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

1. 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. In this paper however, 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, 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 of the network. We present algorithms that suggest locations for such additional nodes. Networks where adding extraneous nodes is feasible are some 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.

This paper is organized as follows: in the next section we describe the problem setting and the underlying assumptions, and define essentially two opti-

[*]We thank the anonymous reviewers of this paper for their constructive suggestions, and Petteri Kaski (TKK) for pointing out the reference [2].
[†]Financially supported by the Finnish Defence Forces Technical Research Centre and in part by a grant from the Nokia Foundation.
[‡]Funded by the EU FP6-507572 project WIDENS.
[§]Funded by the Academy of Finland (grant n:o 202204).

mization problems for the node addition. Section 3 describes the first heuristic algorithm, the Minimum Spanning Tree algorithm. A more efficient algorithm, the Greedy Tessellation algorithm is presented in Section 4, and the last and most evolved algorithm, the Greedy Triangle algorithm in Section 5. Section 6 contains performance analysis of the algorithms, aided by simulation results. Finally, Section 7 concludes the paper with some final remarks.

2. Problem Statements

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 assumptions behind our problem statement are that the locations of the terminal nodes are known and that the location information of these nodes can be collected even though the network is not connected. The motivation behind the latter 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. Within this model, any two nodes are assumed to be directly bidirectionally connected if and only if they are within a common transmission range from each other.

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, $\mathcal{N} = \{\mathbf{x}_i \in \mathcal{S} \mid i = 1, 2, \ldots, N\}$, 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 $\mathcal{N}$: the optimal solution is then to place the relay nodes along the edges of this tree. Finding Steiner minimal trees is known to be $\mathbf{NP}$-hard [3]. 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. 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 $N_\mathrm{r} \leq N_\mathrm{r}^{\max} \in \mathbb{Z}_+$.

Provided that the constraint $N_\mathrm{r} \leq N_\mathrm{r}^{\max}$ actually prevents us from achieving connectivity, this problem can be viewed as the maximization of the chosen utility metric in $\mathcal{S}^{N_\mathrm{r}^{\max}}$. 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.

The choice of the utility metric depends on the target application. In all examples and simulations where applicable in this paper, we have used the number of nodes in the biggest cluster after each step as the utility metric.

3. 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. [4]).

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 $O(N^{4/3} \log^{4/3} N)$ [1]. 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 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.

The Minimum Spanning Tree algorithm is illustrated in Figure 1.

4. 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. Such points can be found from the Voronoi tessellation (or Voronoi diagram) of the nodes: see [1] 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 coincid-

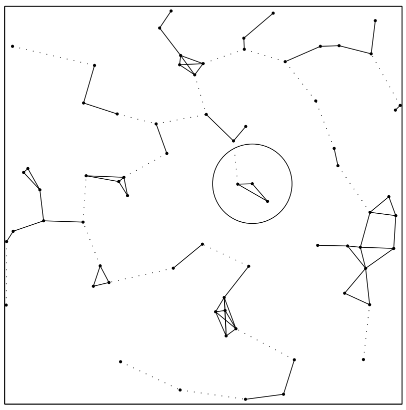

Figure 1. Minimum Spanning Tree algorithm. The initial clusters in this example realization of 70 terminal nodes are connected with solid edges, and the edges to place relay nodes are dotted. The transmission range is 10% of the side of the domain, as illustrated by the circle.

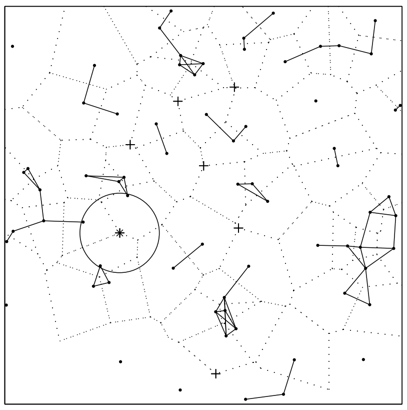

Figure 2. The Greedy Tessellation algorithm, applied to the same realization as in Figure 1. The edges of the Voronoi tessellation are shown with dotted lines, the candidate points for relay node insertion with '+'-signs. The first location to add a relay node is marked with an asterisk.

ing 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 [1]. 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 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. Figure 2 illustrates the algorithm.

