Centre for Geo-Information

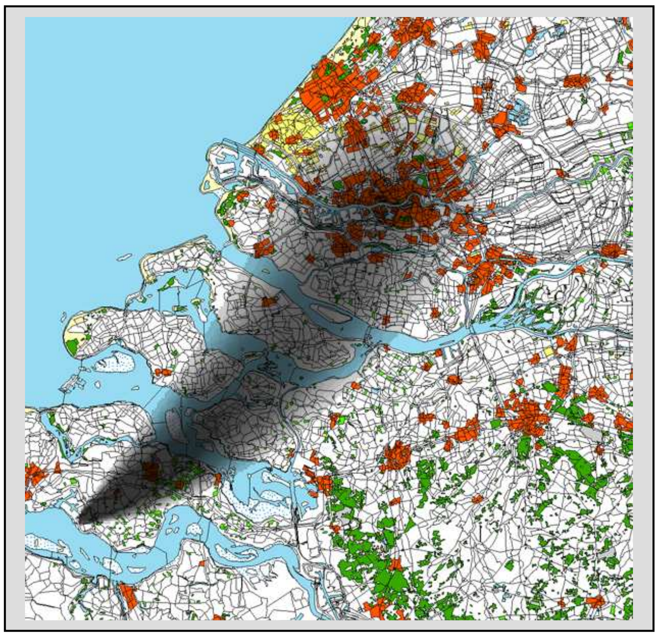
Thesis Report GIRS-2008-07

Dealing with uncertainty in determining the optimal locations of mobile measuring devices

A case study for mapping the dispersion of a radioactive plume in the first phase after a nuclear accident

Johan Beekhuizen

March 2008



WAGENINGEN UNIVERSITY
WAGENINGEN UR

Dealing with uncertainty in determining the optimal locations of mobile measuring devices

A case study for mapping the dispersion of a radioactive plume in the first phase after a nuclear accident

Johan Beekhuizen
Registration number 83 01 27 042 030

Supervisors:

Dr. Ir. G.B.M. Heuvelink
Dr. Ir. S. de Bruin
Dr. S.J. Melles

A thesis submitted in partial fulfilment of the degree of Master of Science at Wageningen University and Research Centre, The Netherlands.

March - 2008
Wageningen, The Netherlands

Thesis code number: GRS-80436
Thesis Report: GIRS-2008-07
Wageningen University and Research Centre
Laboratory of Geo-Information Science and Remote Sensing

Preface

After half a year of hard work, I gladly present my thesis report about the optimisation of mobile sampling designs. This study is a continuation of the thesis work of Zhengkun Jiang, who made a successful first attempt to deal with the optimisation problems. It took me weeks to really understand the complicated methodology worked out by Zhengkun in 2007, but finally I managed to come up with a proposal for extending his work. Many extensions have been implemented successfully and the results of my work gained new insights in the optimisation process, providing enough material for future research. When looking back at the whole period I am glad that I took up the challenge and delighted by the honour to present my work at the GeoEnv2008 conference in Southampton in September 2008.

One of the best experiences of my thesis work was the amount of support I received from various people, and I would like to express my gratitude for all those who have helped me.

First of all I would like to thank my supervisors for their excellent supervision. I had the privilege of working with three professional, motivating and interested supervisors: Gerard Heuvelink, Sytze de Bruin, and last but not least Stephanie Melles. Without your explanations, recommendations and general assistance I would have never come this far.

Of the RIVM I would like to thank the following persons: Arjan van Dijk for his great effort in helping me out with the atmospheric dispersion model NPK-PUFF, the provision of meteorological data and discussions about the modelling problems I had to face, Chris Twenhöfel for his explanation of the Dutch radiation network and general support and Sam Bader for his help with the dose calculations. Furthermore I would like to thank the KNMI for allowing the provision of their meteorological data.

Next I would like to thank Jordi Vila of the chair group Meteorology and Air Quality, an expert on boundary layer atmosphere, for discussing uncertainty in meteorological parameters.

Finally I would like to thank Edzer Pebesma, for the adaptation of his excellent and free geo-statistical library GStat in the programming language R, and Paul Hiemstra for helping me out with R programming issues.

