Department of Informatics
Algorithms

Master Thesis

Localizing Cell Towers from Crowdsourced Measurements

Author:
Johan Alexander Nordstrand Rusvik

Supervisor:
Prof. Jan Arne Telle
Dep. of Informatics
Co-Supervisor:
Prof. Hans K. Hvide
Dep. of Economics

Monday 1st June, 2015

Abstract

Today, several internet sites exist that aim to provide the locations and number of cellular network antennas worldwide. For example [1], [2] and [3]. What makes this task difficult to accomplish is the lack of information available about the whereabouts and number of antennas. Only in a few countries are correct locations for some cellular network antennas known. Otherwise, these sites base their knowledge about cellular network antenna locations on measurement data collected from crowdsourcing. OpenCellID uses a simple and primitive algorithm for estimating antenna locations based on such measurements. In this thesis we suggest an alternative approach to localize cellular network antennas based on data provided by OpenCellID.

We start by giving an introduction to the problem, and give a brief overview of related work. This includes localization of mobile devices in addition to localization of cellular network antennas. We then present some background information for our algorithm development. Next we develop two similar algorithms for localizing cellular network antennas. One utilizes distance between measurements, the other utilizes Received Signal Strength (RSS) values among measurements. We experiment with the two algorithms on theoretical generated test data, and argue that utilizing RSS gives the most accurate estimated antenna locations.

Next we present the OpenCellID data. We explore this data in detail before defining two subsets we will test our two algorithms on. One subset contains measurement data where correct antenna locations are known. The other contains measurement data for antennas in the Bergen City Center area. We then estimate cellular network antenna locations with our two algorithms for the two subsets. Our tests will show that utilizing RSS estimates more accurate antenna locations when correct antenna locations are known and can be compared to. We end the thesis by analyzing two measurement distribution patterns, and propose how the algorithms can be improved.

Acknowledgements

Foremost, I want to thank my supervisor Jan Arne Telle. Without his invaluable contribution of ideas and advice, this thesis would have been impossible to complete.

I would also like to thank my co-supervisor Hans K. Hvide for relevant discussions as well as introducing me to the problem at hand.

I want to thank Markus Semm and Krzysztof Ociepa at OpenCellID.org for providing knowledge and explanations about the OpenCellID data.

Many thanks go to lecturers and the administration at the Departement of Informatics for providing an excellent student environment, both for learning and festivities.

Last but not least, I want to thank my family, friends and the rest of the student community for all the good times during my five years as a student at the University of Bergen.

Contents

List of Algorithms

Chapter 1

Introduction

In this chapter we explain the background and motivation for our work, present our research goals, present some fundamental terminology, and give an outline of the thesis.

1.1 Motivation and Background

Have you ever wondered where your mobile phone receives it signal from? Who can honestly say they have spotted an antenna transmitting such signals? Would the regular citizen recognize such an antenna, even if he saw one?

No one can reject that instant communication through mobile devices are becoming more and more a part of our society. The number of devices connected to the cellular network is increasing and the demand for good reception is growing. In urban areas the general citizen might get frustrated if she is disconnected from the network unwillingly for just a second. Users expect to be able to surf the internet inside car tunnels made of concrete, or make phone calls below the ground while riding the subway. But how is this possible? To communicate with the outside world through the cell phone requires the phone to be connected to the cellular network. This means it needs to receive a signal from somewhere. These signals originate from antennas that are placed all over the world. How many antennas are there and where are they located?

There are several internet sites that attempt to answer this question. For example *opensignal.com* [2], *cellmapper.net* [3] and *opencellid.org* [1]. Their goal is to show exactly where the antennas that broadcast cell signals are located. They provide quite accurate antenna locations but not correct, at least in most cases. What makes this goal difficult to accomplish is that the cellular network providers are not obliged to provide data about the antenna locations. Only in a few specific countries are some correct cellular network antanna locations known. So how can these sites know where the rest of the antennas are located? The answer is *crowdsourcing* [12].

Crowdsourcing involves using a large crowd of people to accomplish a task or solve a problem. The word means *to outsource something to an undefined crowd.* The people in such crowds do not usually know each other, like online communities or

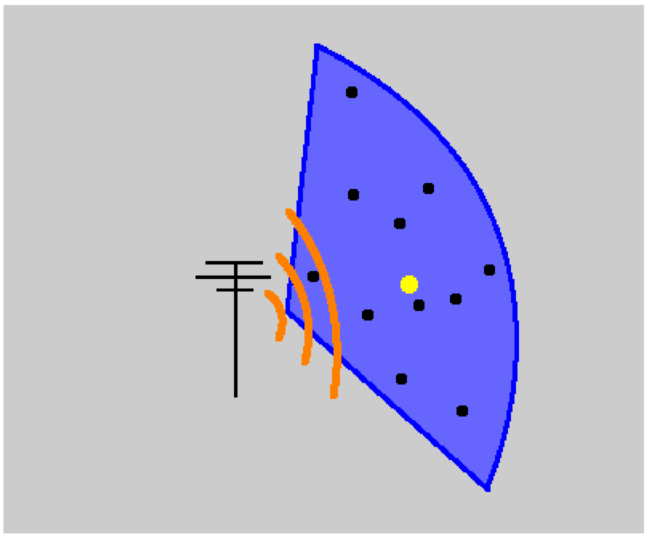


Figure 1.1: A 120° horizontal broadcasting area of a cellular network antenna, also called a *cell* which we will define more thouroughly in section 1.3.1. The black dots are measurements. The yellow dot is the estimated cellular network antenna location as estimated by OpenCellID.

the public in general. When a crowdsourcing process is set in motion, the originator makes a request to the targetted crowd. The request may for example concern ideas, feedback or content.

These sites provide smart phone applications mobile users can download. The applications gather data about the antenna the phone is connected to, the strength of the received signal, and the location of the smart phone. This collection of data, also called a *measurement*, is then transmitted to a database. We will define a measurement more thouroughly in section 1.3.2. Based on this data the sites attempt to locate the antennas. This thesis is concerned with the algorithmic challenge facing these sites: how can we estimate the location of cellular network antennas based on crowdsourced data?

OpenCellID [1] is the worlds largest collaborative community project for collecting GPS locations of cellular network antennas. As of January 2015, their database contained almost 7 million unique antennas and 1.2 Billion crowdsourced measurements. The algorithm used by OpenCellID to compute cellular network antenna locations is very simple. Based on the measurements collected for an antenna, the longitude and latitude coordinates for the antenna is set to be the mean of those measurements' longitudes and latitudes. This is not a good approach since antennas broadcast their signals in the shape of a pizza slize, usually 120° horizontally. Omnidirectional antennas are very rare. This means most antennas in OpenCellID are estimated to be located approximately in the middle of the pizza slice shaped sector, when the correct location is somewhere so that every measurement lies to some direction of the antenna within 120°. See figure 1.1.

Depending on the map scale, this error in antenna localization might not be that significant. We can still get a good idea of how many antennas are within a specific area, and approximately where they are located. But out on the street, in a more practical sense, it is more significant. 500 or even 1,000 meters might look like nothing on a map at a certain scale, but is a significant distance in reality.

1.1.1 Applications

How can we benefit from knowing correct cellular network antenna locations? The most direct application is an alternative to the Global Positioning System (GPS) [17]. GPS provides below 1 meter localization error for devices which receive signals from more than four GPS satellites, but requires special hardware technology which is both expensive and energy-consuming. Say we know the correct locations of every cellular network antenna and which mobile devices every antenna is broadcasting to. Then we can use this information to locate and track individual devices. This can be a cost-effective alternative to GPS. Especially in places where GPS is not available can localization by the cellular network be a valuable asset.

For the individual user, knowledge about cellular network antenna locations can help find the best network provider in areas where the user resides. If the user is concerned with radiation from radiowaves, he can choose his areas of residence based on radio emission data collected by keeping track of where cellular network antennas are located.

In a larger perspective, society can also benefit from cellular network antenna localization of mobile devices. By gathering data about antennas dozens of mobile devices are connected to, for example at certain times of the day or year, or when it is raining or the sun is shining, can help develop infrastructure or cultural visiting places. For example, such data can justify decisions regarding new roads along paths where many people travel, or the building of an amusement park at a place many people travel by in the summer when the sun is shining.

1.2 Research Questions

With this thesis we want to briefly identify techniques and approaches that has been developed to localize cellular network antennas. We want to investigate if these techniques and approaches can be applied to the measurement data collected by OpenCellID, and thereby improve the accuracy of their estimated cellular network antenna locations. We want to develop our own algorithm for locating the antennas, and test it on generated theoretical test data and real data provided by OpenCellID.

1.3 Fundamental Terminology

We now define some fundamental terminology. The problem this thesis addresses is based on the service provided by OpenCellID, so we use the same terminology as them. Other terminolgy will be explained throughout the thesis.

1.3.1 Cell

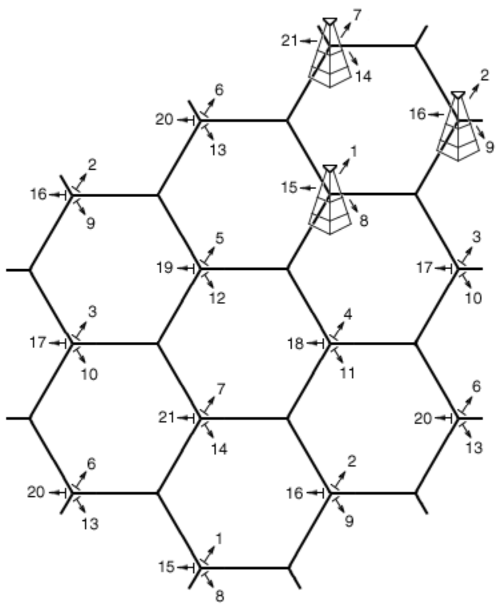


Figure 1.2: Cells as a system of hexagons. [13]

A *cell* is a geographic area that is covered by a cellular network antenna [20, p. 548]. The signal originating from this antenna can potentially reach every mobile device within the cell area. The antenna is placed on a *cell tower*, or more generally a *base station*. Throughout this thesis we will also use the term *cell tower* when referring to a cellular network antenna. To make it easy for the reader we will refer to the coverage area of a cell as the *cell sector*, the angle of the cell sector as the *cell sector angle*, and the edges of the cell sector as the *cell edges*. These components, along with measurements, makes a cell.

The cell sector angle is with few exceptions always 120°. Some cellular network providers also use cells where the cell sector angle is smaller, for example 60°. This may be beneficial in for example urban areas. It is normal that three or more cells share a cell tower to cover everything in a 360° angle around the cell tower. The cell tower will then have several antennas pointing in different directions.

To avoid disconnection and support high demand, the cells and cell towers are in theory organized as shown in figure 1.2. We can think of the cells as a system of hexagons where each hexagon contains at least three different cells. Each hexagon is then covered by at least three different cell towers. Depending on the strength of the cell tower or antenna, it may cover more than its own hexagon. This ensures efficient overlapping and constant connection to the cellular network for the users. In reality, it is difficult to maintain the hexagon system. Cell towers need to be placed at a certain hight, and buildings, mountains or other obstacles must be considered when attempting to cover a specific area. Urban areas may need more than one antenna to cover a small area where there are many obstacles, and rural areas may need fewer than in theory. Today, several cellular network providers exist in every country and each has their own cellular network and own interests.

We will also consider a cell as an object in the OpenCellID database. The data fields of an OpenCellID Cell object contain information about the cell and will be defined in section 6.2.

1.3.2 Measurement

A *measurement* represents a smart phone's registration of data about the cell tower it is currently connected to. OpenCellID obtaines such measurements with the help of smart phone applications and crowdsourcing. In addition to data about the cell the measurement also includes the current coordinates of the phone, and some other information related to the phone's location. The data fields of an OpenCellID Measurement object will be defined in section 6.2. In this thesis we will say that measurements within the same cell belong to that cell.

1.4 Thesis Outline

A brief description of the contents of the remaining chapters:

Chapter 2: Related Work In this chapter we briefly examine the research that has been done when it comes to localizing mobile devices and cell towers. The two are related.

Chapter 3: Background for our Algorithm Development In this chapter we establish some background information for our algorithm development.

Chapter 4: Cell Tower Localization based on Distance (D-CTL) In this chapter we simplify our problem definition and develop the D-CTL algorithm. We then experiment with different parameters involved in the algorithm.

Chapter 5: Cell Tower Localization based on RSS (RSS-CTL) In this chapter we develop the RSS-CTL algorithm. We then experiment with different parameters involved in the algorithm.

Chapter 6: Understanding the OpenCellID Data Set In this chapter we examine the data provided by OpenCellID. We define two datasets on which we will test D-CTL and RSS-CTL on.

Chapter 7: Running the D-CTL and RSS-CTL Algorithms on OpenCellID Data In this chapter we test D-CTL and RSS-CTL on real data provided by OpenCellID, and experiment with different parameters involved in the algorithm. We will estimate cell tower locations with D-CTL and RSS-CTL for cells where the correct cell tower locations are known, to measure their accuracy. And we will estimate cell tower locations for the Bergen City Center, where the correct cell tower locations are not known.

Chapter 8: Analysis and Improvements In this chapter we analyze some cell tower locations estimated by D-CTL and RSS-CTL. We then suggest improvements to the algorithms and end the chapter with a conclusion for the thesis.

Chapter 2

Related Work

We can divide the discussion of localization within cellular networks into two topics. Localization of cell towers, and localization of mobile devices. In academics, the latter is larger and more established, but since our task concerns localization of cell towers we will only give a brief overview of this. The two topics are related because many techniques for locating mobile devices assume that the locations of cell towers are known.

2.1 Localization of Mobile Devices

Localization of mobile devices can be used for many interesting applications. The applications usually involve tracking the mobile device, and thereby the owner of the device. Existing approaches for positioning mobile devices fall into two categories: *Range-Based* positioning and *Range-Free* positioning. A more thorough overview can be found in [28], [21]. A general overview of available localization schemes can be found in [16].

Range-Based Approaches This type of approaches assumes that the mobile devices are equipped with special hardware technology. The type of technology depends on the technique used to obtain the location of the device. Time of Arrival provides the concepts used in GPS [17]. Two other techniques is called Time Difference of Arrival [25] and Angle of Arrival [23]. In addition, Received Signal Strength (RSS) [22] can be utilized. Although range-based approaches can be very accurate, it requires expensive and energy-consuming technology.

Range-Free Approaches This type of approaches are cost-effective alternatives to Range-Based approaches when there are hardware limitations and energy constraints to consider. The trade-off is accuracy and scalability of the localization estimates. This approach is largely based on connectivity measurements with a high density of seeds. The connectivity measurements from multiple sources are used to track the movement of the mobile device. Examples of techniques are Centroid [10], APIT [15], MCL [18] and DV-hop [24].

2.2 Localization of Cell Towers based on Data Collected with Wardriving

When a technique for localizing mobile devices are dependent of cell towers and their locations, these are mostly assumed to be known. The amount of research done on localization of cell towers are significantly smaller than on localization of mobile devices. What we found are techniques based on data collected with *wardriving* [11]. Wardriving involves collecting cellular network data with one mobile receiver while on the move, usualy driving. This way they generate structured trails of measurements and can detect where the mobile receiver finds or loses signals from cell towers. The data provided by OpenCellID are not structured like that. This data contains measurements with random locations within cell sectors. There is no way to know which other cell towers they are within range of and can connect to at the time and place of creation. We present two techniques on localization of cell towers based on wardriving data, given in two different papers.

2.2.1 Paper 1: Accuracy Characterization of Cell Tower Localization [27]

This paper presents the *Bounding Technique*. This is a three-step procedure to improve the existing localization algorithms Strongest RSS and Weighted Centroid. The results are based on measurement data obtained with wardriving.

Strongest RSS This algorithm estimates a cell tower's location as the location of the measurement with the strongest observed RSS in the corresponding cell.

Weighted Centroid This algorithm estimates a cell tower's location as the geometric center of the locations of the measurements belonging to the cell. That is, the mean of the longitudes and latitudes of the measurements. When calculating the geometric center, the coordinates of each measurement are weighted by the signal strength observed by that measurement.

Bounding Technique This technique does not target the algorithms, but the measurement data used to estimate a cell tower location with Strongest RSS or Weighted Centroid. The three steps are called RSS Thresholding, Boundary Filtering and Tower-based Regrouping.

RSS Thresholding In this step, all cells whose strongest RSS observed in a measurement that is lower than a certain threshold, is detected. These cells, with corresponding cell towers and measurements, are filtered out.

Boundary Filtering In this step it is assumed that cell towers far away have their strongest RSS observations in measurements at the boundaries of the wardriving area. These cells, with corresponding cell towers and measurements, are filtered out.

The purpose of RSS Thresholding and Boundary Filtering is to detect the cell towers that can be accurately localized with the current wardriving data. Cell towers observed that are far away may be more accurately localized by gathering data closer to them.

Tower-based Regrouping In this step, measurements within cells that share a cell tower are combined, thus simulating a 360° or omnidirectional cell on which Weighted Centroid and Strongest RSS will perform better.

2.2.2 Paper 2: Base Station Localization in Search of Empty Spectrum Spaces in Cognitive Radio Networks [28]

This paper presents a two-step procedure to localize cell towers called *Localization estimation based Gaussian Mixture Model* (LGMM). The two steps are called Grid-LGMM and Expectation Maximization-LGMM (EM-LGMM).

Grid-LGMM This is an algorithm that estimates the rough locations of the cell towers using a grid-search method and the maximum likelihood estimation. This is accomplished using the Bayesian Information Criterion (BIC). Grid-LGMM works as follows. For each iteration it adds a new cell tower, which it tries to fit to all possible grid points. The BIC value decides which grid point is best. Then all previously added cell towers are readjusted to see if the new cell tower would find better locations for the existing cell towers according to the BIC. The BIC value is updated after each step until it is maximized. This means no more readjustments of the cell towers are beneficial.

EM-LGMM This step refines the grid locations by edging in to the true locations using the EM-method. This is divided into the E-step and M-step, both complex mathematical equations.

2.3 Localizing Cell Towers based on Data Collected with Crowdsourcing

Our task is to estimate the location of cell towers based on measurement data collected with crowdsourcing. Since section 2.1 concerns a different problem, these techniques or approaches are not of much use to us. The OpenCellID data do not have the neccassary properties for these ideas to be utilized. The exception is information about RSS, which we will utilize in chapter 5.

In section 2.2 we described two papers, each presenting a technique to localize cell towers. The drawback is that these techniques are based on wardriving data.

Wardriving data has properties the data from OpenCellID do not have, as described in section 2.2. This means we cannot apply these techniques directly, or in worst case, not at all.

Strongest RSS This algorithm is simple, but inaccurate. Assuming that the measurement with the strongest observed RSS within a cell, also is the measurement closest to the cell tower, is risky. Especially when the OpenCellID measurements are collected from dozens of random mobile devices. RSS can potentially be affected by many factors. For example buildings, hills or other obstacles, other types of radio waves, or the signal receiver in the mobile device. We conclude that this algorithm is useless to us.

Weighted Centroid Weighted Centroid is a simple and effective algorithm for estimating cell tower locations for omnidirectional cells. For cells with a 120° cell sector, it is not effective. Weighted Centroid will estimate the cell tower location to be the geometric center of the measurements belonging to the cell. For cells with a 120° cell sector, the correct location of the cell tower is not the geometric center of the measurements. See figure 1.1. Thus, we cannot rely on this algorithm to estimate accurate cell tower locations based on OpenCellID measurements.

Bounding Technique The purpose of the two first steps of the Bounding Technique, *RSS Thresholding* and *Boundary Filtering*, is to improve measurement data collected through wardriving. We immediately conclude that these two steps are useless to us. The third step, *Tower-Based Regrouping*, is more interesting. If the OpenCellID data provides information that lets ut combine cells that share a cell tower, we can simulate an omnidirectional cell and use Weighted Centroid to estimate the cell tower location.

LGMM The LGMM algorithm is based on a structured distribution of measurements collected with wardriving within a small area. The measurements collected by OpenCellID are not structured in the same way, and most cells have their measurements spread out over much larger areas. Thus, we conclude that the LGMM algorithm is not applicable for estimating cell tower locations based on data provided by OpenCellID.

Chapter 3

Background for our Algorithm Development

Our task concerns the estimation of cell tower locations based on measurement data collected from crowdsourcing. In this chapter we start by exploring the notion of heuristics, and establish this as the basis for our algorithm development. We then define other preliminaries for our algorithm development: the generation of theoretical test data, a measure of accuracy for estimated cell tower locations called *error*, and how we will visualize the errors in charts. We end the chapter by briefly presenting our approach for estimating cell tower locations.

3.1 Heuristics

With this algorithm we want to compute good estimates of cell tower locations. We want to start this chapter by exploring what such a good estimate is.

Recall that a cell broadcasts it's signal in a 120° cell sector. Every measurement belonging to that cell must be located within the same cell sector. This is the most basic requirement for our development. So if we compute a cell sector with a corresponding cell tower location, so that every measurement fits within it, can we automatically say that this cell tower location is the correct location? No, we cannot. There may be several possible 120° cell sectors satisfying this requirement. We cannot know which one is correct, unless we ask the cellular network providers. This is why we cannot aim to compute the correct cell tower location. Instead, we will focus on developing an algorithm that will compute a cell tower location satisfying the requirements for the location of a cell tower. We turn to heuristics for this.

A heuristic algorithm is an algorithm that aims to find an approximate or partial solution to a problem [19]. When a problem occurs where the time or space complexity of the algorithm that finds the most optimal solution is unacceptable, we turn to heuristic algorithms. In reality, it is often sufficient to find an approximate or partial solution, and not the most optimal one due to cost or time constraints. In most cases, several heuristic solutions exist. That is why heuristic algorithms operate with a set of rules to evaluate each approximate or partial solution to decide which one is the best choice. Since we cannot aim to compute correct cell tower locations, we will develop a heuristic algorithm.

In our case we must accept an approximate solution as we cannot know for sure which estimated cell tower location is more correct than the others. For a cell, we are looking for a location of the cell tower such that the cell tower successfully can broadcast it's signal to all the cell's measurements. Our algorithm need to satisfy the following rules:

1. Obviously, every measurement belonging to the cell must fit within the cell sector.
2. The distance from the cell tower to each measurement must be valid with respect to broadcasting technology.
3. The algorithm must be able to compute a good cell tower location given only a random subset of the measurements, given potential time constraints.

We choose to exclude one rule that might otherwise seem obvious: *The algorithm must find a cell tower location such that the corresponing cell sector does not include measurements from other cells.* To include this rule seems only natural if we assume that the cell tower a measurement belongs to, is the closest cell tower to that measurement. Can we make this assumption? First, we must take into consideration the direction of the cell. If the closest cell tower to a mobile device has it's cell sector pointing in the opposite direction, the mobile device would not be able to receive that cell's signal. Second, there are buildings, hills and other obstacles to consider that may block the closest cell tower's signal completely. Third, the cellular network contains cells from several different cellular network providers, even within small areas. If a mobile device subscribes to one specific cellular network provider, it would not be able to receive a signal from a cell belonging to a different cellular network provider, even if the cell was closer. We therefore conclude we cannot make the stated assumption, and cannot include this rule.

3.2 Generating Test Data

As part of developing an algorithm for localizing cell towers, we will want to generate theoretical test data to test it on. When developing the algorithm we want to implement and test it every step of the way to make sure it behaves as desired and required. Generated theoretical test data will help us see how the algorithm performs. Since we are generating it ourselves we can decide it's characteristics to simulate different scenarios, and see how that affects the outcome the algorithm.

The test data needs to represent cells and measurements. When we generate a cell and it's measurements we include the location of a cell tower. This is the most optimal location our algorithm can compute. Real data would not be able to provide the correct location of the cell tower, with some exceptions as explained in section

1.1. It is still important that we generate complete cells so we can compare estimated cell tower locations to correct cell tower locations.

