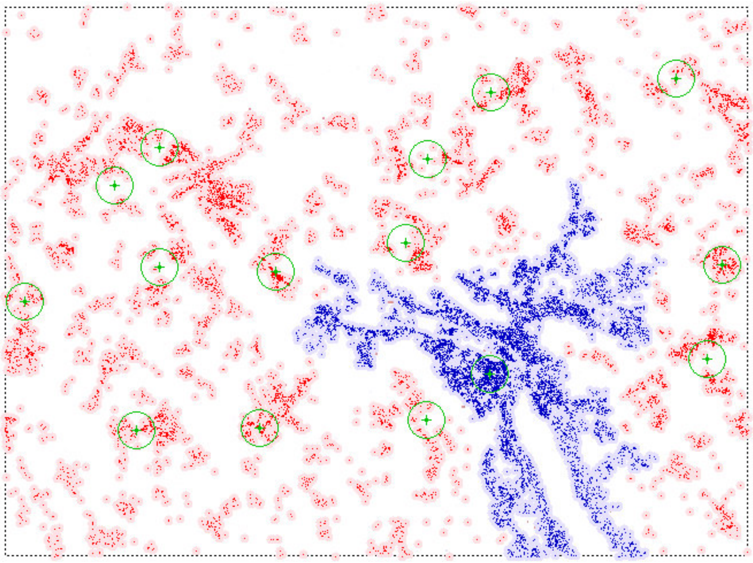
Base Stations in Mobile Ad-Hoc Networks

Thomas Lochmatter
thomas.lochmatter@epfl.ch

July 4, 2004

EX032/2004



Examiner

Prof. Erik Ström
Chalmers University of Technology
Göteborg, Sweden

Supervisors

Dr. Petteri Mannersalo and Prof. Patrick Thiran
EPFL-IC-LCA
Lausanne, Switzerland

Abstract

Multi-hop ad-hoc networks consist of nodes which cooperate by forwarding packets for each other to allow communication beyond the power range of each node. In pure ad-hoc networks, no additional infrastructure is required to allow the nodes to communicate.

Multi-hop hybrid networks are a combination of ad-hoc and cellular networks. As in ad-hoc networks, the nodes forward packets on behalf of other nodes. However, a few base stations are introduced. This enables long-range communication, increases connectivity and allows centralized services.

In our work, we investigate the problem of placing base stations in multi-hop hybrid networks. Since nodes extend the service area by themselves, conventional cellular approaches are not suitable for such networks. We propose the *Cluster Covering Algorithm*, an algorithm which takes into account the percolation phenomenon, and compare it with several greedy algorithms.

We measure the connectivity through different simulations on real population distribution data of Zurich (CH), the Surselva Valley (CH) and Finland. The simulation results show that the Cluster Covering Algorithm outperforms the greedy algorithms.

Acknowledgments

I would like to thank Dr. Petteri Mannersalo, Prof. Patrick Thiran, Prof. Erik Ström and Rico Zenklusen for their guidance, help and discussions along with this thesis.

Contents

Chapter 1

Introduction

In the last ten years, mobile wireless networks have become very popular. Mobile telephony and mobile messaging are only two applications of wireless networks. The most recent mobile networks can transfer virtually any kind of data. These data services (e.g., GPRS or WLAN 802.11) are getting more and more in use.

So far, almost all commercially successful wireless networks are so-called infrastructure networks. They usually require to set up base stations in the area to be serviced. Some networks require additional infrastructure to organize the network, to find mobile devices or to provide gateways to other networks. In the GSM network [1], for example, we identify the Mobile Switching Center (MSC), the Home Location Register (HLR) and the Visitor Location Register (VLR) among other centralized network infrastructure.

Infrastructure networks are most often organized in cells with one or more base stations each. All base stations of a telecommunication operator are interconnected by a high speed backbone network (usually wired). Figure 1.1 shows such a network with 6 base stations. The coverage area in infrastructure networks is determined by the cells. A mobile phone outside the cells has no possibility to access services.

Beyond these popular infrastructure networks, two other types of wireless networks exist: *ad-hoc networks* and *hybrid networks*.

1.1 Ad-Hoc Networks

In a strict sense, ad-hoc (or self-organizing) networks do not require any infrastructure to provide communication services. As soon as mobile devices come close to each other, they detect each other and start to organize themselves without any central authority. A famous example for such a network is bluetooth [2]. WLAN 802.11 [3] and HiperLAN2 [4] also support an ad-hoc mode in which devices talk to each other without the need of a base station.

In ad-hoc networks, mobile devices are also called *nodes* or *terminodes* [5]. We will use these names interchangeably.

Ad-hoc networks can be classified in single-hop and multi-hop ad-hoc networks. The infrared port of a laptop is a typical single-hop network. It can be used to transfer data

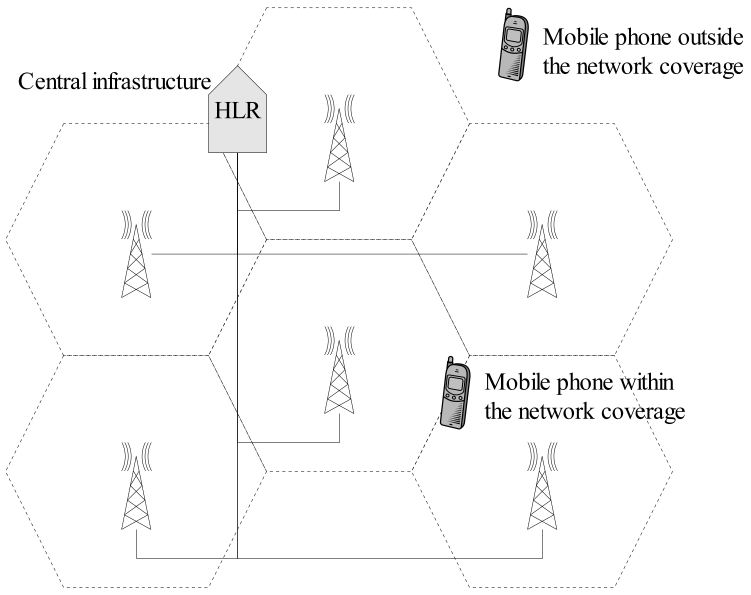


Figure 1.1: An simple infrastructure network with 6 cells. All base stations are connected by a wired network.

between two laptops. However, each laptop can only communicate with its directly connected neighbors (unless special software is used). In multi-hop networks, nodes can use other nodes as intermediate relays to extend their communication area. As an example, Figure 1.2 shows a situation in which laptop A can communicate with a mobile phone E through the intermediate devices B, C and D. Note that A and E are not in each others power range and therefore not directly connected.

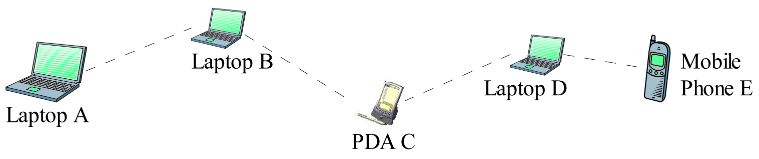


Figure 1.2: A multi-hop connection between nodes A and E.

The service area in multi-hop ad-hoc network is created purely by the nodes that form the network. This area therefore changes when the nodes move. Furthermore, as shown in Figure 1.3, not all nodes can communicate with all other nodes, although every node has neighbors (directly connected nodes). A set of connected nodes is called *cluster*. Communication between clusters is not possible in pure ad-hoc networks. In reality, such clusters could appear very often in large-scale networks. Because of geographical (valleys, lakes, ...) and cultural (cities, villages, streets, ...) reasons, the distribution of the population is non-uniform. It has self-similar (fractal) properties [6], which favors the appearance of clusters.

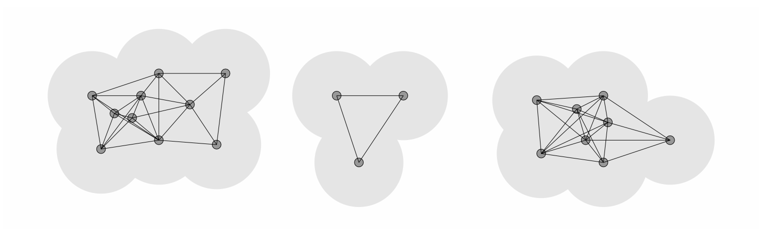


Figure 1.3: Nodes in a multi-hop ad-hoc network with their links. The gray area is determined by one half of the communication range.

Another connectivity problem is related to percolation [7]: If the node density (number of nodes per square meter) falls below a critical value, it is very unlikely to observe big clusters [8]. In this so-called sub-critical density, the advantage of multi-hopping disappears almost completely. Communication to distant nodes is hardly possible. Such situations may appear in alpine areas for example.

In order to profit from multi-hopping, the node density needs to be above this critical value (super-critical density). This is usually the case in cities and villages.

From a protocol point of view, studies in the last years have shown that pure multi-hop ad-hoc networks are difficult to design. A lot of work has been investigated in routing algorithms, but many of them do not scale in large networks (i.e., networks with hundreds or thousands of nodes). An overview of different approaches and more references are given in [9] [10] [11]. Recent papers also address security and fairness issues in different layers and how they can be tackled. An overview on this topic can be found in [12].

1.2 Hybrid Networks

To combine the advantages of pure ad-hoc networks with those of infrastructure networks, hybrid networks have been proposed. Such networks are self-organizing to some extend, but additionally consist of a small set of base stations (see Figure 1.4). Indeed, adding base stations has a number of advantages:

Increased connectivity Base stations can interconnect clusters. They can also cover areas with sub-critical nodes densities, i.e., areas with too few nodes for clusters to appear.

Increased capacity Even if a path between two big cities exists without base stations, the nodes which connect these cities would suffer from high traffic load and therefore be a bottleneck. Base stations interconnected by a high speed wired network can sustain much more traffic.

Decreased delay and jitter Multi-hopping increases delay and jitter of the packets. In practice, the delay is mostly determined by the number of hops. For real-time services (e.g., telephony, gaming) over long distances, this could be a severe problem.

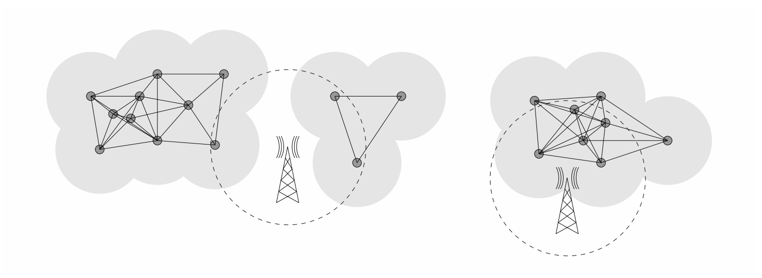


Figure 1.4: A hybrid network.

Increased communication range Some ad-hoc routing protocols use a hop count limit to cope with network flooding or routing loops. In large networks, however, such counters also limit the communication range. Base stations acting as wormholes solve this problem.

Gateways Base stations (or the infrastructure behind) can provide gateways to existing networks, such as the Internet or the telephone network.

More services Some protocols require a central server. For instance, some protocols to improve fairness and security [13] [14] require a central, trusted authority.

Accounting Base stations allow commercial service providers to sell services.

The main advantage of adding multi-hop functionality to the network is the infrastructure cost. Since nodes can be connected without being covered by a base station, fewer base stations are required as compared to pure infrastructure networks. This makes hybrid networks particularly interesting for telecommunication companies and a candidate for future wireless networking.

1.3 Summary and Classification

Table 1.1 shows a summary of the discussed networks.

1.4 Placing Base Stations in Hybrid Networks

In this master thesis, we study the problem of placing base stations in hybrid networks. In principle, the same algorithms as for infrastructure networks could be used. In that case, however, there would be almost no gain by employing multi-hop hybrid networks. The key advantage of hybrid networks is that the same connectivity can be reached with fewer base stations.

We therefore propose new algorithms to place base stations in hybrid networks with non-uniform (but probabilistically known) distribution of the nodes. Beyond some greedy

	With infrastructure	Without Infrastructure
Single-Hop	Today's infrastructure networks, eg. GSM, WLAN in infrastructure mode, HiperLAN2 in infrastructure mode	Some of today's point-to-point networks, eg. infrared port on a laptop or on a mobile phone, WLAN in ad-hoc mode, HiperLAN2 in ad-hoc mode, Bluetooth
Multi-Hop	Hybrid multi-hop networks	Pure multi-hop ad-hoc networks

Table 1.1: Classification of wireless networks with respect to infrastructure and multi-hopping.

attempts (Chapter 4), we introduce the Cluster Covering Algorithm (Chapter 5). It takes into account the percolation phenomenon and puts the base stations at positions where they cover one or more ad-hoc clusters.

To evaluate and compare the performance of the different algorithms, we apply them on real population density data of Zurich (CH), the Surselva valley (CH) and Finland (Chapter 6).

1.5 Related Work

The problem of placing base stations in infrastructure networks has been well studied. Most often, a cellular approach is deployed [1]. Each base station covers a certain area in the network which is called a cell. Hexagonal cell shapes are usually used, since this shape offers the best ratio between the number of base stations and the coverage area. The network capacity is mainly determined by the cell size (or the number of base stations). Though, methods like *cell splitting* or *sectoring* may increase the capacity while keeping the same cell size.

In [15], a demand-based engineering method for planning future cellular mobile communication systems is presented. The *integrated approach* focuses on the network demand and not only on the coverage area. It furthermore addresses RF design aspects. The algorithm presented in the paper is based on so-called demand nodes which are dense in areas of high demand and sparse in regions with low demand. The problem of setting base stations is formulated as a Maximal Covering Location Problem (MCLP) and solved using a variant of a greedy set covering heuristic.

In hybrid networks, only very few papers address the problem of placing base stations. In [8], the effect of base stations on percolation is studied. In the first part of the paper, the nodes are assumed to be uniformly distributed and the base stations are placed in a regular lattice. It turns out that for the case of a 1-dimensional line (or a thin strip), base stations can increase the connectivity significantly. However, in the case of a 2-dimensional plane (or a large strip), the connectivity improvement by base stations remains marginal. In the second part, the nodes are considered to be non-uniformly distributed. Population distribution data of Zurich (CH) and the Surselva valley (CH) is used. The paper shows that the connectivity can be improved by base stations in both areas. In the Surselva valley (almost 1-dimensional distribution of the nodes), connectivity is only achieved if the nodes have a large power range. Deploying a pure ad-hoc network would therefore require a lot of energy from the mobile devices. Introducing base stations changes the picture significantly. Connectivity is achieved for reasonable power ranges. In the region of Zurich (2-dimensional), base stations increase the connectivity as well, although the connectivity gain is not as high as for the Surselva valley.

The impact of percolation on connectivity and interferences has been studied in [8] [16] [17]. In [17], different connection probability functions and different connectivity shapes are compared with regard to percolation. It is conjectured and shown through simulations that discs (circular connection ranges) are worst for percolation. When using other shapes, the critical node density for percolation to appear is below the critical density when using disks.

1.6 Structure of this Thesis

This master thesis is structured as follows: We first define the models and assumptions in Chapter 2. In Chapter 3, we formally state the problem. In Chapter 4, we discuss four greedy algorithms to place base stations. In Chapter 5, we present the Cluster Covering algorithm. Finally, the simulation results of all algorithms are shown and discussed in Chapter 6.

Chapter 2

Models and Assumptions

As mentioned in the introduction (see Chapter 1), a hybrid network consists of an ad-hoc network with base stations. Hence, to model such a network, we mainly need to define

- how nodes are connected to each other (ad-hoc model)
- how base stations interact with the nodes (base station model)

We furthermore introduce the density map model to model the distribution of the nodes (or the population).