Johan Beekhuizen

Wageningen, the Netherlands,
March - 2008

Summary

To prepare for the unlikely event of a nuclear emergency, atmospheric dispersion models are used to assess possible release concentrations and radionuclide deposition rates. In the Netherlands, the NPK-PUFF dispersion model is used to provide critical information about the radioactive plume during the first phase of a nuclear accident. By combining NPK-PUFF model predictions with spatial analyses of real-time gamma (γ) dose rate measurements from the Dutch National Radioactivity Monitoring Network (153 stations and 14 α/β air sampling monitors), informed intervention decisions can be made. In an emergency, additional mobile devices can be deployed to increase the accuracy in spatial estimates of radiation exposure. The main objective of this research was to determine where to optimally locate mobile devices in order to minimize costs associated with uncertain estimates of the radioactive plume. A Monte Carlo approach was used to account for uncertainty in the NPK-PUFF model. First probability distribution functions of model input parameters that contribute most to uncertainty in the predictions were investigated. Next many possible developments of these input parameters could be simulated and used to create various model outputs. To account for uncertainty in the model itself, a stochastic residual was generated and added to the NPK-PUFF model output. This resulted in a set of simulated realities. The optimal location of mobile measuring devices was then determined using spatial simulated annealing. The optimisation criterion selected for this research was the cost of making incorrect decisions due to prediction uncertainty. This uncertainty was quantified by comparing predicted maps with simulated realities. The predicted maps were created by adding a residual to a reference dose map. The residual was calculated by interpolating the differences between observations of the simulated reality and reference dose map using Inverse Distance Weighting. The cost function took population density into account and permitted a weighting of false positive (predicted concentrations greater than intervention thresholds when the actual value is below) and false negative values. Results indicate that optimal placement of mobile devices in the event of an accident tend to be in areas at the edge of the advancing radioactive plume, in densely populated areas. However, the Inverse Distance Weighting interpolation gave unsatisfactory results in improving the predicted maps.

Keywords: Emergency; Uncertainty; Meteorological predictions; Radioactivity; Sampling design; Spatial simulated annealing

Table of contents

List of Figures

1 Introduction

1.1 Background

After the Chernobyl accident the necessity of an early warning monitoring system for radioactive contamination became even more evident. In the Netherlands, this resulted in the improvement of both the national organization for nuclear emergency planning and the technical infrastructure necessary for collection and representation of (radiological) information (Smetsers and van Lunenburg 1994). The first Dutch National Radioactivity Monitoring Network (NRM) was designed, and the National Institute for Public Health and Environment (RIVM) played a crucial role in the development of this network. The monitoring network has improved considerably over the last two decades and in 2005 the operation of the third generation network was initiated (Twenhöfel et al. 2005).

During nuclear emergencies, the RIVM runs the Back Office for Radiological Information (BORI). BORI is part of the Unit Planning and Advice nuclear (EPAn), which organises the national (off-site) nuclear emergency management (Twenhöfel et al. 2005). BORI provides information as good and fast as possible about radiation levels. During the first phase of an accident BORI focuses on issues like the prediction and establishment of the properties and shape of a radioactive plume, whether or not the emission is larger than a critical threshold and what the consequences are for inhabitants of the contaminated area. Estimation of the radiation level is based on model calculations and analysis of radiation measurements (Smetsers 2005).

Model calculations are made by the atmospheric dispersion model NPK-PUFF (Verver et al. 1990). The model simulates the emission of a source by hourly released puffs. The trajectory of each puff is computed by either analysed or forecast wind fields (van Pul 1992). NPK-PUFF can calculate both the concentration and deposition of radionuclides (Eleveld et al. 2007), and is of great importance for the GIS-based decision support system at the RIVM.

The measurements are made by the third generation of the NRM, consisting of 153 ambient γ-monitors and 14 α/β air sampling monitors. Both device types measure ambient dose rates in nSv/h (nanoSievert / hour), which are also the units used for threshold warning levels (Twenhöfel et al. 2005). Monitors cover the whole of the Netherlands, with an increased density around power plants and in the Dutch border region, near foreign power plants (Twenhöfel et al. 2005).

In case of an emergency at a nuclear power plant, the actual release of radioactive nuclides can usually be delayed a few hours. During this delay, RIVM can add extra measuring points to the static network by using mobile devices. RIVM has two vans which transport mobile devices to the area around the emergency (Smetsers 2005). As it is preferred that mobile measuring devices are placed before the anticipated accident, the location of these devices should be determined quickly. This leads to the main objective of this research; where to put mobile devices in order to improve measurements of a radioactive plume in the event of an emergency?