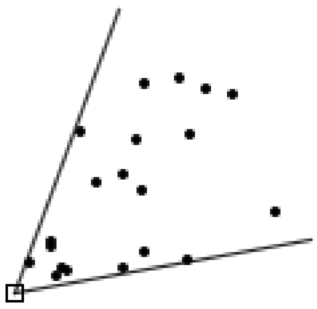


Figure 3.1: A generated cell. The black square represents the correct cell tower location. The angle of the first cell edge is 10° and the angle of the cell sector is 60°. This means we generate the cell edges at 10° and 70°, represented by the black lines. The maximum distance for measurements from the cell tower is 113, which means we also set the length of the cell edges to 113. 20 measurements are generated at random locations within the cell sector, with a maximum distance of 113 from the cell tower. Measurements are represented by black dots.

We use a two-dimensional Cartesian coordinate system [8, page 11] for infrastructure. Measurements and cell towers will then be given locations in the form of x and y coordinates. We use a two-dimensional Cartesian coordinate system to simulate the surface of the real world. To simulate it perfectly would require an environment that matches the earth's spherical shape, along with hills and buildings. We choose to use a Cartesian environment for simplicity.

To satisfy rule number 2 presented in section 3.1, we need to control how far away the measurements will be generated from the cell tower. In reality, cell towers have different range capabilities and cell phones have different reception capabilities. In addition to this, measurements collected for different cells can appear in a lot of different patterns. Sometimes the measurements are really close to the cell tower and sometimes they are really far away from the cell tower. This makes the theoretical maximum distance difficult to set. We will pick a distance that seems reasonable. When we know this maximum distance in addition to the correct cell tower coordinates, the angle of one of the cell edges and the cell sector angle, we can set the cell edges.

If each measurement is to be generated with RSS, we assume the following: If the measurement is close to the cell tower the RSS is strong, and if it is far from the cell tower the RSS is weak. To make theoretical RSS as realistic as possible it is stored as a negative value. In reality, RSS values are negative Decibel-milliwatts (dBm) numbers. The RSS value for a measurement is calculated in the following way: After the location of the measurement has been randomly generated, the RSS value is equal to the distance from the measurement to the correct cell tower location, multiplied by -1.

In section 5.8, we will experiment with deadzones. Deadzones within cell sectors are generated as follows, after measurements has been generated: To create a deadzone we take a random point within a cell sector, and choose a radius from that point. Every measurement within the circle this point and radius forms, are removed.

The actual generation of a cell happens as follows, see figure 3.1:

1. The correct cell tower location is set,

2. The maximum distance for measurements from the cell tower location is set,

3. Given the angle of one of the cell edges and the angle of the cell sector, the cell edges are set,

4. The measurements are generated with random locations within the cell sector and within a maximum distance from cell tower, and optionally with RSS,

5. If the cell is to have a deadzone, this is now generated.

3.3 Error

Throughout this thesis we will need a way to measure the accuracy of an estimated cell tower location when we know the correct cell tower location. We will measure this accuracy by calculating the distance from the estimated cell tower location to the correct cell tower location. We will call this distance the *error*. Say we have estimated the cell tower location of a cell. Then the error of the estimated cell tower location is the distance from the estimated cell tower location to the correct cell tower location. Say we have estimated the cell tower locations of several cells. Then the *average error* of the estimated cell tower locations is the average distance from the estimated cell tower locations to the correct cell tower locations. We can only use this measure of accuracy when the correct cell tower location is known.

3.4 Chart Notations

We will use charts to visualize several average errors throughout this thesis. Each chart will have several graphs, so we need notations to tell them apart. Each graph is presented with the following information to the right of the chart:

- If a chart contains graphs resulting from estimated cell towers locations computed by more than one algorithm, the algorithm is specified.

- Non-constant parameters.

3.5 Implementation Details

Everything we implement is done using Java 7 [4] and Eclipse Integrated Development Environment [5]. When running one of the algorithms or some other part of the code, we set initial heap size and maximum heap size to 512MB. We used a MacBook Air [6] with a 1.7 GHz Intel Core i7 Processor and 8 GB RAM.

3.6 Localizing Cell Towers Theoretically

To estimate the location of a cell tower as effective as possible in real time, we need to develop a heuristic algorithm that works on a random subset of the measurements available. For some cells, not many measurements exist and we may have to use all of them. We assume the measurements of a cell is randomly distributed. To estimate the location of the cell tower we start by computing a line with two measurements as endpoints. This line will give us a good impression of which direction the cell is pointing. This is done with either the D-DL or RSS-DL sub-routine. After the computation of this line we propose two possible cell sector solutions, one for each endpoint. We use the FS sub-routine for this. We then choose one of the estimated cell sectors as the final estimated cell sector solution, and output the corresponding cell tower location, with the help of either D-CD or RSS-CD.

We start by simplifying our problem definition by assuming that RSS is not available and cells broadcast in a cell sector of only $10°$.

Chapter 4

Cell Tower Localization based on Distance (D-CTL)

In this chapter we develop an algorithm for estimating the location of a cell tower, based on the distance between measurements belonging to the cell. We start by simplifying our original problem definition by making some assumptions. Then we present the sub-routines and complete algorithm for estimating a cell tower location based on distance between measurements. The algorithm and it's sub-routines are briefly summarized in table 4.1 and 4.2. After this we will experiment with the different parameters involved in the algorithm. We end the chapter by concluding that utilizing distance between measurements will not estimate accurate cell tower locations for large cell sector angles.

Algorithm	Abbrev.	Description	Sect.
Cell Tower Localization based on Distance	D-CTL	Estimates the cell tower location of a cell by utilizing distance between measurements. Uses the following sub-routines: D-DL, CS, FS, D-CD.	4.5

Table 4.1: Cell Tower Localization based on Distance (D-CTL).

4.1 Small Angles

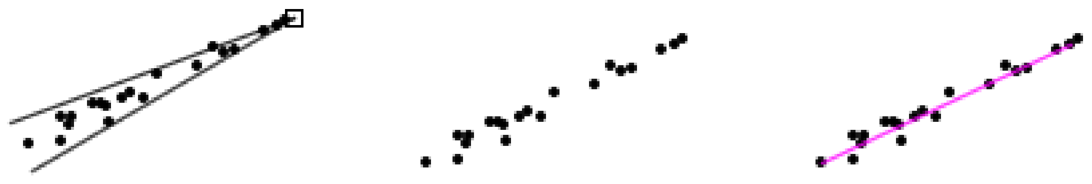
Let us simplify our problem definition by assuming that cells broadcast in a narrow sector, say 10°, and not 120°. We also assume that RSS is not available and that the maximum distance for measurements from the cell tower is 113. In that case the following is a reasonable heuristic.

Distance as Parameter We assume that RSS is not available, so we will utilize distance between the measurements of a cell to estimate the cell tower location.

Sub-routine	Abbrev.	Description	Sect.
Direction Line based on Distance	D-DL	Computes a line between the two measurements that are furthest apart from each other, chosen from n^2 randomly picked pairs of measurements.	4.2
Compute Sector	CS	Used within FS. Based on the line computed by D-DL, computes the actual cell sectors we try to fit the measurements within.	4.3
Find Sector	FS	Based on the line computed by D-DL, finds a cell sector every measurement fits within.	4.3
Cell Direction based on Distance	D-CD	Based on two estimated cell sectors computed by FS, guesses which one is the best choice based on distance.	4.4

Table 4.2: Sub-routines for the D-CTL algorithm.

4.2 Sub-Routine: Direction Line based on Distance (D-DL)


(a) The square represents the cell tower, the two lines represents the cell edges, and the dots represents the measurements.

(b) Only the measurements are shown.

(c) D-DL has computed $l_{direction}$ from the measurements, which is represented by the purple line.

Figure 4.1: A generated cell with a 10° sector angle and 20 measurements.

Figure 4.1a displays a generated cell with a 10° cell sector and 20 randomly distributed measurements. The angles of the cell edges of this particular cell is 200° and 210°. This means the angle of each measurement from the cell tower is between 200° and 210°. We now assume we do not know the location of the cell tower or the angles of the cell edges, as displayed in figure 4.1b. We ask the following question: To witch direction of the measurements is the cell tower located, when we only know the location of the measurements? We can easily guess the answer by looking at the figure, but we need to come up with an answer algorithmically.

We present the first sub-routine of the D-CTL algorithm: *Direction Line based on Distance* (D-DL). The purpose of D-DL is to estimate the direction from the measurements the cell tower is located. It does that by iterating over n^2 pairs of measurements and finds the pair with the largest distance between them. D-DL then forms a line $l_{direction}$ with the two measurements as endpoints. See figure 4.1c.

D-DL takes as input M and n, where M is the set of measurements for a cell. First, it declares the variables ep_1, ep_2 and $d_{difference} = 0$ for storing the two measurements with the current largest distance between them, and that distance. For each of the following n iterations it randomly picks a measurement from M, compares the distance between it to n other randomly picked measurements from M, and stores the largest distance and the pair of measurements representing it in ep_1, ep_2 and $d_{difference}$. The algorithm outputs the line $l_{direction}$ with ep_1 and ep_2 as endpoints. See algorithm 1.

```
Algorithm 1: Direction Line based on Distance (D-DL)
input:  $M$ and $n$
output: $l_{direction}$
Declare variables $ep_1$,$ep_2$ and $d_{difference} = 0$
for 0 to $n$
  randomly pick a measurement $a \in M$
  for 0 to $n$
    randomly pick a measurement $b \in M$ such that $a \neq b$
    if the distance between $a$ and $b$ is larger than the distance currently
       stored in $d_{difference}$,
       store $a$ in $ep_1$, $b$ in $ep_2$, and the distance between them in $d_{difference}$
return $l_{direction}$ with $ep_1$ and $ep_2$ as endpoints
```

Algorithm 1: Direction Line based on Distance (D-DL).

Notations Intruduced in This Section:

M**:** The complete set of measurements for a cell.

n**:** D-DL compares the distance between n^2 pairs of measurements.

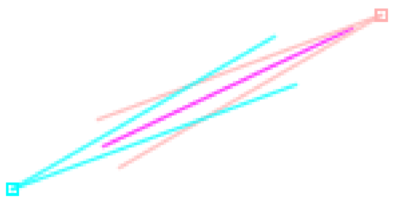
ep_1**,** ep_2**:** In D-DL, this is where the two measurements with the largest distance between them, are stored.

$d_{difference}$**:** In D-DL, this is where the largest distance between two measurements is stored.

$l_{direction}$**:** The line that represents the estimated direction of the cell.

4.3 Sub-Routines: Compute Sector (CS) and Find Sector (FS)

We now assume we have used D-DL to compute $l_{direction}$ with ep_1 and ep_2 as it's endpoints, and that we can extend $l_{direction}$ to either side. $l_{direction}$ provides infor-


(a) The purple line represents $l_{direction}$. The blue and pink squares and lines represent the two estimated cell sectors and cell towers solutions.

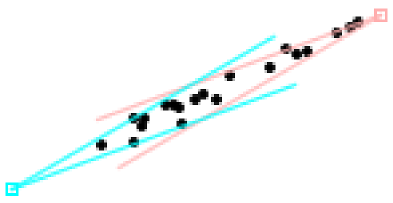

(b) The black dots represent the measurements. The blue and pink squares and lines represent the estimated cell sectors and cell towers solutions.

Figure 4.2: A generated cell with a 10° sector angle and 20 measurements.

mation as to which direction from the measurements we estimate the cell tower to be located. Since we have no more information about the measurements than their locations, we cannot know if the correct cell tower is located by ep_1 or ep_2. We therefore compute estimated cell sectors with cell tower locations for both endpoints. See figure 4.2a. The FS sub-routine is run twice, once for each endpoint. We will describe the sub-routine by assuming the correct cell tower is located by ep_1.

We present the second and third sub-routine of the D-CTL algorithm: *Compute Sector* (CS) and *Find Sector* (FS). The purpose of CS and FS is to compute a cell sector $C_{heuristic}$ by one of the endpoints of $l_{direction}$, so that each measurement $m \in M$ fits within it. See rule number 1 presented in section 3.1. FS computes $C_{heuristic}$ by doing several iterations. For each iteration, if each m does not fit within the currently computed cell sector, FS makes a new call to CS with an extended version of $l_{direction}$, which computes a new cell sector.

CS takes as input $l_{direction}$ and α, where α is the cell sector angle. It starts by rotating $l_{direction}$ around ep_1 in both directions, by and angle of $\frac{\alpha}{2}$. Then, it returns $C_{heuristic}$ with ep_1 as the cell tower location, and the two lines obtained from rotating $l_{direction}$ as cell edges.

```
Algorithm 2: Compute Sector (CS)
input:  $l_{direction}$, $\alpha$
output: $C_{heuristic}$
rotate $l_{direction}$ around $ep_1$ in both directions, by an angle of $\frac{\alpha}{2}$
return $C_{heuristic}$ with $ep_1$ as cell tower location and the two
        lines obtained from the rotations as cell edges
```

Algorithm 2: Compute Sector (CS).

FS takes as input M, $l_{direction}$, d_{extend}, and α, where d_{extend} is a constant. It starts by declaring the variable $C_{heuristic}$. Then it makes a call to CS to compute the initial estimated cell sector with the original ep_1 as the cell tower location, and stores it in $C_{heuristic}$. Then it does several iterations. For each iteration, if each m does not fit within $C_{heuristic}$ it extends $l_{direction}$ by ep_1 by a constant length d_{extend}. We ask the reader to think about it as moving ep_1 a constant length, away from ep_2, along an extended $l_{direction}$. Then a new call is made to CS with the new $l_{direction}$ as input. The iterations stop when each m fits within $C_{heuristic}$.

```
Algorithm 3: Find Sector (FS)
input:  $M$, $l_{direction}$, $d_{extend}$, $\alpha$
output: $C_{heuristic}$
declare variable $C_{heuristic}$
run CS with $l_{direction}$ and $\alpha$ as input, and store output in $C_{heuristic}$
while every measurement $m \in M$ does not fit within $C_{heuristic}$,
  extend $l_{direction}$ by $ep_1$ by constant length $d_{extend}$, and
  run CS with $l_{direction}$ and $\alpha$ as input, and store output in $C_{heuristic}$
return $C_{heuristic}$
```

Algorithm 3: Find Sector (FS).

As stated above, we apply FS on both endpoints of $l_{direction}$, thereby computing two estimated cell sector solutions. See figure 4.2b.

Notations Intruduced in This Section:

$C_{heuristic}$**:** An estimated cell sector.

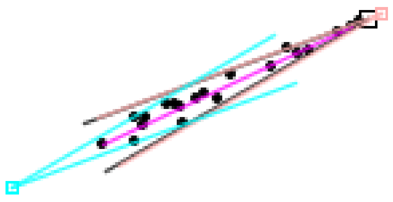
m**:** A measurement $\in M$.

α**:** The cell sector angle.

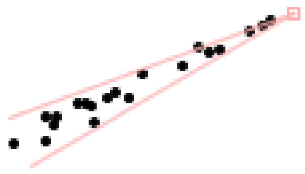
d_{extend}**:** FS extends $l_{direction}$ by the value of d_{extend} for each iteration.

4.4 Sub-Routine: Choose Direction based on Distance (D-CD)

We now assume we have used D-DL to compute $l_{direction}$ with ep_1 and ep_2 as it's endpoints, and used FS to compute two estimated cell sector solutions $C_{heuristic1}$ and $C_{heuristic2}$. Now we want to choose one of them as the final estimated solution for cell sector with corresponding cell tower location. We propose the following hypothesis:



(a) The black square and lines represent the correct cell tower and cell sector. The black dots represent the measurements. The purple line represents $l_{direction}$. The blue and pink squares and lines represent the two estimated cell sectors and cell towers solutions.



(b) The black dots represent the measurements. The pink square and lines represent the chosen estimated cell sector and cell tower solution.

Figure 4.3: A generated cell with a 10° sector angle and 20 measurements.

Hypothesis 1 In a cell sector, the further away from the cell tower we get, the distance between the two lines forming the cell edges l_1 and l_2, increases. See figure 4.4. A perfect line l to give an impression of which direction the cell is pointing would intersect the cell tower and go between l_1 and l_2 such that the distance from l to l_1 and the distance from l to l_2 always are equal. This means that measurements further away from the cell tower may be further away from l, than those close to it. Now consider the two endpoints of l, lep_1 and lep_2, and our randomly distributed measurements. Let S_1 be the subset of measurements closer to lep_1 than lep_2, and S_2 be the subset of measurements closer to lep_2 than lep_1. We calculate the mean $mean_1$ of the least possible distances from the measurements in S_1 to l, and the mean $mean_2$ of the least possible distances from the measurements in S_2 to l. We claim that if $mean_1 < mean_2$, then lep_1 is the endpoint of l at the correct cell tower location. If $mean_2 < mean_1$, then lep_2 is the endpoint of l at the correct cell tower location. We claim the hypothesis works for every $\alpha >= 10$ and $\alpha <= 120$.

We apply this hypothesis to cells we are generating. We are not equipped with such a perfect direction line l, so we use $l_{direction}$ with endpoints ep_1 and ep_2, instead. This means we will calculate the mean of the least possible distances from the measurements closer to ep_1 than ep_2, and the mean of the least possible distances from the measurements closer to ep_2 than ep_1. It is very likely that $l_{direction}$ will not intersect the correct cell tower location, but one of the endpoints will be close to it. We assume the $l_{direction}$ that is passed on to this sub-routine is the one computed by D-DL. The extension part in FS was only performed internally in that sub-routine.

We present the fourth sub-routine of the D-CTL algorithm: *Choose Direction based on Distance* (D-CD). The purpose of D-CD is to determine if $C_{heuristic1}$ or $C_{heuristic2}$ is the better choice for a final estimated cell sector solution. It does this by applying hypothesis 1 on the measurements and $l_{direction}$. The endpoint of $l_{direction}$ corresponding to the lesser of the calculated means is the endpoint presumed to be

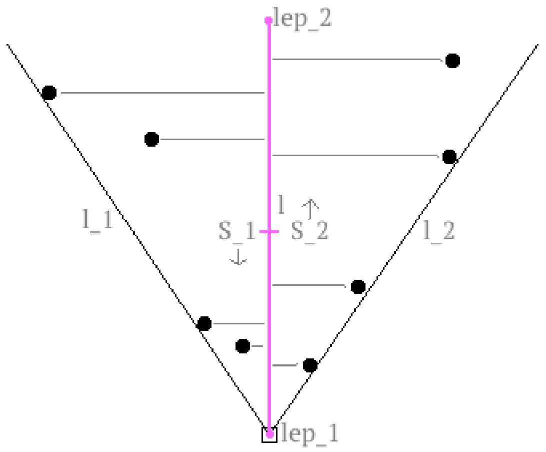


Figure 4.4: The mean of the least possible distances from the measurements in S_1 to l are very likely to be smaller than the mean of the least possible distances from the measurements in S_2 to l.

closest to the correct cell tower location. It compares the distance from this endpoint to the cell tower locations of $C_{heuristic1}$ and $C_{heuristic2}$. See figure 4.3a. Whichever cell tower location is closest will be chosen as the best estimated cell tower location. See figure 4.3b.

D-CD takes as input M, n, $l_{direction}$ with endpoints ep_1 and ep_2, $C_{heuristic1}$ and $C_{heuristic2}$. It starts by selecting n random measurements $S \in M$. It then divides S into two subsets; the measurements that are closer to ep_1 than ep_2, and the measurements that are closer to ep_2 than ep_1. D-CD then calculates the means of the least possible distances from the measurements in the two subsets of S, to $l_{direction}$. For the subset of S with the lesser mean, D-CD chooses the one of $C_{heuristic1}$ or $C_{heuristic2}$ whose cell towers location is closest to the corresponding endpoint of $l_{direction}$. See algorithm 4.

Notations Intruduced in This Section:

n: In this section, n was used differently than in section 4.2. Here, n is the number of randomly chosen measurements $S \in M$

$C_{heuristic1}$, $C_{heuristic2}$: The two estimated cell sectors computed by FS.

l_1, l_2: The two cell edges of the cell in figure 4.4.

l: The line that intersects the correct cell tower location in figure 4.4 and goes exactly in between l_1 and l_2.

lep_1, lep_2: The two endpoints of l.

S: The set of n randomly chosen measurements $\in M$.

```
Algorithm 4: Cell Direction based on Distance (D-CD)
input:  $M$, $n$, $l_{direction}$, $C_{heuristic1}$, $C_{heuristic2}$
output: $C_{heuristic1}$ or $C_{heuristic2}$
pick $n$ random measurements $S \in M$
let the measurements $\in S$ closer to $ep_1$ than $ep_2$ be $S_1$, and the
  measurements $\in S$ closer to $ep_2$ than $ep_1$ be $S_2$
calculate the mean $mean_1$ of the least possible distances from the
  measurements in $S_1$ to $l_{direction}$
calculate the mean $mean_2$ of the least possible distances from the
  measurements in $S_2$ to $l_{direction}$
if $mean_1 < mean_2$,
  return $C_{heuristic1}$ if it's cell tower location is closer to $ep_1$ than the cell
    tower location of $C_{heuristic2}$, or return $C_{heuristic2}$ if it's cell tower
    location is closer to $ep_1$ than the cell tower location of $C_{heuristic1}$
if $mean_2 < mean_1$,
  return $C_{heuristic1}$ if it's cell tower location is closer to $ep_2$ than the cell
    tower location of $C_{heuristic2}$, or return $C_{heuristic2}$ if it's cell tower
    location is closer to $ep_2$ than the cell tower location of $C_{heuristic1}$
```

Algorithm 4: Cell Direction based on Distance (D-CD).

S_1, S_2: The two subsets of measurements where each are closer to one endpoint of l or $l_{direction}$, than the other.

$mean_1$, $mean_2$: The means of the least possible distances from the measurements in S_1 and S_2, to l or $l_{direction}$.

4.5 Algorithm: Cell Tower Localization based on Distance (D-CTL)

We will now combine the algorithms defined in section 4.2, 4.3 and 4.4 into one complete algorithm for localizing cell towers based on distance: The *Cell Tower Localization based on Distance* (D-CTL) algorithm. See algorithm 5.

4.6 Scaling d_{extend}

The Find Sector sub-routine performs several iterations. For each iteration it attempts to fit every measurement within a computed cell sector which is based on $l_{direction}$. If at least one measurement does not fit, FS extends $l_{direction}$ by the value of d_{extend}. See section 4.3. In this section we want to investigate if different values of d_{extend} affect the accuracy of estimated cell tower locations. This knowledge is important in order to improve accuracy of cell tower localization and if time constraints are present.

```
Algorithm 5: Cell Tower Localization based on Distance (D-CTL)
input:  $M$, $n$, $d_{extend}$, $\alpha$
output: An estimated cell tower location
declare varible $l_{direction}$
run D-DL with $M$ and $n$ as input and store the output in $l_{direction}$
declare variables $C_{heuristic1}$ and $C_{heuristic2}$
run FS with $M$, $l_{direction}$, $d_{extend}$ and $\alpha$ as input, compute estimated cell
  sector solution for endpoint $ep_1$ of $l_{direction}$, and store the output in
  $C_{heuristic1}$
run FS with $M$, $l_{direction}$, $d_{extend}$ and $\alpha$ as input, compute estimated cell
  sector solution for endpoint $ep_2$ of $l_{direction}$, and store the output in
  $C_{heuristic2}$
declare variable $C_{heuristic}$
run D-CD with $M$, $n$, $l_{direction}$, $C_{heuristic1}$ and $C_{heuristic2}$ as input and
  store the output in $C_{heuristic}$
return the cell tower location of $C_{heuristic}$
```

Algorithm 5: Cell Tower Localization based on Distance (D-CTL).