2.1 Ad-Hoc Model

We consider an ad-hoc network with n nodes where n is in the order of 10^4 to 10^6. A node has a position (x,y) measured from some reference point. All nodes are assumed to be in the same plane.

Each node has the same deterministic, positive and finite communication range r. In other words, two nodes A and B are *directly connected* (or *neighbors*) if and only if

$$d(A,B) = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2} < r$$

where $d(A,B)$ denotes the physical distance between the two nodes. This implies that the communication channels are symmetric, i.e., if A is a neighbor of B, B is a neighbor of A and vice versa.

We do not consider interference [16] and radio propagation issues [1] in our model. All kind of obstacles that could hinder a wave to propagate and/or reflect it are ignored. We also assume that the antennas are omnidirectional in the plane. There are mainly four reasons that justify these approximations:

- Effects like radio propagation and interference are difficult to consider in a large scale network because they usually appear in small scale. Moving a node by a few centimeters can change the connectivity picture quite much. Very precise data about the environment would be necessary to model them properly and the calculations would be very complex.

- The nature and level of interferences depend on the channel sharing scheme (e.g., TDMA, FDMA, CDMA), the low-layer protocols (e.g., DSSS, CSMA/CA), the type of modulation (e.g., broad-band or narrow-band) and the frequency. The degree of self-organization and the routing mechanisms may also change the picture. For example, it makes a difference if the base stations are allowed to allocate channels to all nodes or if the channels are shared in a completely self-organized way.

- Since we consider very many nodes, we believe that the different connections compensate for each other. The communication range r chosen in the simulation therefore stands for an average communication range in reality.

With the concept of neighbors, an ad-hoc network can be represented as a graph $G(V,E)$. V is the set of nodes (mobile devices) in the network. Links are represented by edges:

$$(A,B) \in E \Longleftrightarrow A \text{ and } B \text{ are neighbors}$$

Figure 2.1 shows an example of such a graph.

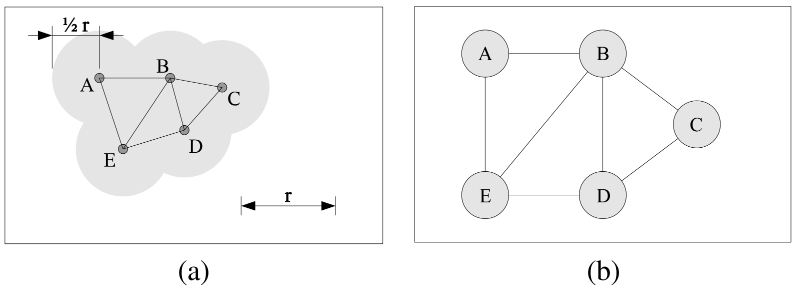


Figure 2.1: A small ad-hoc network (a) and the corresponding graph (b). Each node in the network corresponds to a node in the graph. An edge denotes direct connectivity (neighborhood).

Two nodes are *connected* if there exists a path in the graph G between these two nodes. We do not limit the length of this path to any value, although some routing algorithms do so.

A *cluster* is a set of nodes that are connected all together. In other words, two nodes belong to the same cluster if they are connected (the inverse holds as well).

From percolation theory [7], we know that the appearance of clusters is closely related to the node density (the number of nodes per area). In an infinite, two-dimensional area, there exists a critical density, λ_c, [18] below which an unbounded cluster appears with probability zero. Above this critical density, an unbounded cluster appears with unit probability. However, the presence of an unbounded clusters doesn't imply that all nodes are connected. There may still be bounded (finite) clusters that do not belong to the unbounded cluster. As the density increases, the fraction of the nodes that are not part of the unbounded cluster decreases and finally diminishes to zero for infinite node density.

Since our area is finite and the distribution non-uniform, these results from percolation theory do not hold any more. However, the probability that nodes are connected is still related to the density. If the node density (locally) is much above the percolation threshold, then the nodes are with high probability connected and form a cluster. If the density is much below the percolation threshold, then the nodes will most likely form disjoint sets of very small clusters. In between, there is a smooth transition.

When we represent ad-hoc networks, we often draw gray circles with radius $\frac{r}{2}$ around the nodes. This has nice visual properties:

- If two circles intersect, then the corresponding nodes are directly connected.
- Each gray ”cloud” represents a cluster in the network.

Note that these circles do not correspond to the coverage area. The coverage area would be drawn with circles of radius r.

2.2 Base Station Model

A base station B_i is determined by its position (x_{B_i}, y_{B_i}). We assume that the base stations are in the same plane with the nodes.

All base stations have the same communication radius b, which may be different from the communication radius of the nodes (previously defined as r). In practical cases, $b > r$, because base stations usually have better antennas and more power available. A node A is connected to a base station B_i if and only if

$$d(A, B_i) = \sqrt{(x_A - x_{B_i})^2 + (y_A - y_{B_i})^2} < b$$

We assume symmetric connectivity between nodes and base stations. This is not necessarily the case in hybrid networks. In fact, since the base stations are less power-restricted, the downlink range b_d (from the base stations to the nodes) could be bigger than the uplink range b_u (from the nodes to the base stations). This, however, doesn't affect connectivity per se, as we show in Figure 2.2.

In Figure 2.2 (a), we have drawn the asymmetric case. We observe that node A receives data directly from the base station (downlink) but sends data through two intermediate nodes to the base station (uplink). Node A is therefore connected to the base station. Node B on the other hand is not connected, since it cannot send data back to the base station. In the symmetric case in Figure 2.2 (b), node A is connected as well. This time it sends and receives the data through the intermediate nodes. Node B is obviously not connected.

Therefore, base stations with asymmetric connectivity (b_d, b_u) can be modeled as base stations with symmetric connectivity where $b = \min(b_d, b_u)$ is the smaller of both ranges. (Note that this statement would not be true if interferences or throughput capacity were taken into account.)

For the same reasons as for the communication between nodes, we do not take radio propagation issues into account. We assume that base stations have a circular coverage pattern in the plane of the nodes. This still allows base stations to have several directional antennas.

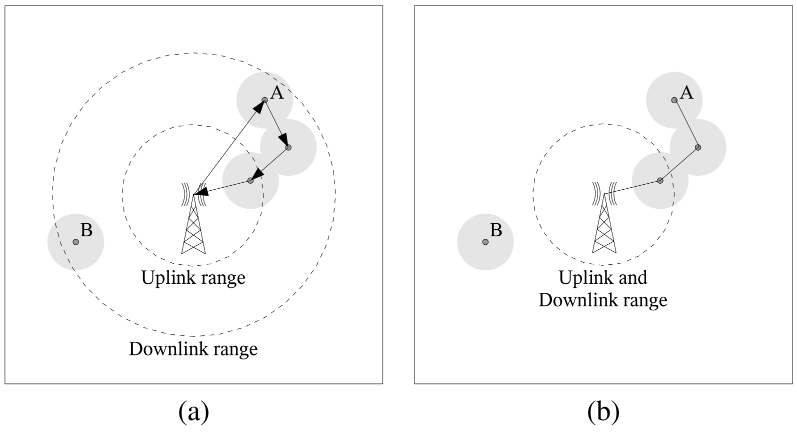


Figure 2.2: Asymmetric (a) and symmetric (b) base station model. The dashed circles delimit the communication ranges of the base stations. The two-way connectivity is the same in both cases.

All base stations are assumed to be interconnected by a high speed wired network. As a consequence, if two nodes are connected to any base stations (not necessarily the same), the are connected.

When we draw base stations in a hybrid network, a dashed circle indicates the communication range. All nodes (not the node communication ranges) inside this circle are *directly connected* and therefore *neighbors* of the base station. A node is *connected* to a base station if at least one node in the same cluster is a neighbor of the base station. In this case, we also say that the corresponding cluster is *covered* by the base station.

2.3 Density Map Model

To model the non-uniform distribution of the nodes, we introduce the density map model.

A density map D is a regular grid of square cells, comparable to a raster image. The cell size (width, height), s, is called resolution and usually lies between $s = 10$ m and $s = 1$ km. All cells in a map have the same size. A better granularity aims more accurate results but is also more difficult to create and to handle. Indeed, time and memory requirements of algorithms that use a density map usually rise at least linearly with the number of cells.

Cells and their properties are often referenced with map coordinates (x, y). In our work, counting always starts at the upper left corner with cell $\gamma_{0,0}$. Thus, the cell in the lower right is γ_{x_m-1,y_m-1}, where x_m and y_m denote the number of cells horizontally resp. vertically.

Each cell $\gamma_{x,y}$ has a non-negative node density value $\lambda_{x,y}$ [m^{-2}] assigned, describing the average number of nodes per square meter. We assume the density to be constant over the cell area.

The node distribution follows a non-stationary Poisson point process (see [19]). However, as cells are disjoint, the process is stationary inside each cell. The number of nodes in the cell, $n_{x,y}$, is thus a Poisson random variable that depends on the density $\lambda_{x,y}$ and the cell size:

$$n_{x,y} \sim \text{Poisson}(s^2 \lambda_{x,y})$$

As an example, the density map of Zurich and its structure is shown in Figure 2.3.

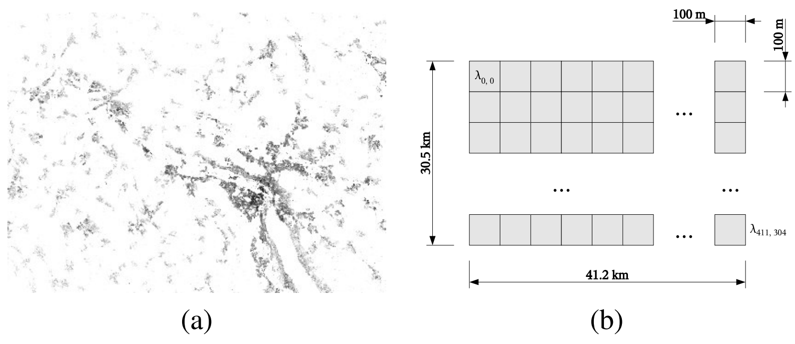


Figure 2.3: The density map (a) of Zurich and its structure (b). It has a resolution of 100 m and spans over an area of 41.2 km x 30.5 km.

Note that the density map model is only a static probabilistic description of the nodes. It is not sufficient to describe moving nodes.

2.3.1 Creating a Density Map

Node density maps can be derived from population density maps. Such data is obtained from statistic offices (e.g., the *Bundesamt für Statistik* [20] in Switzerland). It is natural to assume that the more people are living in a certain area, the more nodes are expected. Hence, we can easily relate

$$\lambda_{\text{nodes}} = \alpha \cdot \lambda_{\text{population}} \tag{2.1}$$

where α describes the estimated number of nodes per human being. For example, if one tenth of the population is expected to carry a mobile device that is switched on, $\alpha = 0.1$.

However, this simple equation has to be used with care:

- Population density maps are usually created by looking at where people have their house. Hence, industrial parts of a city will be underestimated, as very few people live there but many people work there during the day. In contrast, housing areas are overestimated.

- In many cases, only the permanent population is considered. In touristic places, however, the effective density can vary very much and exceed the permanent population by a factor.

- Such maps also neglect social issues. People from higher social classes are more likely to use mobile devices. As an area usually houses people of approximately the same social class, this distorts the map data.

- Streets should be taken into account. Most important are highways and busy streets in a city, because these are almost completely neglected in a population density map. The traffic density of smaller roads can usually be related to the number of people living nearby.

More accurate maps could be created by counting other mobile devices (e.g., mobile phones) in a certain area. People that already use mobile communication devices are likely to switch to a new, better technology sooner or later. Similarly, areas in which traditional mobile devices are used will most likely be areas where future mobile technologies are used.

2.3.2 Generating Nodes from a Density Map

It is quite easy to generate a random set of nodes corresponding to a density map.

We have seen in Chapter 2.3 that the density map describes a non-stationary Poisson point process with stationarity inside each cell. Since cells are non-overlapping and because of the independent scattering property [19], the nodes can be generated independently for each cell.

The generation for each cell γ is done in two steps:

1. Select a random number $n \sim n_\gamma \sim \text{Poisson}(s^2\lambda_\gamma)$.
2. Choose n points (x_i, y_i) with $x_i \sim \text{Uniform}(0,s)$ and $y_i \sim \text{Uniform}(0,s)$ inside the cell.

Recall that s denotes the width and height of a cell.

The total number of nodes generated this way is a random variable, N, and can be expressed as the sum of the nodes in all cells:

$$N \doteq \sum_\gamma n_\gamma$$

Thus, N is a sum of $x_m y_m$ independent Poisson random variables,

$$\begin{aligned} N &\sim \text{Poisson}\left(\sum_\gamma s^2\lambda_\gamma\right) \\ &\sim \text{Poisson}\left(s^2\sum_\gamma \lambda_\gamma\right) \end{aligned}$$

Chapter 3

Problem Statement

As mentioned in the introduction (Chapter 1), the goal of our work is to place base stations in multi-hop hybrid networks. We would like to find an algorithm which places the base stations in a way to connect as many nodes as possible to at least one base station. Recall that a node is connected if there exists a multi-hop path from the node to one of the base stations.

Let us define the *connectivity function*

$$C(m) = \frac{\text{connected nodes with } m \text{ base stations}}{\text{total nodes}}$$

It expresses the *ratio of connected nodes* in the network when m base stations are used. Given a density map, the basic criterion for an algorithm which places base stations can therefore be stated as follows:

$$\max_m \mathrm{E}[C(m)] \quad \forall m \tag{3.1}$$

Since the density map is a probabilistic description of the node distribution, we want to maximize the expectation of the connectivity function $C(m)$.

This is first of all a theoretic problem description which allows us to quantitatively compare different algorithms. We will use this function later in Chapter 6 to evaluate and compare the performance of different algorithms. But the connectivity function is very useful for practical purposes as well. In the following paragraphs, we present three examples.

Maximum Gain Today's economy is most often interested in maximizing the gain. In hybrid networks, it can be written as a trade-off between

- the number of base stations (cost)
- the number of connected nodes (revenue)

Let us assume that a base station involves a fixed investment, c_b. Furthermore, assume that that a connected node brings a fixed revenue of r_c and an unconnected node a revenue of r_u. The following formula then maximizes the gain:

$$\max_m \mathrm{E}[C(m)nr_c + (1 - C(m))nr_u - mc_b]$$

where n denotes the total number of nodes and m the number of base stations.

Connectivity Goal In some applications, we might also have a connectivity goal. This is something that telecommunication companies use to advertise themselves. Furthermore, radio frequencies are sometimes sold with connectivity requirements that the buying company has to fulfill within a certain amount of time. The company therefore wants to connect a certain percentage of the nodes, p_c, with a minimum number of base stations. More formally,

$$\min m \qquad \text{such that } \mathrm{E}[C(m)] > p_c$$

Limited Investment A last problem that might occur is that the investment is limited. In this case, a company would like to connect as many nodes as possible with a maximum of $m_{\max}$ base stations. This problem can be formulated as follows:

$$\max_m \mathrm{E}[C(m)] \qquad \text{such that } m \leq m_{\max}$$

All three optimization problems depend on the same connectivity function $C(m)$. This function is therefore a well-suited candidate to characterize the performance of an algorithm on a given density map.

Chapter 4

Greedy Algorithms

In this chapter, we present four greedy algorithms (K-means, Highest Density, Biggest Clusters, Lattices) to place base stations in hybrid networks and discuss their drawbacks.

By definition, greedy algorithms are simple, straightforward and shortsighted in their approach [21]. Because of these properties, they are usually easy to implement and thus well suited to start with. They provide a simple upper bound in terms of solution cost which can easily be achieved. More sophisticated algorithms should beat the performance of these algorithms.