1.2 Problem definition

The main problem of this research is determining optimal locations of mobile measuring devices, in order to support the monitoring system (i.e. NPK-PUFF + static stations). In this work a possible combination of locations of mobile devices is referred to as a sampling design. As this research was a continuation of the work of Jiang (2007), it followed that the problems were extensions of his work.

In order to find a suitable sampling design it was crucial to examine uncertainty in the NPK-PUFF prediction of the radioactive plume. There are many sources for this uncertainty and these had to be explored and taken into account when determining the optimal location of mobile measuring devices. Obviously, not only the NPK-PUFF output is used in the prediction of the radioactive plume as there are mobile and static measuring devices to support the model. This leads to the question how to use observations of devices to adjust the plume predicted by NPK-PUFF. As we wanted to know real measurement values before actually placing the devices, the observations had to be simulated. These observations were used to improve the prediction of the plume. After implementing a way to adjust the predicted plume, it was possible to compare different (predictions of) sampling designs. This comparison enabled the calculation of the suitability of a sampling design. How good is a sampling design compared with other sampling designs? In other words, what is the optimisation criterion?

Jiang (2007) introduced a cost function which compared a true plume with a predicted plume and assigned costs to the areas that were wrongly predicted (false positives and false negatives). This function was improved by taking into account population density at areas where predictions are wrong. When it was possible to determine the suitability of a sampling design the final step could be taken: implementing a methodology for optimising the sampling design.

In order to test the methodology a fictitious accident at Borssele, a nuclear power plant in the southwest of the Netherlands (Figure 1), was simulated. The accident had a release size of 1e+17 Bq ^{131}I (radioactive iodine). We were interested in the radioactive plume six hours after the release, because we wanted the mobile devices to improve the prediction of the radioactive plume in the first phase after the accident. The wind direction was southwest with a wind speed of four m/s, resulting in a plume heading towards Rotterdam. The study area was defined by the maximum distance radioactive nuclides could be transported in the six hours after the initial release. Figure 1 shows the 100 x 100 km large study area with an overlay of the radioactive plume, which indicates the dispersion of the plume six hours after the accident.

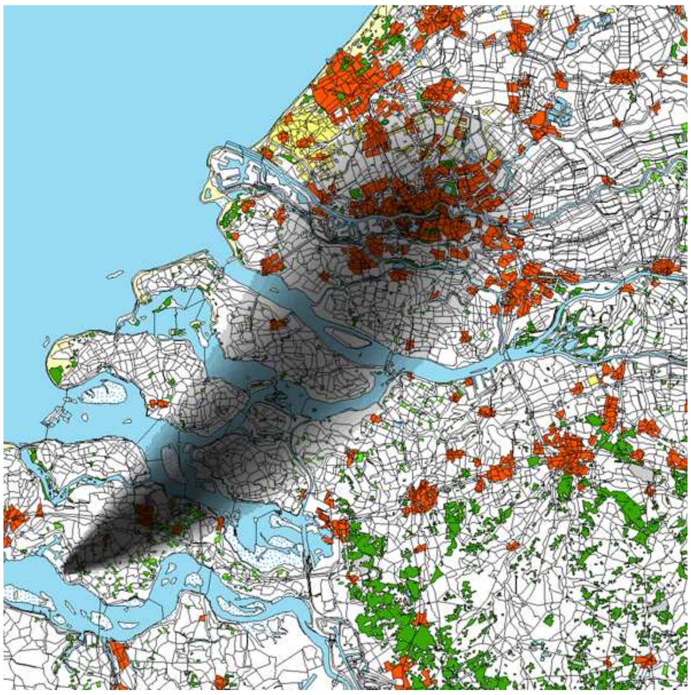


Figure 1: Study area with in grey shades a hypothetical radioactive plume originating from Borssele

1.3 Objective and research questions

Objective:
To minimize the risk of making wrong decisions in the first phase after a nuclear accident by optimizing the locations of mobile measuring devices.

Research questions:
1. What are the sources of uncertainty in the prediction of the radioactive plume and how can these be quantified?
2. How can measurements of mobile devices be used to locate the predicted plume more accurately?
3. How can the costs of wrong decisions associated with the sampling design be calculated?
4. What is a suitable methodology to optimise the locations of mobile devices?
5. How does the optimisation method perform in practice?

