

We will next define the problem rigorously.

III. Problem Definitions

Formally, we assume that the network deployment region (where the terminal nodes are located and the relay nodes may be placed) is a bounded convex subset $\mathcal{S}$ of the Euclidean space $\mathbb{R}^d$, $d > 1$. In all the problems that we are about to define, the problem instance is completely defined by the set of locations of N terminal nodes, $\mathcal{N} = \{\mathbf{x}_i \in \mathcal{S} \,|\, i = 1, 2, \ldots, N\}$, and the transmission range r. Together these imply the pre-existing network topology in the form of an undirected geometric graph $\mathcal{G}(\mathcal{N}, \mathcal{E}(\mathcal{N}, r)) = \mathcal{G}(\mathcal{N}, r)$ with vertex set $\mathcal{N}$ and edge set $\mathcal{E}(\mathcal{N}, r) = \{(\mathbf{x}_i, \mathbf{x}_j) \,|\, \mathbf{x}_i, \mathbf{x}_j \in \mathcal{N},\, i \neq j,\, ||\mathbf{x}_i - \mathbf{x}_j|| \leq r\}$.

A solution to any of the problems is a set of locations to place relay nodes, $\mathcal{N}_\mathrm{r} = \{\mathbf{y}_i \in \mathcal{S} \,|\, i = 1, 2, \ldots, N_\mathrm{r}\}$. Given a configuration of terminal and relay nodes $\mathcal{N} \bigcup \mathcal{N}_\mathrm{r}$ and transmission range r, we call a *cluster* the set of all *terminal nodes* in a single maximal connected component in the graph $\mathcal{G}(\mathcal{N} \bigcup \mathcal{N}_\mathrm{r}, r)$.

We are hereby ready to define the first optimisation problem:

> **Problem 1** Given $\mathcal{N}$ and r, find $\mathcal{N}_\mathrm{r}$ with minimum cardinality that makes the graph $\mathcal{G}(\mathcal{N} \bigcup \mathcal{N}_\mathrm{r}, r)$ connected.

We point out that in the limit $r \to 0$, this problem reduces to finding the Euclidean Steiner minimal tree for the set $\mathcal{N}$: for a given set of points, the Euclidean Steiner minimal tree is

the tree connecting all the points with minimum total edge length, when the addition of new points is allowed before forming the edges. The optimal solution is then to place the relay nodes along the edges of this tree. Finding Steiner minimal trees is known to be **NP**-hard [10]. In the general case, our problem poses the additional complications that we are not connecting single points to each other, but clusters where the best points in the clusters for connecting to other clusters must be chosen, and that the objective function has been discretized from the total length of edges in the Steiner tree to the number of added relay nodes. In the following, we suggest heuristic algorithms that are suboptimal but give results without excessive computing requirements.

A greedy approach to solving the problem is to add new relay nodes trying to get as good an improvement as possible in each step, until the connectivity target has been met. A utility metric is required for this approach, and it should reflect how close we are to achieving the target. To this end, we enumerate the clusters that exist after each step and let $\mathcal{C}_j$ denote the j:th cluster.

The choice of the utility metric depends on the target application. We define several possible metrics:

Metric 1: The number of nodes in the biggest cluster,

$$U_1 = \max_j |\mathcal{C}_j|$$

Metric 2: The number of node pairs in the biggest cluster, up to a multiplication by a constant:

$$\max_j \binom{|\mathcal{C}_j|}{2} \propto \max_j |\mathcal{C}_j|(|\mathcal{C}_j| - 1) = U_2.$$

Metric 3: The sum of all node pairs over all clusters (i.e., the number of possible connections),

$$\sum_j |\mathcal{C}_j|(|\mathcal{C}_j| - 1) = \sum_j |\mathcal{C}_j|^2 - \sum_j |\mathcal{C}_j|,$$

where, since the clusters form a partition of the terminal nodes, the latter term simply equals the number of terminal nodes and can be ignored. Thus,

$$U_3 = \sum_j |\mathcal{C}_j|^2.$$

Metric 4: The amount of traffic carried by the network. Instead of using the coarse-grained but simple approaches of Metrics 1–3, an estimate of a traffic matrix could be used to prioritize connecting different clusters.

The utility metric is also needed for cases where it is not possible to make the network connected, due to having too few relay nodes available. This gives rise to the second problem:

Problem 2 Find $\mathcal{N}_\mathrm{r}$ that maximizes the chosen utility metric U, subject to $N_\mathrm{r} \leq N_\mathrm{r}^\mathrm{max} \in \mathbb{Z}_+$.

Provided that the constraint $N_\mathrm{r} \leq N_\mathrm{r}^\mathrm{max}$ actually prevents us from achieving connectivity, this problem can be viewed as the maximization of the chosen utility metric in $\mathcal{S}^{N_\mathrm{r}^\mathrm{max}}$. This is a difficult task but allows applying, e.g., simulated annealing. Our algorithms can readily be used for greedy approaches to Problem 2 as well as Problem 1.

In all examples and simulations where applicable in this paper, we have used Metric 1. Metrics 1 and 2 are equivalent in light of Problem 2, as discussed briefly in Appendix A.

At least one further problem definition comes into question:

Problem 3 Find $\mathcal{N}_\mathrm{r}$ that maximizes the degree of node-connectivity of the graph $\mathcal{G}(\mathcal{N} \bigcup \mathcal{N}_\mathrm{r}, r)$, subject to $N_\mathrm{r} = N_\mathrm{r}^\mathrm{max} \in \mathbb{Z}_+$.

Recall from graph theory that a k-(node-)connected graph remains connected after the removal of any $k-1$ nodes. In fact, this problem allows for two alternative variants, depending on whether we require robustness against both terminal nodes and relay nodes failing, or only terminal nodes failing.

Problem 3 is computationally expensive, and as we will point out in Section VII, Problems 1 and 2 generalized to k-connectivity become notably more complicated as k increases. We do not study Problem 3 further in this paper.