See rule number 3 presented in section 3.1. If d_{extend} is too large, FS extends $l_{direction}$ too far, and the estimated cell tower location might be further away than neccessary. If d_{extend} is too small, FS does more iterations than neccessary, thus increasing the running time of the algorithm needlessly. We test and compare D-CTL with different values of d_{extend}. The average error values are displayed in the chart in figure 4.5. We use the following values of d_{extend}:

- $d_{extend} = 2$
- $d_{extend} = 4$
- $d_{extend} = 8$
- $d_{extend} = 16$
- $d_{extend} = 32$
- $d_{extend} = 64$

We use the following constant parameters:

- $\alpha = 10°$
- $r_{include}$ (maximum distance for each measurement from the cell tower, will be defined in the next section) = 113

We clearly see from the chart in figure 4.5 that the average error increases when d_{extend} is increasing. From these average error values we conclude that using

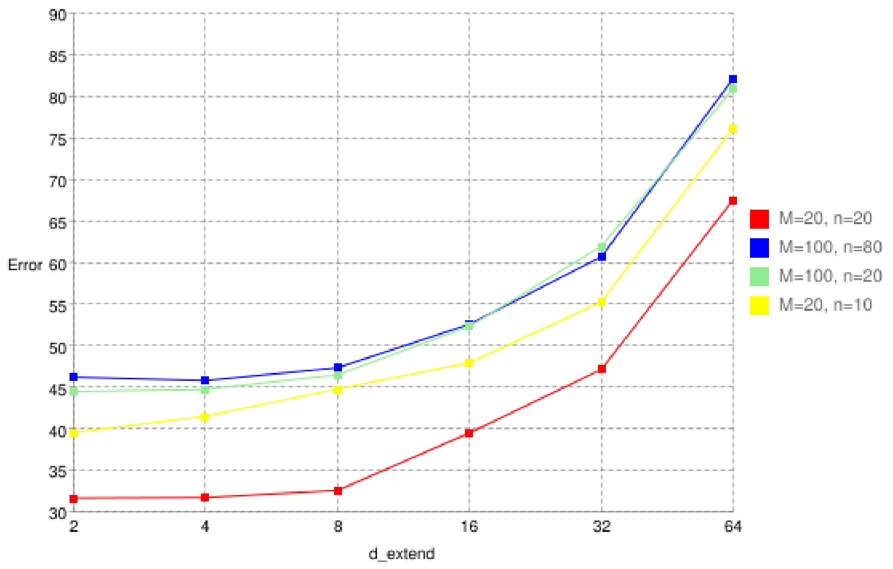


Figure 4.5: The chart displays how the average error increases when d_{extend} is increasing. See section 4.3 for the definition of d_{extend}. Each average error sample is based on the errors of 1000 generated cells. See section 4.2 for the definition of M. See section 4.2 and 4.4 for the definition of n. For example, the red line displays average error values for different values of d_{extend} when cells are generated with $M = 20$ and D-CTL is run with $n = 20$.

$d_{extend} = 2$ or $d_{extend} = 4$ is the best choice when we want to compute as accurate cell tower locations as possible. To decide which one to use further we measure the worst case running time. The assumtion is that D-CTL will run for a longer time with $d_{extend} = 2$ than $d_{extend} = 4$. The worst case running time will occur when we generate cells with $M = 100$, and run the D-CTL algorithm with $n = 80$ and $d_{extend} = 2$. α and $r_{include}$ are still constants. See the blue graph in figure 4.5. The average running time with these parameters was **0.51ms**. Because this running time is so small, we conclude that we can use $d_{extend} = 2$ for further computations of estimated cell tower locations.

4.7 Scaling Maximum Distance from Cell Tower

In the real world, the measurements within a cell can be either very close to the cell tower, or very far away. In this section we want to investigate if different values of a cell tower's maximum range affect the accuracy of the estimated cell tower locations. We do this by scaling the maximum distance that a measurement can by located from the cell tower. We ask the reader to think about this as a radius from the cell tower. Every measurement within this radius will be included in the computations. Will call this radius $r_{include}$. We test and compare D-CTL with different values of $r_{include}$. The average error values are displayed in the chart in figure 4.6. We use the following values of $r_{include}$:

- $r_{include} = 30$
- $r_{include} = 70$
- $r_{include} = 113$
- $r_{include} = 160$
- $r_{include} = 200$

We use the following constant parameters:

- $\alpha = 10°$
- $d_{extend} = 2$

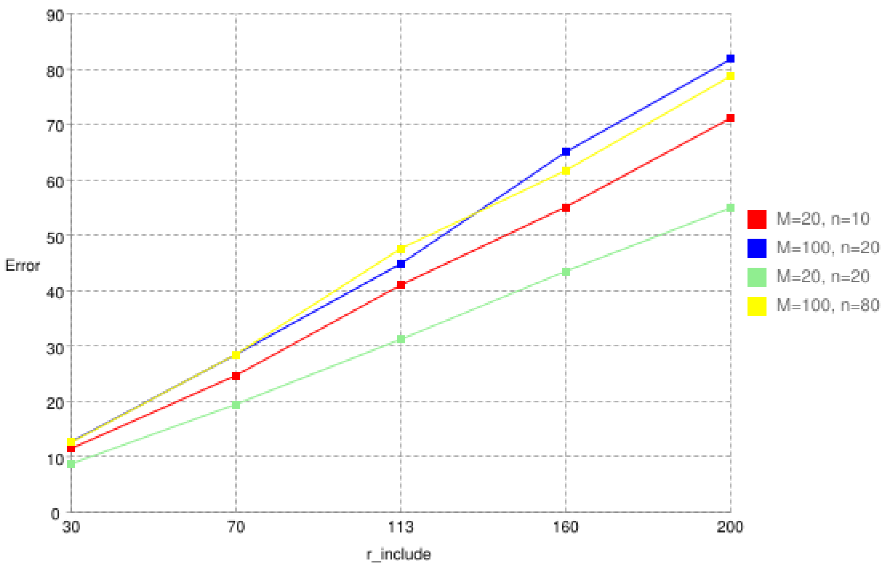


Figure 4.6: The chart displays how the average error increases when $r_{include}$ is increasing. Each average error sample is based on the errors of 1000 generated cells. See section 4.2 for the definition of M. See section 4.2 and 4.4 for the definition of n. For example, the red line displays average error values for different values of $r_{include}$ when cells are generated with $M = 20$ and D-CTL is run with $n = 10$.

We clearly see from the chart in figure 4.6 that the average error increases when $r_{include}$ is increasing. The reason for this is simple. When $r_{include}$ increases, the cell sector grows; the distance between the cell edges increases when $r_{include}$ is increasing. This means the randomly distributed measurements have a larger area to spread out on. This means the FS algorithm potentially needs to extend $l_{direction}$ further to be able to fit them all within a cell sector.

Notations Intruduced in This Section:

$r_{include}$: Every measurement of a cell that is within the value of $r_{include}$ from the correct cell tower location is included when using D-CTL to estimate the cell tower location.

4.8 Increasing Cell Sector Angle

Up until now we have utilized the distance between the measurements of a cell to estimate the cell tower location, and we have only focused on cells with a cell sector angle $\alpha = 10°$. This has only been a simplification of our original problem. The original problem definition involves estimating cell tower locations when $\alpha = 120°$. We ask the following question: How will larger values of α affect the accuracy of estimated cell tower locations computed with D-CTL? We test and compare D-CTL with different values of α. The average error values are displayed in two different charts in figure 4.7 and 4.8. The difference between the charts is the value of $r_{include}$. The chart in figure 4.7 has $r_{include} = 113$. The chart in figure 4.8 has $r_{include} = 200$. We use the following values of α:

- $\alpha = 10°$
- $\alpha = 45°$
- $\alpha = 90°$
- $\alpha = 120°$

We use the following constant parameters for the chart in figure 4.7:

- $d_{extend} = 2$
- $r_{include} = 113$

We use the following constant parameters for the chart in figure 4.8:

- $d_{extend} = 2$
- $r_{include} = 200$

As we can see from figure 4.7 and 4.8, the graphs make a big jump in average error between 45° and 90°. To understand this sudden increase, we must look into the geometry of our problem. Recall that the sub-routine D-DL in the D-CTL algorithm utilizes the longest distance between two measurements to compute $l_{direction}$. Also keep in mind that we still are assuming the measurements within a cell sector are randomly distributed. We also assume that $r_{include}$ is properly large. When a cell sector angle $\alpha = 10°$, the distance between the cell edges is smaller than $r_{include}$, which means the D-DL algorithm will compute a reliable direction line. This is also the case for $\alpha = 45°$. See figure 4.9a. When α increases beyond a certain point, the

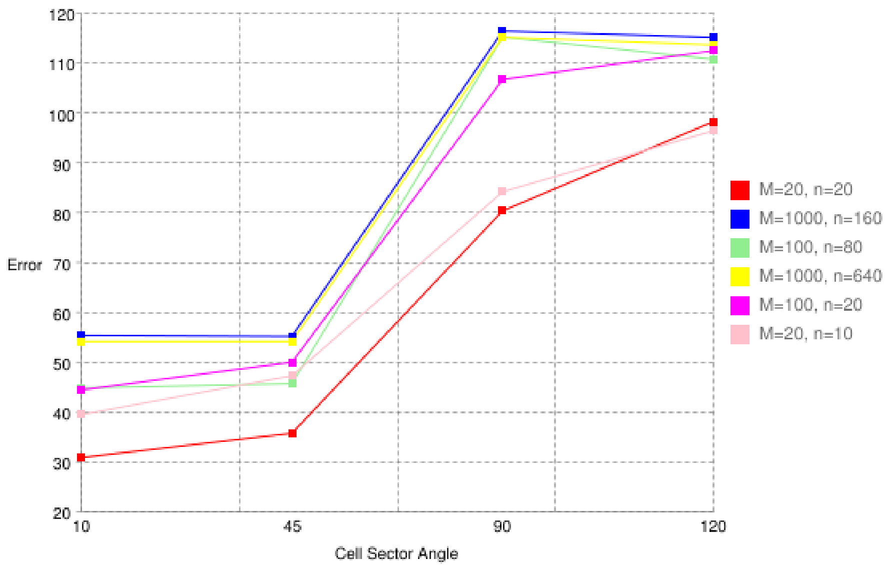


Figure 4.7: The chart displays how the average error increases when α is increasing. See section 4.3 for the definition of α. Each average error sample is based on the errors of 1000 generated cells. See section 4.2 for the definition of M. See section 4.2 and 4.4 for the definition of n. For example, the red line displays average error values for different values of α when cells are generated with $M = 20$ and D-CTL is run with $n = 20$.

distance between the cell edges will be larger than $r_{include}$. See figure 4.9b. This means that D-DL will compute a $l_{direction}$ that gives a completely wrong impression of the direction of the cell. See figure 4.10a and 4.10b. We clearly need to come up with a better approach for estimating cell tower locations for cells with large cell sector angles.

4.9 Chapter Review

In this chapter we have step by step developed an algorithm for estimating a cell tower location based on the distance between the cell's measurements. In addition, we have tested the algorithm on different values of different parameters to get an understanding of how the algorithm scales. Based on the several tests we conclude that D-CTL performs poorly on cells with cell sector angles of 120°.

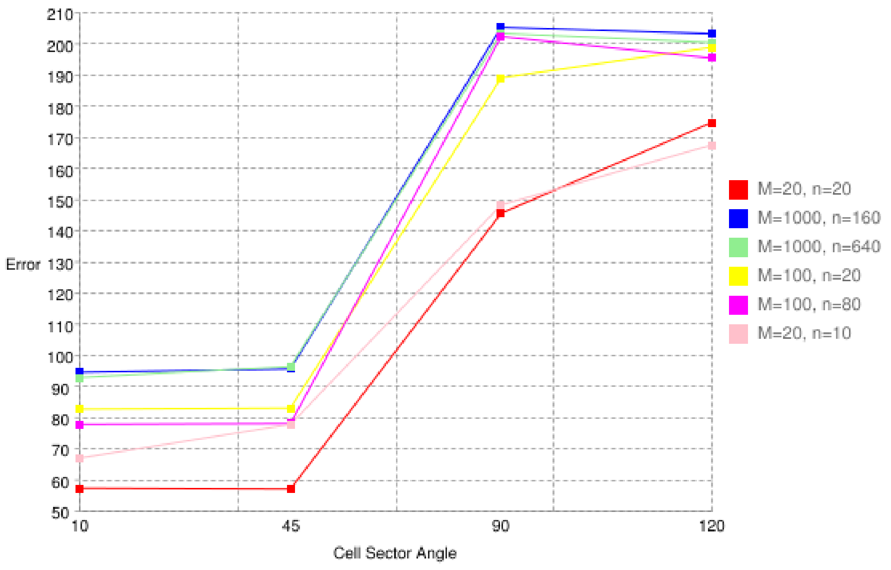
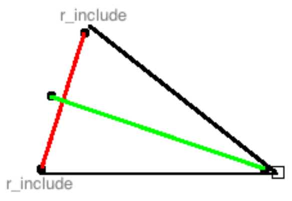
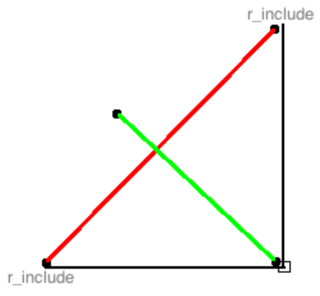


Figure 4.8: The chart displays how the average error increases when α is increasing. See section 4.3 for the definition of α. Each average error sample is based on the errors of 1000 generated cells. See section 4.2 for the definition of M. See section 4.2 and 4.4 for the definition of n. For example, the red line displays average error values for different values of α when cells are generated with $M = 20$ and D-CTL is run with $n = 20$.

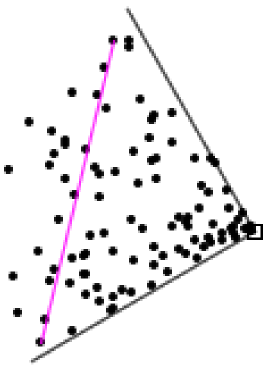


(a) Cell sector with $\alpha = 45°$

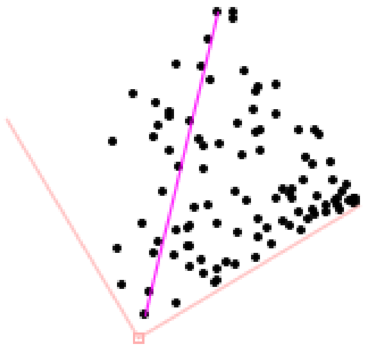


(b) Cell sector with $\alpha = 90°$

Figure 4.9: When α increases beyond a certain point, the distance between the cell edges will get larger than $r_{include}$.



(a) The black square and lines represent the correct cell tower location and cell edges. The black dots represent measurements. The purple line represents $l_{direction}$.



(b) The black dots represent measurements. The purple line represents $l_{direction}$. The pink square and lines represent the estimated cell tower location and cell edges.

Figure 4.10: Generated cell with $\alpha = 90°$ and $M = 100$. We attempt to estimate a cell tower location with the D-CTL algorithm.

Chapter 5

Cell Tower Localization based on Received Signal Strength (RSS-CTL)

In this chapter we develop an algorithm for estimating the location of a cell tower, based on the RSS values among measurements belonging to the cell. We start by presenting RSS as the new parameter. Then we present the sub-routines and complete algorithm for estimating a cell tower location based on RSS values among measurements. The algorithm and it's sub-routines are briefly summarized in table 5.1 and 5.2. After this we will experiment with the different parameters involved in the algorithm, and deadzones. We end the chapter by concluding that utilizing RSS estimates more accurate cell tower locations than by utilizing distance.

Algorithm	Abbrev.	Description	Sect.
Cell Tower Localization based on RSS	RSS-CTL	Estimates the cell tower location of a cell by utilizing difference in RSS among measurements. Uses the following sub-routines: RSS-DL, CS, FS, RSS-CD.	5.5

Table 5.1: Cell Tower Localization based on RSS (RSS-CTL).

5.1 Large Angles: Estimating Cell Tower Location based on Received Signal Strength (RSS)

We will now step away from the simplification of our problem definition. The original problem definition concerns cell sector angles $\alpha = 120°$. As we saw in section 4.8, the average error increased when α increased, when we used D-CTL to estimate cell tower locations. For $\alpha = 120°$ D-CTL performed poorly. This means we need to come up with another way to estimate cell tower locations for large values of α. We

Sub-routine	Abbrev.	Description	Sect.
Direction Line based on RSS	RSS-DL	Computes a line between the two measurements whose difference in RSS values are the largest, chosen from n^2 randomly picked pairs of measurements.	5.2
Compute Sector	CS	Used within FS. Based on the line computed by RSS-DL, computes the actual cell sectors we try to fit the measurements within.	4.3
Find Sector	FS	Based on the line computed by RSS-DL, finds a cell sector every measurement fits within.	4.3
Cell Direction based on RSS	RSS-CD	Based on two estimated cell sectors computed by FS, guesses which one is the best choice based on RSS.	5.4

Table 5.2: Sub-routines for the RSS-CTL algorithm.

introduce the parameter *Received Signal Strength* (RSS). For the duration of this chapter we will assume that $\alpha = 120°$ and that RSS is available.

RSS as Parameter We assume that each measurement is equipped with RSS. We will now utilize RSS instead of distance between measurements to estimate cell tower locations. To do this we need to give each measurement a RSS value upon generation. See section 3.2. The sub-routines that utilize RSS are very similar to the ones described in section 4.2, 4.3 and 4.4.

5.2 Sub-Routine: Direction Line based on RSS (RSS-DL)

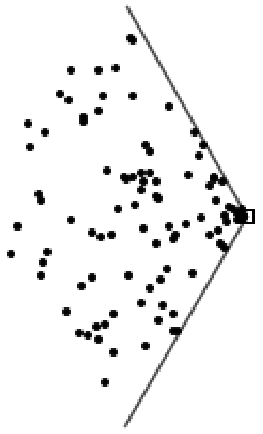
Just as in section 4.2, we want to estimate to which direction of the measurements the cell tower is located. See figure 5.1a and 5.1b. Measurements close to the cell tower have large RSS values, and measurements far away have small RSS values. We utilize this property to compute $l_{direction}$ for cells with $\alpha = 120°$.

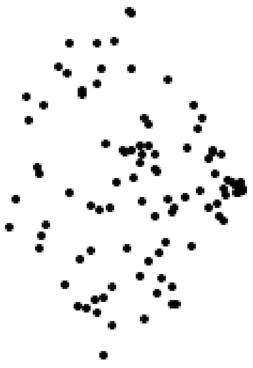
We present the first sub-routine: *Direction Line based on RSS* (RSS-DL). The purpose of RSS-DL is to estimate the direction from the measurements the cell tower is located. It does that by iterating over n^2 pairs of measurements and finds the pair with the largest difference in RSS between them. RSS-DL then forms a line $l_{direction}$ with the two measurements as endpoints. See figure 5.1c.

The only difference from the D-DL sub-routine is that RSS-DL compares RSS values between two measurements instead of distance. See algorithm 6.

Notations Intruduced in This Section:

RSS**:** The strength of the signal from a cell tower that is received by a mobile device.


(a) The square represents the cell tower, the two lines represent the cell edges, and the dots represent the measurements.


(b) Only the measurements are shown.

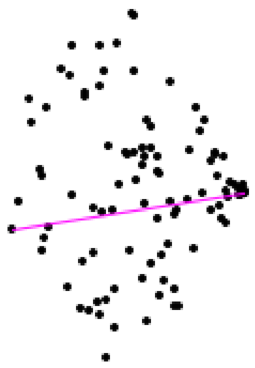

(c) RSS-DL has computed $l_{direction}$ from the measurements, which is represented by the purple line.

Figure 5.1: A generated cell with $\alpha = 120°$ and $M = 100$.

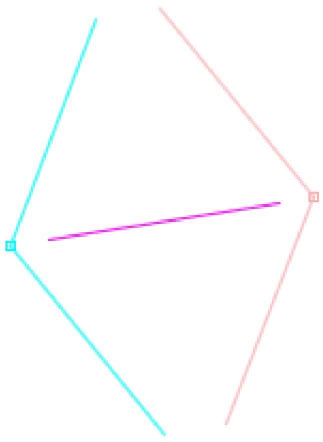
```
Algorithm 6: Direction Line based on RSS (RSS-DL)
input:  $M$, $n$
output: $l_{direction}$
Declare variables $ep_1$,$ep_2$ and $rss_{difference} = 0$
for 0 to $n$
  randomly pick a measurement $a \in M$
  for 0 to $n$
    randomly pick a measurement $b \in M$ such that $a \neq b$
    if the difference in RSS between $a$ and $b$ is larger than the difference in
       RSS currently stored in $rss_{difference}$, store $a$ in $ep_1$, $b$ in $ep_2$, and the
       difference in RSS between them in $rss_{difference}$
return $l_{direction}$ with $ep_1$ and $ep_2$ as endpoints
```

Algorithm 6: Direction Line based on RSS (RSS-DL).

$rss_{difference}$: The difference in RSS between two measurements.

5.3 Sub-Routines: Compute Sector (CS) and Find Sector (FS)

We now assume that we have used RSS-DL to compute $l_{direction}$. Just as in section 4.3 we use the CS and FS sub-routines to compute two estimated cell sectors. See figure 5.2a and 5.2b. We do not alter these algorithms in any way.


(a) The purple line represents $l_{direction}$. The blue and pink squares and lines represent the two estimated cell sectors and cell towers solutions.

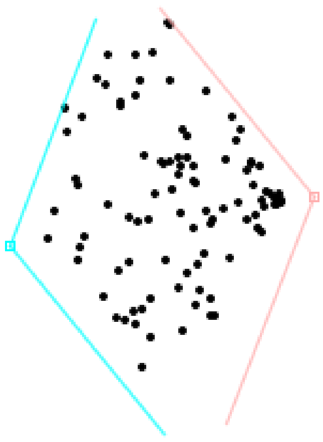
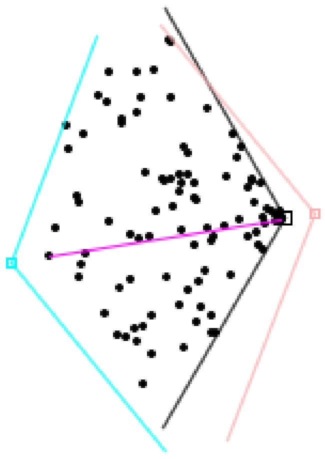

(b) The black dots represent measurements. The blue and pink squares and lines represent the two estimated cell sectors and cell towers solutions.

Figure 5.2: A generated cell with $\alpha = 120°$ and $M = 100$.

5.4 Sub-Routine: Choose Direction based on RSS (RSS-CD)


(a) The black square and lines represent the correct cell tower and cell sector. The black dots represent the measurements. The purple line represents $l_{direction}$. The blue and pink squares and lines represent the two estimated cell sectors and cell towers solutions.

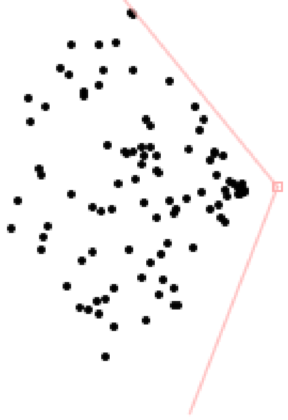

(b) The black dots represent the measurements. The pink square and lines represent the chosen estimated cell sector and cell tower solution.

Figure 5.3: A generated cell with a 10° sector angle and 20 measurements.

We now assume we have used RSS-DL to compute $l_{direction}$ and used FS to compute two estimated cell sector solutions $C_{heuristic1}$ and $C_{heuristic2}$. Just as in section 4.4 we want to choose one of them as the final estimated cell sector solution with a corresponding cell tower location. See figure 5.3a and 5.3b. To do this we take advantage of hypothesis 1 described in section 4.4, with some alterations. In the hypthesis we calculate the mean of the least possible distances from measurements to l. l was the line that would perfectly give an impression of which direction the cell tower was pointing. Now, we take advantage of the fact that measurements closer to the cell tower have a stronger RSS than those further away.