Chapter 2 discusses the first four research questions, forming the methodology of this study. The results (research question 5) of this study are given in chapter 3. After showing the results the major issues that were raised during this research are discussed in chapter 4. The final chapter provides recommendations for further research and concluding remarks.

2 Methodology

A Monte Carlo simulation approach (Heuvelink et al. 2008) was used for representing the uncertainty in NPK-PUFF model outcomes. A large number of possible realities were simulated and the costs of each of these simulated realities were calculated. If the amount of simulated realities was sufficient the average costs would be a reasonable indication for the suitability of the sampling design. Figure 2 provides an overview of the methodology to determine the costs, along with references to the sections where the separate steps are explained in more detail.

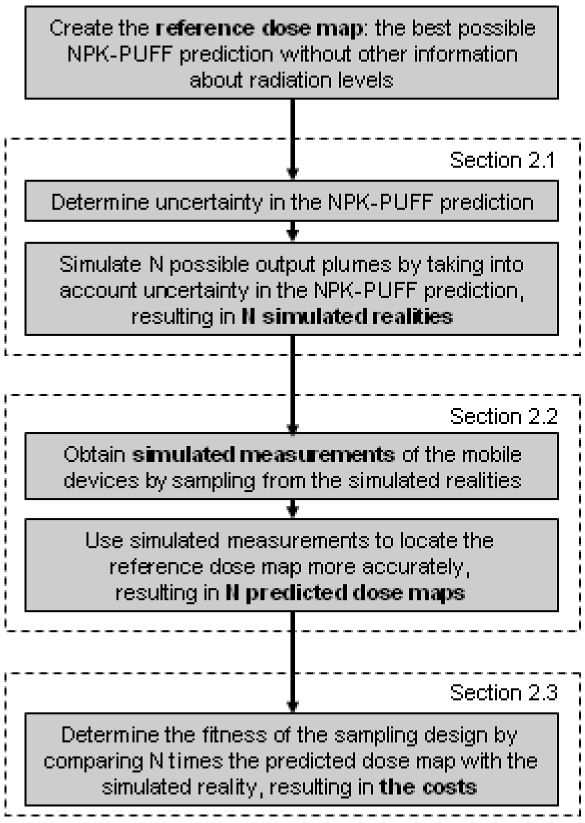

Create the **reference dose map**: the best possible NPK-PUFF prediction without other information about radiation levels
Section 2.1
Determine uncertainty in the NPK-PUFF prediction
Simulate N possible output plumes by taking into account uncertainty in the NPK-PUFF prediction, resulting in **N simulated realities**
Section 2.2
Obtain **simulated measurements** of the mobile devices by sampling from the simulated realities
Use simulated measurements to locate the reference dose map more accurately, resulting in **N predicted dose maps**
Section 2.3
Determine the fitness of the sampling design by comparing N times the predicted dose map with the simulated reality, resulting in **the costs**

Figure 2: Overview of the methodology to determine the suitability of a sampling design; N represents the number of Monte Carlo simulations

Assessing the suitability of the sampling design was crucial for determining the optimal configuration of mobile devices. This chapter concludes with an explanation of the method used to optimise the sampling design (section 2.4).

2.1 Determining and quantifying sources of uncertainty in the predicted radioactive plume

The atmospheric dispersion model NPK-PUFF was used to assess the concentration and deposition of radionuclides (Eleveld et al. 2007). But how certain of this output are we? We distinguish between two sources of uncertainty in the model output: model uncertainty and input uncertainty. Input uncertainty is associated with uncertainty in values of model input parameters, whereas model uncertainty is associated with discrepancies between the computational model and real processes (Karssenberg and De Jong 2005). For this research, model uncertainty was represented by a stochastic residual, which was simulated as a spatially correlated random field. This residual was added to the output of the NPK-PUFF model and an uncertainty analysis was performed to examine the effects of uncertainty in model input parameters. The dose map is the sum of a deterministic dispersion process model and a stochastic residual, i.e.:

Dose(x, t) = NPK-PUFF(x, t, input) + residual(x, t) [1]

where *input* represents NPK-PUFF input parameters, *x* a location in space and *t* a moment in time. Figure 3 shows a visualization of the generation of various dose maps.