IV. Minimum Spanning Tree Algorithm

Our first algorithm arises naturally if we only require that each relay node or contiguous chain of relay nodes connect exactly two clusters of the graph $\mathcal{G}(\mathcal{N}, r)$. Under this limitation, the optimal solution is to place the relay nodes along the edges of the Euclidean minimum spanning tree (MST) calculated for the clusters, when the distance between two clusters is defined as the shortest distance between two terminal nodes in these distinct clusters.

In fact, it is not difficult to show that this MST consists of exactly those edges that are longer than the transmission range r, in the MST calculated for all the terminal nodes. This can be seen by considering Kruskal's algorithm for finding the MST (see e.g. [11]).

The steps of the algorithm are thus as follows:

1) Calculate the Euclidean minimum spanning tree for $\mathcal{N}$.
2) Place the relay nodes on the edges of the minimum spanning tree that are longer than r. If there are too few relay nodes available to span all such edges, select the edges that result in maximizing the chosen utility metric.

In two dimensions, step 1 can be completed in $O(N \log N)$ time by utilizing the Delaunay triangulation; when $d = 3$, the complexity of finding the minimum spanning tree has at least been brought down to $O(N^{4/3} \log^{4/3} N)$ [12]. In a higher number of dimensions, step 1 is likely to require exhaustively calculating the distance matrix of the terminal nodes, which is a quadratic task. Step 2 is linear in N if all the necessary edges can be spanned, since the whole minimum spanning tree has $N-1$ edges. If, on the other hand, not all necessary edges can be spanned, the optimal selection of edges generally requires going through all possibilities. In this case, we propose the greedy method of selecting edges in the order of added utility (with respect to the initial clusters) per used relay node. In this method, the initial clusters can be found in linear time by traversing the minimum spanning tree (edges longer than r

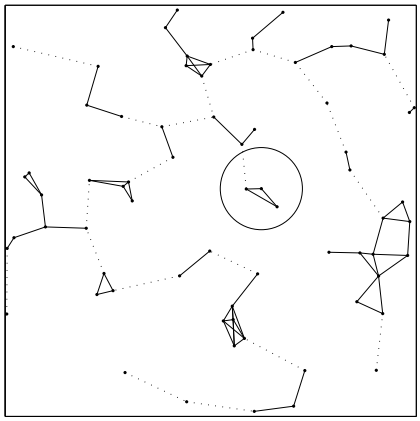


Fig. 2. Minimum Spanning Tree algorithm. The initial clusters in this example realization of 70 terminal nodes are connected with solid edges, and the edges to place relay nodes are dotted. The transmission range is 10% of the side of the domain, as illustrated by the circle.

in the tree separate different clusters), and rating and sorting the $O(N)$ potential edges takes $O(N \log N)$ time. With this approach, the overall complexity of the algorithm is in any case determined by step 1.

Figure 2 illustrates the Minimum Spanning Tree algorithm.

V. Greedy Tessellation Algorithm

The stricter requirement that a single relay node should, when possible, connect more than two clusters suggests points that are equally distant from several clusters as potential points of placement. A useful tool for finding such points is the Voronoi tessellation: for a given set of n points, or *sites*, in the plane, their Voronoi tessellation (or Voronoi diagram) partitions the plane into n convex sets, so that all points forming a given set have the same site as their nearest site. The definition is analogous in higher dimensions, but we restrict the following description to the planar case. See [12] for a rather comprehensive survey on Voronoi diagrams.

What makes the Voronoi tessellation interesting for our problem is that it efficiently captures the geometric neighbor relationships of the nodes: points equally distant from three clusters are a subset of the vertices, i.e. the coinciding corners of the convex sets also called cells, of the Voronoi tessellation of the nodes. Note that in practise, points equally distant to more than three nodes do not exist. However, placing a relay node at a vertex close to other vertices may well result in connecting more than three clusters.

For this reason, we examine coinciding corners of Voronoi cells that contain nodes all in different clusters, and the corner where inserting a new node yields the maximal increase in the chosen utility metric is selected as the place to insert the next relay node. To sum up:

1) Find the maximal connected components and clusters of the graph $\mathcal{G}(\mathcal{N} \bigcup \mathcal{N}_\mathrm{r}, r)$. (Initially, $\mathcal{N}_\mathrm{r} = \emptyset$.)
2) Construct the Voronoi tessellation of $\mathcal{N} \bigcup \mathcal{N}_\mathrm{r}$.
3) Regard as candidate points the coinciding corners of such Voronoi cells that contain nodes all in different connected components, excluding corners further than r from the nodes and corners not in $\mathcal{S}$.
4) Add to $\mathcal{N}_\mathrm{r}$ the candidate point that yields maximal increase in the chosen utility metric.
5) If there were more than one candidate points in step 3 and the problem constraints allow further addition of points, go to step 1.
6) If allowed by the constraints and the graph $\mathcal{G}(\mathcal{N} \bigcup \mathcal{N}_\mathrm{r}, r)$ is not yet connected, finish with the Minimum Spanning Tree algorithm.

The last step is required since connected components can be too far apart to be connected with the addition of a single relay node.

Our complexity analysis is mainly based on results gathered in [12]. On the first run through steps 1–5, step 1 amounts to finding and traversing the minimum spanning tree of the nodes, as described in the previous section. The computational complexity of constructing the Voronoi tessellation in step 2 is $O(N \log N)$ in the plane, quadratic in three dimensions, and increases exponentially with the number of dimensions, along with the maximal size of a diagram. The number of vertices in the tessellation to consider potential candidate points in steps 3 and 4 is $O(N)$ in the plane and $O(N^2)$ in $\mathbb{R}^3$.