There is another reason to start with greedy algorithms: if the problem could be solved (optimally) with such an algorithm, there would be no need to go further and to develop more complicated algorithms. However, we found out that the greedy algorithms we studied all have certain disadvantages.

4.1 K-means

The basic K-means algorithm approximates the least-mean-square problem which is defined [22] as

$$\min \sum_{n=1}^{N} \|\mathbf{x}_n - Q(\mathbf{x}_n)\|^2 \tag{4.1}$$

where $\mathbf{x}_1, ..., \mathbf{x}_N$ is a set of N points in an d-dimensional space ($d > 0$) and $Q(\mathbf{x}_1)$, ..., $Q(\mathbf{x}_N)$ their approximation. The algorithm starts with a set of non-optimal points and improves their position in a step-by-step fashion. The points converge towards a local optimum which is not necessarily the global optimum.

Variants of this algorithm are used in various fields of telecommunications. In vector quantization, for example, it is known as the LBG algorithm [22]. It has also been used to minimize the costs of wired telecommunication networks [6]. This is certainly a reason to consider it for ad-hoc networks as well.

We therefore implemented a very basic k-means algorithm in 2 dimensions with the following considerations:

- We start with m base stations, $\mathbf{b}_1, ..., \mathbf{b}_m$, randomly distributed over the density map.

Variables

$\mathbf{x}_1, ..., \mathbf{x}_N$:	demand nodes
$\mathbf{b}_1, ..., \mathbf{b}_m$:	base stations

Algorithm

```
# Initialization
for $j = 1, ..., m$
    $\mathbf{b}_j$ = random point on the density map
end for

# Optimization
$d = \infty$
repeat until $d < d_{threshold}$
    for $j = 1, ..., m$
        $\mathbf{a}_j = (0,0)$
        $n_j = 0$
    end for
    for $i = 1, ..., N$
        select $j$ such that $\|\mathbf{x}_i - \mathbf{b}_j\| \leq \|\mathbf{x}_i - \mathbf{b}_k\| \qquad \forall k \neq j$
        $\mathbf{a}_j = \mathbf{a}_j + \mathbf{x}_i$
        $n_j = n_j + 1$
    end for
    $d = 0$
    for $j = 1, ..., m$
        if $n_j = 0$
            $d = \infty$
            $\mathbf{b}_j$ = random point on the density map
        else
            $d = d + \|\mathbf{b}_j - \frac{\mathbf{a}_j}{n_j}\|$
            $\mathbf{b}_j = \frac{\mathbf{a}_j}{n_j}$
        end if
    end for
end repeat
```

Table 4.1: Sketch of the K-means algorithm implementation.

- $\mathbf{x}_1, ..., \mathbf{x}_N$ are *demand nodes* of the density map. The demand nodes are representative points for the node density (or population density). This is similar to the demand node concept in [15].

- The approximation of a demand node i is its nearest base station, i.e.,

$$Q(\mathbf{x}_i) = \mathbf{b}_j \qquad \text{such that } \|\mathbf{x}_i - \mathbf{b}_j\| \leq \|\mathbf{x}_i - \mathbf{b}_k\| \quad \forall k \neq j \tag{4.2}$$

  To each base station, we assign the set Ω_j of corresponding demand nodes, which is equivalent to

$$i \in \Omega_j \Longleftrightarrow Q(\mathbf{x}_i) = \mathbf{b}_j \tag{4.3}$$

- In each step, every base station is moved to the center of gravity of all its corresponding demand nodes. Formally,

$$\mathbf{b}_j = \frac{1}{|\Omega_j|} \sum_{n \in \Omega_j} \mathbf{x}_n \tag{4.4}$$

As mentioned above, we initialized the base stations randomly. To increase the quality of the result, we run the algorithm multiple times and chose the best result only. There are mainly two ways to generate the demand nodes. One possibility consists in generating random nodes (see Chapter 2.3.2) and using them as demand nodes with unit weight. Another possibility is to generate a lattice of nodes over the whole density map and to weight them according to the density. For the latter method, we have to replace Equation (4.4) by

$$\mathbf{b}_j = \frac{1}{\sum_{n \in \Omega_j} w_n} \sum_{n \in \Omega_j} \mathbf{x}_n w_n \tag{4.5}$$

where w_n denotes the weight of the corresponding demand node.

We found that - independent of a good or a bad initialization and independent of the way to generate the demand nodes - this algorithm has two important drawbacks:

Overconnected Clusters First of all, it tends to place many base stations in areas with high node density (clusters). By the nature of ad-hoc networks, however, these areas are already well connected. A single base station would suffice to connect all nodes of the cluster. Such a situation is shown in Figure 4.1. The grayed shapes represent ad-hoc clusters (high node density). All three base stations were placed in the big cluster on the left, although a single base station would be enough. The cluster on the right hand side is not covered at all.

This problem appears because the k-means algorithm doesn't care about clusters.

Stonehenge Problem The second problem is drawn in Figure 4.2. Seven clusters form a circle which is big enough to host a base station. If the nodes are more or less equally distributed around this circle, its center of gravity happens to be in the center of the circle. Although the k-means algorithm proposes this as an optimal point, the base station is useless. Because of its shape, we call this the *Stonehenge problem*.

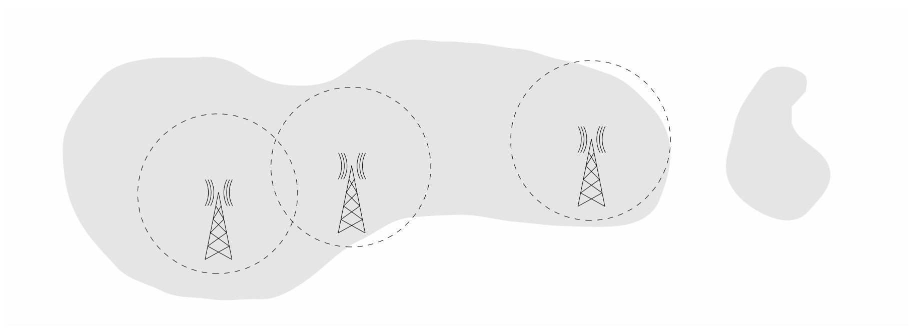


Figure 4.1: The problem of overconnected clusters. One cluster is covered by three base stations whereas the other cluster is not covered at all.

From Figure 4.2, we might think that this is a very constructed situation. However, this situation appears quite frequently in density maps based on real population data, especially with few base stations or small base station ranges. This problem appears because no step of the k-means algorithm takes into account the communication distance of the base station or the nodes.

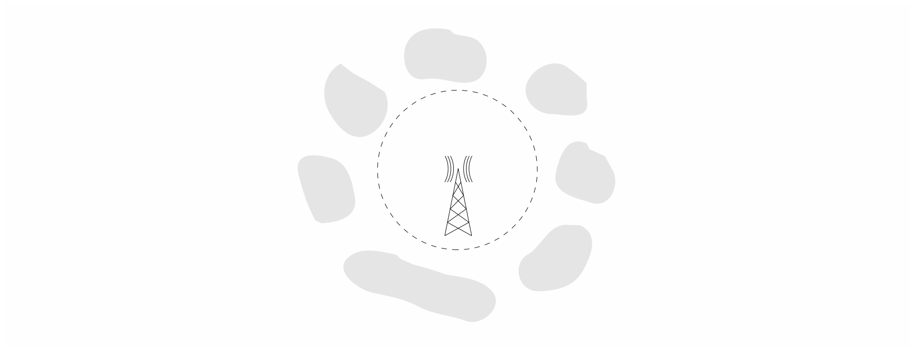


Figure 4.2: The Stonehenge problem. A base station is put in between clusters but doesn't cover any of them.

Both these problems could possibly be diminished by better approximation functions $Q(\mathbf{x})$ or modified distance measures. For example, $Q(\mathbf{x})$ could be chosen as the connected base station (if any) instead of the closest base station. We however believe that there is no simple method to drastically increase the performance of this algorithm.

4.2 Highest Density

Our second attempt deals with the cells of the density map. We have defined in Chapter 2.3 that each cell has a density value λ associated which corresponds to the expected number

Variables

$d_{x,y}$:	density of cell (x,y)
$c_{x,y}$:	cell coverage (1: free, 0: covered)
s:	cell width and height
b:	base station communication radius
$\mathbf{b}_1$, ..., $\mathbf{b}_m$:	base stations

Algorithm

```
# Initialize cells
for each $(x,y)$ in the density map
    $c_{x,y} = 1$
end for

# Place $m$ base stations
for $i$ = 1, ..., $m$
    $(x,y) = \arg\max_{(x,y)} c_{x,y} \cdot d_{x,y}$
    $\mathbf{b}_i = (xs + \frac{s}{2}, ys + \frac{s}{2})$
    for each $(x_1,y_1)$ in the density map
        if $\sqrt{(x_1-x)^2+(y_1-y)^2} < b$
            $c_{x_1,y_1} = 0$
        end if
    end for
end for
```

Table 4.2: Sketch of the Highest Density algorithm implementation.

of nodes in this cell.

The Highest Density algorithm puts the base stations into cells with a high density value. More specifically, the algorithm executes the following steps to place one base station:

1. Among the cells that are not yet covered, choose the (non-empty) cell with the highest density value. (Initially, no cells are covered.)

2. Add a base station in the center of this cell.

3. Mark all cells within the communication range of this base station as covered.

These three steps can be repeated m times, where m denotes the number of base stations (see also Chapter 3). As soon as all non-empty cells (i.e., cells with $\lambda > 0$) are covered, the algorithm stops. In Figure 4.3, an example with three base stations is shown.

This algorithm performs in general badly on real population data. Its tends to over-connect the big cities, since their population density is much higher than in small villages. Furthermore, the criterion not to put a base station within the range of another is not strict

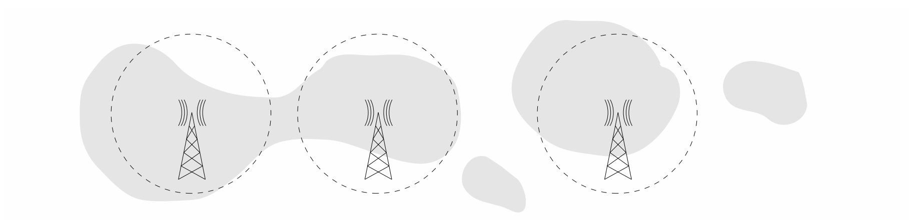


Figure 4.3: $m = 3$ base stations placed with the highest density algorithm. The two main clusters of the network are covered, although an optimal placement would cover the two smaller clusters as well.

enough. In densely populated areas, the distances between the base stations are often very close to the coverage radius. This leads to a very uneven distribution of the base stations in which a lot of small clusters remain uncovered.

Let us anyway note that this basic algorithm could be tuned in several ways. It might for example be advantageous to blur (or down sample) the density map before applying the algorithm. Furthermore, the whole cluster could be marked as being covered instead of the base station range only. Other improvements could possibly be achieved by locally optimizing the base station positions in the neighborhood of the high density cells.

We did not develop this algorithm any further because even with some improvements by preprocessing the density map or post processing the base station positions, it remains short-sighted.

4.3 Biggest Clusters

A similar algorithm is based on the clusters of the density map. How to cluster a density map is discussed later in Chapter 5.4.

Assume we have a density map with n_c clusters and we want to place $m < n_c$ base stations. The size of a cluster, $s_1, ..., s_{n_c}$, refers to the expected number of nodes. The Biggest Clusters algorithm sorts the clusters by their size and puts one base station in the m biggest clusters, as shown in Figure 4.4. Inside the clusters, the base stations are placed in the center of the cell with the highest density.

This algorithm does not overconnect clusters. It decides to either cover a cluster or not. Obviously, the Stonehenge problem doesn't appear neither.

However, the algorithm is very short-sighted. The main weakness is that it considers each cluster to be completely isolated from all other clusters. Figure 4.4 shows that the third base station accidentally covers the second-biggest cluster as well. But the algorithm doesn't take this into account. Recall that an optimal solution would cover all four clusters.

Variables

$C_1, ..., C_{n_C}$:	list of clusters
$d_{x,y}$:	density of cell (x,y)
s:	cell width and height
$\mathbf{b}_1, ..., \mathbf{b}_m$:	base stations

Algorithm

```
sort $\{C_1,...,C_{n_C}\}$ by the cluster size
for $i$ = 1, ..., $m$
    $(x,y) = \arg\max_{(x,y)\in C_i} d_{x,y}$
    $\mathbf{b}_i = (xs + \frac{s}{2}, ys + \frac{s}{2})$
end for
```

Table 4.3: Sketch of the Biggest Clusters algorithm implementation.

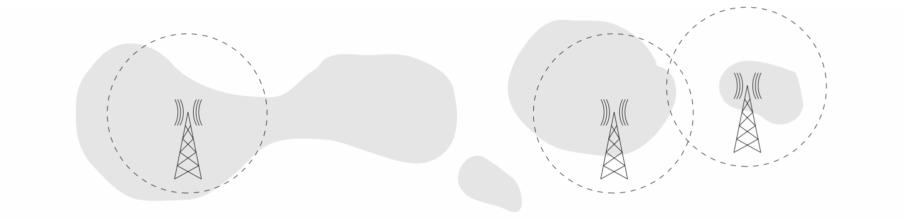


Figure 4.4: A density map with $n_c = 4$ clusters covered by $m = 3$ base stations. The fourth cluster is not covered. By accident, the third base station covers also a tiny part of the second-biggest cluster.

4.4 Lattices

As a last way of greedy base station placement, we considered two kinds of base station lattices: the rectangular and the hexagonal lattice.

Figure 4.5 shows both lattice types. Adjacent base stations are distant by d, as indicated in the figure. In cellular networks, this distance is determined by the communication radius of the base station. In hybrid networks, however, it makes sense to choose bigger distances in the hope that the uncovered nodes reach the base station via intermediate nodes (multi-hopping).

Lattices are certainly not a good way of placing base stations in hybrid networks. They do not take advantage of any properties of these networks. We consider them for comparison purposes only.

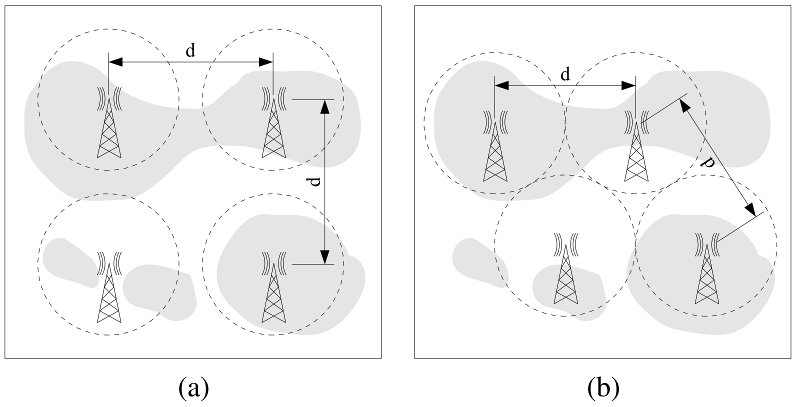


Figure 4.5: Base stations placed in a rectangular lattice (a) and a hexagonal lattice (b).

4.5 Summary of the Greedy Algorithms

All algorithms introduced in this Chapter are easy to implement but have important drawbacks.

The k-means algorithm and the lattices do not consider important properties of hybrid networks, such as the communication ranges or the appearance of clusters due to percolation.

The Highest Density algorithm and the Biggest Clusters algorithm are based on these properties. Their drawback, however, is the lack of a global view. The base stations are placed in a step-by-step fashion. Furthermore, high density cells resp. clusters are considered as isolated from each other.