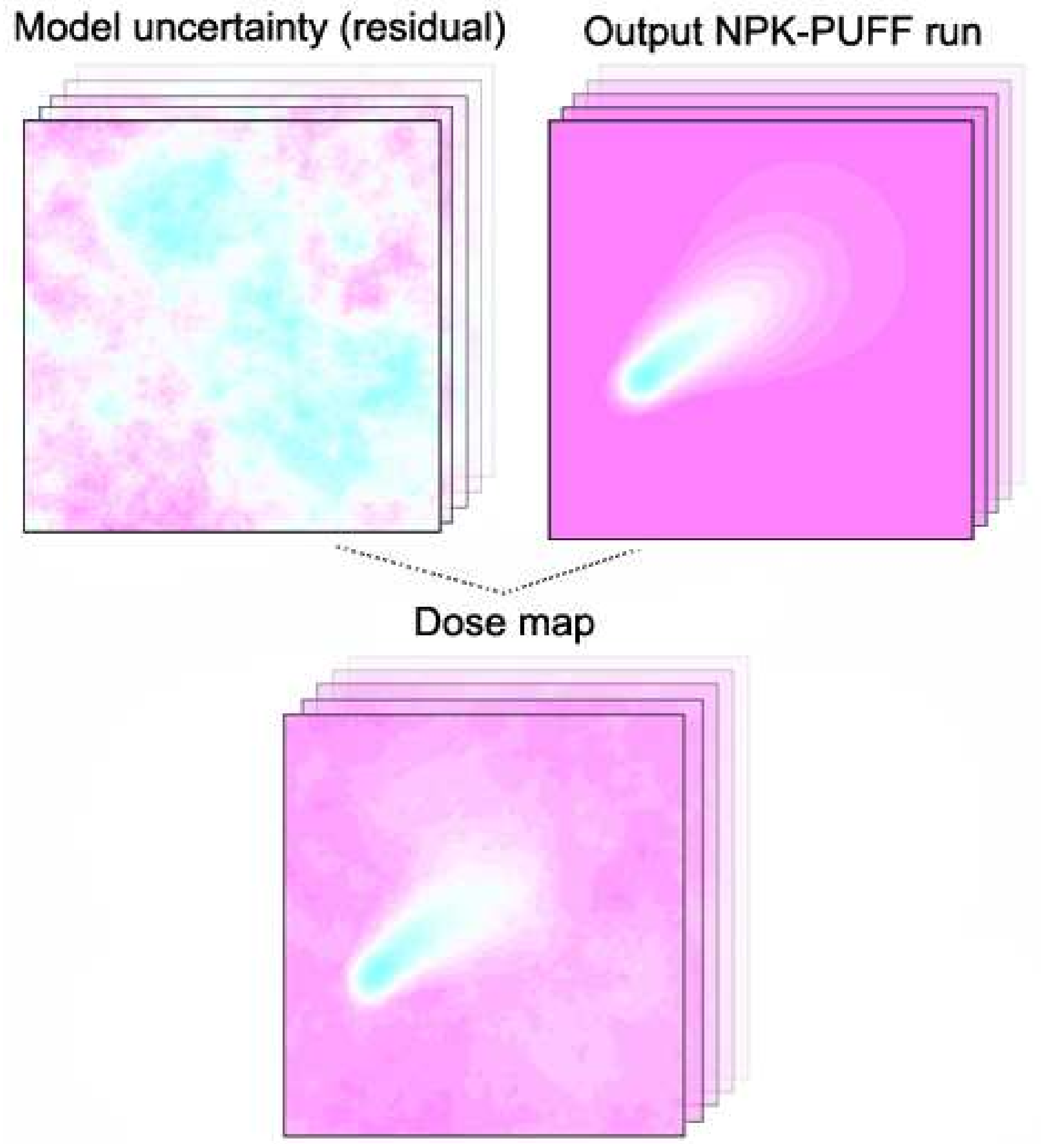


Figure 3: Realization of uncertain dose maps, which are the sum of NPK-PUFF output as a function of a realized set of input values and a sampled residual

Please be aware that **the dose map is a simulation of a possible reality**; the actual dose map cannot be known, as it is impossible to look ahead in time to see what the real radiation levels will be and we can never measure the actual radiation levels at all locations. Therefore, the dose map will be referred to with the term *simulated reality*. In short, we focus on simulating possible realities, taking into account as much knowledge as possible about uncertainty in the prediction of the output plume.