On subsequent rounds, the addition of new points to $\mathcal{N}_\mathrm{r}$ can be updated to the connected components and the tessellation without having to find them from scratch. Updating the tessellation is the more complicated task but takes only $O(n)$ time, where $n = N + N_\mathrm{r}$, when $d = 2$ or $d = 3$. Although updating the candidate points should also be a light task, the increase in utility must still be checked for each one in step 4, on every round. The number of rounds made (i.e., the number N_r before proceeding to the last step) with fixed N depends on the density of the network: a very sparse network is unlikely to result in any addition due to too large distances between

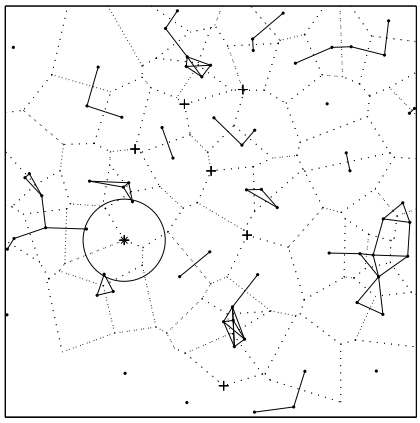


Fig. 3. An example of the Greedy Tessellation algorithm, when applied to the same realization as in Figure 2. The edges of the Voronoi tessellation are shown with dotted lines, the candidate points for relay node insertion with '+'-signs. The first location to add a relay node is marked with an asterisk.

clusters, as is a very dense network due to a high probability of being connected. With fixed average density of terminal nodes and transmission range, the number of additions is $O(N)$, since it is bounded by the number of initial clusters.

As a conclusion, because of the $O(N)$ repetitions of step 4, the overall running time of this algorithm before the final step is $O(N^2)$ in $\mathbb{R}^2$ and $O(N^3)$ in $\mathbb{R}^3$, which also dominates the final step. The algorithm is illustrated in Figure 3.

VI. Greedy Triangle Algorithm

The Greedy Tessellation algorithm uses points that are equally distant from different clusters as potential places for relay nodes. However, with a closer look we see that this is not always optimal: the point equally distant from three given terminal nodes may fall outside the triangular convex hull of their locations, in which case it cannot be the optimal place for a relay node to connect the three nodes (optimal in the sense that the range required from the relay node to connect the terminal nodes is minimized). For example, the point marked in Figure 3 as the place for the first relay node is such a point. Hence, taking only to the vertices of the Voronoi tessellation into account, one may not find all the places where connecting three clusters with a single relay node is possible.

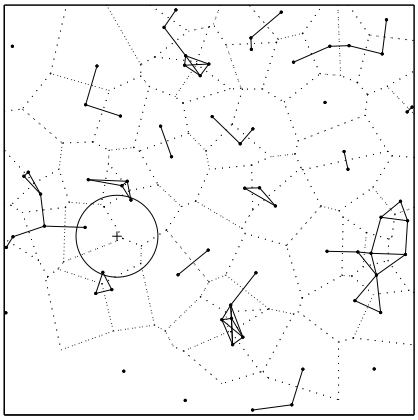
Having made this observation, we may simply select triplets of nodes where the nodes are pairwise at most $2r$ apart and all belong to different clusters, as corners of *candidate triangles*. The point equally distant from the corners of a candidate triangle is a candidate point for node insertion only if this point is inside the triangle; if the point is outside the triangle, the midpoint of the longest side of the triangle is the candidate point (see Figure 4(a) and compare with Figure 3). Finally, it needs to be checked whether the distance from the candidate point to each corner of the triangle is less than r. Of these feasible candidate points, the one yielding the maximal increase in the chosen utility metric is chosen as the location of the next added relay node.

It is of course possible for a single relay node to connect more than three distinct clusters: in $\mathbb{R}^2$, a single node could in theory connect five clusters (given the same transmission range for all nodes), and in higher-dimensional spaces even more. The occurrence of such cases is, however, rare, and thus deliberately seeking these cases is omitted in order to simplify the algorithm. It should be noted however that a proper candidate triangle can still result in connecting more than three clusters.

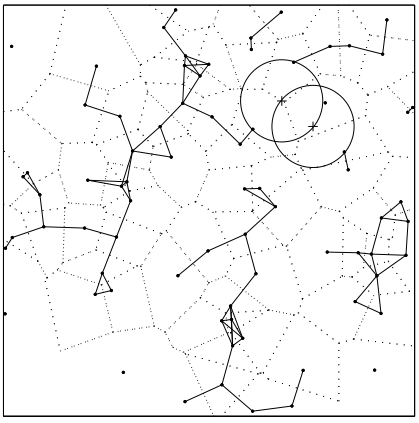
This method is easily extended to handle triangles whose vertices are too far apart to be connected by a single relay node. The idea is to place two nodes optimally in order to connect the clusters. Consider an addition of two nodes, targeting in connecting three clusters: find a candidate triangle with no side longer than $4r$, and find jointly optimal points for two relay nodes (optimality being defined as above). Where to add these two nodes optimally is divided into different cases, depending on the shape of the triangle; these cases are discussed exhaustively in Appendix B. As in the case of single node placement, it must be checked whether placing the new nodes will actually establish connectivity. Again, the placement of the two nodes can actually connect more than three clusters simultaneously, but for the sake of simplicity, checking this has been omitted.

Like the Greedy Tessellation algorithm, this algorithm must also be finished with the Minimum Spanning Tree algorithm. The Greedy Triangle algorithm has thus the following phases:

1) Find the maximal connected components and clusters of the graph $\mathcal{G}(\mathcal{N}, r)$, and the candidate triangles.
2) Find the point (if any exist) where adding a single relay node results in connecting the candidate triangle that yields the maximal increase in the chosen utility metric, and add this point to $\mathcal{N}_\mathrm{r}$. Maintaining the connected components, the clusters, and the candidate triangles, repeat this as long as new candidate triangles can be connected and the problem constraints permit.
3) Repeat the previous step, now adding to $\mathcal{N}_\mathrm{r}$ pairs of points where relay nodes connect candidate triangles.