We present the fourth sub-routine: *Choose Direction based on RSS* (RSS-CD). The purpose of RSS-CD is to determine if $C_{heuristic1}$ or $C_{heuristic2}$ is the better choice for an estimated cell sector solution. The sub-routine is very similar to the D-CD sub-routine, but we calculate the means differently. We calculate the mean $mean_1$ of the RSS values of the measurements in S_1 and the mean $mean_2$ of the RSS values of the measurements in S_2. See algorithm 7.

```
Algorithm 7: Cell Direction based on RSS (RSS-CD)

input:  $M$, $n$, $l_{direction}$, $C_{heuristic1}$, $C_{heuristic2}$
output: $C_{heuristic1}$ or $C_{heuristic2}$

pick $n$ random measurements $S \in M$
let the measurements $\in S$ closer to $ep_1$ than $ep_2$ be $S_1$, and the
    measurements $\in S$ closer to $ep_2$ than $ep_1$ be $S_2$
calculate the mean $mean_1$ of the RSS values of the measurements in $S_1$
calculate the mean $mean_2$ of the RSS values of the measurements in $S_2$
if $mean_1 < mean_2$,
    return $C_{heuristic1}$ if it's cell tower location is closer to $ep_1$ than the cell
        tower location of $C_{heuristic2}$, or return $C_{heuristic2}$ if it's cell tower
        location is closer to $ep_1$ than the cell tower location of $C_{heuristic1}$
if $mean_2 < mean_1$,
    return $C_{heuristic1}$ if it's cell tower location is closer to $ep_2$ than the cell
        tower location of $C_{heuristic2}$, or return $C_{heuristic2}$ if it's cell tower
        location is closer to $ep_2$ than the cell tower location of $C_{heuristic1}$
```

Algorithm 7: Cell Direction based on RSS (RSS-CD).

Notations Intruduced in This Section:

***$mean_1$*, *$mean_2$*:** In this section, $mean_1$ and $mean_2$ was used differently than in section 4.4. Here, $mean_1$ and $mean_2$ are the means of the RSS values of the measurements in S_1 and S_2, respectively.

5.5 Algorithm: Cell Tower Localization based on RSS (RSS-CTL)

We will now combine the sub-routines defined in section 5.2, 4.3 and 5.4 into one complete algorithm for localizing cell towers based on RSS: The *Cell Tower Localization based on RSS* (RSS-CTL) algorithm. See algorithm 8.

```
Algorithm 8: Cell Tower Localization based on RSS (RSS-CTL)
input:  $M$, $n$, $d_{extend}$, $\alpha$
output: An estimated cell tower location
declare varible $l_{direction}$
run RSS-DL with $M$ and $n$ as input and store the output in $l_{direction}$
declare variables $C_{heuristic1}$ and $C_{heuristic2}$
run FS with $M$, $l_{direction}$, $d_{extend}$ and $\alpha$ as input, compute estimated cell
  sector solution for endpoint $ep_1$ of $l_{direction}$, and store the output in
  $C_{heuristic1}$
run FS with $M$, $l_{direction}$, $d_{extend}$ and $\alpha$ as input, compute estimated cell
  sector solution for endpoint $ep_2$ of $l_{direction}$, and store the output in
  $C_{heuristic2}$
declare variable $C_{heuristic}$
run RSS-CD with $M$, $n$, $l_{direction}$, $C_{heuristic1}$ and $C_{heuristic2}$ as input and
  store the output in $C_{heuristic}$
return the cell tower location of $C_{heuristic}$
```

Algorithm 8: Cell Tower Localization based on RSS (RSS-CTL).

5.6 Scaling d_{extend}

In section 4.6 we ran the D-CTL algorithm on cells with $\alpha = 10°$, for different values of d_{extend}. Now do the same on cells with $\alpha = 120°$. In addition, we will run both D-CTL and RSS-CTL for comparison. We test and compare D-CTL and RSS-CTL with different values of d_{extend}. The average error values are displayed in the chart in figure 5.4. We use the following values of d_{extend}:

- $d_{extend} = 2$
- $d_{extend} = 4$
- $d_{extend} = 8$
- $d_{extend} = 16$
- $d_{extend} = 32$
- $d_{extend} = 64$

We use the following constant parameters:

- $\alpha = 120°$
- $r_{include} = 113$

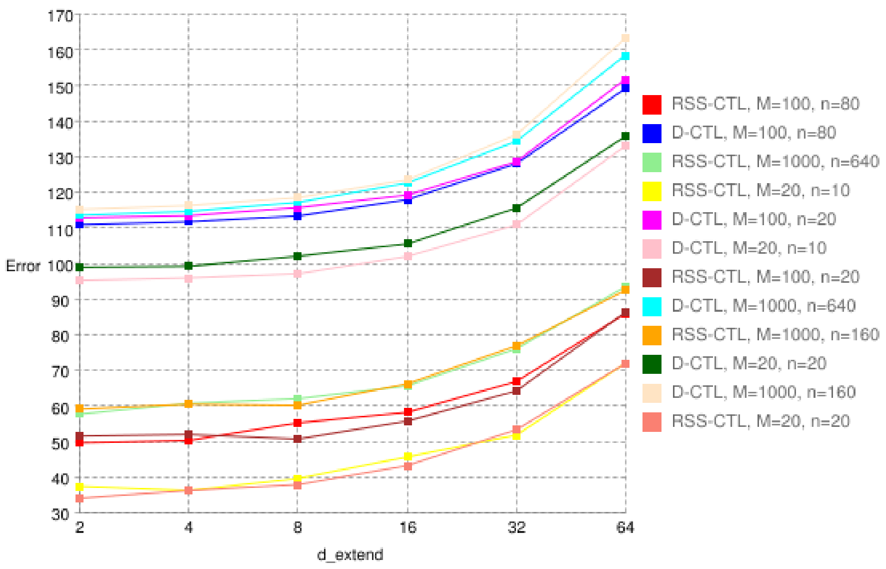


Figure 5.4: The chart displays how the average error increases when d_{extend} is increasing, for both the D-CTL and RSS-CTL algorithms. See section 4.3 for the definition of d_{extend}. Each average error sample is based on the errors of 1000 generated cells. See section 4.2 for the definition of M. See section 4.2 and 4.4 for the definition of n. For example, the red line displays average error values for different values of d_{extend} when we run RSS-CTL with $n = 80$ and cells are generated with $M = 100$.

We clearly see from the chart in figure 5.4 that the average error increases when d_{extend} is increasing, just as in the chart in figure 4.5. The average error values from running D-CTL are higher than those from running RSS-CTL, but this is just as expected when $\alpha = 120°$. Recall that we found the average running time when we generated cells with $M = 100$, $r_{include} = 113$ and $\alpha = 10°$ and ran D-CTL with $n = 80$ and $d_{extend} = 2$, in section 4.6. $r_{include}$ and α were constants. This was the most time consuming scenario, and was to justify further use of $d_{extend} = 2$. The average running time for this scenario was **0.51ms**.

Now we do the same for cells with $\alpha = 120°$. The assumption is that the D-CTL and RSS-CTL algorithms will run for a longer time with $d_{extend} = 2$, but on the other hand will compute the most accurate estimated cell tower locations. The worst case time will occur when we generate cells with $M = 1000$, and run the D-CTL and RSS-CTL algorithms with $n = 640$ and $d_{extend} = 2$. α and $r_{include}$ are still constants. See the light blue and light green graphs in the chart in figure 5.4. The average running time with these parameters for D-CTL was **11.88ms**. The average running time for RSS-CTL was **10.24ms**. Because these average times are so small, we conclude that we can use $d_{extend} = 2$ for further computations of estimated cell tower locations.

5.7 Scaling Maximum Distance from Cell Tower

In section 4.7 we ran the D-CTL algorithm on cells with $\alpha = 10°$, for different values of $r_{include}$. Now we do the same on cells with $\alpha = 120°$. In addition, we will run both D-CTL and RSS-CTL for comparison. We test and compare D-CTL and RSS-CTL with different values of $r_{include}$. The average error values are displayed in the chart in figure 5.5. We use the following values of $r_{include}$:

- $r_{include} = 30$
- $r_{include} = 70$
- $r_{include} = 113$
- $r_{include} = 160$
- $r_{include} = 200$

We use the following constant parameters:

- $\alpha = 120°$
- $d_{extend} = 2$

The graphs in the chart in figure 5.5 are very similar to the graphs in the chart in figure 4.6. It is interesting to see that the average error for RSS-CTL increases more slowly than the average error for D-CTL. This strengthen our conclusion that RSS-CTL computes more accurate cell tower locations than D-CTL for cells with $\alpha = 120°$.

5.8 Deadzones

To properly simulate a cell we need to consider that there may be areas within the cell sector where no measurements are recorded. We call such areas *deadzones*. In reality, deadzones can occur for several different reasons. For example, a deadzone can occur behind a tall hill that completely blocks the signal. Or if there is a lake within the cell sector, this may potentially form a deadzone. Due to potential deadzones' variation in size, shape and location, they are difficult to simulate. For simplicity we will simulate deadzones as circular shapes, but vary the size and location within the cell sector. See figure 5.6a and 5.6b, and section 3.2. This strategy also stimulates the fact that for example only 40% of a lake is within a sell sector.

We ask the following question: Does the D-CTL and RSS-CTL algorithms perform worse when deadzones are present within cell sectors? We test and compare D-CTL

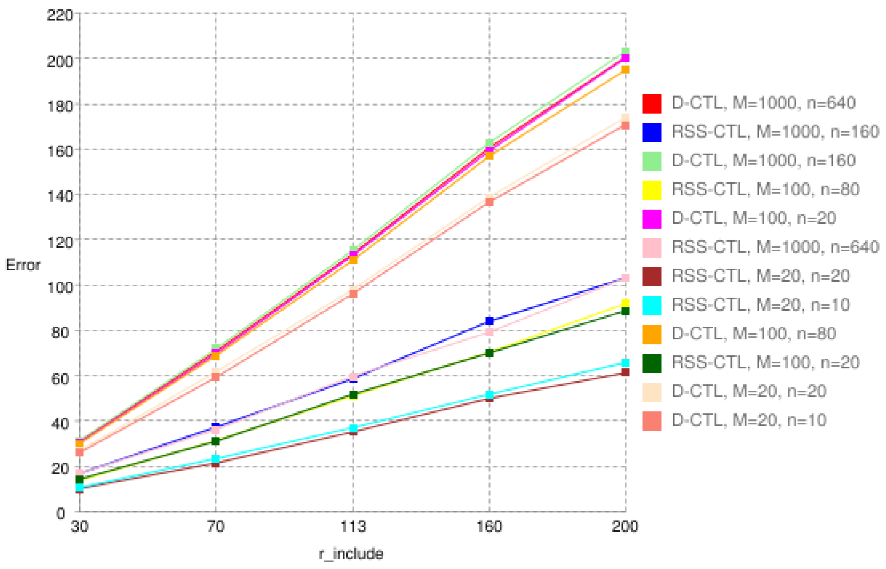
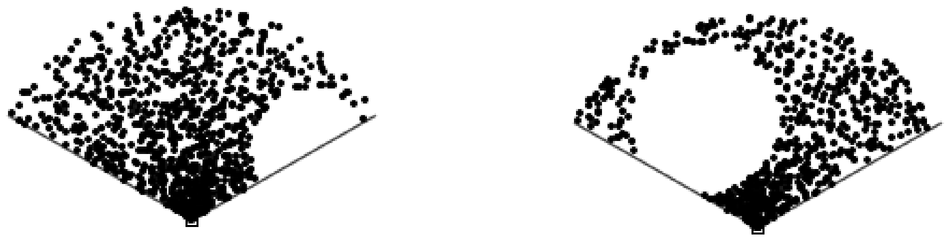


Figure 5.5: The chart displays how the average error increases when $r_{include}$ is increasing, for both the D-CTL and RSS-CTL algorithms. See section 4.7 for the definition of $r_{include}$. Each average error sample is based on the errors of 1000 generated cells. See section 4.2 for the definition of M. See section 4.2 and 4.4 for the definition of n. For example, the red line displays average error values for different values of $r_{include}$ when we run D-CTL with $n = 640$ and cells are generated with $M = 1000$..



(a) $r_{dz} = 30$ (b) $r_{dz} = 40$

Figure 5.6: Two generated cells, each with a deadzone.

and RSS-CTL with different values of the deadzone radius r_{dz}. The average error values are displayed in two different charts in figure 5.7 and 5.8. The difference between the charts is the value of $r_{include}$. The chart in figure 5.7 has $r_{include} = 113$. The chart in figure 5.8 has $r_{include} = 200$. We use the following values of r_{dz}:

- $r_{dz} = 10$
- $r_{dz} = 20$
- $r_{dz} = 30$
- $r_{dz} = 40$

We use the following constant parameters for the chart in figure 5.7:

- $\alpha = 120°$
- $d_{extend} = 2$
- $r_{include} = 113$

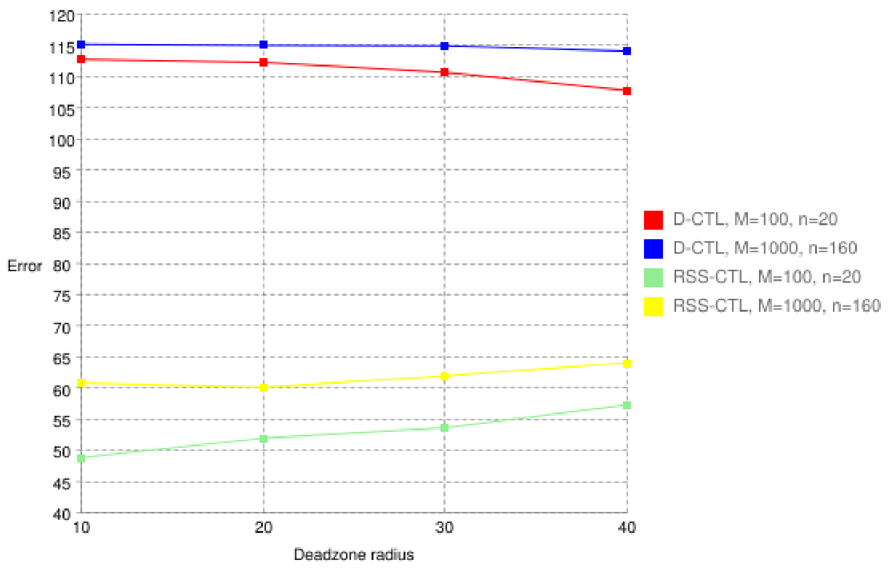


Figure 5.7: The chart displays how the average error does not change significatly when r_{dz} is increasing, for both the D-CTL and RSS-CTL algorithms. Each average error sample is based on the errors of 1000 generated cells. See section 4.2 for the definition of M. See section 4.2 and 4.4 for the definition of n. For example, the red line displays average error values for different values of r_{dz} when we run D-CTL with $n = 20$ and cells are generated with $M = 100$.

We use the following constant parameters for the chart in figure 5.8:

- $\alpha = 120°$
- $d_{extend} = 2$
- $r_{include} = 200$

As we can see from figure 5.7 and 5.8, the average error does not change significantly when we include larger and larger deadzones in the cell sectors. We therefore conclude that both D-CTL and RSS-CTL can estimate just as good cell tower locations when deadzones are present, as when deadzones are not present.

Notations Intruduced in This Section:

r_{dz}**:** The radius deciding the size of a deadzone.

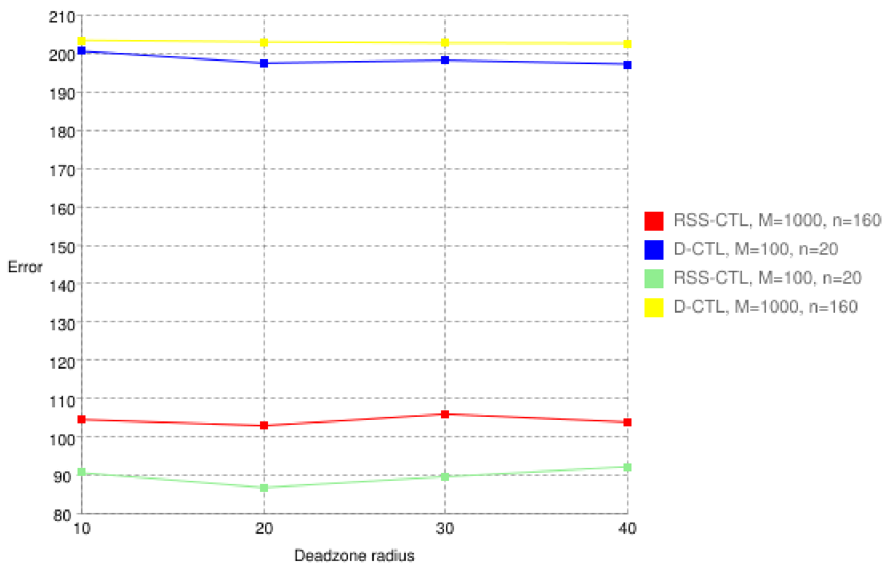


Figure 5.8: The chart displays how the average error does not change significatly when r_{dz} is increasing, for both the D-CTL and RSS-CTL algorithms. Each average error sample is based on the errors of 1000 generated cells. See section 4.2 for the definition of M. See section 4.2 and 4.4 for the definition of n. For example, the red line displays average error values for different values of r_{dz} when we run RSS-CTL with $n = 160$ and cells are generated with $M = 1000$.

5.9 Chapter Review

In this chapter we have step by step developed an algorithm for estimating cell tower locations based on the RSS values among each cell's measurements. In addition, we have tested the algorithm on different values of different parameters to get an understanding of how the algorithm scales. Based on the several tests we conclude that RSS-CTL estimates more accurate cell tower locations than D-CTL. In chapter 7 we will test both algorithms on real data. We want to keep testing D-CTL in addition to RSS-CTL as this can be an alternative if RSS is not available in the measurements.

Chapter 6

Understanding the OpenCellID Data Set

We have developed two algorithms for localizing cell towers: the D-CTL and RSS-CTL algorithms. Now we want to test them on real data. In this chapter we explain how we retrieve data from OpenCellID.org. We discuss the format and structure of the data, and how we select proper subsets to test the two algorithms on.

The subsets we test D-CTL and RSS-CTL on was downloaded April 4th 2015. As new measurements are added and since OpenCellID always strives to improve their data, the OpenCellID database is constantly updated.

6.1 Initiating Data Retrieval

To access the data we need an API-key, which is easily retrievable from the OpenCellID administrators. This key needs to be included every time we want access to data on OpenCellID.org. Obtaining the key is very simple. We only need to register on the site with our name and email.

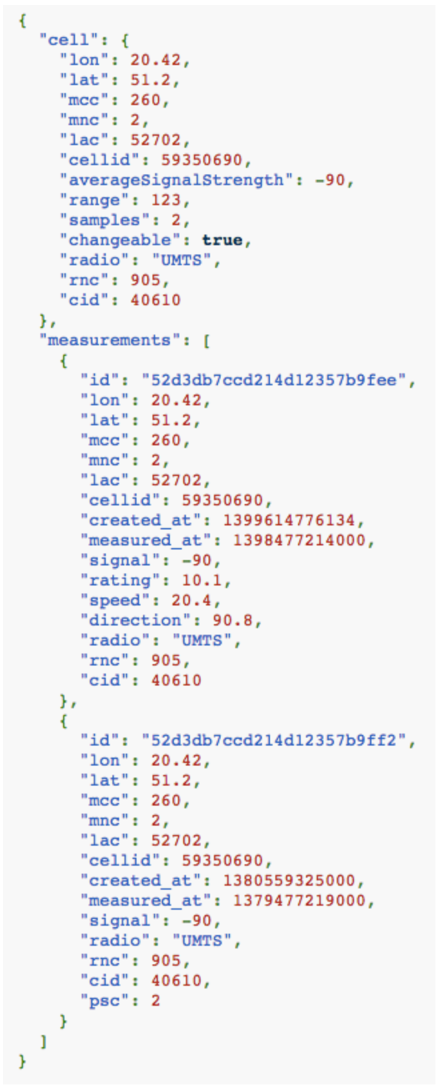
When we have the API-key there are two ways to access the data. The first way is to download a copy of the entire database to a computer piece by piece. The only restriction on this approach is a specific number of times a day we are allowed to download different pieces of the database for free. The number of times vary for the different pieces, but we can download all of the pieces at least once a day.

The second approach is performing HTTP GET or POST requests to the OpenCellID database. This approach lets us query for very specific parts of the database by adding parameters to the request. Those contributing to OpenCellID can do this as much as they want. Others need to pay for it. We were considered contributors for doing the research contained in this thesis.

We take advantage of both approaches. The first approach is handy when we want to compute statistics and get an overview of the data. For example finding out how many cells have been registered in Norway, or how many that use the broadcasting technology GSM worldwide. The second approach is more useful when we want to target small subsets of the database. Then we do not wish to traverse every piece of the entire database every time.

6.2 The Data Objects

The OpenCellID data is structured as follows. There are cell and measurement objects. Measurement objects are created from data collected by users, as mentioned in section 1.1. When a collected measurement is for a cell that has not been seen before, a new cell object is created. Several of the cell object's data fields are updated each time a new measurement for the cell is added. See figure 6.1 for an example of a cell with two measurements. Table 6.1 and 6.2 display the cell and measurement objects with corresponding data fields.



```
{
  "cell": {
    "lon": 20.42,
    "lat": 51.2,
    "mcc": 260,
    "mnc": 2,
    "lac": 52702,
    "cellid": 59350690,
    "averageSignalStrength": -90,
    "range": 123,
    "samples": 2,
    "changeable": true,
    "radio": "UMTS",
    "rnc": 905,
    "cid": 40610
  },
  "measurements": [
    {
      "id": "52d3db7ccd214d12357b9fee",
      "lon": 20.42,
      "lat": 51.2,
      "mcc": 260,
      "mnc": 2,
      "lac": 52702,
      "cellid": 59350690,
      "created_at": 1399614776134,
      "measured_at": 1398477214000,
      "signal": -90,
      "rating": 10.1,
      "speed": 20.4,
      "direction": 90.8,
      "radio": "UMTS",
      "rnc": 905,
      "cid": 40610
    },
    {
      "id": "52d3db7ccd214d12357b9ff2",
      "lon": 20.42,
      "lat": 51.2,
      "mcc": 260,
      "mnc": 2,
      "lac": 52702,
      "cellid": 59350690,
      "created_at": 1380559325000,
      "measured_at": 1379477219000,
      "signal": -90,
      "radio": "UMTS",
      "rnc": 905,
      "cid": 40610,
      "psc": 2
    }
  ]
}
```

Figure 6.1: A cell object with two measurement objects, given in JSON [7].

Data field	Description
radio	Network type. Either GSM, UMTS, LTE or CDMA.
mcc	Mobile Country Code.
net	Mobile Network Code (MNC) for GSM, UMTS and LTE. System IDentification number (SID) for CDMA.
area	Location Area Code (LAC) for GSM and UMTS. Tracking Area Code (TAC) for LTE. Network IDentification number (NID) for CDMA.
cell	Cell ID (CID) for GSM and LTE. UTRAN Cell ID/LCID for UMTS. Base station IDentifier number (BID) for CDMA.
unit	Primary Scrambling Code (PSC) for UMTS. Physichal Cell ID (PCI) for LTE. Empty for GSM and CDMA.
lon	Longitude in degrees between -180.0 and 180.0.
lat	Latitude in degrees between -90.0 and 90.0.
range	Estimated cell range, in meters.
samples	Total number of the cell's measurements.
changeable	If 1: The lon,lat values have been calculated from available measurements. If 0: The lon,lat values are correct - no measurements have been used to calculate it.
created	The first time the cell was seen and added to the database.
updated	The last time the cell was seen, and thus updated.
averageSignal	Average signal strength for all the cell's measurements.

Table 6.1: Overview of the OpenCellID Cell object, as defined at [1].

6.3 Overview of the Data

As we can see from table 6.1 and 6.2, both cells and measurements contain several data fields. To us, they are not all relevant.