The following sections explain how input uncertainty and model uncertainty were modelled. First a selection of uncertain input sources of the NPK-PUFF model was made (2.1.1), along with data needed to analyze uncertainty in these sources (2.1.2). Next uncertainty in the NPK-PUFF parameters was modelled using probability distribution functions (pdfs), discussed in section 2.1.3. After modelling the pdfs, random simulation was used to generate model input parameters using a geostatistical approach (section 2.1.4). The last section of this chapter discusses the implementation of model uncertainty (2.1.5).

2.1.1 Selection of sources of uncertainty

A first step in quantifying the sources of uncertainty was to determine which inputs can be adjusted in the NPK-PUFF model. It was important to understand which and to what extent input values can be varied, because this limited the possibilities of the uncertainty analysis.

The NPK-PUFF model was used to simulate a number of possible realities by modifying the model input text file to read relevant parameters for determining the output plume. There are two categories of input parameters – source parameters and meteorological parameters. The source parameters, e.g. height and strength of the emission can only be set once in an input file, and therefore had to be constants.

The meteorological parameters can be either manually altered in the input file, or the NPK-PUFF model can be instructed to use output of meteorological models for the prediction of the plume. We chose to manually create the input files, because we wanted to vary these meteorological parameters to account for uncertainty. Manual input allows the user to set meteorological parameters in time steps of an hour, thus providing the opportunity for implementing temporal variation. Unfortunately the manual input file does not provide the possibility to vary meteorological parameters in space. In other words, NPK-PUFF assumes that these parameters are constant in space. Therefore, we focused on temporal variability.

After consulting experts (Twenhöfel[1] and van Dijk[2], *pers. comm*) it became clear that the most important model inputs contributing to uncertainty in the prediction of a radioactive plume are:

- wind speed
- wind direction
- mixing layer height
- atmospheric stability
- release height
- emission rate / release strength

The first four parameters are related to meteorological circumstances and the last two are related to the source of radionuclide emission. The NPK-PUFF model calculates the dispersion of the radioactive plume using wind-vectors at three different heights (10, 382 and 1387 meters), thus uncertainty in the wind-vector parameters are to be estimated at each of these heights.

2.1.2 Data-analysis of sources of uncertainty

In order to research uncertainty in the meteorological parameters, an analysis of meteorological data was performed. The data used were the same data that the RIVM uses to predict a potential radioactive plume with the NPK-PUFF model. KNMI (the Royal Dutch Meteorological Institute) provides RIVM with meteorological data every six hours, which consists of an estimation of the current and predicted meteorological situation. The predictions are made for several time steps of six hours, thus a prediction is made for six hours ahead in time, 12 hours, 18 hours and so on. For this research the six-hour prediction is most relevant as this prediction covers the first phase of a nuclear accident. The predictions of the meteorological parameters are made by the High Resolution Limited Area Model (HIRLAM), a numerical short-range weather forecasting system (About HIRLAM Programme, n.d.). HIRLAM calculates values of the meteorological parameters for each cell on a grid with a cell size of 55 by 55 kilometres.

The KNMI provided a dataset with the data of various meteorological parameters for one year. This dataset contained both the six-hour HIRLAM prediction and the actual value estimated six hours later - for each parameter and only for the grid-cell on which the nuclear power plant of Borssele is located. The HIRLAM-dataset included all parameters which contribute to uncertainty in the prediction of the radioactive plume; wind speed and wind direction (at 10, 382 and 1387 meters height), mixing layer height and atmospheric stability. By comparing the six-hour prediction and the actual value estimated six hours later, the accuracy of the six-hour prediction was analyzed.

Because meteorological conditions vary structurally both throughout the year and during the day, temporal variation and prediction accuracy are time-dependent. For instance, it is known that the development of mixing layer height is different in summer than in winter, and variance in wind direction

[1] C.J.W. Twenhöfel, RIVM, Bilthoven, Personal Communication, July 16, 2007
[2] A. van Dijk, RIVM, Bilthoven, Personal Communication, October 11, 2007

during the day differs from the variance during the night (Wieringa and Rijkoort 1983). Therefore, we selected a certain time period for our analysis, and the statistical properties of only part of the dataset were determined. The spring-day scenario was chosen, which restricted the data to daytime values (8:00 to 18:00) for March, April and May. The practical implication of using this differentiation in scenarios is that the statistical models estimated from this dataset are only valid for this time-period.