(a)



(b)

Fig. 4. Applying the Greedy Triangle algorithm to the realization of Figure 2. (a): The first point to place a relay node, as determined in step 2 and indicated by the '+'-sign. Note the difference from Figure 3 in the placement. (b): The first pair of points to place relay nodes, as determined in step 3, after several relay nodes have been added in step 2. Note that in this case, four clusters are connected.

4) If allowed by the constraints and the graph $\mathcal{G}(\mathcal{N} \bigcup \mathcal{N}_\mathrm{r}, r)$ is not yet connected, finish with the Minimum Spanning Tree algorithm.

The Greedy Triangle algorithm can be used as such in a Euclidean space with an arbitrary number of dimensions. However, because among n nodes there are altogether $\Theta(n^3)$ triplets of nodes, it is again a good idea to utilize the geometric neighbor relationships of the nodes in finding sensible candidate triangles, at least in the two-dimensional case. In this case, we propose requiring that at most one pair in any considered triplet of nodes not have neighboring cells in the Voronoi tessellation of the nodes, which limits the number of triplets to examine down to $O(N)$. (We found requiring all three nodes to have pairwise neighboring cells, i.e. considering only the triangles in the Delaunay triangulation, too restrictive.) With this choice, the complexity of the algorithm is the same as that of the Greedy Tessellation algorithm, namely, $O(N^2)$ in $\mathbb{R}^2$ and $O(N^3)$ in $\mathbb{R}^3$. The phases of the algorithm are illustrated in Figure 4.

VII. On the Generalization to k-connectivity

A natural generalization of the problems is to require k-connectivity. In a k-(node-)connected network there are at least k node-disjoint paths between every pair of nodes or, equivalently, there are no $k-1$ nodes whose removal would disconnect the network. Thus, the higher k, the more tolerant the network is to node failures.

As in the case of simple connectivity, we may start with the idea of connecting clusters pairwise with chains of relay nodes, which leads us to a related problem known as graph augmentation. In commonly used terms, the task in the *minimum augmentation* problem is to add to a given graph the set of edges with minimum total weight so that the resulting graph is k-connected. When $k = 1$, the problem reduces to finding the MST, but for any $k > 1$ the problem is known to be **NP**-hard: see [13] and the references therein.

Thus, even with the simplest approach to making a network k-connected with $k > 1$, we immediately run into complex problems. In what follows, we examine the problem of making a given connected network biconnected as the simplest case to demonstrate the difficulties that appear. The solutions to this problem can be applied on top of the previous algorithms to make a disconnected network biconnected, although it is not difficult to see that this kind of incremental strategy – even if the individual steps were solved optimally – can be highly suboptimal in increasing the degree of connectivity by more than one. A somewhat related problem has been studied in [14], where an ad hoc network of mobile robot nodes is already assumed to be connected, and the goal is to move the robots to make the network biconnected so that the total distance travelled by the robots is minimized.

The task in making a connected network biconnected is thus to make sure that there are no single nodes whose removal would disconnect the network; such nodes are sometimes called *articulation points*, and they can be found from a given network using, for example, the recursive depth-first-search (see e.g. [11]). Note that in this particular problem, the assumption of an initially connected network removes the ambiguity mentioned in Section III as to which nodes to regard as articulation points: a relay node as an articulation point either contradicts this assumption or implies unnecessarily added relay nodes.

The articulation points can be eliminated by considering the network without them, and reconnecting this network with relay nodes. However, as explained shortly, this can result in placing unnecessary relay nodes, and such nodes should be excluded from the final solution. Thus, the steps of our algorithm to make a connected network biconnected are as follows:

1) Find the set of articulation points $\mathcal{A} \subset \mathcal{N}$ of the graph $\mathcal{G}(\mathcal{N}, r)$ (assumed connected). If no articulation points are found, the network is already biconnected and no relay nodes need to be added.
2) Apply any of the algorithms presented earlier with input $\{\mathcal{N} \backslash \mathcal{A}, r\}$ to find a set of points $\mathcal{N}_\mathrm{r}$ that, depending on the constraints, makes the graph $\mathcal{G}(\mathcal{N}_\mathrm{r} \bigcup \mathcal{N} \backslash \mathcal{A}, r)$ connected or maximizes the chosen utility metric (with clusters defined as in Section III).
3) Remove unnecessary points possibly added in the previous step: for each point $\mathbf{y} \in \mathcal{N}_\mathrm{r}$, check whether the graph $\mathcal{G}(\mathcal{N} \bigcup \mathcal{N}_\mathrm{r} \backslash \{\mathbf{y}\}, r)$ has any other articulation points than those in $\mathcal{G}(\mathcal{N} \bigcup \mathcal{N}_\mathrm{r}, r)$. (The latter articulation points exist only if the problem constraints prevented making the graph $\mathcal{G}(\mathcal{N}_\mathrm{r} \bigcup \mathcal{N} \backslash \mathcal{A}, r)$ connected in the previous step.) If not, let $\mathcal{N}_\mathrm{r} = \mathcal{N}_\mathrm{r} \backslash \{\mathbf{y}\}$.

It is essential that in step 2, all the articulation points are considered absent, because this way chains of several articulation points require in general less relay nodes to be bypassed.

An example of when unnecessary points are included in $\mathcal{N}_\mathrm{r}$ in step 2 and the last step is needed is when the network is a chain of several biconnected clusters (thus each two successive clusters separated by an individual articulation point) and the ends of the chain may be connected – and the network made biconnected – with the insertion of a single relay node: a lot more than this single required point will be output by the algorithm called in step 2. Figure 5 shows an example of such a network.