Identification A cell is identified with four different values; *mcc*, *net*, *area* and *cell*. These tell us which country, which cellular network provider, the area within that country, and which ID belongs to the cell, respectively. Both cell and measurement objects have these four data fields. Each measurement belonging to a cell will have the same identification values. The only way to tell two measurements apart is by comparing for example *longitude* and *latitude* values, or *created* values. On the other hand, we never need to target one measurement individually. We are always iterating over a set of measurements. The longitude and latitude values of the measurements will clearly be very important to us when estimating cell tower locations.

Data field	Description
mcc	Mobile Country Code.
net	Mobile Network Code (MNC) for GSM, UMTS and LTE. System IDentification number (SID) for CDMA.
area	Location Area Code (LAC) for GSM and UMTS. Tracking Area Code (TAC) for LTE. Network IDentification number (NID) for CDMA.
cell	Cell ID (CID) for GSM and LTE. UTRAN Cell ID/LCID for UMTS. Base station IDentifier number (BID) for CDMA.
lon	Longitude in degrees between -180.0 and 180.0.
lat	Latitude in degrees between -90.0 and 90.0.
signal	Signal level in dBm or as defined in [14, section 8.5].
measured	When the measurement was registered.
created	When the measurement was added to the database.
rating	GPS quality/accuracy information in metres.
speed	Speed of the phone when the measurement was registered.
direction	Heading direction of the phone when the measurement was registered.
radio	Network type. Either GSM, UMTS, LTE or CDMA.
ta	Timing advance; only for GSM and LTE.
rnc	Radio Network Controller; only for UMTS.
cid	Cell ID (short); only for UMTS.
psc	Primary Scrambling Code; only for UMTS.
tac	Tracking Area Code; only for LTE.
pci	Physical Cell ID; only for LTE.
sid	System Identifier; only for CDMA.
nid	Network Identifier; only for CDMA.
bid	Base station ID; only for CDMA.

Table 6.2: Overview of the OpenCellID Measurement object, as defined at [1].

Known Correct Cell Tower Locations As mentioned in section 1.1, OpenCellID estimates the cell tower location for a cell by calculating the mean of the longitudes and latitudes of the cell's measurements. But there are some exceptions. The *changeable* data field tells us whether the cell tower location has already been computed using available measurements or not. If the changeable value is 0, the location given for the cell tower is correct. In this case OpenCellID has been given access to data about correct cell tower locations from certain cellular network providers in certain countries. The countries this concerns is Russia, Germany and Poland. We learned this by writing a piece of code that iterates over every cell in the OpenCellID database, checks the changeable value for ever cell, and stores the mcc value of every cell where this value is 0. Among the stored mcc values were only the ones for Russia, Germany and Poland. To know that certain cells contain the correct location of the cell tower will be very valuable to us when testing the D-CTL and RSS-CTL algorithms. This lets us compare our estimated cell tower locations to the correct cell tower locations so we can know to which degree the algorithms work.

6.3.1 Filtering the Data

We need a strategy for how to create subsets of the OpenCellID data to test the D-CTL and RSS-CTL algorithms on. As of January 2015, their database contained almost 7 million unique cell towers and 1.2 Billion crowdsourced measurements. This means we must rule out parts of the data we do not need, thus narrowing down to what we do need. When we have traversed the pieces of the downloadable database and found the cells we want to use, we will download their corresponding measurements through HTTP GET.

GSM To test D-CTL and RSS-CTL we need to choose subsets from the database to test them on. Considering the *radio* data field is a good first step. This field tells us if a cell uses either the GSM, UMTS, LTE or CDMA technology to broadcast it's signal. Table 6.3 shows how the entire dataset of available cells is distributed with respect to the radio data field. We learned these numbers by writing a piece of code that counts all the cells using GSM technology, counts all the cells using UMTS technology, and so on. As we can see from table 6.3, the OpenCellID database contains most cells that uses GSM technology. Since we have no deep knowledge about the data, we conclude that this type of cell is the most favorable type to test the D-CTL and RSS-CTL algorithms on.

radio	Number of cells
GSM	4,098,870
UMTS	3,342,507
LTE	32,782
CDMA	30,016
Total	**7,504,176**

Table 6.3: Distribution of OpenCellID cells over broadcasting technology.

Known Correct Cell Tower Locations We are also interested in cells where the data field changeable has the value 0, since these cell tower locations are correct. We wrote a piece of code that found the distribution of GSM cells, whose changeable values are 0, over the three countries Russia, Germany and Poland. The distribution is displayed in table 6.4.

Measurements per Cell Each cell object has a *sample* field. We will use this when downloading cells through HTTP GET. This field gives us control over the amount of measurements each cell has, and we can adjust our subsets of test cells

Country	Number of cells
Russia	48,444
Germany	41,374
Poland	235,121
Total	**324,939**

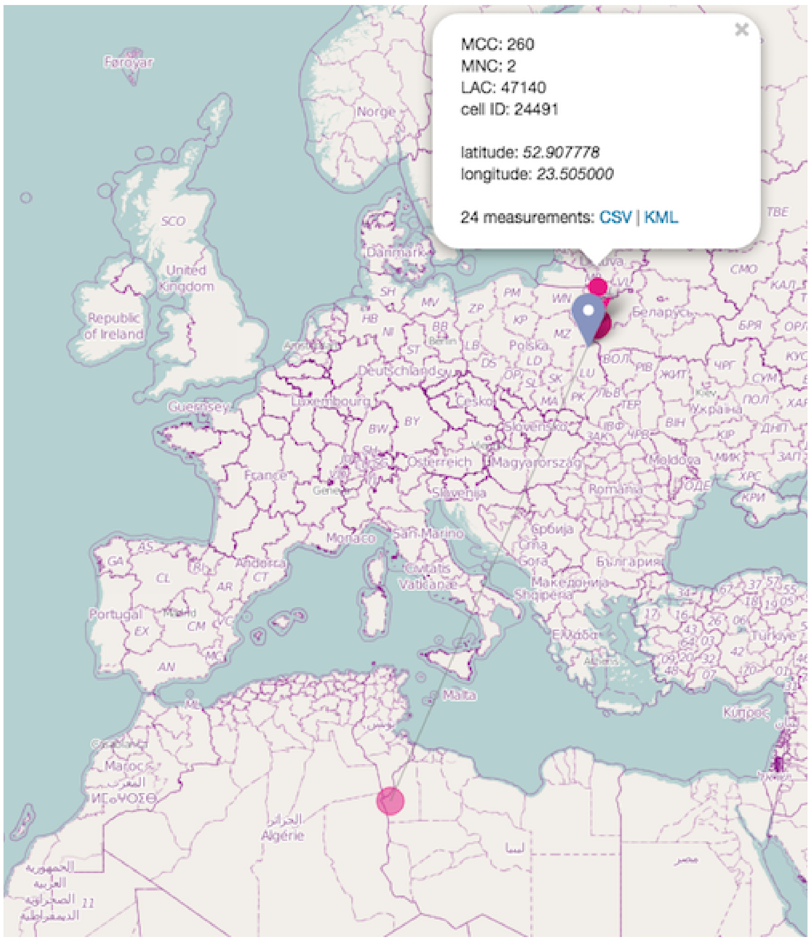
Table 6.4: Distribution of OpenCellID GSM cells per country whose changeable values are 0.

accordingly by configuring the parameters in the HTTP GET requests. When choosing the subsets to test the D-CTL and RSS-CTL algorithms on we keep a similar tactic as when we tested them on generated data in chapter 4 and 5. We want to see how they perform when different amounts of measurements are available per cell. One problem arised when downloading data by performing HTTP GET requests. The sample values of the cells in the downloadable pieces of the database were in many cases not equal to the actual number of measurements available when performing HTTP GET requests. For example, if a cell in the downloadable database had the sample value 100, the actual amount of available measurements for that cell when requesting them through HTTP was 500. This made it difficult to request cells with the intended amount of measurements. In addition to this, the maximum number of measurements that can be retrieved for a cell thrugh HTTP is 1000. When we request measurements for a cell that has more than 1000, we get back 1000 randomly chosen measurements.

6.3.2 Cleaning Up and Validating the Data

Before we use D-CTL and RSS-CTL on the data we need to clean it up due to the fact that the OpenCellID database contains a lot of bad measurements. When each measurement for a cell has been through the clean-up process we will also validate the complete cell to ensure it has the correct properties.

$r_{include}$ **Validates Distance from Measurement to Cell Tower** We look at the location of where the measurement was registered. A cell tower cannot broadcast infinitely far, and some measurements are too far away from the cell it belongs to for it to be valid. See for an example figure 6.2. That is why we re-introduce the parameter $r_{include}$. If a measurement is further away from the correct cell tower location than the value of $r_{include}$, we do not include it when running the D-CTL and RSS-CTL algorithms. When we do not know the correct cell tower location, we calculate the mean of the measurements' longitudes and latitudes, and use this geometric center of the measurements as origo for $r_{include}$. Macro-cell is the type of cell with the strongest broadcasting capabilities. The standard theoretical range for the macro-cell is 35 kilometers [26]. We use this distance as a maximum $r_{include}$ value when validating a measurement's distance to it's cell tower. We will also experiment with lower limits of this parameter.


MCC: 260
MNC: 2
LAC: 47140
cell ID: 24491

latitude: *52.907778*
longitude: *23.505000*

24 measurements: CSV | KML

Figure 6.2: A cell in Poland with one measurement in Libya

Received Signal Strength We need to validate each measurement's RSS. As described in table 6.2, the RSS of a measurement is in either dBm or a number between 0 and 31 (both included) as defined in [14, section 8.5]. The dBm value of cell signals are always negative. [14, section 8.5] defines a mapping from the negative dBm values to positive numbers. See table 6.5. As we can see from this table, RSS can have the positive values 0 to 31 in addition to negative dBm values. Measurements with RSS values larger than 31 will not be considered valid. When running the D-CTL and RSS-CTL algorithms on cells that have measurements with RSS values from 0 to 31, we translate these to dBm values. The formula is simple. If x is a positive RSS value not smaller than 0 and not larger than 31, the dBm value of x can be calculated like this: $(2 \times x) - 113$.

Received Signal Strength Values among a Cell's Measurements We need to consider the total number of different RSS values among every measurement belonging to a cell. If we are using RSS-CTL to compute an estimated cell tower location, we need at least two different RSS values among the measurements. See algorithm 6. Since the pairing of measurements in RSS-CTL is done randomly we

[14, section 8.5]	dBm
0	-113 dBm or less
1	-111 dBm
2...30	-109...-53 dBm
31	-51 dBm or greater
99	not known or not detectable

Table 6.5: Received Signal Strength comes in two formats. Here is the mapping between them.

may need more than two different RSS values, especially when n is small. That is why we use 3 as a threshold when validating the number of RSS values among a cell's measurements.

m_{min} and m_{max} We want to control the number of measurements per cell when estimating cell tower locations. That is why we operate with a minimum number m_{min} and maximum number m_{max} of measurements per cell when runing D-CTL and RSS-CTL on a set of cells.

Notations Intruduced in This Section:

m_{min},m_{max}: The minimum and maximum number of measurements per cell when runing D-CTL and RSS-CTL on a set of cells.

6.4 Test Data: Cells where Correct Cell Tower Locations are Known

For our test data of cells with correct cell tower locations, we downloaded 2036 of the 324,939 cells described in table 6.4, including corresponding measurements. We form the set $S_{correct}$ with these 2036 cells. The amount of measurements per cell ranges from approximately 70 to 1000.

We now summarize the validation procedure for a cell $\in S_{correct}$. For each measurement belonging to the cell, it is approved if the following two properties hold:

- the distance to the correct cell tower location is less than or equal to $r_{include}$.
- the RSS value is less than or equal to 31 (if it is greater than or equal to 0 and less than or equal to 31, we translate it to the corresponding dBm value).

After evaluating all the measurements belonging to the cell, the cell is approved if the following two properties hold:

- the total number of different RSS values among the measurements is greater than or equal to 3.

- the total number of measurements is not less than m_{min} and not greater than m_{max}.

Notations Intruduced in This Section:

$S_{correct}$**:** A set containing 2036 random real cells with correct cell tower locations.

6.5 Test Data: Cells where Correct Cell Tower Locations are Not Known - Bergen City Center

We want to estimate cell tower locations for a familiar area. To accomplish this task it was very natural for us to choose the Bergen City Center. To locate the cells corresponding to this area, we had to decide minimum and maximum longitude and latitude values, thereby creating a square around the Bergen City Center. Then we traversed all of the cells in the OpenCellID database, and located the ones within the square area. We tried to find a way to narrow down the Bergen City Center cells by using the *area* data field. See table 6.1. According to various blogposts, the different cellular network providers use different area code schemes, and usually do not publish these. In addition, the area codes change regularly. Thus we could not use the area data field to accomplish this.

The area confined within the square area resulted in 597 cells, with corresponding measurements. We form the set S_{bergen} with these cells. The amount of measurements per cell ranges from 1 to 1000. We use almost the same validation procedure as for correct cells in section 6.4.

We now summarize the validation procedure for a cell $\in S_{bergen}$. For each measurement belonging to the cell, it is approved if the following two properties hold:

- the distance to the geometric center of all measurements is less than or equal to $r_{include}$.
- the RSS value is less than or equal to 31 (if it is greater than or equal to 0 and less than or equal to 31, we translate it to the corresponding dBm value).

After evaluating all the measurements belonging to the cell, the cell is approved if the following two properties hold:

- the total number of different RSS values among the measurements is greater than or equal to 3.
- the total number of measurements is not less than 10 (D-CTL and RSS-CTL need a minimum amount of measurements to work properly).

Notations Intruduced in This Section:

S_{bergen}**:** A set containing 597 real cells located in the Bergen City Center.

6.6 Chapter Review

In this chapter we have described the real cell and measurement data provided by OpenCellID. We have discussed the process of how to retrieve it from OpenCellID's database, and what we must do with it before actually using it to test the D-CTL and RSS-CTL algorithms. The entire database is large, complex, and has many flaws. Understanding it's complexity and problems was very time-consuming.

Chapter 7

Running the D-CTL and RSS-CTL Algorithms on OpenCellID Data

We will now estimate cell tower locations of real cells by using D-CTL and RSS-CTL on the OpenCellID data subsets $S_{correct}$ and S_{bergen}. We define the following two primary goals for this chapter:

1. With the cells in $S_{correct}$, we want to test the accuracy of our two algorithms by comparing their estimated cell tower locations to the correct ones.

2. With the cells in S_{bergen}, we want to propose new estimated cell tower locations as alternatives to the existing suggested cell tower locations.

Even though we made strong arguments that RSS-CTL results in more accurate results than D-CTL in chapter 5, we still want to run D-CTL on real cells as this algorithm can be an alternative if RSS is not available in the measurements. We remind the reader that an overview and description of D-CTL and corresponding sub-routines can be found in table 4.1 and 4.2. An overview and description of RSS-CTL and corresponding sub-routines can be found in table 5.1 and 5.2.

We will compare the performance of our two algorithms with the algorithm currently used by OpenCellID. This computes a cell tower location by calculating the mean of the longitudes and latitudes of the cells' measurements. As we discussed in section 1.1 this approach is no good as the estimated cell tower will be located in the middle of the measurements. The cell tower should be located so that each measurement lies to some direction of the cell tower within a 120° cell sector, as described in figure 1.1. That is why we also will calculate the maximum number of a cell's measurements that can be fit within a 120° cell sector, when the cell tower location is estimated with the approach currently used by OpenCellID. When we show results we will provide the average percentage M_{miss} of the number of measurements that could not be fit within a 120° cell sector. The D-CTL and RSS-CTL algorithms are designed so that M_{miss} always is 0%.

Notations Intruduced in This Section:

M_{miss}**:** The average percentage of the number of measurements that could not be fit within a 120° cell sector, for multiple cells.

7.1 Parameters

There are several parameters we wish to scale when running D-CTL and RSS-CTL. These are the same paremeters we experimented with in chapter 4 and 5.

n In the sub-routines D-DL, D-CD, RSS-DL and RSS-CD, n plays important roles. See table 4.2 and 5.2. When running D-CTL and RSS-CTL we want to learn if small values of n result in as accurate estimated cell tower locations as large values of n.

d_{extend} In the sub-routine FS, $l_{direction}$ is extended by a distance d_{extend} until every measurement of a cell fits within the cell sector. When running D-CTL and RSS-CTL we want to learn if large values of d_{extend} result in as accurate estimated cell tower locations as small values of d_{extend}. When estimating cell tower locations for real cells, the values of d_{extend} are much lower than when we estimated cell tower locations for generated cells in chapter 4 and 5. The reason for this is the difference between a Cartesian coordinate system and longitude/latitude coordinates. For example, in a Cartesian coordinate system 0.001 is a very small distance, but in longitude/latitude coordinates it is 111.2 meters.

$r_{include}$ When validating measurements, we decline those that are further away than $r_{include}$ kilometers. See section 6.3.2. When running the D-CTL, RSS-CTL and OpenCellID algorithms we want to learn if using small values of $r_{include}$ result in more accurate estimated cell tower locations than using large values of $r_{include}$. Even though measurements far away from the correct cell tower location are completely valid, we want to exclude them from the computations to see how that affects the errors of estimated cell tower locations.

m_{min} **and** m_{max} When running the D-CTL, RSS-CTL and OpenCellID algorithms on real cells we want to learn if the accuracy of estimated cell tower locations differs when the cells have different amounts of measurements. See section 6.3.2. If $m_{min} = 100$ and $m_{max} = 200$, we run the D-CTL, RSS-CTL and OpenCellID algorithms on cells that have not less than m_{min} and not more than m_{max} measurements. In section 7.3, 7.4 and 7.5 we experiment with the set of real cells $S_{correct}$. In these sections we will use the following values of m_{min} and m_{max}: $m_{min} = 100/m_{max} = 200$, $m_{min} = 450/m_{max} = 550$ and $m_{min} = 900/m_{max} = 1000$. In section 7.6 we experiment with the set of real cells S_{bergen}. In this section we will estimate cell tower locations for each cell in the set that has not less than 10 measurements. This means we use $m_{min} = 10/m_{max} = 100$.

7.2 The Haversin Formula

Our planet has the shape of a sphere. To calculate distances from one point to another on the surface of a sphere, we need a bit more complex formula than the one we use with a regular Cartesion coordinate system. We will use the Haversin Formula [9, page 161]:

$$\text{haversin}(\tfrac{d}{r}) = \text{haversin}(\phi_2 - \phi_1) + \cos(\phi_1)\cos(\phi_2)\,\text{haversin}(\lambda_2 - \lambda_1)$$

where

- *haversin* is the haversin function: $\text{haversin}(\theta) = \sin^2(\frac{\theta}{2}) = \frac{1-\cos(\theta)}{2}$,
- **d is the distance between the two points,**
- r is the radius of the sphere,
- ϕ_1 and ϕ_2 is the latitude of point 1 and latitude of point 2, and
- λ_1 and λ_2 is the longitude of point 1 and longitude of point 2

We will not incorporate this directly into the D-CTL and RSS-CTL algorithms. When running the two algorithms on real cells we are imagining that we are dealing with a flat surface. But we will use this formula when calculating the errors and when excluding measurements with respect to $r_{include}$.

7.3 Testing on Cells where Correct Locations are Known: Constant d_{extend} and $r_{include}$

In this section we test our algorithms on cells where the correct cell tower locations are known. The purpose of these tests are to measure the accuracy of the estimated cell tower locations by comparing them to the correct locations. We keep d_{extend} and $r_{include}$ constant to get an understanding of how only different values of n affect the accuracy of the estimated cell tower locations. We will experiment with these parameters in section 7.4 and 7.5.

We run and compare the D-CTL, RSS-CTL and OpenCellID algorithms on $S_{correct}$. We use the following consant parameters:

- $d_{extend} = 0.001$
- $r_{include} = 35\text{km}$

n	D-CTL	RSS-CTL
10	7.9km	4.7km
20	8.1km	4.0km
40	7.8km	3.8km
80	7.6km	3.9km

(a) Overview of average errors when running the D-CTL and RSS-CTL algorithms.

OpenCellID	M_{miss}
2.8km	38.9%

(b) Overview of average error and M_{miss} when running the OpenCellID algorithm.

Table 7.1: Overview of average errors of estimated cell tower locations of cells in $S_{correct}$ when $m_{min} = 100$ and $m_{max} = 200$. These m_{min} and m_{max} values amounted to 241 cells.

n	D-CTL	RSS-CTL
10	13.8km	10.1km
20	14,3km	9.9km
40	14.5km	9.7km
80	14.2km	9.7km
160	13.5km	9.8km
320	13.2km	9.8km

(a) Overview of average errors when running the D-CTL and RSS-CTL algorithms.

OpenCellID	M_{miss}
2.2km	39.4%

(b) Overview of average error and M_{miss} when running the OpenCellID algorithm.

Table 7.2: Overview of average errors of estimating cell tower locations of cells in $S_{correct}$ when $m_{min} = 450$ and $m_{max} = 550$. These m_{min} and m_{max} values amounted to 266 cells.

n	D-CTL	RSS-CTL
10	16.4km	13.3km
20	17.3km	13.0km
40	17.6km	12.7km
80	17.2km	12.7km
160	16.6km	12.7km
320	16.1km	12.8km
640	15.8km	13.2km

(a) Overview of average errors when running the D-CTL and RSS-CTL algorithms.

OpenCellID	M_{miss}
2.7km	37.7%

(b) Overview of average error and M_{miss} when running the OpenCellID algorithm.

Table 7.3: Overview of average errors of estimated cell tower locations of cells in $S_{correct}$ when $m_{min} = 900$ and $m_{max} = 1000$. These m_{min} and m_{max} values amounted to 308 cells.

Analysis

The OpenCellID algorithm has the lowest average error for all three m_{min}/m_{max} intervals. See table 7.1, 7.2 and 7.3. The M_{miss} values tells us that this algorithm estimated cell tower locations that on average will not cover 38.9%, 39.4% and 37.7%, respectively.

As we can see there is a clear gap between the average error values of D-CTL and RSS-CTL. We can also see that the average errors increase when the amount of measurements per cell increases. This confirms our results from running the two algorithms on generated test data in chapter 4 and 5.

There are no strong correlation between the value of n and average error values of D-CTL, or n and average error values of RSS-CTL. In table 7.1a, 7.2a and 7.3a, the average errors for both D-CTL and RSS-CTL change very little with respect to the different values of n. There seems to be no pattern to predict whether the average error values will increase or decrease when scaling n. Though we might argue that both D-CTL and RSS-CTL have slightly lower average errors at maximum n than minimum n.

These unpredictable average error values from D-CTL and RSS-CTL tells us several things. First, our randomization procedure when computing $l_{direction}$ in algorithm 1 and 6 works. Even with a low n and a large amount of measurements per cell, the algorithms manage to compute viable direction lines. We can see this by looking at the small and unpredictable changes in average error as n increases. If the randomization procedure did not work, we would expect the average errors to be much higher for small values of n than for large values of n. Secondly, the average error values increase when m_{min} and m_{max} increase. This means that a larger number of measurements per cell results in larger error values. This again means that, on average, a cell with a large amount of measurements has them spread out on a larger area than a cell with a smaller amount of measurements. This conclusion is based on how much algorithm 3 has to extend $l_{direction}$ to be able to compute a cell sector every measurement will fit within.

The average error from the OpenCellID algorithm is also unpredictable as it decreases from table 7.1b to table 7.2b, but increases from table 7.2b to table 7.3b.

This unpredictability for all three algorithms tells us that the measurements are randomly distributed within their cells. There are no patterns explaining the distribution.

7.4 Testing on Cells where Correct Locations are Known: Scaling d_{extend} and Measuring Time

In this section we test our algorithms on cells where the correct cell tower locations are known. The purpose of these tests are to measure the accuracy of the estimated cell tower locations by comparing them to the correct locations. We will now experiment with different values of d_{extend} to get an understanding of how this parameter affects the accuracy of the estimated cell tower locations.