Chapter 5

Cluster Covering Algorithm

In the previous chapter, we have discussed some greedy algorithms and their drawbacks. In this chapter, we propose a more sophisticated algorithm.

5.1 Introduction

As stated in Chapter 3, we want to maximize the connectivity with a minimum number of base stations in a density map (see Chapter 2.3) which models the non-uniform distribution of the nodes. We have seen in Chapter 2.1 that clusters will appear in areas with high node densities. In the design of our algorithm, we take advantage of this property.

Recall the Biggest Clusters algorithm from Chapter 4.3. We have seen that it has no global view of the whole density map. Let us explain this with the example shown in Figure 5.1. Two clusters of approximately the same size (number of nodes) shall be covered.

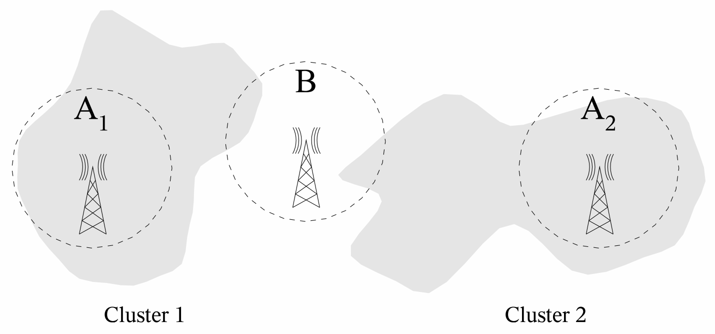


Figure 5.1: Example: Two clusters shall be covered. A greedy solution consists in putting one base station in each cluster (A_1 and A_2). However, a single base station (B) is enough to cover both clusters.

The Biggest Clusters algorithm would put one base station in each cluster (A_1 and A_2). This solution clearly connects all nodes. However, a base station in between the two clusters

(*B*) would be enough to reach the same level of connectivity. This is a solution that the Biggest Clusters algorithm does not consider. In fact, it places the base stations where the nodes *are* and not where they *can be reached*. This is the first major difference between the Cluster Covering algorithm and the Biggest Clusters algorithm.

The second major difference is that the Cluster Covering algorithm doesn't set the base stations in a step-by-step fashion. Instead, it first checks out potential locations for base stations and then chooses a subset of these locations to put base stations. In the example (Figure 5.1), three base station locations were proposed and two subsets of them were identified as solutions to cover both clusters.

By looking at connectivity only, the second solution is clearly better. If the capacity is considered as well, it is not obvious anymore which situation to prefer. Indeed, the few nodes in the communication range of the base station *B* might be a bottleneck. Other issues that we neglect in our models (see Chapter 2) might influence the choice as well, e.g.,

- The physical ability to put base stations. *B* for example might be on a lake, at the top of a mountain or behind a hill.
- Radio propagation and interference issues.
- The required capacity and the service degradation due to capacity bottlenecks.
- The desired quality of service.

In the rest of this chapter, such issues are not taken into account. The discussion focuses on connectivity only.

5.2 Covering Clusters

The algorithm we propose tries to cover the clusters only. Sub-critical regions, i.e. regions in which the nodes do not form big clusters by percolation, are neglected. There are two reasons for doing so:

- The scientific work of the last years has shown that adding multi-hopping to a network requires quite a big amount of additional complexity and protocol overhead. Therefore, deploying multi-hop networks only makes sense if this additional feature can be used by a large fraction of the nodes. As a consequence, nodes in future multi-hop networks are likely to have a power range large enough for clusters to appear. From this point of view, the most important thing is to cover these clusters properly.
- In sub-critical areas, the advantage multi-hopping diminishes. Thus, there is no difference to a single-hop wireless network and the same algorithms as for cellular networks can be used if these areas need to be served.

5.3 Main Steps of the Algorithm

The Cluster Covering algorithm performs in the three steps outlined in Figure 5.2. Starting from a density map, we first need to find out where the nodes percolate and clusters appear. From the clusters, we then search for potential base station locations. The last step consists in choosing an optimal subset of these potential base stations.

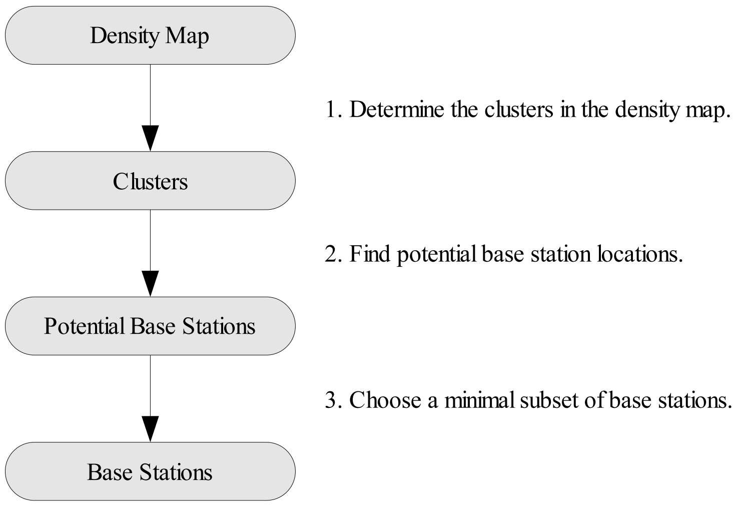

Density Map
1. Determine the clusters in the density map.
Clusters
2. Find potential base station locations.
Potential Base Stations
3. Choose a minimal subset of base stations.
Base Stations

Figure 5.2: The main steps of the Cluster Covering algorithm.

In the rest of this chapter, we discuss each of these steps in more details and give some hints for the implementation.

5.4 Clustering a Density Map

The first step of the Cluster Covering algorithm aims to determine the clusters. Recall from Chapter 2.1 that a cluster in an ad-hoc network is a set of nodes that are connected to each other, either directly or indirectly (through other nodes). If we know the position of the nodes and the connectivity conditions (e.g., the maximum connection distance), we can easily find out clusters with graph algorithms such as deep search or broad search. If the connectivity function is deterministic (two nodes are either connected or not connected), then the clusters are disjoint sets of nodes. In other words, the set of nodes is partitioned into clusters.

In a density map, however, things are slightly more complicated. A density map is just a probabilistic description of the node distribution, but it does not say anything about the exact position of the nodes. We must therefore play with the cells of the density map, i.e. we want to find sets of cells (cell clusters) in which nodes are likely to be connected.

In contrast to node clusters, the cell clusters need not to be disjoint. A cell may be part of different clusters without connecting them. Let us visualize this with the two density maps shown in Figure 5.3. Cells with a high node density are marked gray. White cells indicate a low node density. In Figure 5.3 (a), our eye immediately detects the two clusters. Indeed, the small cluster in the lower left corner to too far away to be connected to the bigger cluster on the right side.

In Figure 5.3 (b), it is not obvious which cells belong to the same cluster. We observe four groups with four density map cells each. Inside the groups, the nodes are very likely to be connected. Neighboring groups are connected with some probability, say $\gamma = 0.95$. Then the leftmost group and the rightmost group are connected with probability $\gamma^3 = 0.95^3 = 0.8574$ only.

Assume we use a probability threshold method with a threshold of $\gamma_{th} = 0.9$, i.e., all groups connected with a probability $\gamma > \gamma_{th}$ belong to the same cluster. From the point of view of the leftmost group, there are two clusters: one cluster including the first three groups (counted from left to right) and another cluster including the rightmost group of cells. From the point of view of one of the groups in the middle, however, all four groups belong to the same cluster (since $\gamma^2 = 0.95^2 = 0.9025$).

Obviously, the transitivity property does not hold. Therefore, the probability threshold based method leads to non-disjoint clusters.

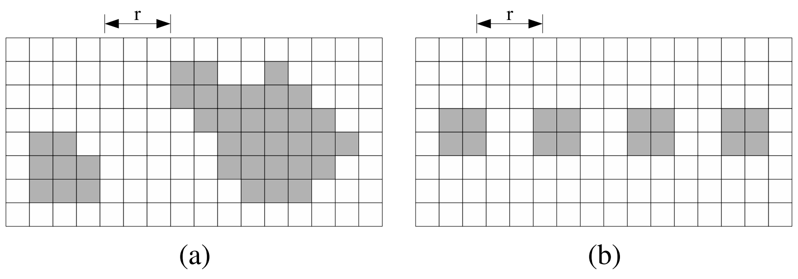


Figure 5.3: (a) An example where disjoint cell clusters can be determined. (b) An example where the cell clusters are not disjoint.

Fortunately, real density maps usually resemble more the case in Figure 5.3 (a). At the city borders, the population density drops quite quickly and stays at a low level until the next city or town is reached. Since the intercity distances are much bigger than the node connection range, an ad-hoc connection between the cities is not possible at all.

In the following chapters, we show two algorithms which can be used to determine the clusters. Both algorithms partition the density map into disjoint clusters and area with subcritical density where no (or only small) clusters appear.

5.4.1 Density Threshold Method

The first algorithm we propose (and implemented in the simulation program) is based on percolation theory [7]. As mentioned in Chapter 2.1, an unbounded cluster appears above a density threshold λ_c (which depends on the node communication range).

To determine the cluster, we look at each pair of cells and check whether they are connected or not. We consider two density map cells, γ_1 and γ_2, to be connected if the distance between them is smaller than r and if

$$\lambda_{\gamma_1} \geq \lambda_t \quad \text{and} \quad \lambda_{\gamma_2} \geq \lambda_t$$

Since the existence of a cluster doesn't imply that all nodes are part of it, we add a safety margin m_s to the percolation threshold, i.e.

$$\lambda_t = \lambda_c + m_s$$

This is similar to the fill-in tool known from raster graphics editing software, where adjacent points with the same or a similar color are masked and then repainted with a new color. Here, neighboring cells with a high enough connection probability are put together in the same cluster. Different algorithms for this problem exist. In this section, we present one which is suitable to determine all clusters (not just a single cluster) in a density map.

The algorithm proceeds in two steps:

1. Initialize all cells with their own individual cluster. Hence, there are as many clusters as cells.

2. Merge the clusters that are connected (according to the above rules).

If n is the number of cells in the density map, there are n^2 cell pairs. However, all pairs of cells distant by more than r (node communication distance) do not need to be considered. Their direct connection probability is always zero. Assume the current cell has index (x, y). Only the following surrounding cells (x_n, y_n) need to be checked:

$$x - \left\lceil \frac{r}{s} \right\rceil \leq x_n \leq x + \left\lceil \frac{r}{s} \right\rceil \text{ and } y - \left\lceil \frac{r}{s} \right\rceil \leq y_n \leq y + \left\lceil \frac{r}{s} \right\rceil$$

where s denotes the width and height of a cell (density map resolution). Furthermore, the connectivity between two cells is symmetric. We can take that into account by restricting the checks to cells with a lower id only. Figure 5.4 shows these cells graphically.

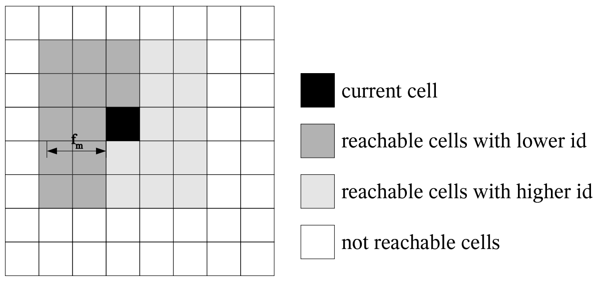


Figure 5.4: Reachable cells and cells with a lower id than the current cell.

Important for the performance is the way to represent clusters in the memory. Since we assume clusters to be disjoint sets of cells, we need a good disjoint set data structure [21].

We used a structure where each cluster is represented by a tree. This can be implemented with a simple index table. Each cell stores the index of its parent cell in the cluster. A cell at the tree root stores its own index in the table. Merging two trees only requires to set the parent of one root to a cell in the other tree.

With these optimizations, the algorithm becomes:

1. Initialize the tree table with the cell indices: each cell is a tree root and thus a cluster by itself.

2. Iterate through each cell of the density map. For each of the surrounding cells with a lower cell id than the current cell:

    (a) If the cell distance is less than r and both cells densities are above the threshold λ_t, merge the two corresponding trees.

Each cluster is now a tree in the graph and can easily be transformed to any other cluster representation structure.

The complexity of this algorithm is $\mathcal{O}(\frac{r^2}{s^2}n^2)$ in the worst case when the tree becomes one long chain. In most practical cases however, the tree is closer to a n-ary tree. Thus the algorithm rarely takes more than $\mathcal{O}(\frac{r^2}{s^2}n\log n)$ time and $\mathcal{O}(r^2+n)$ memory.

This method of determining the clusters is simple but not very accurate at the cluster borders. Especially low density cells nearby high density cells are not added to the cluster, although the nodes in there will almost certainly be connected to the nodes in the high density cells.

5.4.2 Iterative Cluster Search

A more exact way to determine the clusters is possible when the direct connection probabilities between the cells are known.

Assume we know the connection probability, $p_c(\gamma_1, \gamma_2)$, between each pair of cells, γ_1 and γ_2, in the density map. We can represent this information as a graph with the cells as nodes and their connections as edges. Each edge has a probability value associated. An illustrative example is shown in Figure 5.5.

Let us now start with an arbitrary clustering of the density map, e.g., the clustering obtained using the percolation based approach. We can apply the following algorithm to improve the clustering:

1. Start with an arbitrary clustering.

2. For each cell γ_i and each cluster C_j,

    - Calculate the probability that the cell γ_i is connected to the cluster C_j.
    - If the probability exceeds a threshold, add the cell to the cluster (and possibly merge clusters if the cell already belongs to another cluster).
    - If the probability is below a threshold, remove the cell from the cluster (and check if this divides the cluster).

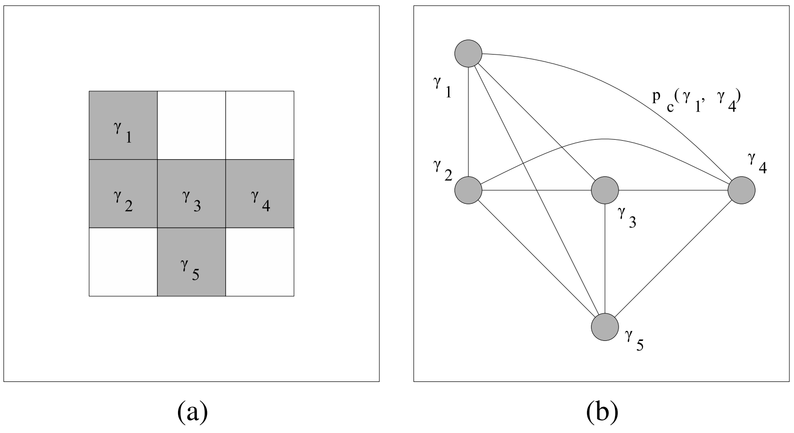


Figure 5.5: A small density map (a) and its corresponding connectivity graph (b). Note that cells with zero density (white) don't need to be represented in the graph.

3. Repeat step 2 until a stable solution is reached.

The probability that a cell γ_i is connected to the cluster C_j can be expressed as

$$p(\gamma_i) = 1 - \prod_{\gamma_k \in C_j} (1 - p_c(\gamma_i, \gamma_k))$$

where we used the notation C_j to denote the set of cells belonging to the cluster. This probability depends on the clustering, which may change at each iteration. Therefore, we need to run the algorithm until the solution is reasonable stable.

Note that this algorithm still creates disjoint cluster sets with the side effects discussed at the beginning of Chapter 5.4.