In particular, whether a given point in $\mathcal{N}_\mathrm{r}$ is deemed unnecessary in step 3 is independent of the existence of the other unnecessary points, and therefore the order in which the points are checked and removed does not matter. To see this, think of the separate, connected clusters of the graph $\mathcal{G}(\mathcal{N} \backslash \mathcal{A}, r)$ as single vertices and the original articulation points as edges connecting these vertices. (Such an edge may consist of several articulation points.) Because a continuous chain of articulation points – or even a single one – may have connected more than two clusters, these edges may be branched. As articulation points, they have formed a tree connecting the vertices, and because they may be branched, we may think of the tree as a Steiner tree. Thus, relay nodes at the points $\mathcal{N}_\mathrm{r}$ found in step

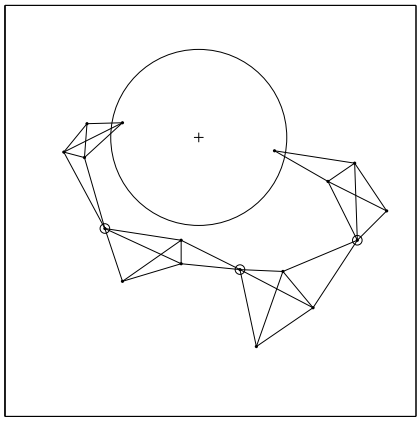


Fig. 5. An example of a connected network that can be made biconnected by adding a single relay node, e.g. at the point marked with the '+'-sign. The articulation points are circled.

2 form another such tree between the vertices, and removing any relay node cuts at least a branch of an edge from this tree. Now, if the union of these two trees has no cycles, the removal of any relay node will leave the connectivity of the vertices up to the original tree and hence the original articulation points, so in this case there are no unnecessary relay nodes. In fact, an unnecessary relay node is such that the branch that its removal cuts only connects vertices that belong to the same cycle in the union of the trees, and even without this branch the cycle would exist in the union. Thus, all unnecessary points in $\mathcal{N}_\mathrm{r}$ can be found and removed independently of each other. This also shows that the network will remain biconnected at the end of the algorithm.

Although intuitively appealing, this algorithm may also give very poor results. Consider the example network of Figure 5: there is no guarantee that the single required point to place a relay node will be found in step 2. Instead, the final result may just as well be that one relay node is added next to each pre-existing articulation point.

Finally, we remark that the same strategy in making a biconnected network triconnected, i.e. figuratively removing all separation pairs at once and reconnecting the network, only rules out separation pairs comprised of terminal nodes, as the resulting network will not in general be triconnected with respect to the added relay nodes. For triconnectivity with respect to all nodes, the separation pairs would have to be removed one at a time, but then the final number of added relay nodes can depend on the order of removal.

VIII. Performance Analysis

In this section, we present and discuss results from applying our three algorithms for simple connectivity to simulated realizations of randomly and uniformly distributed terminal nodes in a square-shaped domain in the plane. The purpose is partly to compare the performance of the algorithms relative to each other, and in part to gain some idea on how close to optimal their solutions are.

The latter is a problematic task, as finding the optimal solution for a general realization is very difficult. As mentioned in Section III, when the transmission range is infinitely shrunk, the optimal solution to Problem 1 is to cover the edges of the Euclidean Steiner minimal tree for the terminal nodes with chains of relay nodes. In this limit, we know the so-called Steiner Ratio: for any set of points in the plane, the total edge length of their Euclidean minimum spanning tree is at most $2/\sqrt{3} \approx 1.15$ times the optimal solution, i.e. the total edge length of their Euclidean Steiner minimal tree [15]. However, with a non-negligible transmission range the case is completely different: as a simple example, consider a regular pentagon whose vertices are on a circle with radius equal to the transmission range, and assume one terminal node at each of these vertices. These initially disconnected terminal nodes can be connected with a single relay node placed at the center of the circle, whereas utilizing the minimum spanning tree results in placing four relay nodes.

Nonetheless, we used as a benchmark for our algorithms the method of placing the relay nodes on those edges of the Euclidean Steiner minimal tree that connect different clusters. This method should be close to optimal with sparse networks, i.e. when the transmission range is small compared to the typical distance between neighboring terminal nodes. Figure 6 shows the average number of relay nodes needed to connect random configurations with varying number of terminal nodes using each of the different algorithms. The transmission range was set to 10% of the side of the square domain, in order to demonstrate a "feasible" scenario where the number of relay nodes needed is still a fraction of the number of terminal nodes, making the addition of relay nodes sensible. As expected, our three algorithms produce gradually better solutions. The two greedy algorithms also outperform utilizing the Steiner tree with these parameters, as the Steiner minimal tree simply optimizes the wrong measure from our problem's viewpoint.

The gain from utilizing the Steiner tree is captured in Figure

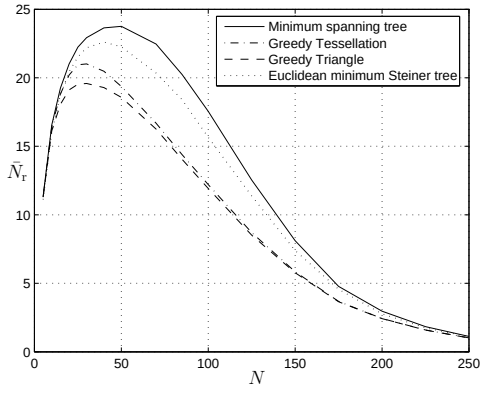


Fig. 6. Average number of relay nodes needed to connect the network, as a function of the number of terminal nodes initially in the network, taken over 1000 random realizations. The transmission range is 10% of the side of the square-shaped domain.