We run and compare the D-CTL and RSS-CTL algorithms with different values of d_{extend}, on $S_{correct}$. We use the following constant parameters:

- $r_{include} = 35\text{km}$

Using a too small value of d_{extend} is one of the factors that potentially can increase the time it takes to estimate a cell tower location due to the increased number of iterations in algorithm 3. We experimented with and discussed d_{extend} theoretically in section 4.6 and 5.6. We use the following five values of d_{extend}:

- $d_{extend} = 0.1$
- $d_{extend} = 0.01$
- $d_{extend} = 0.001$
- $d_{extend} = 0.0001$
- $d_{extend} = 0.00001$

We will also measure the average running times for D-CTL and RSS-CTL. To narrow down the amount of average running times we have to keep track of, we focus on the worst case. That is when the average running time will be largest. In this case the worst case is when we run the two algorithms with $d_{extend} = 0.00001$ and a large value of n.

For the following estimations of cell tower locations we will not use the OpenCellID algorithm. The parameter d_{extend} does not affect the cells, only the D-CTL and RSS-CTL algorithms, so the OpenCellID algorithm would produce exactly the same average errors as in section 7.3.

n	D-CTL	RSS-CTL
10	12.1ms	23.8ms
20	8.3ms	24.0ms
40	5.4ms	24.0ms
80	3.1ms	23.2ms

Table 7.4: Overview of average running times for D-CTL and RSS-CTL when $d_{extend} = 0.00001$. These are for $m_{min} = 100/m_{max} = 200$.

Analysis

In the charts in figure 7.1, 7.2 and 7.3, the average errors for $d_{extend} = 0.01$ and $d_{extend} = 0.00001$ is approximately equal. This means using $d_{extend} = 0.01$ will result in just as accurate estimated cell tower locations as when using $d_{extend} = 0.00001$. This is very relevant with regard to time constraints. We want the computation of a cell tower location to take as little time as possible, so we wish to use the largest value of d_{extend} as possible. In table 7.4 and 7.5, the average running times

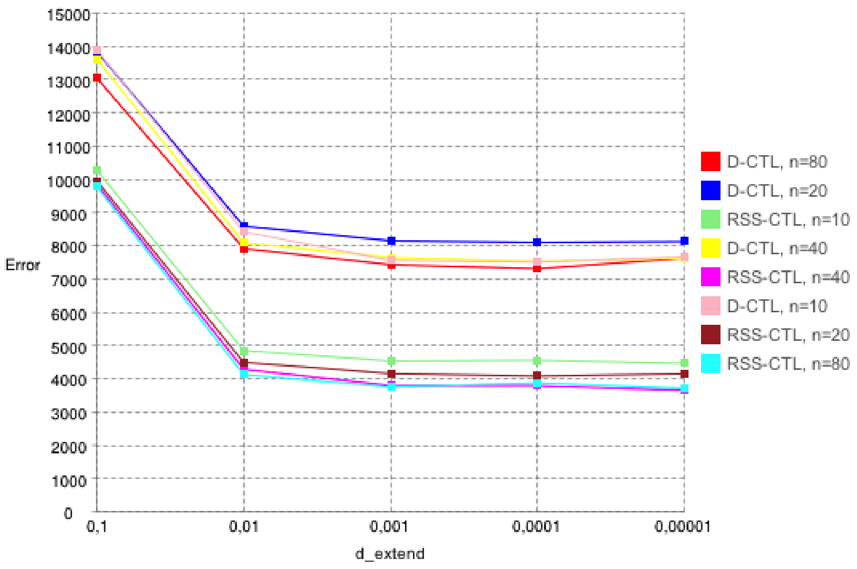


Figure 7.1: Overview of average errors of estimated cell tower locations of cells in $S_{correct}$, when $m_{min} = 100$ and $m_{max} = 200$. These m_{min} and m_{max} values amounted to 241 cells.

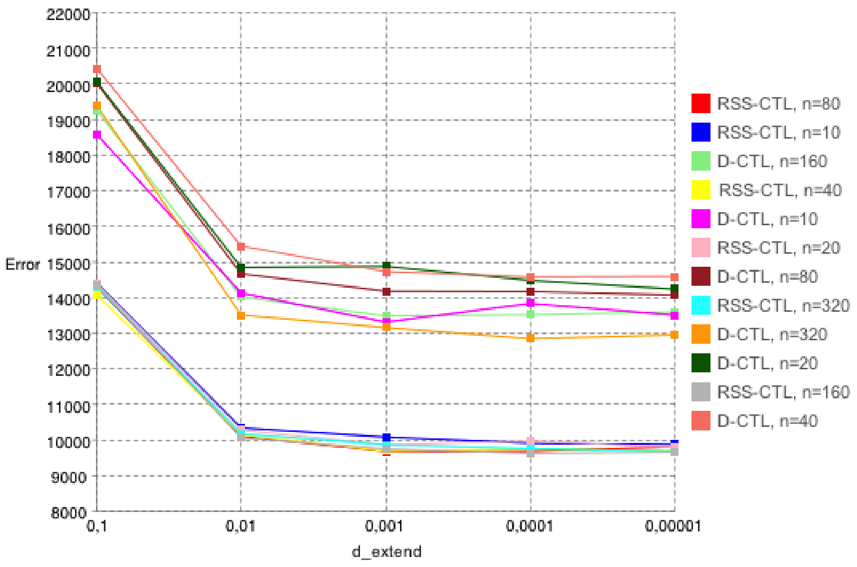


Figure 7.2: Overview of average errors of estimated cell tower locations of cells in $S_{correct}$, when $m_{min} = 450$ and $m_{max} = 550$. These m_{min} and m_{max} values amounted to 266 cells.

for $d_{extend} = 0.00001$ are not large enough for us to consider using a larger value of d_{extend}. But in table 7.6a and 7.6b the average running times for $d_{extend} = 0.00001$ is getting high. That is why we will use $d_{extend} = 0.0001$ for further computations of estimated cell tower locations with D-CTL and RSS-CTL.

In the charts in figure 7.2 and 7.3, the average errors from running D-CTL are spread out more than in the chart in figure 7.1, with respect to *Error*. We cannot see

n	D-CTL	RSS-CTL
10	87.9ms	130.3ms
20	67.3ms	125.9ms
40	39.7ms	127.4ms
80	33.2ms	129.6ms
160	25.2ms	129.0ms
320	17.8ms	128.3ms

Table 7.5: Overview of average running times for D-CTL and RSS-CTL when $d_{extend} = 0.00001$. These are for $m_{min} = 450/m_{max} = 550$.

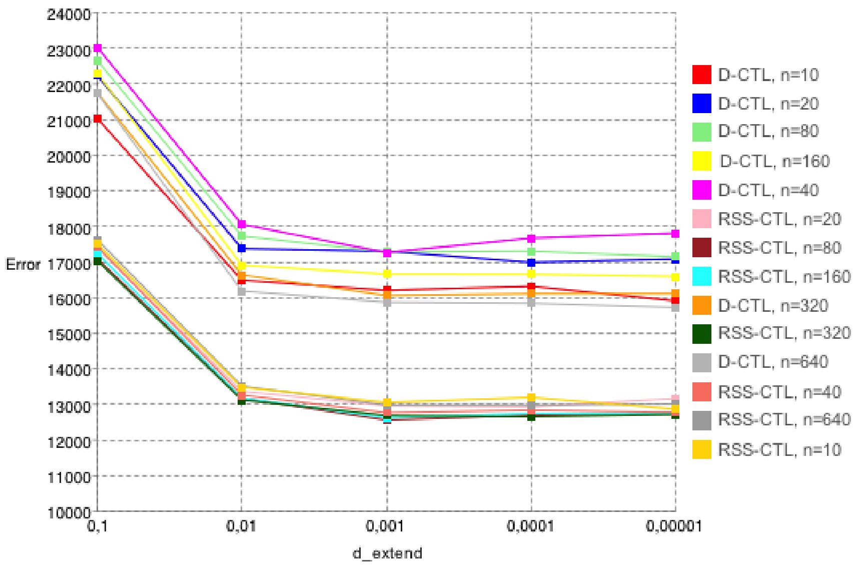


Figure 7.3: Overview of average errors of estimated cell tower locations of cells in $S_{correct}$, when $m_{min} = 900$ and $m_{max} = 1000$. These m_{min} and m_{max} values amounted to 308 cells.

$n \backslash d_{extend}$	0.00001	0.0001
10	428.9ms	42.0ms
20	285.8ms	30.9ms
40	195.8ms	20.1ms
80	146.4ms	15.3ms
160	120.3ms	13.4ms
320	109.9ms	12.6ms
640	66.2ms	17.6ms

(a) Average running times for D-CTL.

$n \backslash d_{extend}$	0.00001	0.0001
10	537.0ms	56.3ms
20	508.5ms	50.6ms
40	505.3ms	51.2ms
80	526.9ms	49.9ms
160	507.8ms	53.0ms
320	536.8ms	56.2ms
640	522.7ms	63.9ms

(b) Average running times for RSS-CTL.

Table 7.6: Overview of average running times for D-CTL and RSS-CTL when $d_{extend} = 0.00001$ and $d_{extend} = 0.0001$. These are for $m_{min} = 900/m_{max} = 1000$.

any pattern where the lowest n results in the highest average error and the largest n results in the lowest average error, or vice versa. This means the spread simply must be caused by the heuristic choices throughout the D-CTL and RSS-CTL algorithms.

7.5 Testing on Cells where Correct Locations are Known: Scaling $r_{include}$

In this section we test our algorithms on cells where the correct cell tower locations are known. The purpose of these tests are to measure the accuracy of the estimated cell tower locations by comparing them to the correct locations. We will now experiment with different values of $r_{include}$ to get an understanding of how this parameter affects the accuracy of the estimated cell tower locations.

We run and compare the D-CTL, RSS-CTL and OpenCellID algorithms with different values of $r_{include}$, on $S_{correct}$. We use the following constant parameters:

- $d_{extend} = 0.0001$

How will the average errors look if we use values lower than 35 kilometers for $r_{include}$? Recall that the value of $r_{include}$ decides if a measurement is too far away from the correct cell tower location or not, to be included when estimating the cell tower location. Initially it was to help exclude invalid measurements as figure 6.2 shows an example of. Forcing the D-CTL and RSS-CTL algorithms to exclude measurements that are valid but just far away from the cell tower, will most likely result in lower average errors. This is because algorithm 3 will not need to extend $l_{direction}$ as far to be able to fit the remaining measurements. On the other hand, the amount of available measurements to compute $l_{direction}$ from, in the first place, is reduced just as much. We experimented with and discussed $r_{include}$ theoretically in section 4.7 and 5.7. We use the following five values of $r_{include}$:

- $r_{include} = 35\text{km}$
- $r_{include} = 25\text{km}$
- $r_{include} = 15\text{km}$
- $r_{include} = 10\text{km}$
- $r_{include} = 5\text{km}$
- $r_{include} = 2\text{km}$

We will not time these computations since we can be sure we will not exceed the running times given in section 7.4, as we are not lowering d_{extend}.

For these computations, the amount of cells used will vary for each reduction of $r_{include}$. This is because we are excluding more and more of each cells' measurements the lower $r_{include}$ gets. This means for example that a cell satisfying m_{min} and m_{max}

at $r_{include} = 35$ kilometers might not at $r_{include} = 25$ kilometers. The number of cells used to compute estimated cell tower locations for each value of $r_{include}$ is given in parenthesis below the value of $r_{include}$ in the charts.

Since the amount of measurements for a cell is changing when $r_{include}$ decreases, the average errors from running the OpenCellID algorithm is also changing. The line in the graph called *OpenCellID* shows the average errors from running the OpenCellID algorithm for each value of $r_{include}$. This also means the value of M_{miss} changes. The values of M_{miss} for each value of $r_{include}$ for all three m_{min}/m_{max} intervals are shown in table 7.7.

$r_{include} \backslash m_{min} - m_{max}$	100-200	450-550	900-1000
35km	38.9%	39.4%	37.7%
25km	39.2%	40.0%	38.6%
15km	39.4%	41.1%	38.8%
10km	39.6%	42.6%	38.5%
5km	40.9%	44.1%	37.3%
2km	43.1%	43.8%	39.5%

Table 7.7: M_{miss} for each value of $r_{include}$ when $m_{min} = 100$ and $m_{max} = 200$, $m_{min} = 450$ and $m_{max} = 550$ and $m_{min} = 900$ and $m_{max} = 1000$.

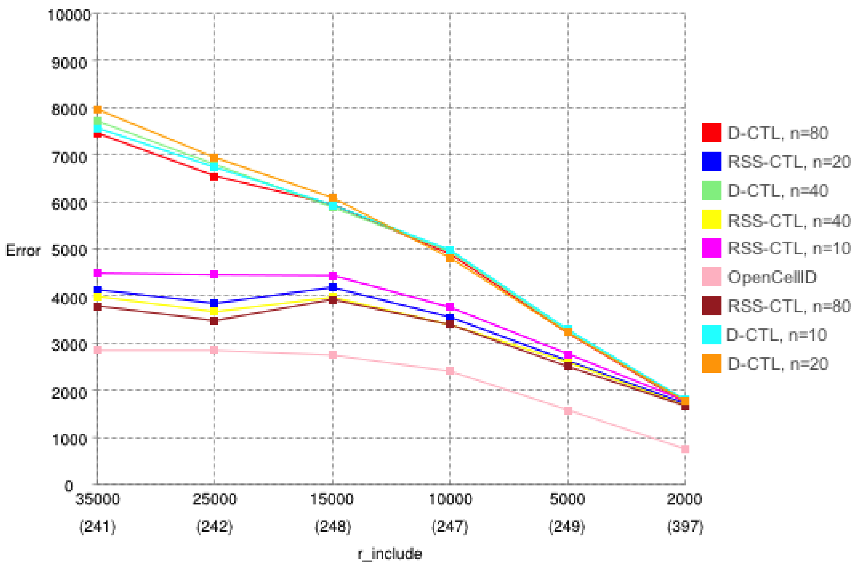


Figure 7.4: Overview of average errors of estimated cell tower locations of cells in $S_{correct}$, when $m_{min} = 100$ and $m_{max} = 200$.

Analysis

For some values of $r_{include}$, the number of cells used to compute average errors is lower than preferred, especially for $r_{include} = 2$ kilometers in the chart in figure 7.6. Recall

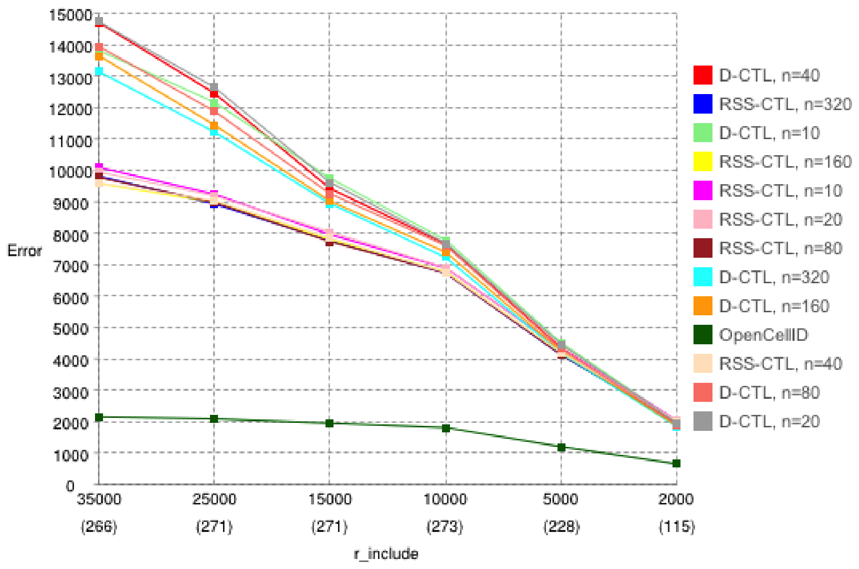


Figure 7.5: Overview of average errors of estimated cell tower locations of cells in $S_{correct}$, when $m_{min} = 450$ and $m_{max} = 550$.

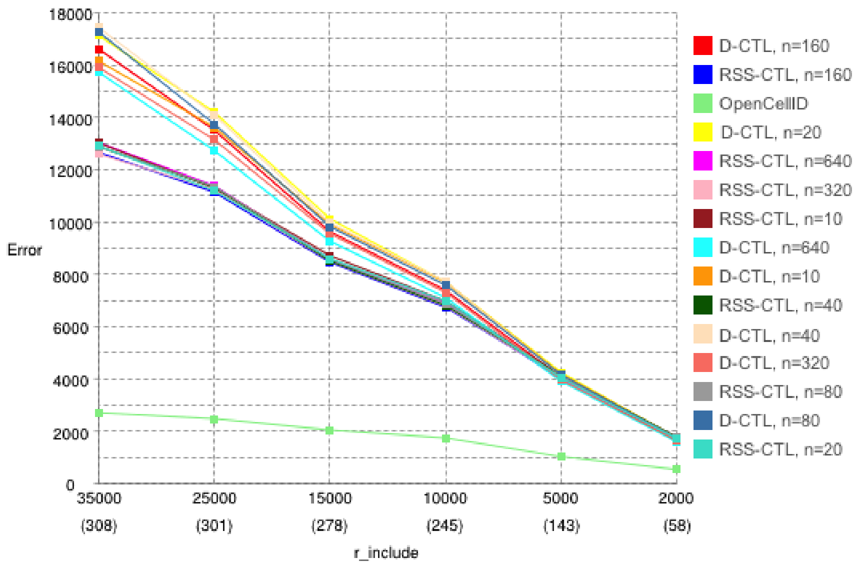


Figure 7.6: Overview of average errors of estimated cell tower locations of cells in $S_{correct}$, when $m_{min} = 900$ and $m_{max} = 1000$.

that the maximum number of measurements we can download for one cell through HTTP is 1000, even if OpenCellID has more measurements for that cell in their database. When we are running D-CTL and RSS-CTL with $r_{include} = 2$ kilometers and $m_{min} = 900$ we are dependent of cells that have 900 or more measurements within a radius of 2,000 meters from the correct cell tower location. As we can see from figure 7.6, only 58 of the cells in $S_{correct}$ had that property. On the other hand, the average error graphs in all three charts in figure 7.4, 7.5 and 7.6 behave similar with respect to *Error*. Thus we can argue that 58 cells was enough.

The average errors computed throughout this section are giving us very valuable information. Just as expected there is a gap between the average errors from D-CTL and RSS-CTL for $r_{include} = 35$ kilometers. We already knew this from our findings in section 7.3 and 7.4. What we could only project, but not know, is that the graphs for the two different algorithms would converge for lower values of $r_{include}$. This means that a lot of the pairs of measurements we choose when computing $l_{direction}$ in D-DL with $r_{include} = 35$ kilometers, form lines between them that are far from projecting the actual directions of the cells. We discussed this in theory in section 4.8. When we reduce $r_{include}$ it seems that we prevent D-DL from choosing two measurements that form such a poor direction line. Or we have excluded all of the measurements that could cause such a poor direction line.

Reducing $r_{include}$ results in lower average errors which are better results if we want to estimate cell tower locations as close as possible to the correct cell tower locations. This is essential if the goal is to use the estimated cell tower locations in other applications, such as localizing mobile devices. See section 1.1.1 and chapter 2. Now a dilemma occurs. Is it correct to exclude measurements to produce low errors? First we must consider the fact that the purpose of D-CTL and RSS-CTL is to estimate the location of cell towers we do not know the correct locations of. So how would we know if a measurement is far away from the cell tower or not, in that case? One possibility is to look at the signal strength, but this might not be a safe parameter. RSS can be affected by obstacles like buildings, other signals flowing through the air, or even the signal receiver in the mobile device. In short, this parameter may lie. Another possibility is to calculate the mean of the longitudes and latitudes of each cells' measurements, and use this as an origo for excluding measuremens that are too far away.

But do we want to exclude measurements from the computations? We must remember that we are developing heuristics. In most cases it is very difficult to say how far away our estimated cell tower locations are from the correct ones. We must base our results and conclusions on the data that is available to us. If we manage to estimate a cell tower location so that every measurement belonging to the cell are not too far away from this location, and fits within the cell sector, our heuristic works. See the rules in section 3.1. We can still calculate the distance from each measurement to the estimated cell tower location after this has been computed. If some measurements seems to be too far away from the estimated cell tower location, we can exclude them and run the algorithm again.

7.6 Bergen City Center: Estimating Cell Locations when These are Not Known

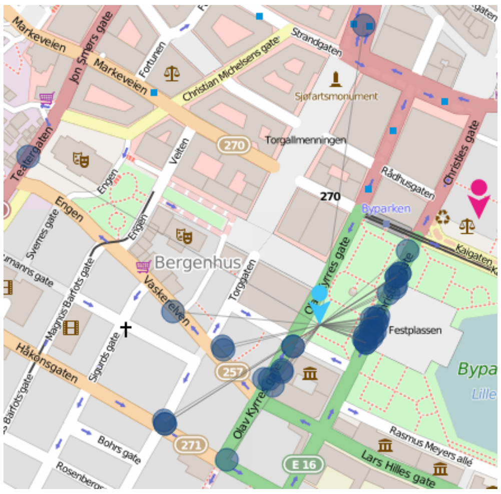
In this section we run RSS-CTL on cells where the correct cell tower locations are not known. The purpose of these tests are to suggest alternative cell tower locations to the suggested locations that already exist for the test cells. We will suggest new cell tower locations for the cells in S_{bergen}.

We use the following constant parameters:

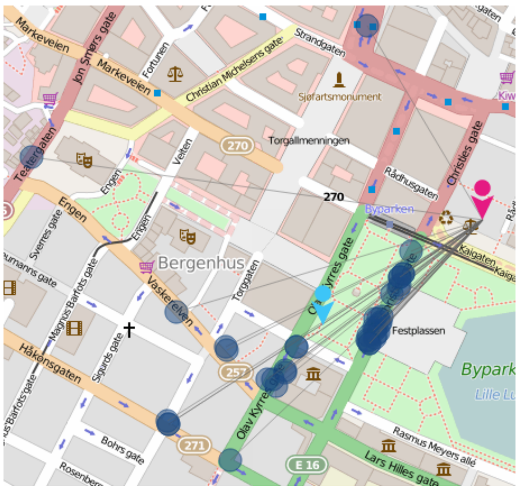
- $m_{min} = 10$
- $m_{max} = 1000$
- $d_{extend} = 0.0001$
- $r_{include} = 35\text{km}$
- if a cell's amount of measurements $|M|$ is equal to or larger than 80, $n = 80$
- if a cell's amount of measurements $|M|$ is less than 80, $n = |M|$

In section 6.5 we described S_{bergen}. The number of cells in this set is 597. After running the validation procedure on each of the 597 cells, we are left with 120 cells. This is mostly because many of the cells has less than 10 measurements. The upside of this is that the map of Bergen will not be overly crowded with cell tower locations.

Analyzing Two Estimated Cell Tower Locations



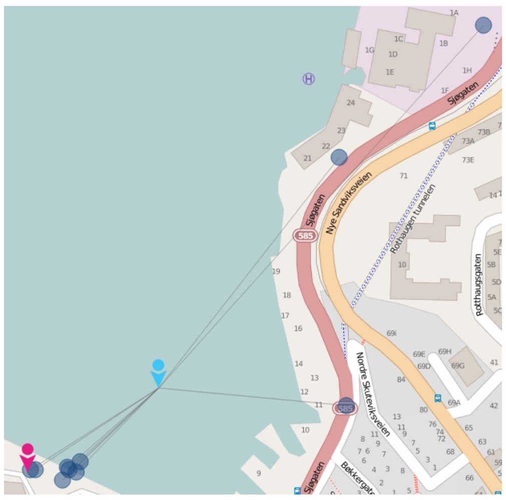
(a) The focus is on the cell tower location as estimated by OpenCellID.



(b) The focus is on the cell tower location as estimated by RSS-CTL.