A problem that we have not discussed so far is how to find out the probability that two cells are connected. In the two following paragraphs, we propose two ways to deal with this problem. We first derive an analytical approximation. The second method uses a Monte Carlo simulation to measure the connection probability statistically.

Cell Connection Probability (analytical)

The probability that the nodes of two distinct cells, γ_1 and γ_2, are connected can be written as

$$p_c(\gamma_1, \gamma_2) = \sum_{a=0}^{\infty} \sum_{b=0}^{\infty} \mathrm{P}(a_r = a)\mathrm{P}(b_r = b) p_d(\gamma_1, \gamma_2, a, b)$$

where the last expression

$$p_d(\gamma_1, \gamma_2, a, b) = \mathrm{P}(\min_{\substack{i=1,\ldots,a \\ j=1,\ldots,b}} \left|\mathbf{x}_i - \mathbf{y}_j\right| < r)$$

denotes the probability that there is at least one pair of cells between the two clusters close enough to be connected. The $\mathbf{x}_i$ resp. $\mathbf{y}_j$ stand for the position of the nodes in γ_1 resp. γ_2.

Unfortunately, this equation cannot be calculated explicitely. We therefore derive an approximation for it in the following paragraphs.

One node in each cell Let us first have a look at two cells γ_1 and γ_2 with exactly one node each, randomly set somewhere inside the cell (using a 2-dim uniform distribution). This situation is shown in Figure 5.6.

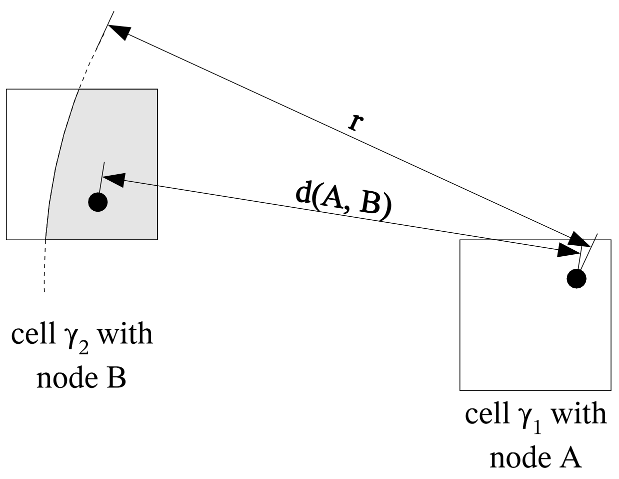


Figure 5.6: Calculating the connection probability between two cells with one node each.

Assume we know the position of node A in γ_1. Since B is uniformly distributed in γ_2, the connection probability between A and B is

$$p_A(A,\gamma_2) = \frac{1}{s^2}\iint_{\gamma_2} f(d(A,B))dB$$

where $f(x)$ denotes the connection probability between two nodes in function of their distance. With the model we consider,

$$f(x) = \begin{cases} 1 & \text{if } x \le r \\ 0 & \text{if } x > r \end{cases} \tag{5.1}$$

p_A describes the intersection area of a square and a circle (the gray area in Figure 5.6).

Therefore, the connection probability between the two cells with one node each is given by

$$p(\gamma_1,\gamma_2) = \frac{1}{s^2}\iint_{\gamma_1} p_A(A,\gamma_2)dA \tag{5.2}$$

$$= \frac{1}{s^4}\iint_{\gamma_1}\iint_{\gamma_2} f(d(A,B))dBdA \tag{5.3}$$

This equation only depends on the distance between the two nodes A and B and thus only on the relative positioning of the two cells γ_1 and γ_2. Therefore, we sometimes abbreviate the notation:

$$p(\gamma_{x,y}, \gamma_{x+x_d,y+y_d}) = p(\gamma_{0,0}, \gamma_{x_d,y_d}) \hat{=} p(x_d, y_d) \tag{5.4}$$

Moreover, all symmetric constellations of the two cells yield the same result:

$$p(x,y) = p(-x,y) = p(x,-y) = p(-x,-y) \tag{5.5}$$

$$p(x,y) = p(y,x) \tag{5.6}$$

To obtain an analytically exact solution of Equation (5.3), we have to integrate $f(d(A,B))$ four times. It turns out that the problem becomes very hard to solve. However, there are different methods to calculate an estimate p_e for p numerically.

- One method consists in taking n^2 points (usually $3 \leq n \leq 5$) from a regular lattice inside each cell. The connection probability between each pair of points (n^4) is calculated and averaged. For smooth f, p_e is unbiased but does not necessarily approach p steadily when n is increased. As the method is deterministic and symmetric, it always gives the same result for the same and all symmetric constellations.

- Another method is based on a Monte Carlo simulation [21]. Two points A and B are selected randomly in γ_1 and γ_2 respectively and the connection probability $f(d(A,B))$ is calculated. The experiment is repeated m times and the average is assigned to p_e. This estimate is unbiased and for high m normally distributed around the true value p. But the method is not deterministic, which might be a disadvantage in certain applications. Furthermore, it requires $2m$ random numbers.

We have chosen the first method with $n = 4$ for all our simulations. This gives a reasonable precision in almost all practical cases.

A deterministic number of nodes in each cell From the connection probability $p(\gamma_1, \gamma_2)$ between two cells with one node each, we can now derive an approximation for the probability of connection p_d between two cells γ_1 and γ_2 with a resp. b nodes. This situation is drawn in Figure 5.7.

It is easy to see that each node A_i ($i \in \{1,2,3,...,a\}$) of γ_1 may get a connection to each node B_j ($j \in \{1,2,3,...,b\}$) of γ_2. Although the nodes are uniformly and independently distributed inside the cell, the connections that they form are not independent. The connections (A_1, B_1) and (A_1, B_2), for example, are dependent since both of them include the same node A_1.

To our knowledge, there is no simple way of coping with this dependence. We therefore approximate our calculation by assuming that the connections are independent. The connection probability between two cells with a resp. b nodes is approximately

$$p_d(\gamma_1, \gamma_2, a, b) \approx 1 - \prod_{i=1}^{a}\prod_{j=1}^{b}(1 - p(\gamma_1, \gamma_2)) \tag{5.7}$$

$$= 1 - (1 - p(\gamma_1, \gamma_2))^{ab} \tag{5.8}$$

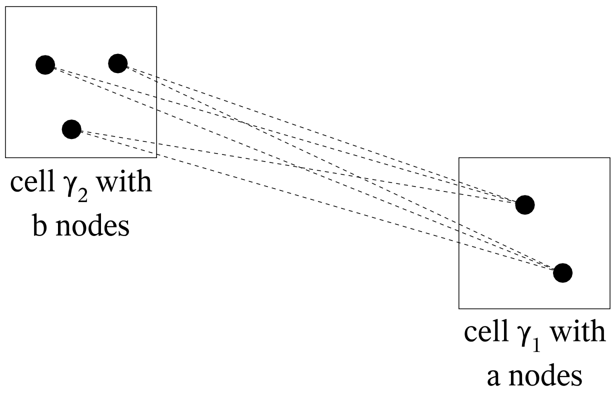


Figure 5.7: Calculation of the approximate connection probability between two cells with a deterministic number of nodes. Each node in γ_1 may get a connection to to each node of γ_2.

where we used the probability of no connection $(1 - p(\gamma_1, \gamma_2))$.

Two special cases are worth being discussed. If one of the cells is empty ($a = 0$ or $b = 0$), no connection is possible. Indeed,

$$p_d(\gamma_1, \gamma_2, 0, b) = 1 - (1 - p(\gamma_1, \gamma_2))^0 = 0$$

If both cells contain one node each ($a = b = 1$), we obtain the result from step 1:

$$p_d(\gamma_1, \gamma_2, 1, 1) = 1 - (1 - p(\gamma_1, \gamma_2))^1 = p(\gamma_1, \gamma_2)$$

In both these cases, the approximation becomes an equality. In the first case, there are no connections which could be dependent from each other. The second case consists of a single connection only.

Interestingly, we can calculate this approximation for $p_d(\gamma_1, \gamma_2, a, b)$ for all a and b from one single starting value $p(\gamma_1, \gamma_2)$ (or $p_e(\gamma_1, \gamma_2)$). The algorithmic complexity to derive a good estimate for $p(\gamma_1, \gamma_2)$ is thus less important. Once calculated, this value can be stored and reused many times.

A probabilistic number of nodes in each cell Recall that each cell in a density map has a density value λ assigned. The number of nodes in a cell is Poisson distributed with the corresponding density value.

Let us now define the random variable a_r as the number of nodes in cell γ_1,

$$a_r \sim \text{Poisson}(\lambda_{\gamma_1})$$

and correspondingly b_r as the number of nodes in the γ_2,

$$b_r \sim \text{Poisson}(\lambda_{\gamma_2})$$

To calculate the connection probability, we have to weight and sum up all combinations of possible outcomes for a_r and b_r, or more formally

$$\begin{aligned} p_c(\gamma_1,\gamma_2,a_r,b_r) &= \sum_{a=0}^{\infty}\sum_{b=0}^{\infty} P(a_r=a)P(b_r=b)p_d(\gamma_1,\gamma_2,a,b) && (5.9)\\ &\approx \sum_{a=1}^{\infty}\sum_{b=1}^{\infty} \frac{\lambda_{\gamma_1}^{a}\exp^{-\lambda_{\gamma_1}}}{a!}\frac{\lambda_{\gamma_2}^{b}\exp^{-\lambda_{\gamma_2}}}{b!}(1-(1-p(\gamma_1,\gamma_2))^{ab}) && (5.10)\\ &= 1-\exp^{-\lambda_{\gamma_1}-\lambda_{\gamma_2}}\sum_{a=1}^{\infty}\sum_{b=1}^{\infty}\frac{\lambda_{\gamma_1}^{a}}{a!}\frac{\lambda_{\gamma_2}^{b}}{b!}(1-p(\gamma_1,\gamma_2))^{ab} && (5.11)\end{aligned}$$

This equation can be calculated numerically.

This analytical result is of limited utility only. The independence assumption in Equation (5.8) yields a rather bad approximation for our connection function $f(x)$ defined in Equation (5.1). It is neither an upper nor a lower bound. Furthermore, the formula with factorials, exponentiations and summations is quite calculation intensive.

Cell Connection Probability (numerical)

As a second method for measuring the connection probabilities between two cells γ_1 and γ_2, we propose a Monte Carlo simulation [21]. We repeat the following experiment:

1. Choose a random number of nodes in both cells, n_1 and n_2, according to a Poisson distribution with the corresponding cell density value. Formally,

$$n_1 \sim \text{Poisson}(\lambda_{\gamma_1})$$
$$n_2 \sim \text{Poisson}(\lambda_{\gamma_2})$$

2. Randomly put n_1 nodes in γ_1 and n_2 nodes in γ_2, both uniformly distributed inside the cell.

3. Check if there is at least one connection between a node from γ_1 and a node from γ_2.

After having repeated this experiment several times, the connection probability is determined statistically as the ratio of connection between the two cells and the number of experiments. The precision (and confidence) of this method depends on the number of experiments.

5.5 Finding Potential Base Station Locations

In the previous step of the Cluster Covering algorithm, we have determined the clusters in a density map. In this second step, we want to find potential locations for base stations.

Recall from Chapter 2.2 that each base station has a communication distance b. This means that nodes inside a circular range around a base station are directly connected to the base station. As mentioned in the introduction of Chapter 5, potential base stations locations are

- inside a cluster, to cover a single cluster,
- between two or more clusters, to cover several clusters.

In the following paragraphs, we present an algorithm to find such positions.

Let us start with some definitions. We define Ω_i to be the set of density map cells belonging to cluster C_i. The distance between two clusters C_1 and C_2, $d(C_1,C_2)$, is defined as the length of the shortest segment still able to connect both clusters. The distance between a cluster C and a point P, $d(C,P) = d(P,C)$, is the length of the shortest segment able to connect the point with the cluster. Figure 5.8 may help to understand these definitions.

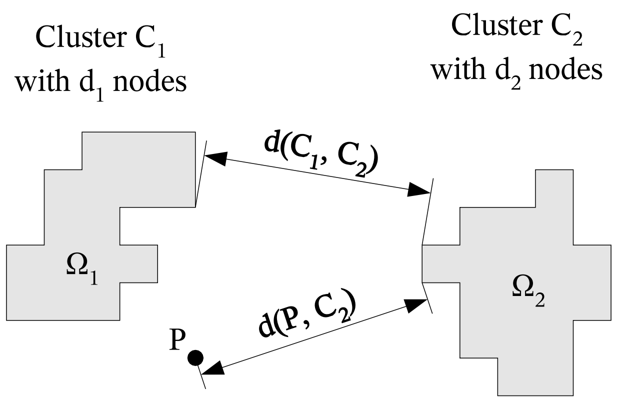

Cluster C_1 with d_1 nodes
Cluster C_2 with d_2 nodes
$d(C_1, C_2)$
Ω_1
Ω_2
P
$d(P, C_2)$

Figure 5.8: Examples of distances between two clusters and between a point and a cluster.

Assume we have found n_C clusters in the density map. Each non-empty subset of clusters, Ψ, may have a common point, denoted $B(\Psi)$, that is able to connect these clusters. Whether or not such a point exists can be expressed as follows:

$$\exists B(\Psi) \iff \Psi \neq \emptyset \text{ and } \exists P \text{ such that } \forall C \in \Psi, d(C,P) < b \tag{5.12}$$

This condition is fairly easy to understand: If a point exists from which a connection to each cluster in the subset is possible, this point is a potential place for a base station. An implementation, however, is not straightforward.

To check whether a common point for a subset of clusters exists, two different methods can be distinguished:

Scanning One can iteratively look at all points in the surrounding of the clusters in question and check whether they fulfill the condition. The problem with this approach is that there are infinitely many points to check. If not all of them are checked, a solution can possibly be overlooked. This could happen, for example, if the distance between the two clusters is exactly $2b$.

Geometric Objects Another possibility consists in dilating[1] all clusters by the base station connection distance b and finding the intersection area. This is a priori feasible.

[1]Dilation: operator of mathematical morphology

However, the dilation of clusters is not a simple geometric object and an intersection algorithm would be complex. Furthermore, this would only tell us where a connection is possible, but not where to optimally place the potential base station point.

We have chosen to combine simplified versions of both methods. For each cluster, we first calculate the smallest rectangle including the cluster dilation with b. The intersection of these rectangles (if it exists) is again a rectangle and an outer bound for the area we are looking for. An example with two clusters is shown in Figure 5.9.

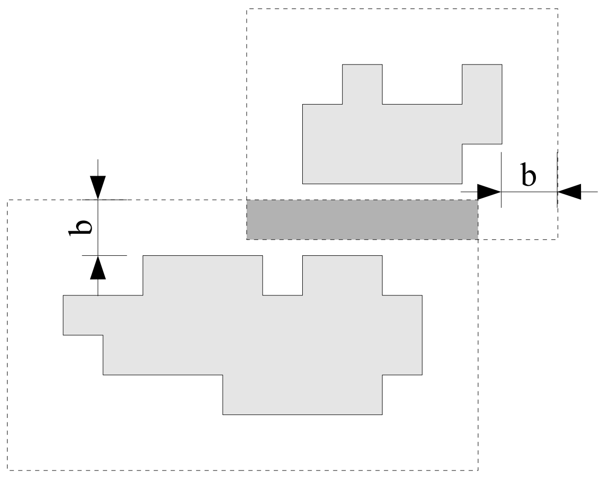


Figure 5.9: A simple outer bound (dark gray) for the intersection of clusters dilations.

If no such intersection exits, then condition (5.12) definitely doesn't hold. We can skip this cluster subset and all subsets that contain it.