5. 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: for example, the point marked in Figure 2 as the place for the first relay node falls outside the triangle spanned by the three terminal nodes to be connected, meaning that it cannot be the optimal place for a relay node to connect the three terminal nodes (optimal in the sense that the range required from the relay node to connect the terminal nodes is minimized). Hence, taking only 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 3(a) and compare with Figure 2). 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. 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; we omit the analysis of these cases in this paper due to space limitations.

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.

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,

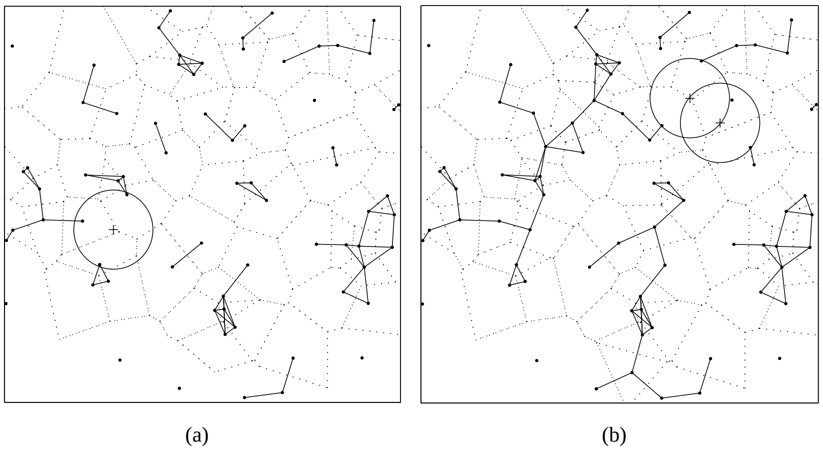

Figure 3. Applying the Greedy Triangle algorithm to the realization of Figure 1. (a): The first point to place a relay node, as determined in step 2 and indicated by the '+'-sign. Note the difference from Figure 2 in the placement. (b): The first pair of points to place relay nodes, as determined in step 3, after several relay nodes have been added in step 2. Note that in this case, four clusters are connected.

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 3.

6. Performance Analysis

In this section, we present and discuss results from applying our three algorithms 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 2, 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 [2]. 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 4 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.

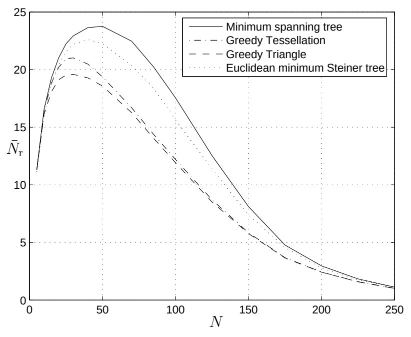

Figure 4. Average number of relay nodes needed to connect the network, as a function of the number of terminal nodes initially in the network, taken over 1000 random realizations. The transmission range is 10% of the side of the square-shaped domain.

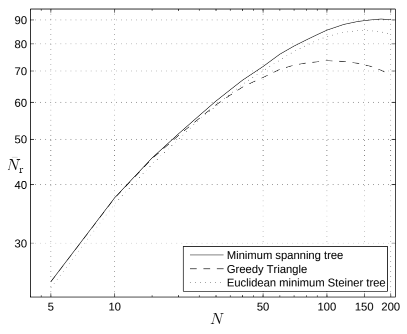

Figure 5. Average number of relay nodes needed to connect the network, taken over 1000 random realizations and plotted on log-log -scale. The transmission range is 5% of the side of the domain. The Greedy Tessellation algorithm has been omitted for clarity.

The gain from utilizing the Steiner tree is captured in Figure 5 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.

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.

7. Discussion

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 in our assumptions. The approach of adding relay nodes in optimized locations has little application if all the terminal nodes tend 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.

References

[1] F. Aurenhammer. Voronoi diagrams – a survey of a fundamental geometric data structure. *ACM Comput. Surv.*, 23(3):345–405, 1991.

[2] 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*, volume 1, pages 76–85, Oct. 1990.

[3] M. R. Garey, R. L. Graham, and D. S. Johnson. The complexity of computing Steiner minimal trees. *SIAM J. Appl. Math.*, 32(4):835–859, June 1977.

[4] R. Sedgewick. *Algorithms in C.* Addison Wesley, 1990.