Figure 7.7: Cell 242-1-11011-12302. The blue circles represent measurements. The light blue marker represents the cell tower location as estimated by OpenCellID. The pink marker represents the cell tower location as estimated by RSS-CTL.

In figure 7.7a and 7.7b we can see one of the cells from S_{bergen}. Measurements, estimated cell tower location by OpenCellID and estimated cell tower location by RSS-CTL are shown. There is really no way of knowing how accurate the location estimated by RSS-CTL is. But if we look closely, we can see that it estimated the cell tower to be located just where *Gulating Lagmannsrett* is located. This is a large building that contains a court of law. A rooftop seems like an excellent spot for a cellular network provider to place a cell tower, because of the hight. The cell tower location estimated by OpenCellID is much less likely. The green area where OpenCellID estimated the location to be is a lawn, with no high structures. We therefore conclude that RSS-CTL estimated a likely cell tower location for this cell.


(a) The focus is on the cell tower location as estimated by OpenCellID.

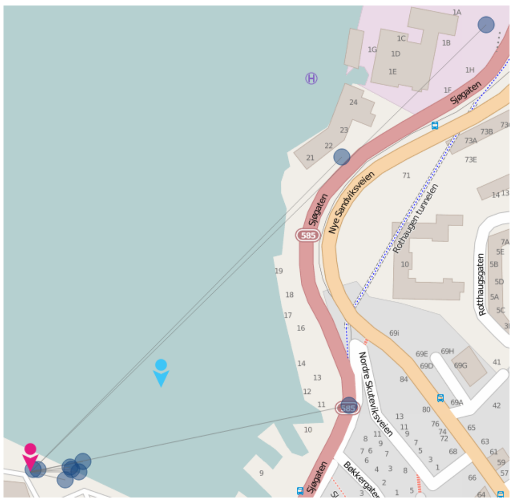

(b) The focus is on the cell tower location as estimated by RSS-CTL.

Figure 7.8: Cell 242-2-11011-40768. The blue circles represent measurements. The light blue marker represents the cell tower location as estimated by OpenCellID. The pink marker represents the cell tower location as estimated by RSS-CTL.

In figure 7.8a and 7.8b we can see another cell from S_{bergen}. OpenCellID estimated the cell tower location of this cell to be in the middle of the water. That is not an accurate estimation. We can see that RSS-CTL estimated the cell tower location of this cell to be on dry land, which makes it a more likely location.

Estimating Cell Towers for the Bergen City Center

Figure 7.9 shows a map of the Bergen City Center. Not all of the 120 estimated cell tower locations are shown. Some are outside of the figure boundaries. We chose to not show the old cell tower locations estimated by OpenCellID as that would have made the map very crowded. In addition, it would have been very difficult to clarify which new and old estimated cell tower locations belonged to the same cell.

We immediately see that some of the cell tower locations estimated by RSS-CTL cannot be very accurate as they are located in water areas. There may be several reasons for this, one being the algorithm itself. Based on the available measurements, RSS-CTL has not been able to estimate a valid cell tower location. Another possible reason is the distribution of the measurements belonging to each cell. It is clear that RSS-CTL works best when a cell's measurements are randomly distributed within the cell sector. If the measurements are distributed in any other way, that affects the accuracy of the estimated cell tower location. Especially combined with a low

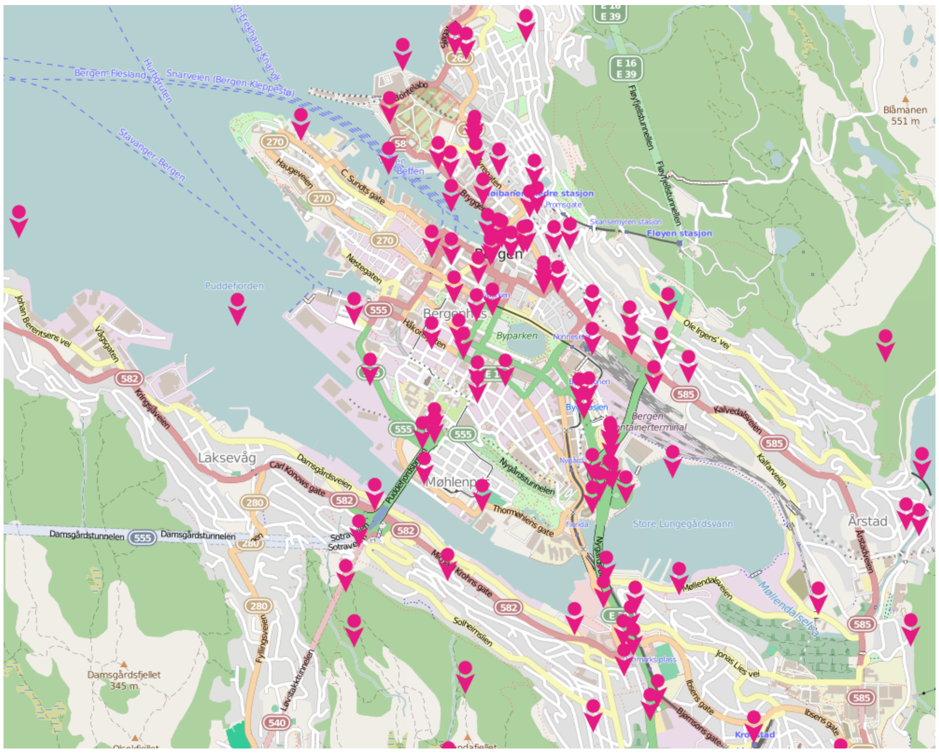


Figure 7.9: Overview of 97 estimated cell tower locations of cells in S_{bergen}. The estimated cell tower locations are computed by RSS-CTL.

number of measurements. Last, RSS-CTL base it's estimated cell tower location on RSS values among measurements. If these values do not reflect the distance each measurement is located from the cell tower, that will affect the estimated cell tower location.

7.7 Chapter Review

In this chapter we have presented and compared average errors of estimated cell tower locations of real cells, computed by the D-CTL, RSS-CTL and OpenCellID algorithms. We have experimented with different values for several parameters to learn how the algorithms scale. We have also used RSS-CTL to estimate cell tower locations in the Bergen City Center. To improve the D-CTL and RSS-CTL algorithms we need to analyze some cells and consider some of the more regular measurement distribution patterns. We also need to pay attention to the individual measurements that are causing high errors. These are probably the ones furthest away from the correct cell tower locations. Or the ones far from the other measurements belonging to the same cell, possibly close to the cell edges.

Chapter 8

Analysis and Improvements

In this chapter we analyze two cells with different measurement distribution patterns. We run the D-CTL, RSS-CTL and the OpenCellID algorithms on both cells and anylyze their estimated cell tower locations. Next we propose how D-CTL and RSS-CTL can be improved. We end the chapter by concluding the thesis.

8.1 Analyzing Measurement Distributions

In chapter 7 we presented average errors of estimated cell tower locations from running the D-CTL, RSS-CTL and OpenCellID algorithms. We ask the following question: What is affecting the average errors of estimated cell tower locations? We will now analyze the measurement patterns and estimated cell tower locations of two different cells with correct cell tower locations. One where the measurements are scattered and one where they are more structured.

The following parameters are used when running the algorithms on the two cells:

- $n = 80$
- $d_{extend} = 0.0001$
- $r_{include} = 35$km

8.1.1 Pattern 1: Cell with Scattered Measurements

We analyze the following cell, which is displayed in figure 8.1:

MCC: 262

NET: 7

AREA: 30605

CELL: 342

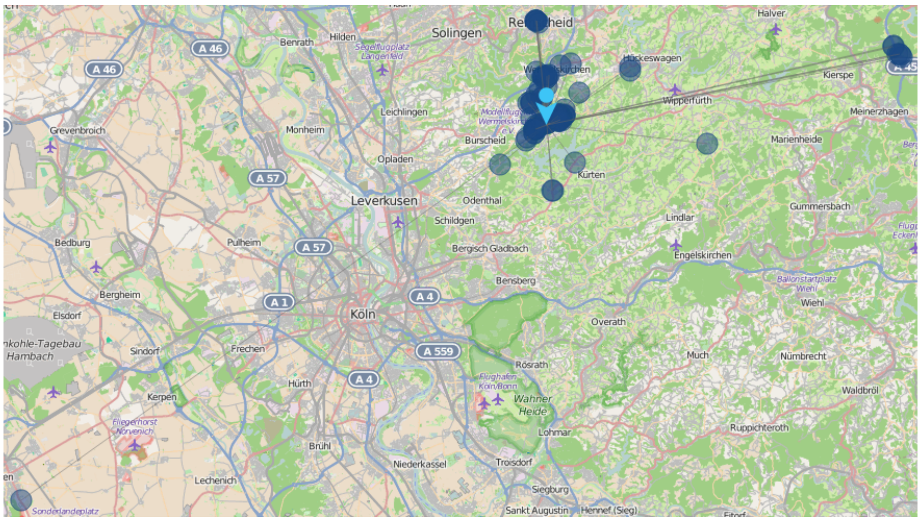


Figure 8.1: Cell 262-7-30605-342.

Measurements Analysis As we can see from figure 8.1, the measurements of this cell is scattered to all directions of the correct cell tower location, which is represented by the light blue marker. We immediately notice the measurement far away from the cell tower, in the lower left corner. This measurement is approximately 60 kilometers away from the correct cell tower location. Assuming that $r_{include}$ is active, with maximum value equal to 35 kilometers, this measurement will be marked as invalid and will not be considered when running D-CTL, RSS-CTL or the OpenCellID algorithm. See section 6.3.2. The measurements in the upper right corner is approximately 32 kilometers away from the correct cell tower location, and will be considered when $r_{include} = 35$ kilometers.

D-CTL

We run D-CTL on the cell. The result is displayed in figure 8.2, where the pink marker represents the estimated cell tower location. The error is 33.5 kilometers.

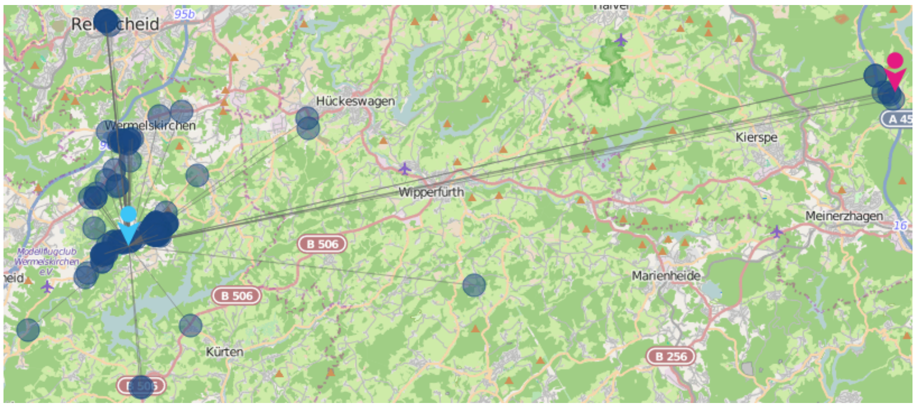


Figure 8.2: Cell 262-7-30605-342. D-CTL has estimated a cell tower location.

By examining the estimated cell tower location we can see that the sub-routine D-DL has computed a $l_{direction}$ between the two measurements furthest apart from each other. After this the sub-routine FS computed two cell sectors. The sub-routine D-CD has chosen one of the two cell sectors with corresponding cell tower location, but the wrong one. We designed D-CD so it would pick the cell sector whose cell tower location was closest to the correct cell tower location. It clearly did not do that. Figure 8.3 shows both of the two estimated cell tower locations computed by FS. In theory, the one to the left in the figure was supposed to be chosen by D-CD. Why was it not?

To answer this question we must look into D-CD. This sub-routine assumes that the measurements are randomly distributed within the correct cell sector. In this cell, they are not. Look at the estimated cell tower location to the right in figure 8.3. The little cluster of measurements at that location is really close to that estimated cell tower location. The rest of the measurements are closer to the estimated cell tower location to the left, and more spread out. Say we calculate the mean of the least possible distances from the few mesurements closer to the estimated cell tower location to the right in the figure, to $l_{direction}$. Then we calculate the mean of the least possible distances from the mesurements closer to the estimated cell tower location to the left in the figure, to $l_{direction}$. It is very likely that the former mean is lesser than the latter. That is why D-CD fail.

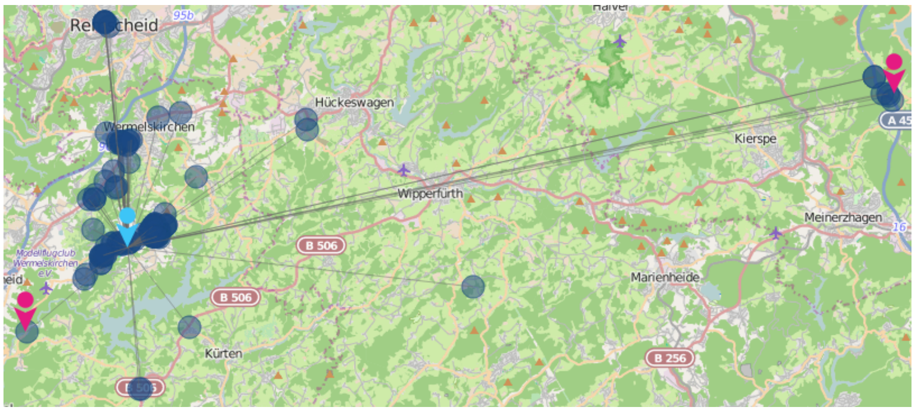


Figure 8.3: Cell 262-7-30605-342. We show both estimated cell tower locations computed by FS during the execution of D-CTL.

RSS-CTL

We run RSS-CTL on the cell. The result is displayed in figure 8.4 where the pink marker represents the estimated cell tower location. The error is 22.6 kilometers. Here, RSS-DL computed a completely different $l_{direction}$ than D-DL. We can see that the two sub-routines computed two versions of $l_{direction}$ that are close to be perpendicular to each other.

RSS-CTL worked as it was supposed to. RSS-DL found the two measurements with the largest difference in RSS. Then two possible estimated cell sectors with

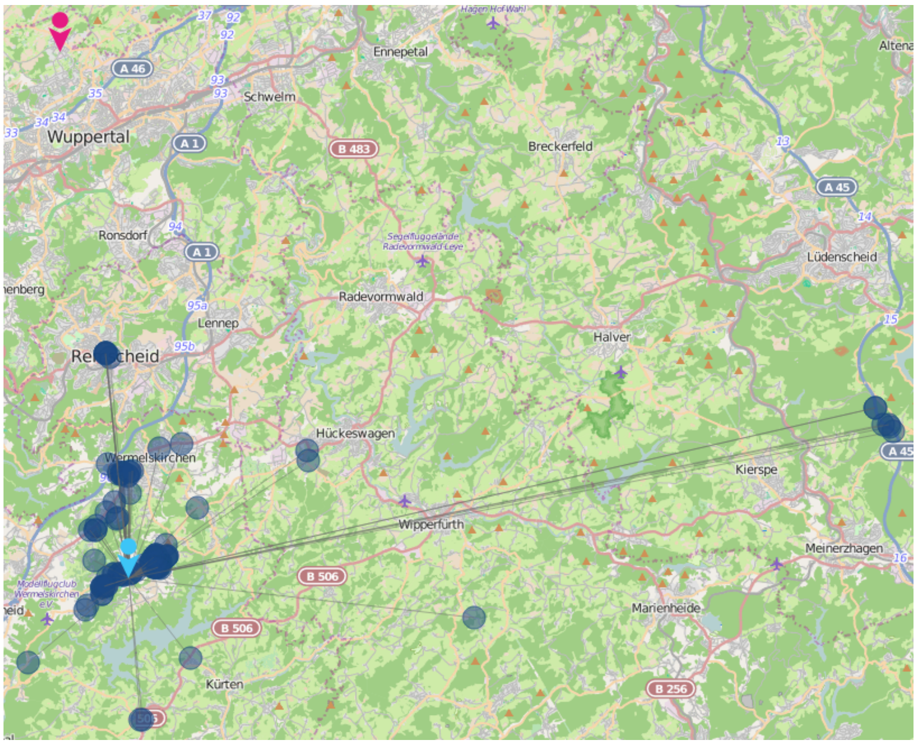


Figure 8.4: Cell 262-7-30605-342. RSS-CTL has estimated a cell tower location.

corresponding cell tower locations were computed by FS. One of them were chosen by RSS-CD. But the error is still very large. Why is that?

Figure 8.5 shows both of the two estimated cell tower locations computed by FS. We clearly see that RSS-CD in fact chose the estimated cell tower location closest to the correct cell tower location. The reason for the large error can be found by examining the measurements. After RSS-DL computed $l_{direction}$ in the current way, FS had to extend it by a large distance to be able to fit the group of measurements far to the right in figure 8.5, within a 120° cell sector. If RSS-DL had computed a $l_{direction}$ looking more like the one D-DL computed in figure 8.3, the error would have been decreased.

OpenCellID Algorithm

We run the OpenCellID algorithm on the cell. The result is displayed in figure 8.6 where the pink marker represents the estimated cell tower location. The error is 2.6 kilometers.

It is not strange this approach gave the estimated cell tower location with the lowest error. The measurement distribution of this cell, where they are scattered to every direction of the correct cell tower location, is perfect for this kind of estimation approach.

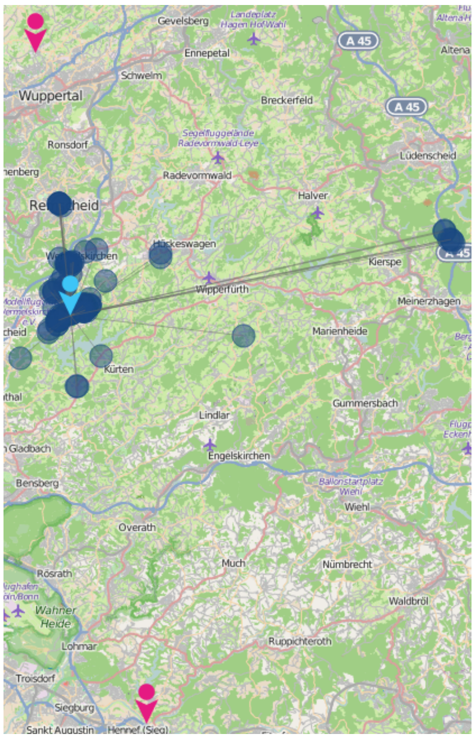


Figure 8.5: Cell 262-7-30605-342. We show both estimated cell tower locations computed by FS during the execution of RSS-CTL.

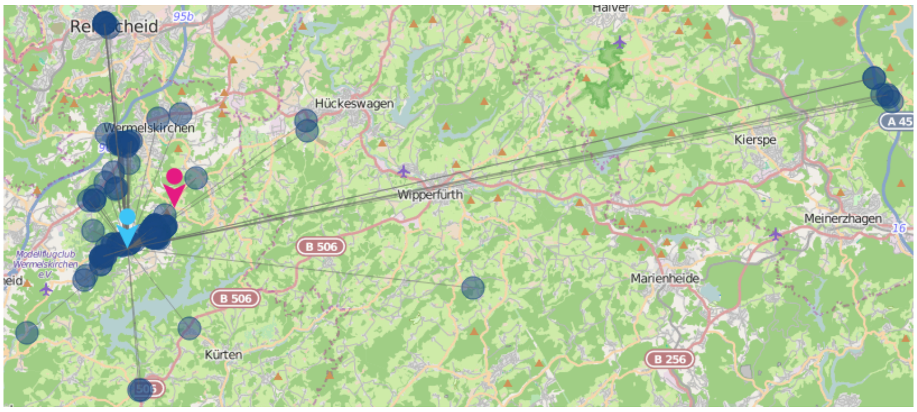


Figure 8.6: Cell 262-7-30605-342. The OpenCellID algorithm has estimated a cell tower location.

Conclusion

This cell is clearly omnidirectional. We draw this conclusion from the fact that the measurements are localized in every direction from the correct cell tower location, and not within a 120° cell sector. The D-CTL and RSS-CTL algorithms were not designed to estimate cell tower locations for this kind of cell. Estimated cell tower locations from these two algorithms on this kind of cell will very often have large errors. For omnidirectional cells, Weighted Centroid is a better alternative. See section 2.2.1.

8.1.2 Pattern 2: Cell with Structured Measurements

We analyze the following cell, which is displayed in figure 8.7:

MCC: 260

NET: 1

AREA: 29001

CELL: 22095

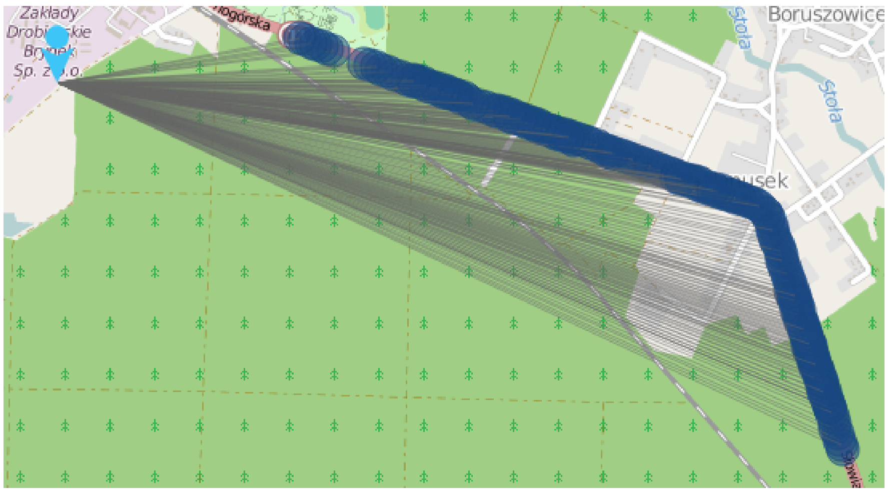


Figure 8.7: Cell 260-1-29001-22095.

Measurement Analysis As we can see from figure 8.7, the measurements of this cell are more structured than the cell analyzed in section 8.1.1. They are clearly within a 120° cell sector from the correct cell tower location, which is represented by the light blue marker. This cell almost appear forged by the way these measurements form an almost straight path.

D-CTL

We run the D-CTL algorithm on the cell. The result is displayed in figure 8.8, where the pink marker represents the estimated cell tower location. The error is 0.925 kilometers.

We can see that the D-CTL algorithm has worked as it is supposed to. The estimated cell tower is located by the measurement at the end of the measurent path close to the upper boundary of the figure. This means D-DL chose the two measurements furthest apart from each other to be the endpoints of $l_{direction}$. Then

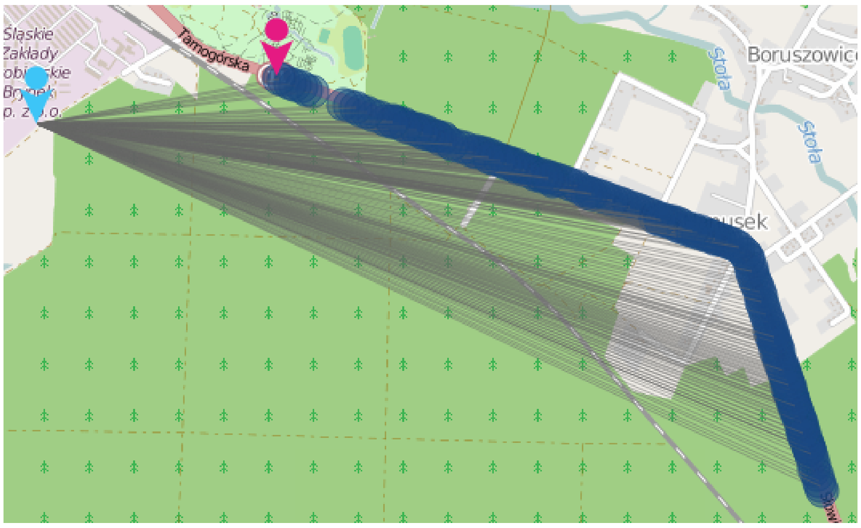


Figure 8.8: Cell 260-1-29001-22095. D-CTL has estimated a cell tower location.