If an intersection rectangle exists, this does not prove that the clusters are connectable, but it is worth checking if they do. We therefore scan through the intersection area of the two dilation rectangles (see Figure 5.9). As mentioned before, we cannot check all possible points even in this intersection area since there are infinitely many points. Instead, we only check the points of a regular grid (rectangular lattice) and choose the point with the best quality. (We will discuss later in Chapter 5.5.1 how to define the quality of a potential base station point.) As shown in Figure 5.10, we have chosen the same horizontal and vertical distance d_c between grid points, where d_c is in the order of the density map resolution. Typically,

$$d_c = \frac{1}{2}s$$

where s denotes the width and height of a density map cell.

There are mainly two reasons for doing so:

- The density map has a finite resolution. Hence, it models the effective node distribution with a certain precision only. The quality difference of two very close potential base station points is therefore not of big importance in reality.

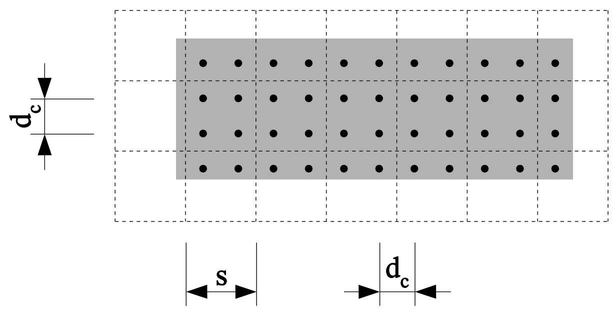


Figure 5.10: The grid points to scan. The gray field represents the intersection area which needs to be checked. The black dots are the scan points. In this example, the distance between two scan points is $d_c = \frac{1}{2}s$. The fine dotted grid shows the density map cells.

- In reality, base stations cannot be placed exactly at the mathematically best positions. Therefore, a rough approximation of the base station points is enough.

Note that we cannot use a gradient search method since this would find suboptimal potential base station points.

The procedure so far applies to base stations that cover more than one cluster. This is not always possible. A clusters may be far away from all other clusters, so that a base station must be dedicated for this single cluster. In addition to the potential base stations between the clusters, the algorithm therefore proposes a potential base station point inside each cluster. To find this point, the whole cluster area is scanned in the same way as described before.

5.5.1 Rating of Potential Base Station Locations

When we scan the intersection of the dilation rectangles (see Figure 5.9), we most probably find several points in which a base station would cover all involved clusters. From the point of view of pure connectivity, all these points would be suitable.

The real quality of the points, however, is certainly not the same. Many of the real-world issues that we neglect in our model (see Chapter 2) could intervene here. For example, topographical map data could give us hints on the feasibility to build a base station. Knowledge about radio propagation could further help determining good positions. The overall quality of a location is finally a weighted sum of several such issues.

In the scope of this thesis, we have chosen the throughput capacity to rate the points. We consider two quality criteria:

1. A point is good if it is expected to cover many nodes in each cluster. Therefore, the expected number of covered nodes is one measure of quality.

2. A point is good if it covers approximately the same fraction of nodes in all clusters (that it is supposed to cover).

The first quality criteria is pretty straightforward. The second rule probably requires further explanation. Assume we want to cover two nearby clusters with one base station and think about the case in which one cluster has a high node density and the other cluster has a rather low node density (just enough for percolation to occur). This situation is drawn in Figure 5.11. According to the first rule, the optimal point for a base station would be A. Indeed, it covers both clusters and maximizes the expected number of covered nodes. However, this point is not a satisfactory solution since the low density cluster is only badly covered. We therefore need the second criterion to prevent such situations.

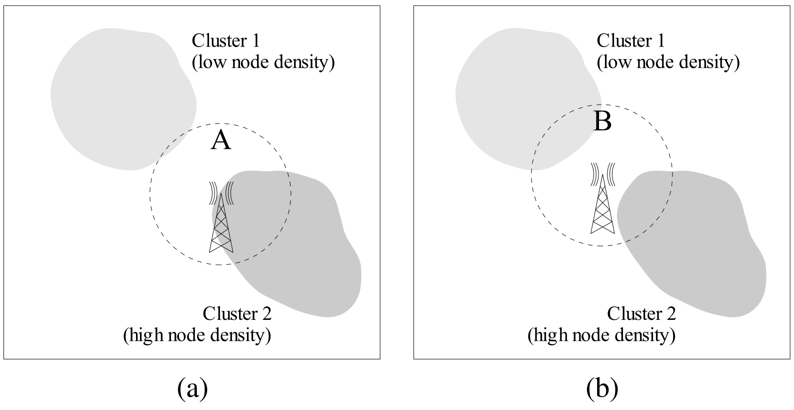


Figure 5.11: Selecting a base station location between two clusters.

In our simulations, we rate the points with the formula

$$q = \frac{f_1}{n_1} \cdot \frac{f_2}{n_2} \cdots \frac{f_{|\Psi|}}{n_{|\Psi|}}$$

where f_i represents the expected number of covered points and n_i the expected total number of points in the corresponding cluster. This formula takes both criteria into account.

5.5.2 Complexity

The complexity of the potential base station search algorithm mostly depends on the clusters and on the base station radius.

Recall that we find potential base stations between subsets of clusters. The number of cluster subsets grows exponentially with the number of clusters n_C. Without the empty set, there are $2^{n_C} - 1$ different subsets to check.

However, if a subset Ψ_i doesn't fulfill Equation (5.12), all subsets that contain Ψ_i certainly don't fulfill this equation neither. Fortunately, this rule very often applies in real situations and cuts away many possibilities.

Let Υ be the set of all cluster subsets Ψ_i which can be covered with a single base station. For each of these subsets, we get one potential base station. The number of potential base stations, n_B, is therefore

$$n_B = |\Upsilon| \leq 2^{n_C} - 1$$

If $v = \max_{\Psi_i \in \Upsilon} |\Psi_i|$ is the number of clusters in the biggest subset, then we can derive the upper bound

$$n_B \leq \binom{n_C}{1} + \binom{n_C}{2} + \binom{n_C}{3} + \ldots + \binom{n_C}{v}$$

This last equation describes the number of different cluster subsets of size smaller or equal than v. This must be bigger than n_B, otherwise Υ would contain at least one subset with cardinality greater than v.

Note that the cardinality of the subsets (notably v) depends on the base station radius, b, and on the cluster size and distance. A bigger base station radius leads to larger subsets and therefore to more potential base stations.

We can now calculate a complexity bound for the potential base station search algorithm. For each solution subset, we have to scan grid points of an intersection rectangle. The size of the intersection rectangle is bounded by the dilation rectangle of the biggest cluster. Let us denote the width and height of this rectangle by r_w and r_h respectively. The scan resolution (horizontal and vertical distance between grid points) was previously defined as d_c.

For each of these grid points, we have to consider all density map cells within the coverage area of a base station which is proportional to $\frac{b^2}{s^2}$. Recall that s denotes the width and height of a density map cell and b the base stations radius.

The complexity of the algorithm can therefore be written as

$$\mathcal{O}\left(\frac{r_w r_h b^2}{d_c^2 s^2} n_B\right)$$

The base station radius is an important factor in this equation since it is also a dominant factor in n_B. In our simulations, we experienced a big increase in calculation time when increasing it.

5.5.3 Avoidance of Useless Potential Base Stations

The presented algorithm finds out two types of potential base stations which are useless for further calculations.

Low Quality Potential Base Stations Some of the potential base stations cover only tiny parts of the clusters determined from the density map. Such base stations are useless because they do not provide the necessary throughput. Furthermore, there is a high risk that none of the nodes in that cluster is covered by the base station.

We therefore introduced a cluster coverage threshold, $0 \leq f_{th} < 1$. If a potential base station cannot cover more than a fraction f_{th} of the nodes in the cluster (i.e. $\frac{f_i}{n_i} < f_{th}$) then we skip the potential base station.

In our simulations, we used $0.1 \leq f_{th} \leq 0.5$.

Inferior Potential Base Stations Some potential base stations are always worse than others. For instance, a potential base station fully covering a single cluster is always worse than a nearby base station fully covering the same cluster but some other clusters as well.

In our simulations, we therefore skip a potential base stations if

- another potential base station covers at least the same clusters
- the coverage ratio $\frac{f_i}{n_i}$ of each cluster is the same or worse

Both these optimizations make it possible to get rid of many useless potential base stations. This saves much computation time and, we believe, improves the final result by facilitating the work in the next step.

5.6 Choosing the Base Stations

In the two previous steps of the Cluster Covering algorithm, the clusters were determined and potential base station points were found. The last step consists in choosing an optimal subset of these potential base stations.

This task can be modeled as a set covering problem (SCP) [23]. Let us call C_i ($0 \leq i < n_C$) the clusters and B_j ($0 \leq j < n_B$) the potential base stations. Recall that each potential base station was placed in a way to cover a subset of clusters. This relation between the potential base stations and the clusters can be written as a binary matrix $\mathbf{L}$ with n_C rows and n_B columns, where

$$\mathbf{L}_{i,j} = \begin{cases} 1 & \text{if } B_j \text{ covers } C_i \\ 0 & \text{otherwise} \end{cases}$$

Furthermore, let v_b be a binary vector of the selected base stations. This vector has length n_B and contains a 1 for all selected base stations and a 0 for all other base stations. The cluster coverage vector, v_c, can be expressed as

$$v_c = \mathbf{L} v_b$$

The vector v_c has length n_C. Its values correspond to the number of times a cluster is covered.

Assume we want to cover all clusters with as few base stations as possible. This is the standard set covering problem, which can be written as

$$\begin{array}{lll} \text{Minimize} & \sum_{j=0}^{n_B-1} (v_b)_j & \\ \text{Subject to} & \sum_{j=0}^{n_B-1} \mathbf{L}_{i,j}(v_b)_j = (v_c)_i > 0 & \forall 0 \leq i < n_C \end{array}$$

For the cluster covering algorithm, we do not need to cover all clusters. Instead, we want to cover as many nodes as possible with a given number of base stations. If w_i denotes the expected number of nodes in cluster i, the corresponding problem can be formulated as follows:

$$\begin{aligned} &\text{Maximize} && \sum_{i=0}^{n_C-1} w_i \cdot \operatorname{sign}((v_c)_i) && (5.13)\\ &\text{Subject to} && v_c = \mathbf{L} v_b && (5.14)\\ & && \sum_{j=0}^{n_B-1} (v_b)_j = m \qquad (v_b)_j \in \{0,1\} && (5.15) \end{aligned}$$

where m stands for the number of base stations. Note that $\operatorname{sign}((v_c)_i)$ is 1 if $(v_c)_i > 0$ and 0 if $(v_c)_i = 0$.

The standard set covering problem (SCP) has been proven to be NP complete [23]. By limiting the number of base stations, however, the problem is transformed into polynomial complexity in the number of potential base stations, n_B. Indeed, the number of possibilities to choose m base stations among a set of n_B potential base stations is

$$n_P = \binom{n_B}{m} = \frac{n_B!}{m!(n_B - m)!} \le \frac{n_B^m}{m!} \le n_B^m$$

for strictly positive n_B and m.

A simple (optimal) algorithm would check all these possibilities and return the best one. The complexity of such an algorithm is

$$\mathcal{O}\left(\binom{n_B}{m}\right) < \mathcal{O}\left(n_B^m\right)$$

which is exponential in m, but polynomial in n_B.

Despite this fact, checking all possibilities is infeasible. Consider a (rather small) problem with $n_B = 100$ potential base stations among which one would like to choose $m = 10$ base stations. The set of possibilities

$$n_P = \binom{100}{10} \approx 1.73 \cdot 10^{13}$$

is so large that it would take years to check all of them.

There are many ways to solve this problem differently. Commercial software is available to solve (or approximate) the discrete maximization problem given in Equation (5.13). There also exist heuristic algorithms for the set covering problem (e.g. genetic algorithms [23]) which converge to suboptimal solutions. In our simulations, we use a greedy algorithm which we present in the following paragraphs.

The greedy algorithm first places m base stations in a step-by-step fashion. At each step, it checks all potential base stations and selects the one with the biggest connectivity improvement. The improvement of a potential base station j can be calculated as

$$\omega_j = \sum_{i=0}^{n_C-1} w_i \cdot (1 - \operatorname{sign}((v_c)_i)) \mathbf{L}_{i,j} \tag{5.16}$$

```
Variables
    bs_sel:      list of selected base stations

Functions
    biggest_connectivity_improvement(list): returns the base station with the biggest
                 connectivity improvement according to Equation (5.16) if the base sta-
                 tions in list are already selected

Algorithm
    bs_sel = {}

    # Phase 1
    repeat m times
        x = biggest_connectivity_improvement(bs_sel)
        add x to bs_sel
    end repeat

    # Phase 2
    bs_todo = bs_sel
    foreach y in bs_todo
        remove y from bs_sel
        x = biggest_connectivity_improvement(bs_sel)
        add x to bs_sel
    end repeat
```

Table 5.1: Sketch of the greedy Set Covering algorithm used in the simulation program.

where v_b denotes the vector of the already selected base stations and $v_c = \mathbf{L}v_b$.

This formula sums up the size, w_i, of all clusters which can be covered by the potential base station j and which are not yet covered by any other base station. At each step, the potential base station with the highest improvement is selected and marked in the vector v_b.

In a second phase, the algorithm goes through all selected base stations in the order they were found. Each base station is removed from the solution and the base station with the biggest improvement is added. This eliminates the most obvious non-optimalities.

This algorithm finds suboptimal solutions. For small number of base stations m, the solution is often even optimal. For larger m, the solution quality decreases. In our simulations, however, we found that it is still acceptable.

5.7 Summary of the Cluster Covering Algorithm

In this chapter, we have presented an algorithm which places base stations in a way to cover one or several clusters in the ad-hoc network.

In the first step, we *determined the clusters* in the density map. It turns out that this is the analytically hardest part. We made some assumptions to simplify the problem.

In the second step, we *located the potential base station*. These are points where a base station would cover one or more clusters. Depending on the clusters and on the base station radius, this part can consume quite a lot of calculation time.

In the third and last step, we *selected a subset of the potential base stations* using a greedy algorithm.

Note that due to the different assumptions we have made in all steps, the Cluster Covering algorithm does not provide optimal solutions.

Chapter 6

Simulation

As mentioned in previous chapters already, we implemented a simulation program with the presented algorithms.

There are a number of network simulators available on the Internet. The most famous is probably the Network Simulator ns-2 [24]. Unfortunately, this simulator works only up to roughly 100 nodes [25]. An improved version, fast-ns-2 [25], can cope with up to 3000 nodes. This is still very limiting for our purposes, since we require to simulate between 10000 and 500000 nodes. Other simulation programs which can simulate bigger sets of nodes are available as well. Most of them were however created for other purposes than simulating connectivity in large networks. We therefore decided to implement a simulation program from scratch.

In this chapter, we first give some explanations on our implementation. We then present how we measured and compared the quality of the algorithms. Finally, we discuss and compare the simulation results.

6.1 Some Notes about the Simulation Program

For reasons of speed and portability, we have chosen to implement the program in C++. To create the graphical user interface (GUI), we used the wxWindows library [26] which is freely available on the web. Programs written with this library can be compiled for Linux (and many other Unix systems), Macintosh and Windows.

We wrote the Simulator in a fully object-oriented manner. The main part is divided into four class libraries:

AdHocLib These classes provide the main simulation environment. There are classes to handle density maps, clusters, potential base station, base stations and nodes. A simulation class holds a whole simulation scenario together and provides functionality to save and load simulation files.