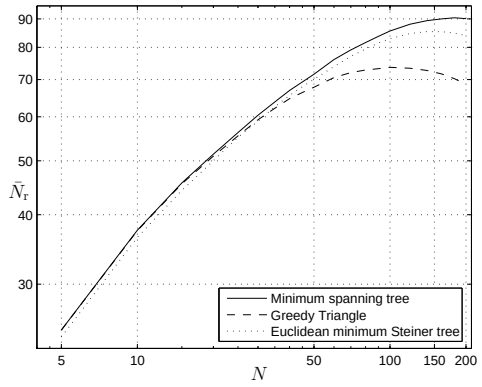


Fig. 7. Average number of relay nodes needed to connect the network, taken over 1000 random realizations and plotted on log-log -scale. The transmission range is 5% of the side of the domain. The Greedy Tessellation algorithm has been omitted for clarity.

7 which shows corresponding results with the transmission range set to 5% of the side of the domain. In a very sparse initial network, the existence of suitable candidate triangles is unlikely, and the Greedy Triangle algorithm practically reduces to the Minimum Spanning Tree algorithm, while the Steiner tree yields the best results. As the density of the initial network increases, the Greedy Triangle algorithm surpasses the Steiner tree method in performance. For almost throughout the range of the figure, the curve of the MST algorithm is a constant shift from that of the Steiner tree method on the logarithmic scale, implying a constant approximation ratio (in terms of the average number of relay nodes that has been plotted). This ratio was approximately 103%.

The quantity that best describes what we referred to as the density of the network is the average number of other terminal nodes directly connected to a random terminal node in the initial configuration. Not accounting for boundary effects, this quantity is given by $N/A \cdot \pi r^2$ where A is the area of the domain. In essence, this quantity determines which method yields the best results, and for example the two greedy algorithms bring significant advantage to using the MST at proper intermediate values of this quantity, when suitable candidate triangles are likely to exist. It is interesting to note in both figures that the average number of relay nodes needed increases with the number of terminal nodes up to the point where $N/A \cdot \pi r^2 \approx 1$: when $r^2/A = (10\%)^2$, this point is at $N \approx 32$, and with $r^2/A = (5\%)^2$ at $N \approx 127$. This is especially true with the two greedy algorithms.

IX. Summary and Discussion

We defined and studied two connectivity problems for static ad-hoc networks, which involve adding new nodes to the network. The first problem is achieving connectivity with a minimal number of additional nodes, which reduces in a limit case to the Euclidean Steiner minimal tree problem. The second problem is maximizing the utility of the network given the number of additional nodes available. Our algorithms for solving the second problem can be used to define greedy approaches for solving the first problem. We also considered the generalization of the problem to k-connectivity, which is closely related to the graph augmentation problem.

We presented three heuristic algorithms for achieving connectivity of a randomly dispersed ad-hoc network by adding relay nodes. The Minimum Spanning Tree algorithm connects clusters pairwise with the shortest chains of relay nodes. The Greedy Tessellation algorithm and the Greedy Triangle algorithm aim at connecting several clusters at each step. The greedy algorithms resort to the Minimum Spanning Tree algorithm in case the clusters are too far apart.

We assumed throughout this paper that all the nodes have equal transmission range. It would also be reasonable to assume that the relay nodes can have a larger range when communicating with each other than when communicating with the terminal nodes. It takes only slight modifications to adjust our algorithms to this relaxed assumption.

We also precluded the mobility of terminal nodes from our assumptions. The approach of adding relay nodes in optimized locations has little application in a scenario where each terminal nodes tends to move all over the network region. However, by keeping track of the locations of terminal nodes over time, it should be possible to recognize those nodes that are nearly stationary and place relay nodes to connect these nodes. Studying this question is left as further work.

Acknowledgements

We thank the anonymous reviewers for the suggestion of Problem 3, and Petteri Kaski (TKK) for pointing out the reference [15]. Henri Koskinen has been financially supported by the Finnish Defence Forces Technical Research Centre and in part by a grant from the Nokia Foundation. The work of Jouni Karvo was funded by the EU FP6-507572 project WIDENS, and Olli Apilo was funded by the Academy of Finland (grant n:o 202204).

References

[1] O. Dousse, P. Thiran, and M. Hasler, "Connectivity in ad-hoc and hybrid networks," in *Proc. INFOCOM'02*, vol. 2, New York, June 2002, pp. 1079–1088.

[2] M. Sánchez, P. Manzoni, and Z. J. Haas, "Determination of critical transmission range in Ad-Hoc Networks," in *Proceedings of Multiaccess Mobility and Teletraffic for Wireless Communications 1999 Workshop (MMT'99)*, Oct. 1999.

[3] 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*, pp. 547–566, 1998.

[4] C. Bettstetter, "On the connectivity of ad hoc networks," *The Computer Journal, Special Issue on Mobile and Pervasive Computing*, vol. 47, no. 4, pp. 432–447, Jul 2004.

[5] H. Koskinen, "A simulation-based method for predicting connectivity in wireless multihop networks," *Telecommunication Systems*, vol. 26, no. 2-4, pp. 321–338, June 2004. [Online]. Available: http://dx.doi.org/10.1023/B:TELS.0000029044.31054.5a

[6] P.-J. Wan and C.-W. Yi, "Asymptotic critical transmission radius and critical neighbor number for k-connectivity in wireless ad hoc networks," in *Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*. ACM Press, 2004, pp. 1–8.

[7] H. Koskinen, "Quantile models for the threshold range for k-connectivity," in *MSWiM '04: Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*. ACM Press, 2004, pp. 1–7.