2.1.3 Modelling the uncertainty

In order to model uncertainty in the input values it was necessary to explore how uncertainty of these values could be defined. Heuvelink et al. (2007) described a framework in which uncertain environmental variables are represented and simulated with pdfs, which is one of the most frequently used approaches for representing uncertainty. He distinguishes between positional uncertainty and attribute uncertainty of variables. Because the location of the radioactive source is fixed and the meteorological parameters cannot be defined by certain locations, describing positional uncertainty is ineffectual. The defined attribute uncertainty is useful however. Heuvelink et al. (2007) distinguished between space and time variability of an attribute. Because the NPK-PUFF model only provides the possibility to vary meteorological parameters in time, we focused on the temporal variability of the meteorological parameters. The source parameters (height and strength of the emission) were fixed in the input file of the NPK-PUFF model, and thus could only be varied between different input files.

Temporal variability in the meteorological variables represents possible changes of the variables over time. For instance, wind speed can be four m/s at a certain moment in time, but one hour later it can be five m/s. The probability that wind speed will change from four to eight m/s within one hour is small in the Netherlands. By modelling temporal variability it is possible to simulate developments of meteorological variables in time. Temporal variability is discussed in section 2.1.3.1.

In addition to uncertainty in the development of meteorological NPK-PUFF input variables, there is uncertainty in the six-hour meteorological prediction that RIVM receives from KNMI. How certain can we be of these HIRLAM-predictions? What is the pdf of the uncertainty in this prediction? Section 2.1.3.2 deals with this question.

Figure 4 illustrates the overall approach to modelling input uncertainty. The blue dashed lines show the range in which 95% of the normally distributed input values (e.g. wind speed) fall. As can be seen from the graph there is no uncertainty in the value at t = 0, i.e. it is assumed that the initial value is exactly known. Up till a temporal distance of three hours, the input value becomes more and more uncertain. The range of possible values, represented by the area within the blue dashed lines, increases. But uncertainty does not grow continuously, as the six-hour prediction is known, which limits the range of possible values six hours ahead in time. The red dotted line represents the mean value of all simulations of this parameter.

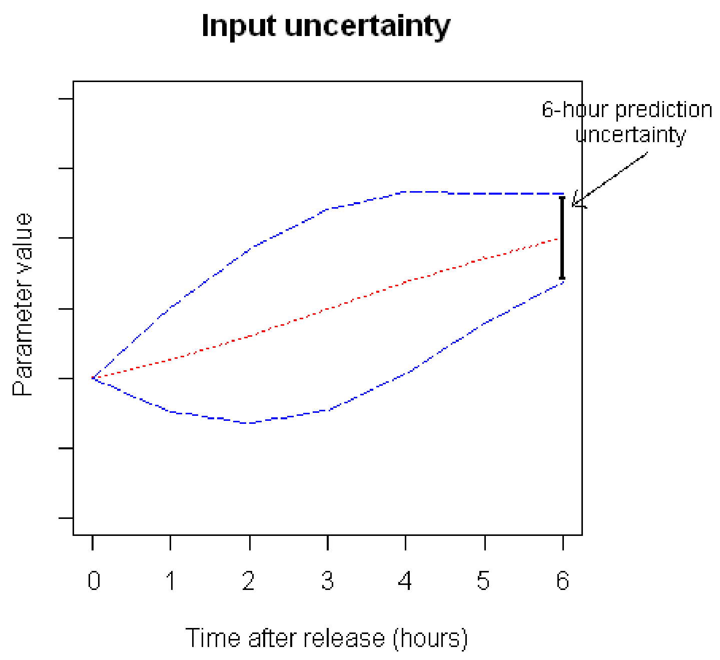


Figure 4: Input uncertainty caused by temporal variation and uncertainty in the six-hour HIRLAM prediction

2.1.3.1 Modelling the temporal variability

The temporal variability in the input values was estimated by means of a variogram. Normally a variogram is used to describe spatial continuity as a function of distance and direction (Isaaks and Srivastava 1989). However it is also possible to estimate a variogram for changes in temporal continuity, by replacing distance with time. As time is a one-dimensional variable, direction is irrelevant. The GStat R-package (Pebesma 2004) was used to calculate variograms of the meteorological parameters.