AdHocSim These files contain everything related to the GUI. They can draw simulation scenarios and invoke simulation functions.

BaseStationAlgorithm Classes in this namespace contain the base station algorithms. Each algorithm is encapsulated in one or more classes.

Sim This library contains specific simulation classes. These macro-like methods merely use the other libraries to perform simulations and to store the results in text files, which can be loaded and analyzed in Matlab or similar software.

Beyond that, there are a few general-purpose utility classes.

The simulation program (including a class documentation) is freely available.

6.2 Measuring the Algorithm Quality

The main purpose of the simulation program is to measure and compare the quality of the different algorithms.

Recall the connectivity function, $C(m)$, which we defined in Chapter 3 as

$$C(m) = \frac{\text{connected nodes with m base stations}}{\text{total nodes}}$$

The connected nodes are those which reach at least one base station in a multi-hop fashion. The better the algorithm, the higher is the ratio of connection nodes. We can therefore compare the different algorithms by comparing their connectivity function.

To get the connectivity function of an algorithm, we repeat the following procedure for different m:

1. Run the algorithm for m base stations.

2. Generate r sets of random nodes according to the node distribution given by the density map.

3. For each set of nodes, $0 \leq i < r$, count the number of connected nodes, c_i, and the number of total nodes, t_i.

The connectivity value for m base stations can then easily be estimated as

$$\hat{\text{E}}[C(m)] = \frac{1}{r}\sum_{i=0}^{r-1}\frac{c_i}{t_i}$$

If the algorithm is deterministic (i.e., for a given m, it always returns the same set of base stations), we can safely repeat the above procedure for many different m to get the connectivity function. We used $r = 100$ in such cases.

For non-deterministic algorithms, however, we need to repeat the whole procedure several times for the same number of base stations m. We therefore run the simulation 10 times for each value of m and kept the average connectivity of the best run only. To decrease the simulation time, we used $r = 10$.

6.3 Simulation Results

In this chapter, we discuss the simulation results for the different density maps.

6.3.1 Zurich

With its 364,000 residents, Zurich is the biggest city in Switzerland. Including the surroundings, we count even 1,100,000 inhabitants[1].

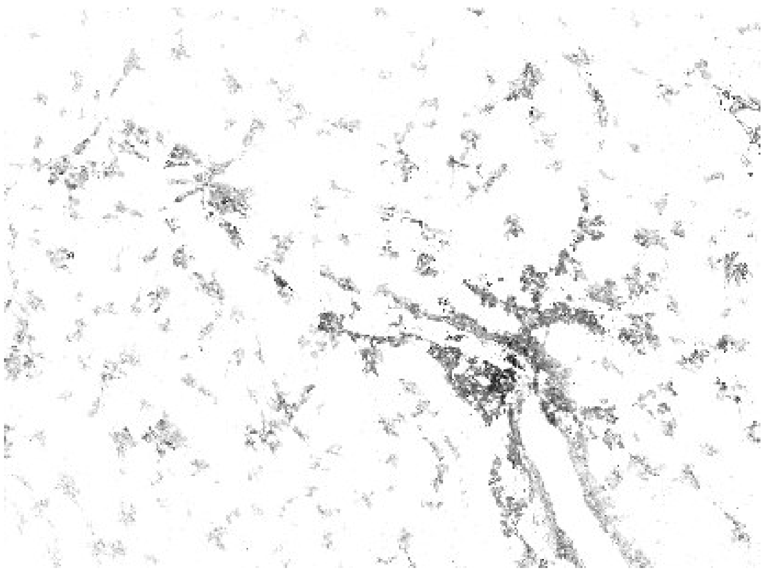


Figure 6.1: The density map of Zurich.

In the population density map of the area of Zurich[2] (Figure 6.1), there are 1041079 inhabitants. The data has a resolution of 100 m x 100 m and is therefore quite precise. The area is about 41.2 km x 30.5 km large and the population is distributed over the whole area. In the lower right part, one can see the lake of Zurich. In the north of this lake is the main city with the highest population density.

We have chosen 2% of all inhabitants to carry a device able to take part in the network. This corresponds to 20,822 devices.

Zurich with Small Base Stations

In a first experiment, we selected a node radius of $r = 500$ m and a base station radius of $b = 1000$ m.

To find the clusters, we used the density threshold method (see Chapter 5.4.1) with $\lambda_t = 5 \cdot 10^{-5}$ [nodes/m^2]. This density threshold value was derived experimentally by trying

[1]Data from 2002, *Stadt Zürich*, http://www.stzh.ch

[2]Data from 1990, *Statistik Schweiz, Bundesamt für Statistik*, [20]

different values and comparing them to the clusters formed by random nodes. We only considered clusters with an expectation of at least 10 nodes. This configuration led to 85 clusters.

To avoid low quality potential base stations, we used the optimization presented in Chapter 5.5.3 with $f_{th} = 0.1$ and found 157 potential base stations.

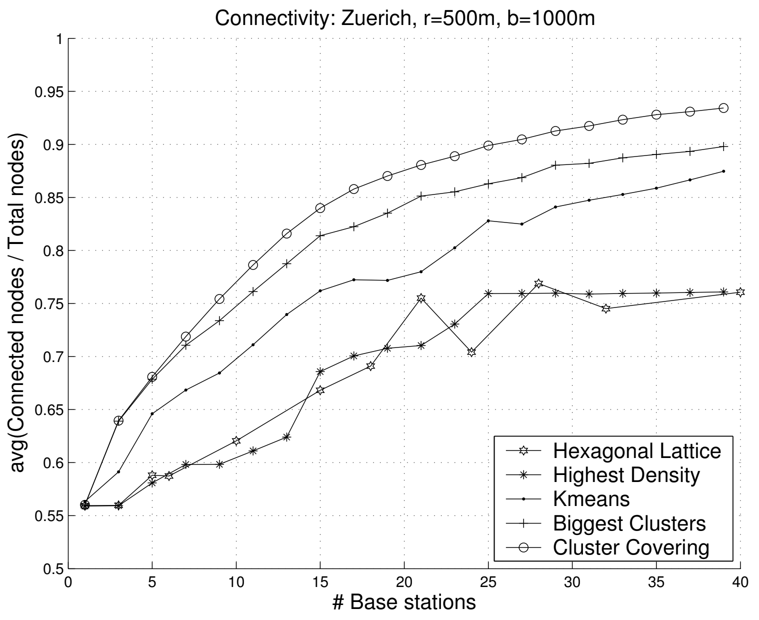


Figure 6.2: Connectivity functions for the density map of Zurich with $r = 500$ m and $b = 1000$ m.

Figure 6.2 shows the connectivity function of the different algorithms.

All algorithms start with covering 55% of the nodes with the first base station. This is the size of the biggest cluster appearing in the main city and around the lake.

The Cluster Covering algorithm clearly performs best. It reaches 90% coverage with 26 base stations only. All other algorithms need 40 or more base stations to reach the same degree of connectivity.

For few base stations, the Biggest Clusters algorithm attains the same connectivity as Cluster Covering algorithm. Both algorithms cover the clusters with the highest number of nodes first. In the interesting region above 75% connectivity, however, it reaches between 2% and 4% less connectivity.

The K-means algorithm is again a few percent worse.

The remaining two algorithms perform substantially worse. To cover the whole area with a cellular approach, many more base stations would be necessary. This is why the Hexagonal Lattice algorithm reaches only 76% coverage with 40 base stations. Note that it reaches the same connectivity for 21 base stations already. This is due to the fact that the grid is blindly placed onto the density map, without optimizing anything.

The Highest Density algorithm has the big drawback of putting many base stations in the same clusters. Therefore, its connectivity almost doesn't improve between 25 and 39 base stations.

Zurich with Big Base Stations

In a second simulation, we selected a node radius of $r = 500$ m and a base station radius of $b = 4000$ m. To avoid low quality potential base stations, we used the optimization presented in Chapter 5.5.3 with $f_{th} = 0.5$.

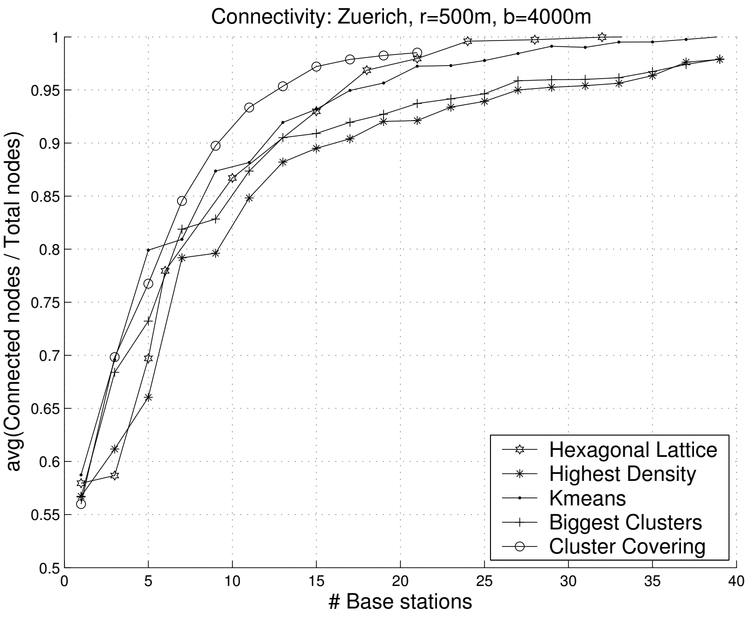


Figure 6.3: Connectivity functions for the density map of Zurich with $r = 500$ m and $b = 4000$ m.

Figure 6.3 first of all shows that it is possible to cover the whole area with a hexagonal lattice of roughly 30 base stations. Even with 23 base stations, the ratio of unconnected nodes is below 1%. None of the other algorithms reaches this with less than 35 base stations.

Between 6 and 22 base stations, the Cluster Covering algorithm again reaches the best connectivity. Below 6 base stations, the K-means algorithm connects more nodes. This underlines that the Cluster Covering algorithm is not optimal and the K-means algorithm can - due to its random initialization - beat it in some rare cases.

It is remarkable that the Cluster Covering algorithm stops at roughly 98% connectivity. The reason is that the other 2% of the nodes are not part of any cluster. Since the algorithm considers the clusters only, it does not cover these nodes.

The worst algorithm is again the Highest Density algorithm. This time, however, the K-means clearly beats the Biggest Cluster algorithm because the latter has the problem of covering the same cluster multiple times.

Zurich with Small Node Radius

In this simulation, we selected a node radius of $r = 200$ m only. The base station radius was set to $b = 1000$ m.

To find out the clusters, we used the density threshold method (see Chapter 5.4.1) with $\lambda_t = 4 \cdot 10^{-5}$ and found 182 clusters with an expected size of more than 10 nodes.

To avoid low quality potential base stations, we used the optimization presented in Chapter 5.5.3 with $f_{th} = 0.1$. We counted 521 potential base stations after optimization.

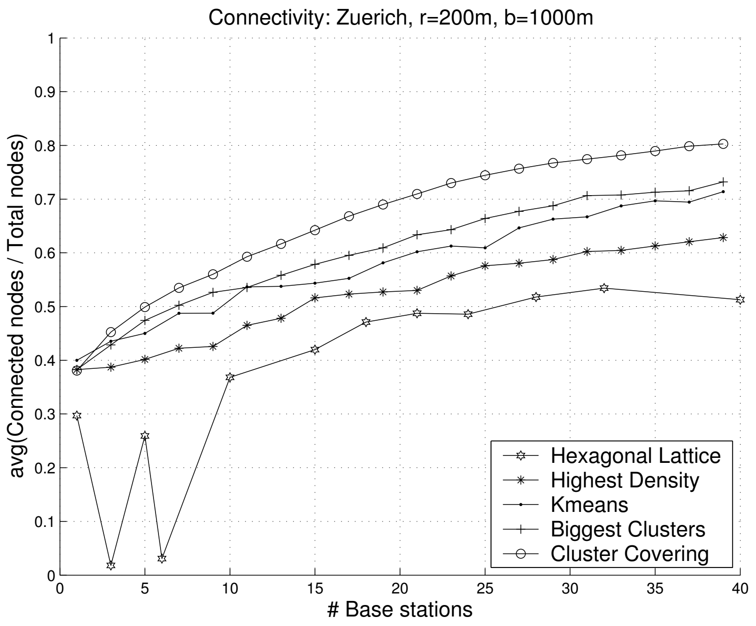


Figure 6.4: Connectivity functions for the density map of Zurich with $r = 200$ m and $b = 1000$ m.

Except for the case of a single base stations, the Cluster Covering algorithm reaches up to 8% more coverage than the Biggest Clusters algorithm and up to 10% more coverage than the K-means algorithm.

6.3.2 Surselva Valley

The Surselva Valley is a small valley in the Swiss alps with 31342 inhabitants[3]. Most of the inhabitants live in 110 villages situated in the valley itself, along the young Rhine River whose source is at the top of the valley[4].

We have chosen 10% of the inhabitants to carry a device. This corresponds to 3134 devices in the valley. The corresponding density map is shown in Figure 6.5.

[3]Data from 1990, *Statistik Schweiz, Bundesamt für Statistik*, [20]

[4]Source: *Tourist Information Brigels*, http://www.brigels.ch

Figure 6.5: The density map of the Surselva Valley.

Surselva Valley with Small Base Stations

We first simulated the case with a node radius of $r = 500$ m and a base station radius of $b = 1000$ m. The other parameters are the same as for the corresponding simulation with the density map of Zurich (Chapter 6.3.1).

The connectivity functions are shown in Figure 6.6. The Cluster Covering algorithm dominates the picture. It covers between 5% and 10% more nodes than the Highest Density algorithm, the Biggest Clusters algorithm and the K-means algorithm.

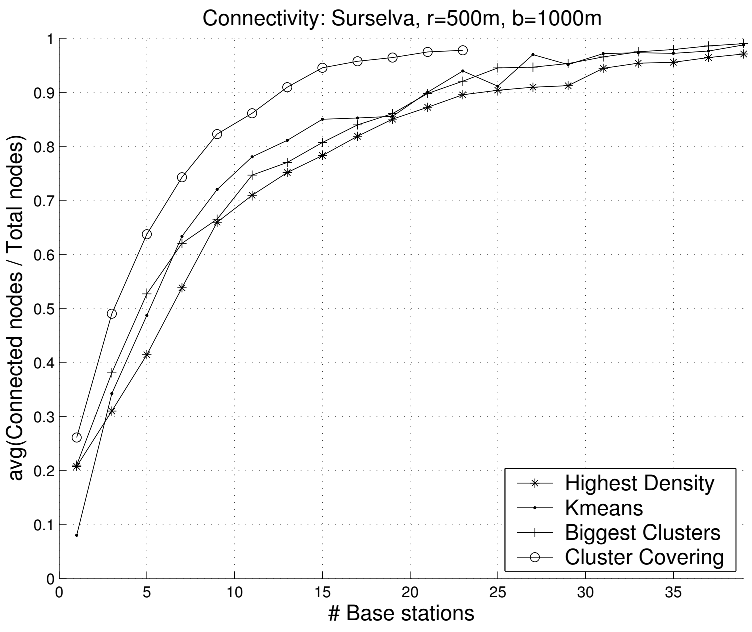


Figure 6.6: Connectivity functions for the Surselva Valley with $r = 500$ m and $b = 1000$ m.

Surselva Valley with Big Base Stations

The results for the Surselva valley with $r = 500$ m and $b = 4000$ m are shown in Figure 6.6.