[8] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–114, Aug. 2002.
[9] C. Savarese, J. M. Rabaey, and J. Beutel, "Locationing in distributed ad-hoc wireless sensor networks," in *Proc. Acoustics Speech and Signal Processing, ICASSP'01*, vol. 4, Salt Lake City, Utah, May 2001, pp. 2037–2040.
[10] M. R. Garey, R. L. Graham, and D. S. Johnson, "The complexity of computing Steiner minimal trees," *SIAM J. Appl. Math.*, vol. 32, no. 4, pp. 835–859, June 1977.
[11] R. Sedgewick, *Algorithms in C*. Addison Wesley, 1990.
[12] F. Aurenhammer, "Voronoi diagrams – a survey of a fundamental geometric data structure," *ACM Comput. Surv.*, vol. 23, no. 3, pp. 345–405, 1991.
[13] T.-S. Hsu, "Graph augmentation and related problems: Theory and practice," Ph.D. dissertation, University of Texas at Austin, 1993. [Online]. Available: citeseer.ist.psu.edu/hsu93graph.html
[14] P. Basu and J. Redi, "Movement control algorithms for realization of fault-tolerant ad hoc robot networks," *IEEE Network*, vol. 18, no. 4, pp. 36–44, Jul 2004.
[15] D.-Z. Du and F. Hwang, "An approach for proving lower bounds: solution of Gilbert-Pollak's conjecture on Steiner ratio," in *Proceedings of 31st Annual Symposium on Foundations of Computer Science*, vol. 1, Oct. 1990, pp. 76–85.

APPENDIX

A. Equivalence of Utility Metrics 1 and 2

Let $\mathcal{V}$ denote the set of all possible solutions to Problem 2. This set is partially ordered using U_1. The optimal solution to Problem 2 is any one of the solutions $v \in V \subset \mathcal{V}$, where V is the smallest set of $v \in \mathcal{V}$ such that $\forall w \notin V,\ w \in \mathcal{V}\ : U_1(w) < U_1(v)$.

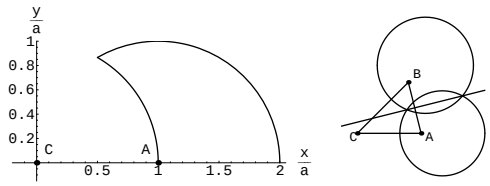
Consider Metric 2: $U_2 = \max_i |\mathcal{C}_i|(|\mathcal{C}|_i - 1) = \max_i(|\mathcal{C}_i|^2 - |\mathcal{C}_i|)$. Since the expression inside $\max(\cdot)$ is strictly increasing with $|\mathcal{C}_i|$ when $|\mathcal{C}_i| \geq 1$, this expression equals $(\max_i |\mathcal{C}_i|)^2 - \max_i |\mathcal{C}_i|$. Thus, this metric is strictly increasing as a function of $\max_i |\mathcal{C}_i|$, and thus a partial order of $\mathcal{V}$ using U_2 is the same as the partial order using U_1. Thus, the smallest set $V' \subset \mathcal{V}$ such that $\forall w \notin V',\ w \in \mathcal{V}\ : U_2(w) < U_2(v)$ equals V.

Thus, the optimal solution for problem 2 using utility function U_1 is the same as the optimal solution for problem 2 using utility function U_2. The same reasoning can be used to analyse greedy algorithms and show that the step that a greedy algorithm would take is the same independently of which of these two utility functions is used.

B. Optimal placement of two relay nodes to connect three terminal nodes

We examine here the problem of connecting three terminal nodes at given locations by placing two relay nodes so that the transmission range required from the relay nodes is minimized.

Let us name the locations of the three terminal nodes as points A, B, and C, forming the vertices of a triangle ABC. Without loss of generality, we assume that $|AB| \leq |CA| \leq |CB|$. We will choose Euclidean coordinates in $\mathbb{R}^2$ so that point C is chosen as the origin and the first dimension is in the direction $\overrightarrow{CA}$, so that the position vector $\bar{r}_A = (a \quad 0)^T$. Let $\bar{r}_B$ denote the position vector of terminal node B. Given the assumption $|AB| \leq |CA| \leq |CB|$, the triangle can always be flipped and rotated so that point B is located inside the bounded set depicted in Figure 8(a). Also, note that any triplet



(a) (b)

Fig. 8. The set of possible locations of terminal node B (a); example triangle (b)

of points in $\mathbb{R}^d$ is located on a $\mathbb{R}^2$ plane, and thus this algorithm generalizes easily to $\mathbb{R}^d$.

The optimal solution for the locations of the two relay nodes is either

A to use the relay nodes to split in half the two shortest sides (CA, AB) of the triangle ABC, or

B first, to connect two terminal nodes (A and B) with one relay node and then place the second relay node midway between the first relay node and the remaining terminal node C to be connected.

It can be deduced that whenever case **B** is optimal, the first relay node must connect the two terminal nodes closest to each other. This is because by the possibility of case **A**, the least required transmission range cannot exceed half the second-shortest distance between two of the three terminal nodes.

For now, assume that case **B** above is optimal (we will derive the conditions for this later). Under this assumption, our task is to optimize the location of the first relay node, point P, so as to minimize the required range $\max\{f_1(P) = |CP|/2, f_2(P) = |AP|, f_3(P) = |BP|\}$. For an explanation of $f_1(P)$, recall that in case **B**, the second relay node is placed in the middle of edge CP. We know that the optimal P must lie inside the triangle ABC, for otherwise it would be possible to decrease the distance to all the points A, B, and C by moving P. Furthermore, in the optimum we must have $f_i(P) = f_j(P) \geq f_k(P)$, for some $i \neq j \neq k; i, j, k \in \{1, 2, 3\}$, otherwise P could again be improved. Here, strict inequality applies if the two equal functions have attained their least possible common value.

The equation $f_1(P) = |CP|/2 = f_2(P) = |AP|$ is satisfied by points P located on the circle with radius $\frac{2}{3}|CA|$, centered at $\frac{4}{3}\bar{r}_A$. Another circle is defined accordingly by the equation $f_1(P) = f_3(P)$. The solutions of the equation $f_1(P) = f_2(P) = f_3(P)$ are hence the intersections of these two circles, which are on the line $|AP| = |BP|$, as shown by Figure 8(b). It is easy to show that under the assumption $|AB| \leq |CA| \leq |CB|$, these intersections always exist. In light of the above, the optimal P is located at such an intersection if it is inside the triangle, otherwise it is at the intersection of the line or circle $f_i(P) = f_j(P)$ and the side of the triangle where the value $f_i(P) = f_j(P)$ is smallest.