FS computed the two estimated cell sectors, and D-CD picked the one whose cell tower location was closest to the correct cell tower location. Figure 8.9, which displays both of the two estimated cell tower locations computed by FS, confirms this hypothesis.

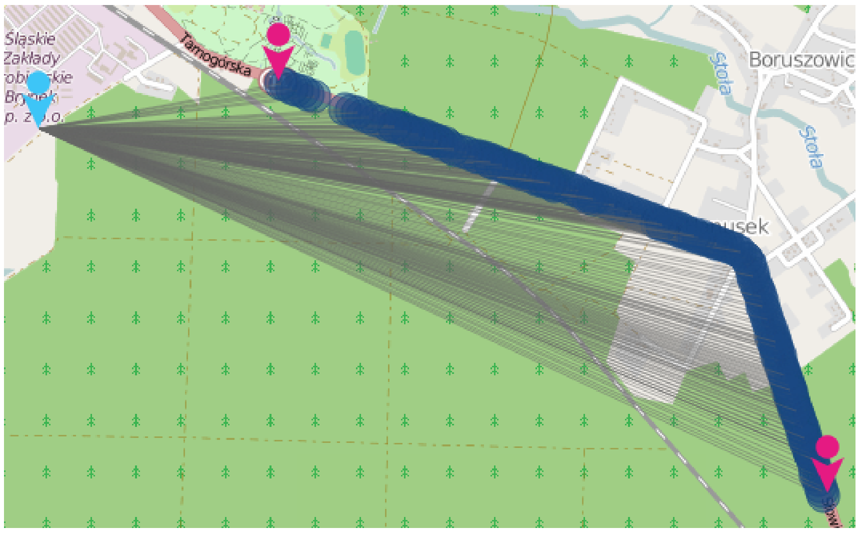


Figure 8.9: Cell 260-1-29001-22095. We show both estimated cell tower locations computed by FS during the execution of D-CTL.

Even though the error is small and the D-CTL algorithm works as expected, there are still room for improvements. Recall that the D-CTL and RSS-CTL algorithms assume measurements are randomly distributed within their respective cell sectors. When the measurements form an almost straight path such as these do, with no measurements above or below the path, the D-CTL algorithm will always estimate the cell tower location to be at one of the ends of the path. This time, the error is only small because the correct cell tower location is close to the one of the ends of the path. If some measurements were to be located away from the path, the algorithm would have had other measurements to base $l_{direction}$ on.

RSS-CTL

We run RSS-CTL on the cell. The result is displayed in figure 8.10, where the pink marker represents the estimated cell tower location. The error is 0.922 kilometers.

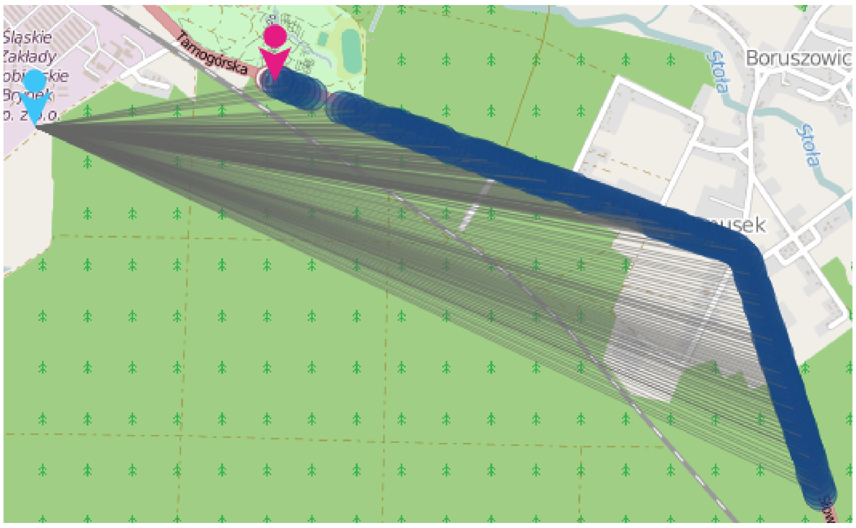


Figure 8.10: Cell 260-1-29001-22095. RSS-CTL has estimated a cell tower location.

RSS-CTL estimated the cell tower location to be almost at the same spot as D-CTL did. This is as expected, as the measurements at that spot are the closest ones to the correct cell tower location, and thus should have the strongest RSS. This means that RSS-DL successfully computed a $l_{direction}$ between the pair of measurements with the largest difference in RSS between them. Then RSS-CD chose the one of the two estimated cell sectors whose cell tower location was closest to the correct cell tower location. Figure 8.11, which displays both of the two possible estimated cell tower locations, confirms this.

The problem is still the assumption of randomly distributed measurements. When the measurements form an almost straight path, the RSS-CTL algorithm will estimate the cell tower location to be at one of the ends of the path. RSS-DL will create $l_{direction}$ from two measurements in the path. FS will extend $l_{direction}$ to both directions to compute two cell sectors. When $l_{direction}$ is extended as far as to one of the ends of the path, every measurement will fit within a cell sector.

OpenCellID Algorithm

We run the OpenCellID algorithm on the cell. The result is displayed in figure 8.12 where the pink marker represents the estimated cell tower location. The error is 2.3 kilometers.

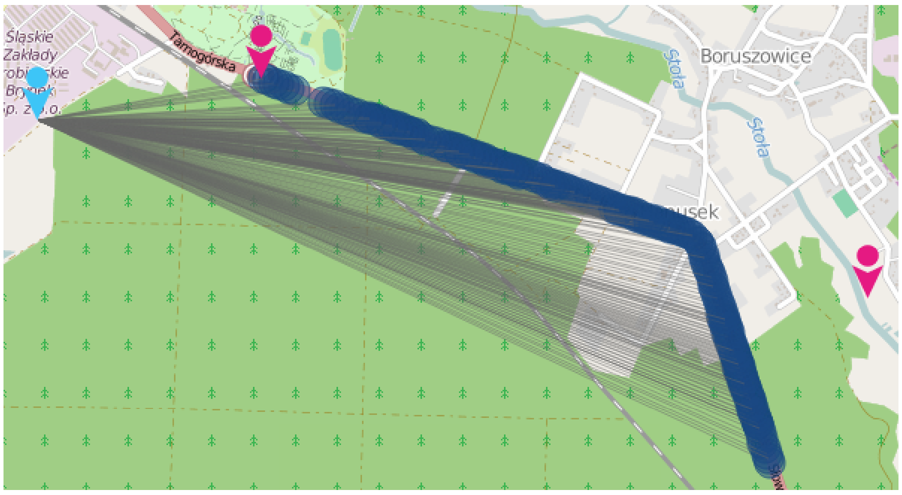


Figure 8.11: Cell 260-1-29001-22095. We show both estimated cell tower locations computed by FS during the execution of RSS-CTL.

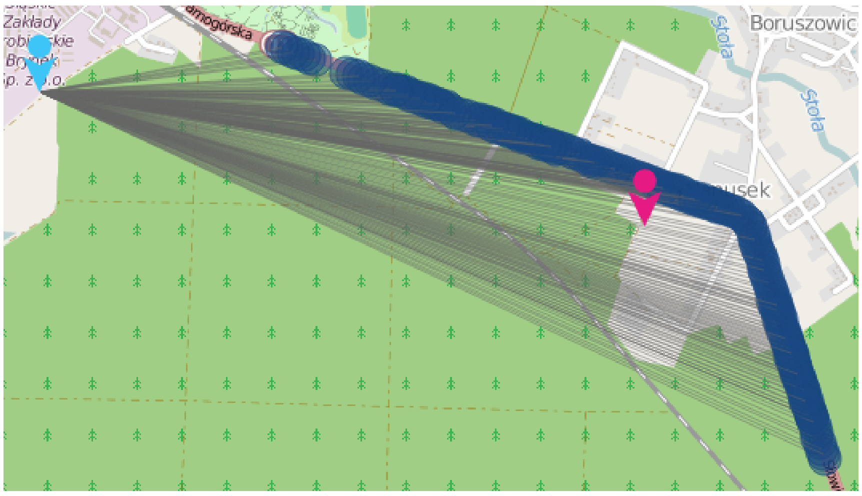


Figure 8.12: Cell 260-1-29001-22095. The OpenCellID algorithm has estimated a cell tower location.

This approach does not work well for this kind of cell. We can see that the estimated cell tower by this approach is located close to the middle of the path of measurements. For this kind of measurement pattern the D-CTL and RSS-CTL algorithms estimate better locations.

Conclusion

D-CTL and RSS-CTL work well on this cell. They manage to estimate cell tower locations close to the measurements and close to the correct cell tower location. There are still room for improvements. Estimated cell tower locations at one of the ends of such a path-like measurement pattern is not necessarily good. For this specific cell, D-CTL and RSS-CTL estimated cell tower locations not far from the correct cell tower location. But the correct cell tower location might not always be located that close to the measurements.

8.2 Improvements

We can improve the D-CTL and RSS-CTL algorithms i several ways. Every subroutine can be improved so that they all execute their tasks better.

Measurement Distribution

The most fundamental improvement we can do involves learning more about the measurement distribution for each cell. Say we can algorithmically find out how the measurements are distributed for a cell. Then we can base choices done throughout the algorithms on this information. It is clear that D-CTL and RSS-CTL works better on some cells than others. For example would Weighted Centroid work much better on omnidirectional cells than our two algorithms. See section 2.2.1 and 2.3. This means information about measurement distribution even will help with the choice of which algorithm that would be most effective.

Another reason for learning more about each cell's measurement distribution is to detect the individual measurements within cells that will affect the accuracy of estimated cell tower locations negatively. We saw examples of such measurements in figure 8.1. We did try one approach for detecting such measurements. The purpose of $r_{include}$ is to exclude measurements that are too far away to be valid. See section 6.3.2. We also experimented with different values of $r_{include}$ in section 7.5 to see how that would affect the accuracy of estimated cell tower locations. But this approach is simple and rough. We need an approach that can single out individual measurements that will affect the accuracy of the algorithms negatively, no matter how far away they are or to which direction they are located, from the correct cell tower location.

It would also help if the information included in measurements were more complete. In advance of running D-CTL and RSS-CTL on real cells in chapter 7, we had to run a validation procedure for each measurement and each cell. The validation procedures are described in section 6.4 and 6.5. These procedures lead to an exclution of many measurements and cells. Recall that several data fields for a cell are dependent of information included in measurements. So if measurements were more complete, they could have provided additional and more accurate information about the cell tower location. Take for example the *ta* (Timing Advance) data field included in the measurement object described in table 6.2. This is one of the data fields very few measurements has recorded values for. The timing advance value corresponds to the length of time a signal takes to reach the cell tower from a mobile device. To utilize this could have been an even more accurate way than utilizing RSS, to estimate cell tower locations.

D-DL/RSS-DL

D-DL and RSS-DL are the two most important sub-routines in D-CTL and RSS-CTL, respectively. These sub-routines form the basis for where a cell tower location will be estimated to be. These sub-routines compute the line $l_{direction}$ from two chosen measurements. Sometimes they compute very poor lines, often as a result of bad luck when randomly pairing the measurements. One improvement is to upgrade the sub-routines to sample more than one $l_{direction}$. By comparing multiple direction lines we can find those that stand out and decrease the chance of choosing a poor $l_{direction}$. Or we can take multiple direction lines and compute some sort of average direction line from these.

FS/CS

CS has a very specific task, and is difficult to improve. FS on the other hand can be improved. Recall that FS performs several iterations. For each iteration we check if each measurement of a cell fits within a cell sector computed by CS based on $l_{direction}$. If not, then $l_{direction}$ is extended and a new cell sector is computed by CS. When checking if each measurement fits within a computed cell sector, an improvement is to also rotate the cell sector around the corresponding estimated cell tower location. Then FS also checks if each measurement fits when the cell sector rotates. When changing the angles of the cell edges like this, we might elliminate the need to extend $l_{direction}$ by a good distance. The reason for not being able to fit each measurement is not always because they cannot be fit within a 120° cell sector from the current estimated cell tower location. It is often because of the angles of the cell edges. This is just because of an unlucky computation of $l_{direction}$.

D-CD/RSS-CD

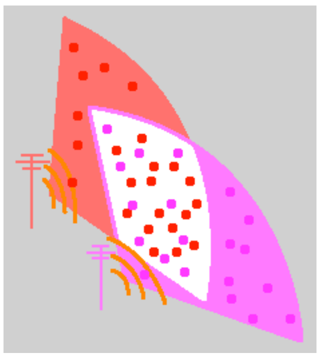
D-CD and RSS-CD are not part of the process of computing the estimated cell sectors and cell tower locations. The purpose of these sub-routines is to decide which one of the estimated cell sectors with corresponding cell tower locations, are the best. Improving these sub-routines would involve giving them more information to base their decisions on. As of now, D-CD and RSS-CD operate with the parameter n to choose a subset of measurements to use when deciding which cell sector is the best. In section 7.4, we measured running times for D-CTL and RSS-CTL. We found out that computing an estimated cell tower location with one of the algorithms, takes very little time. We can therefore with good conscience use every measurement belonging to a cell, and not just a subset, when deciding which estimated cell sector to pick. This also applies to D-DL and RSS-DL.

Multiple Cells Considered Together During Computation

Until now we have only considered cells individually when computing estimated cell tower locations. There are several ways to consider multiple cells together to potentially improve the accuracy of the locations. The techniques described in section 2.2.1 and 2.2.2 are examples of techniques that include the consideration of multiple cells together. The difference from the problems those techniques are solutions to, and our problem, is that they are based on measurement data collected with wardriving. We ask the following question: How can we consider multiple cells together based on measurement data from OpenCellID?

Cells Share Cell Tower We can apply the third step in the Bounding Technique described in section 2.2.1, Tower-based Regrouping, to our problem. In this step, cells that share a cell tower is considered together, thus simulating a 360° cell. When the measurements within these cells are combined, the Weighted Centroid algorithm can estimate the cell tower location with good accuracy. The problem with the data provided by OpenCellID is that we cannot know which cells share a cell tower. OpenCellID states that the identification number for each cell says something about which cell tower it belongs to, but there are too many identification schemes among the cellular network providers in each country to be able to know how to look for it. This is the data field *cell* in table 6.1. In addition to this, it is impossible to know if cells are missing or just do not exist as OpenCellID does not have an overview of every cell in the world. When searching for a cell to complete a 360° radius around a cell tower, we cannot know whether it exists in the database or not.

Grid or Hexagon Another way to improve the estimated cell tower locations is to look at the estimated locations of several cell towers within an area. We evaluate the estimated locations of each individual cell tower relative to the other cell towers. Recall that section 2.2.2 describes a technique where each cell tower is placed on a grid, and every cell tower's location is re-evaluated when a new one is added to the grid. In addition, we described a theoretical structure of cell towers as a system of hexagons in section 1.3.1. Evaluating cells as an organized system can help us estimate individual cell tower locations based on more than individual cell data. The measurements provide valuable information for this task. Lets assume the closest cell tower to a large subset of the measurements within a cell, is the cell tower the measurements belong to. Then, if we have estimated another cell tower's location to be closer to each measurement in that subset, we get a contradiction. See figure 8.13a. Naturally, the direction of the cell sectors also has to be considered. See figure 8.13b. Thus, the cell tower locations can be adjusted so that the system of cells is valid. But can we assume cells are organized relative to each other in the first place? In section 1.3.1 we briefly discussed theoretical locations of cell towers compared to locations in reality. We mentioned several factors affecting the practical locations of cell towers. In addition to this, time is important to consider. The demand for good reception has been growing vastly the last decades, and new cell towers must be added to satisfy this demand. Adding new cell towers over time makes it difficult to maintain a theoretical system of cells. For example, new infrastructure like buildings or tunnels may prevent a former cell tower location from still being beneficial, or even an option. Or a sudden growth in population within an area may require a new cell tower location that does not match the theoretical system. This means we cannot take a theoretical cell system for granted in reality.


(a) Too much overlap. Cell tower locations need to be re-adjusted.

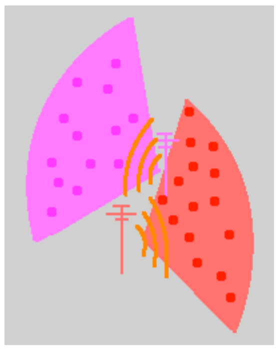

(b) Measurements are closer to another cell tower than their own, but the cells have different directions.

Figure 8.13: Examples of how two cell towers can and cannot be located relative to each other.

8.3 Conclusion

In section 1.2 we defined the research questions for this thesis. With this thesis we wanted to:

- briefly identify existing techniques and approaches for localizing cell towers, and
- develop our own algorithm for estimating cell tower locations and test it on real data provided by OpenCellID.

In chapter 2 we presented two papers, each containing a technique for localizing cell towers based on measurement data obtained from wardriving. We could not find any related work concerning localization of cell towers based on measurement data obtained from crowdsourcing. But we discussed if and how the wardriving data techinques could be applied to crowdsourced data. The conclusion was that the crowdsourced data provided by OpenCellID did not have the neccessary properties for this to be possible.

In chapter 4 and 5 we developed two algorithms, D-CTL and RSS-CTL, for localizing cell towers based on data obtained from crowdsourcing. The two algorithms are very similar. D-CTL utilizes the distance between a cell's measurements. RSS-CTL utilized the RSS values among a cell's measurements. We have tested the two algorithms on theoretical generated test data in chapter 4 and 5, and on real data provided by OpenCellID in chapter 7. We knew the correct cell tower locations on both occasions. We measured the accuracy of the algorithms by comparing the estimated cell tower locations to the correct cell tower locations. With both generated and real data, RSS-CTL estimated the most accurate cell tower locations. We also estimated cell tower locations with RSS-CTL for the Bergen City Center area.

In this chapter we have analyzed two cells with different measurement distribution patterns. We let D-CTL, RSS-CTL and the OpenCellID algorithm estimate cell tower locations for both cells. We then analyzed how the algorithms computed the estimated locations and compared them to each other. In the end we proposed several ways of how D-CTL and RSS-CTL can be improved further.

Bibliography

[1] URL: www.opencellid.org.

[2] URL: www.opensignal.com.

[3] URL: www.cellmapper.net.

[4] URL: http://www.oracle.com/technetwork/java/javase/downloads/index.html.

[5] URL: http://www.eclipse.org.

[6] URL: http://www.apple.com/macbook-air.

[7] URL: www.json.org.

[8] Robert A. Adams and Christopher Essex. *Calculus, A Complete Course.* 7th. Pearson Education, 2009.

[9] Glen van Brummelen. *Heavenly Mathematics: The Forgotten Art of Spherical Trigonometry.* Princeton University Press, 2013.

[10] N. Bulusu, J. Heidemann, and D. Estrin. "GPS-less low-cost outdoor localization for very small devices". In: *Personal Communications, IEEE* 7.5 (Oct. 2000), pp. 28–34. ISSN: 1070-9916. DOI: 10.1109/98.878533.

[11] Ruth Cowell. *War Dialing and War Driving: An Overview.* Paper. Global Information Assurance Certification. URL: http://www.giac.org/paper/gsec/863/war-dialing-war-driving-overview/101791.

[12] Enrique Estellés-Arolas and Fernando González-Ladrón-De-Guevara. "Towards an Integrated Crowdsourcing Definition". In: *Journal of Information Science* 38.2 (Apr. 2012), pp. 189–200. ISSN: 0165-5515. DOI: 10.1177/0165551512437638. URL: http://dx.doi.org/10.1177/0165551512437638.

[13] R.H. Frenkiel. "Cellular radiotelephone system structured for flexible use of different cell sizes". Pat. US4144411 A. Incorporated Bell Telephone Laboratories. Mar. 1979. URL: http://www.google.no/patents/US4144411.

[14] ETSI 3rd Generation Partnership Project. *TS 127 007.* Tech. rep. European Telecommunications Standards Institute (ETSI).

[15] T. He et al. "Range-free Localization Schemes for Large Scale Sensor Networks". In: *Proceedings of the ACM 9th Annual International Conference on Mobile Computing and Networking.* MobiCom '03 (2003), pp. 81–95. DOI: 10.1145/938985.938995. URL: http://doi.acm.org/10.1145/938985.938995.

[16] J. Hightower and G. Borriello. "Location systems for ubiquitous computing". In: *Computer* 34.8 (Aug. 2001), pp. 57–66. ISSN: 0018-9162. DOI: 10.1109/2.940014.

[17] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins. *Global Positioning System: Theory and Practice*. 5th ed. Springer-Verlag Wien, 2001. ISBN: 978-3-7091-6199-9. DOI: 10.1007/978-3-7091-6199-9.

[18] L. Hu and D. Evans. "Localization for Mobile Sensor Networks". In: *Proceedings of the ACM 10th Annual International Conference on Mobile Computing and Networking*. MobiCom '04 (2004), pp. 45–57. DOI: 10.1145/1023720.1023726. URL: http://doi.acm.org/10.1145/1023720.1023726.

[19] Natallia Kokash. "An Introduction to Heuristic Algorithms". 2005. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.105.8050&rep=rep1&type=pdf.

[20] Kurose and Ross. *Computer Networking: A Top-Down Approach*. 6th. Pearson Education, 2013.

[21] Mo Li and Yunhao Liu. "Rendered Path: Range-Free Localization in Anisotropic Sensor Networks With Holes". In: *Networking, IEEE/ACM Transactions on* 18.1 (Feb. 2010), pp. 320–332. ISSN: 1063-6692. DOI: 10.1109/TNET.2009.2024940.

[22] L.M. Ni et al. "LANDMARC: indoor location sensing using active RFID". In: *Pervasive Computing and Communications, 2003. (PerCom 2003). Proceedings of the First IEEE International Conference on*. Mar. 2003, pp. 407–415. DOI: 10.1109/PERCOM.2003.1192765.

[23] D. Niculescu and B. Nath. "Ad hoc positioning system (APS) using AOA". In: *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*. Vol. 3. Mar. 2003, 1734–1743 vol.3. DOI: 10.1109/INFCOM.2003.1209196.

[24] Drago Niculescu and Badri Nath. "DV Based Positioning in Ad Hoc Networks". English. In: *Telecommunication Systems* 22.1-4 (2003), pp. 267–280. ISSN: 1018-4864. DOI: 10.1023/A:1023403323460. URL: http://dx.doi.org/10.1023/A%3A1023403323460.

[25] Andreas Savvides, Chih-Chieh Han, and Mani B. Strivastava. "Dynamic Fine-grained Localization in Ad-Hoc Networks of Sensors". In: *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*. MobiCom '01. Rome, Italy: ACM, 2001, pp. 166–179. ISBN: 1-58113-422-3. DOI: 10.1145/381677.381693. URL: http://doi.acm.org/10.1145/381677.381693.

[26] A. Varshavsky et al. "Are GSM Phones THE Solution for Localization?" In: *Mobile Computing Systems and Applications, 2006. WMCSA '06. Proceedings. 7th IEEE Workshop on*. Aug. 2006, pp. 34–42. DOI: 10.1109/WMCSA.2006.2.

[27] Jie Yang et al. “Accuracy Characterization of Cell Tower Localization”. In: *Proceedings of the 12th ACM International Conference on Ubiquitous Computing*. UbiComp ’10. Copenhagen, Denmark: ACM, 2010, pp. 223–226. ISBN: 978-1-60558-843-8. DOI: `10.1145/1864349.1864384`. URL: `http://doi.acm.org/10.1145/1864349.1864384`.

[28] Yuan Zhang et al. “Base Station Localization in Search of Empty Spectrum Spaces in Cognitive Radio Networks”. In: *MSN 2009, The Fifth International Conference on Mobile Ad-hoc and Sensor Networks, Wu Yi Mountain, Fujian, China , December 14-16, 2009*. 2009, pp. 94–101. DOI: `10.1109/MSN.2009.66`. URL: `http://dx.doi.org/10.1109/MSN.2009.66`.