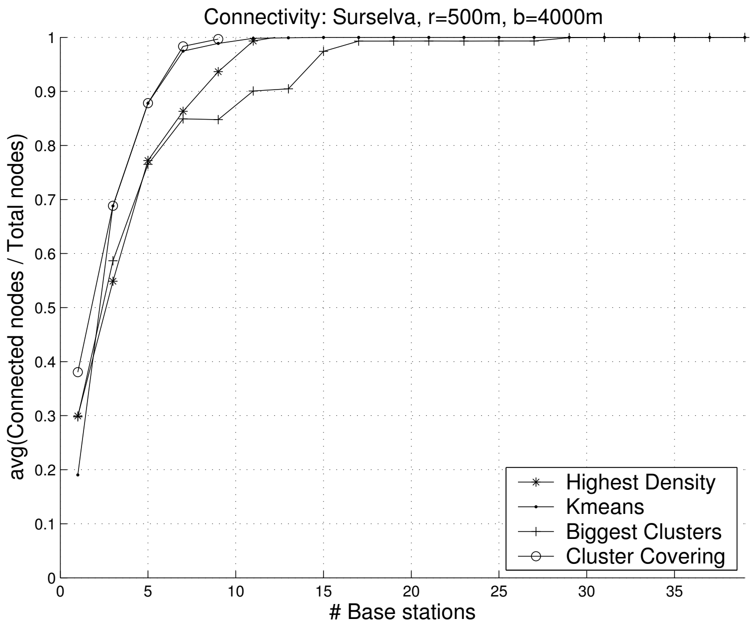


Figure 6.7: Connectivity functions for the Surselva Valley with $r = 500$ m and $b = 4000$ m.

The K-means algorithm has some troubles at the beginning but then reaches about the same connectivity than the Cluster Covering algorithm. Both algorithms cover almost the whole valley with 10 base stations.

Interestingly, the Highest Density algorithm outperforms the Biggest Clusters algorithm in this case. This comes from the fact that the Highest Density algorithm never puts a base station inside the range of another base station.

6.3.3 Finland

We finally applied the algorithms on the density map of Finland.

Finland has more than 5 million inhabitants, most of them living in the south. The biggest city is the capital Helsinki with 560,000 residents. Other important cities are Espoo (216,000), Tampere (198,000), Vantaa (179,000), and Turku (173,000). All of them are situated in the south of Finland. The biggest city in the north is Oulu with 123,000 residents[5].

Most of Finlands area is covered by forests. In between, there are small villages and towns. The distance between neighboring towns is typically between 20 km and 50 km.

[5]Source: *Virtual Finland*, http://virtual.finland.fi

Since the population density is close to zero between the towns and villages, there is no hope that percolation between them occurs for reasonable node connection ranges.

We scaled the population density by $\alpha = 0.1$ to derive a node density map. This means that we expect one tenth of the population to possess a turned on device. The node range was chosen to be $r = 500$ m and the base station range $b = 1000$ m.

To cluster the density map, we used a density threshold of $\lambda_t = 6 \cdot 10^{-6}$ and considered only clusters with an expected size of 50 or more nodes. This leads to 537 clusters.

Since the clusters are well separated from each other, only few combinations of them can be covered with a single base station. Most of them require a base station for their own to be covered. Using $f_{th} = 0.1$, we found 562 potential base stations.

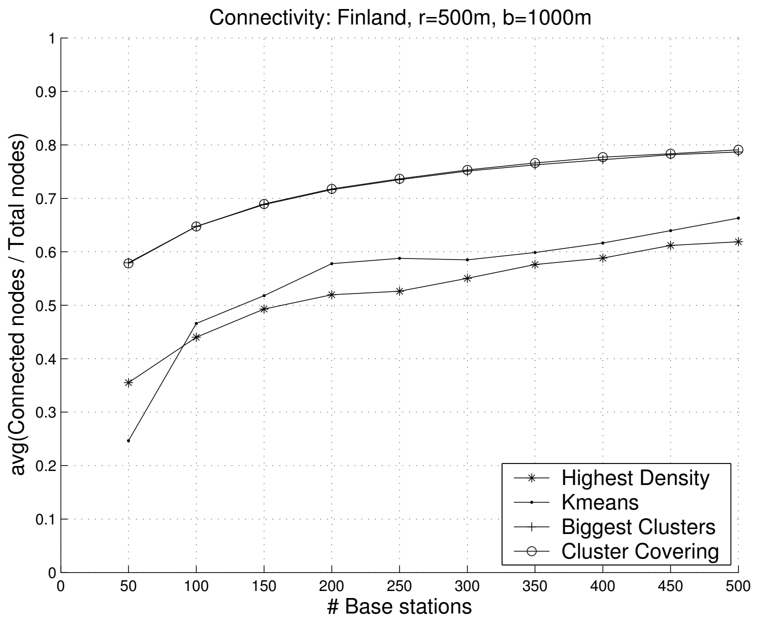


Figure 6.8: Connectivity functions for Finland with $r = 500$ m and $b = 1000$ m.

The connectivity functions for the different algorithms is shown in Figure 6.8. Note that for speed reasons, we only averaged over 10 measurements in this simulation. As expected, the Cluster Covering algorithm and the Biggest Cluster algorithm both have approximately the same performance.

The K-means algorithm is clearly worse. Many base stations are placed between towns. We called this the Stonehenge problem (see Chapter 4.1).

The Highest Density algorithm is disturbed by the big cities in the south. It wastes many base stations there.

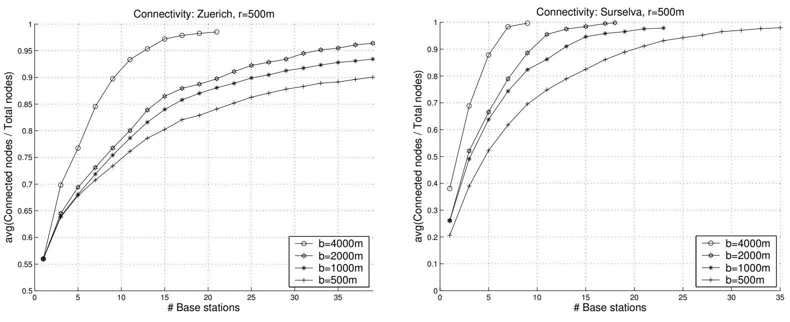
6.3.4 Comparison

By comparing the previously discussed results, we can draw three conclusions.

Influence of the Base Station Radius *b*

In the figures of Zurich and the Surselva Valley, we observe that the number of base stations required to reach a certain coverage depends very much on the base station radius. The increase in coverage is much steeper with large base stations. This is not astonishing at all, since large base stations cover more area.

In Figure 6.9, we compare the connectivity of the Cluster Covering algorithm for different base stations ranges. The big difference between $b = 2000$ m and $b = 4000$ m has to do with the cluster distances in the density map. A radius of 2 km is just not enough to cover many clusters with a single base station.



(a) Zurich (b) Surselva

Figure 6.9: Connectivity functions of the Cluster Covering algorithm for different base station ranges. The node range is $r = 500$ m.

Even though large base stations seem to be beneficial, this is probably not the best choice for multi-hop networks. First of all, the advantage multi-hopping diminishes because large base stations anyway cover most of the area. Second, they are more expensive to build (e.g., antennas) and to run (e.g., power consumption). Finally, they cause more interference for the communication between the nodes and might therefore decrease the overall network capacity.

Influence of the Node Radius *r*

A second observation can be made by looking at the results for Zurich with different node ranges, namely Figure 6.2 with $r = 500$ m and Figure 6.4 with $r = 200$.

We put the two curves of the Cluster Covering algorithm together in Figure 6.10 and added the corresponding curves for $r = 300$ m and $r = 400$ m. The difference is quite significant. To cover 80% of the nodes, we only need about 12 base stations if the node radius is $r = 500$ m. If the node radius decreases to $r = 200$ m, we need 39 base stations. A similar relation can be found for any coverage percentage between 65% and 80%.

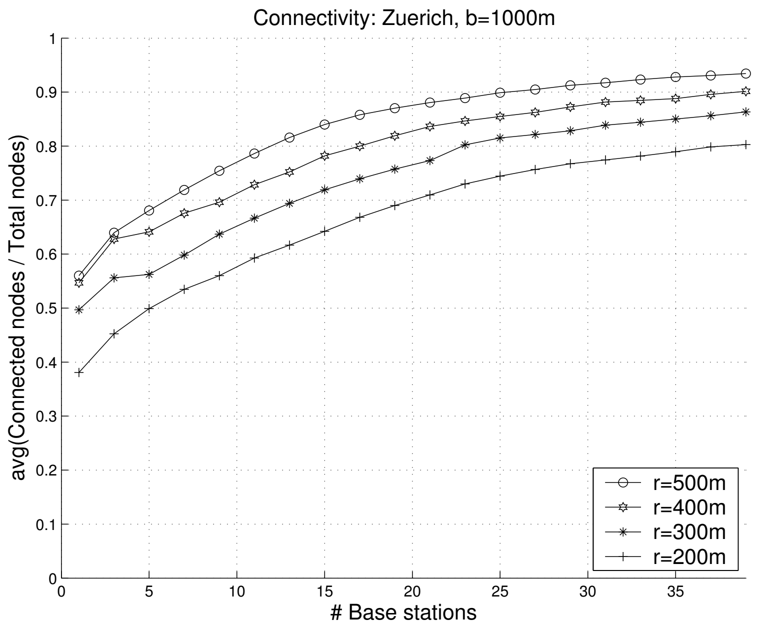


Figure 6.10: Connectivity functions of the Cluster Covering algorithm for different node ranges. The base station range is $b = 1000$ m.

Influence of the Density Map

As we have already mentioned earlier, the results depend on the density map as well. One observation was the big gap between $b = 2000$ m and $b = 4000$ m in Figure 6.9 (a) which is due to the cluster distance.

The cluster distance also has a big influence in the map of Finland. The villages and towns there are about 20 km - 50 km apart from each other and between them, the node density is very low. This leads to important connectivity differences between the algorithms which use the clusters (Cluster Covering, Biggest Clusters) and those which do not (K-means, Highest Density).

The picture is different for Zurich and the Surselva Valley. Although the difference in connectivity between the algorithms applied is quite big as well, there is no clear separation between the cluster based algorithms and the other algorithms. For big base stations, the K-means algorithm even reaches better results than the Biggest Cluster algorithm.

6.3.5 Summary

In all simulation scenarios, the Cluster Covering algorithm yields the best performance, although we have chosen a simple implementation. A more accurate cluster search in step 1 and a better implementation for the set covering problem in step 3 would again improve the results.

For small base station ranges (as compared to the cluster distance), the Biggest Clusters algorithm can be used as a cheaper (in terms of calculation time) alternative. The K-means algorithm is not a good choice in this case.

For big base station ranges, the K-means algorithm outperforms the Biggest Clusters algorithm.

The Highest Density algorithm is in any case not a good choice. It fails by putting many base stations in the big clusters. Its connectivity is significantly worse compared to the Cluster Covering algorithm.

Chapter 7

Conclusion

In this work, we have discussed where to place base stations in multi-hop hybrid networks. We have first presented three greedy algorithms. The K-means algorithm minimizes the square distance to all nodes. The Highest Density algorithm puts base stations in places with high population density. The Biggest Clusters algorithm determines the clusters and puts the base stations in those with the most nodes. All these algorithms are simple to implement, but have some drawbacks.

We then presented the Cluster Covering algorithm which performs in three steps. It determines the clusters in the density map, finds potential base station places and finally selects a subset of these places to put base stations.

Through simulations, we have shown that the Cluster Covering algorithm performs best. In some density maps, the K-means and the Biggest Clusters algorithm yield good results as well. The Highest Density algorithm performed poorly in all our simulation setups.

From the results of the Cluster Covering algorithm, we believe that multi-hop hybrid networks could decrease the number of base stations significantly as compared to cellular networks. Since we neglected many real-world issues in our simulations, the exact numbers have to be taken with a grain of salt. Despite these simplifications, the infrastructure cost for multi-hop networks will probably be much lower than the cost for the current cellular networks.

Bibliography

[1] T. S. Rappaport. *Wireless Communications: Principles and Practice*. Prentice Hall, 2nd edition, 2002.

[2] Bluetooth, http://www.bluetooth.com.

[3] WLAN 802.11 Working Group, http://grouper.ieee.org/groups/802/11/.

[4] HiperLAN2, http://www.hiperlan2.com/.

[5] J. P. Hubaux, J. Y. Le Boudec, Th. Gross, and M. Vetterli. Towards self-organizing mobile ad-hoc networks: the terminodes project. *IEEE Comm Mag*, 39(1):118 –124, January 2001.

[6] A. Appleby. Estimating the cost of a telecommunications network using the fractal structure of the human population distribution. In *IEE Proceedings: Communications*, pages 142(3):172–178, 1995.

[7] K. Christensen. *Percolation Theory*. Blackett Laboratory, Imperial College London, Prince Consort Road, SW7 2BW London, UK, October 2002.

[8] O. Dousse, P. Thiran, and M. Hasler. Connectivity in ad-hoc and hybrid networks. In *Proceedings of IEEE Infocom 2002*, pages 1079–1088, New York, June 2002.

[9] Ch. E. Perkins, editor. *Ad Hoc Networking*. Addison-Wesley, 2001.

[10] X. Hong, K. Xu, and M. Gerla. Scalable routing protocols for mobile ad hoc networks. *IEEE Network Magazine*, pages 11–21, July-August 2002.

[11] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Mobile Computing and Networking*, pages 85–97, 1998.

[12] Levente Butty&#225;n and Jean-Pierre Hubaux. Report on a working session on security in wireless ad hoc networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 7(1):74–94, 2003.

[13] S. Zhong, J. Chen, and R. Yang. Sprite: A simple, cheat-proof, credit-based system for mobile ad-hoc networks. In *Proceedings of IEEE Infocom 2003*, pages 1987–1997, 2003.

[14] N. Ben Salem, L. Butty&#225;n, J.-P. Hubaux, and M. Jakobsson. A charging and rewarding scheme for packet forwarding in multi-hop cellular networks. In *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pages 13–24. ACM Press, 2003.

[15] K. Tutschku. Demand-based radio network planning of cellular mobile communication systems. Technical Report 177, Institute of Computer Science, University of Würzburg, Am Hubland, 97074 Würzburg, Germany, July 1997.

[16] O. Dousse, F. Baccelli, and P. Thiran. Impact of interferences on connectivity of ad hoc networks. In *Proc. IEEE Infocom*, San Francisco, April 2003.

[17] M. Franceschetti, L. Booth, M. Cook, R. Meester, and J. Bruck. Percolation in multi-hop wireless networks. Technical report, California Institute of Technology, Parallel and Distributed Systems Group, September 2003.

[18] R. Meester and R. Roy. *Continuum percolation*, volume 119 of *Cambridge tracts in mathematics*. Cambridge University Press, 1996.

[19] D. Stoyan, W. S. Kendall, and J. Mecke. *Stochastic Geometry and its Applications*. John Wiley & Sons, Chichester, 2nd edition, 1995.

[20] Statistik Schweiz, Bundesamt für Statistik, http://www.statistik.admin.ch/.

[21] G. Brassard and P. Bratley. *Fundamentals of Algorithmics*. Prentice Hall, 1st edition, 1996.

[22] K. Sayood. *Introduction to Data Compression*. Morgan Kaufmann, 2nd edition, 2000.

[23] J. Beasley and P. Chu. A genetic algorithm for the set covering problem. In *European Journal of Operational Research*, pages 392–404, 1996.

[24] The Network Simulator - ns-2, http://www.isi.edu/nsnam/ns/.

[25] V. Naoumov and Th. Gross. Simulation of large ad hoc networks. In *Proceedings of The Sixth ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM 2003)*, September 2003.

[26] wxWindows, http://www.wxwindows.org.