We know that one of the intersections $f_1(P) = f_2(P) = f_3(P)$ always falls outside the triangle ABC. In order that the other intersection not fall outside the side AB, the midpoint of AB must lie inside the two circles. In fact, it suffices to

write this condition for one circle only, since it implies the other, so we get

$$\left|\frac{\bar{r}_A+\bar{r}_B}{2}-\frac{4}{3}\bar{r}_A\right|<\frac{2}{3}|CA| \Leftrightarrow \left|\bar{r}_B-\frac{5}{3}\bar{r}_A\right|<\frac{4}{3}|CA|, \quad (1)$$

i.e. point B must lie inside the circle with radius $\frac{4}{3}|CA|$, centered at $\frac{5}{3}\bar{r}_A$. If this condition is not satisfied, the optimal P — given the assumption that case **B** really is optimal — is located midway between A and B, at $(\bar{r}_A+\bar{r}_B)/2$. On the other hand, the condition for the other intersection not falling outside the side CB is

$$\left|\frac{2}{3}\bar{r}_B-\frac{4}{3}\bar{r}_A\right|>\frac{2}{3}|CA| \Leftrightarrow |\bar{r}_B-2\bar{r}_A|>|CA|, \quad (2)$$

i.e. point B must lie outside the circle with radius $|CA|$, centered at $2\bar{r}_A$. If this condition is not satisfied, the optimal P — given the assumption that case **B** really is optimal — is located on the segment CB at $\frac{2}{3}\bar{r}_B$. (Under the assumption $|AB| \le |CA| \le |CB|$, the other intersection cannot fall outside the side CA.)

In general, case **B** is optimal if the optimal P presented above satisfies $|CP|<|CA|$. It is easy to check that whenever the optimal P is at $(\bar{r}_A+\bar{r}_B)/2$, this condition is always satisfied. When it is at $\frac{2}{3}\bar{r}_B$ (i.e. when $|\bar{r}_B-2\bar{r}_A|<|CA|$), the condition becomes

$$\frac{2}{3}|CB|<|CA| \Leftrightarrow |CB|<\frac{3}{2}|CA|. \quad (3)$$

Finally, let us derive the condition $|CP|<|CA|$ for the intersection $f_1(P)=f_2(P)=f_3(P)$ falling inside the triangle ABC. Let $\bar{r}_B=(x \quad y)^T$. A general point P on the line $|AP|=|BP|$ is then at $\bar{r}_P=((a \quad 0)^T+(x \quad y)^T)/2+(y-0 \quad -(x-a))^T t$. The value of the scalar t corresponding to the intersection $f_1(P)=f_2(P)=f_3(P)$ falling inside the triangle ABC can be solved to be

$$t=\{4\,a\,y-[16\,a^2\,y^2-\left(3\,a^2-10\,a\,x+3\,x^2+3\,y^2\right)\cdot \left(12\,a^2-24\,a\,x+12\,x^2+12\,y^2\right)]^{1/2}\} / \left(12\,a^2-24\,a\,x+12\,x^2+12\,y^2\right).$$

On the other hand, the value of t that makes $|CP|=|CA|$ and results in the greater x-coordinate for P, is

$$t=[-2\,a\,y +(3\,a^4-8\,a^3\,x+6\,a^2\,x^2-x^4+6\,a^2\,y^2-2\,x^2\,y^2-y^4)^{1/2}] /[2\,\left(a^2-2\,a\,x+x^2+y^2\right)].$$

We get the boundary for the condition of interest by setting these two values equal. The resulting equation is satisfied on the circles with radii $|CA|/2$, centered at $\left(\frac{7}{8}a \quad \pm\frac{\sqrt{15}}{8}a\right)^T$. Only the upper one of these creates a boundary in the domain of interest. Simple experimenting shows that given the conditions (1) and (2), case **B** is optimal if

$$\left|\bar{r}_B-\left(\frac{7}{8}a \quad \frac{\sqrt{15}}{8}a\right)^T\right|<|CA|/2. \quad (4)$$

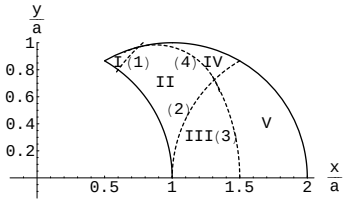
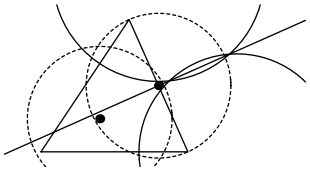
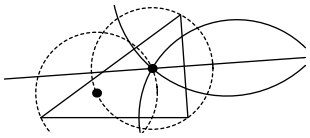
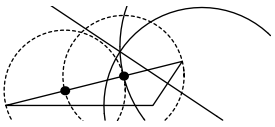


Fig. 9. The division of the possible locations of terminal node B according to the optimal placement of two relay nodes

The labels of the conditions (1) to (4) have been placed near their respective boundaries in Figure 9, so that each label is on the side of the boundary where the condition is satisfied. The resulting five subsets of the possible locations of terminal node B have been labelled using Roman numerals, and the optimal solution in each subset is summarized in Table I.

TABLE I

OPTIMAL PLACEMENT OF RELAY NODES ACCORDING TO THE LOCATIONS OF TERMINAL NODE B IN FIGURE 9

Subset	Solution and example
I	Place first node P at $\bar{r}_P=(\bar{r}_A+\bar{r}_B)/2$, place second node midway between C and P 
II	Place first node P in the intersection $f_1(P)=f_2(P)=f_3(P)$ inside the triangle, place second node midway between C and P 
III	Place first node P at $\bar{r}_P=\frac{2}{3}\bar{r}_B$, place second node midway between C and P 
IV,V	Place one node midway between C and A and the other node midway between A and B