After calculating temporal semivariance, a variogram model was fitted. Variogram models were necessary to randomly simulate many possible developments of meteorological parameters. This general approach was suitable for estimating variogram models for wind speed at three different heights, shown in Appendix 1. The variogram models of the other meteorological parameters were more difficult to estimate.

Wind direction: Calculating the variogram model for wind direction caused problems because it is a circular variable (i.e. between 0 and 360 degrees). In order to make a variogram of wind direction, the data were transformed into two continuous variables, representing “northness” and “eastness” (Palmer, n.d.). This was done by calculating the sine and cosine of wind direction. A variogram model for both the sine and cosine of wind direction was estimated; together these models represented the temporal variability in wind direction. After the calculations, the components were back transformed to wind direction in degrees. Appendix 1 shows the variogram models for cosine(wind direction) and sine(wind direction).

Mixing layer height: Estimating the variogram model for mixing layer height was problematic because the quality of the data was low. The mixing layer height data were only rough estimates; the hourly increase during daytime was constant, there was no gradual decrease between day and night, and the six-hour prediction was equal to the estimated current value six hours later. Therefore another approach to determining the variogram model was taken. First four sets of possible developments of the mixing layer height, using four different variogram models with varying sill and range, were generated. Based on expert judgment, the sill and range of the set which best resembled reality was chosen. This resulted in the following model: *variogram(mixing layer height) = 10000 * Exp(6)*

Atmospheric stability: The Pasquill-classification was used to model the stability of the atmosphere. This is a classification of atmospheric stability in six (or sometimes seven) ordinal classes (Pasquill 1974 in (Kok 2005)). Using the HIRLAM-dataset the temporal semivariances of the Pasquill values were calculated and the following variogram model was estimated:

*variogram(atmospheric stability) = 0.62 * Linear(10)*

2.1.3.2 Modelling the uncertainty in the HIRLAM-prediction

Next to uncertainty in the development of input parameters in the NPK-PUFF model, there is uncertainty in the six-hour HIRLAM predictions. The statistical properties of the uncertainty were estimated by comparing measured and predicted values of the same time-instance, using the HIRLAM-dataset that KNMI provided. This was done by determining the histograms of differences between predicted and actual values. The analyses were based on a spring-day scenario; therefore only the daytime data of March, April and May were compared.

Appendix 2 shows SPSS-results of this analysis. The mean and standard deviations shown in the appendix were used to generate, using a normal distribution, different possible predictions. The assumption of a normal distribution was statistically valid for the wind speed parameters, but the prediction error of the wind direction was not normally distributed. The distribution of the prediction error in wind direction had a positive kurtosis, but was not significantly skewed. However, improving the distribution function for the wind direction would be complex and the assumption of a normal distribution resulted in similar prediction errors as the errors obtained by the six-hour HIRLAM predictions.

There was no histogram for mixing layer height, as all prediction values were exactly the same as the measured values. This is not realistic, as it is unfeasible that all predictions of this complicated parameter are 100% accurate. For the distribution of the prediction error a normal distribution with a mean of 0 (no bias) and a standard deviation of 50 meters was assumed.

The last histogram shown in Appendix 2 depicts the error in the prediction of atmospheric stability using the Pasquill-classification. The usage of the statistical parameters of this histogram is questionable, as the Pasquill values are on an ordinal scale while the data are treated as continuous and normally distributed values. But the HIRLAM predictions were quite good, as the prediction error had a small standard deviation and a negligible bias. Therefore a normal distribution was assumed with a standard deviation of 0.4 and a mean of 0.

Representing uncertainty in both mixing layer height and atmospheric stability input parameters using a Gaussian pdf was simplistic. However, as this study focuses on the overall methodological framework, the used approach was thought to be a reasonable starting point. Nevertheless, limitations along with suggestions for further research are addressed in chapter 5.

2.1.3.3 Modelling uncertainty in the source variables

Modelling of uncertainty in the source variables (source height and emission rate) was more straightforward, as these were assumed to be constant in time. For emission height, the same approach was used as in (Kok 2005). He made a distinction between a low and a high emission source scenario. The values used for a low emission source were in the range of 10-30m, and values for the high source between 200-600m, both uniformly distributed.

The uncertainty in the source emission rate is substantial. According to Twenhöfel[3], very little is known about the emission rate in early phase risk estimates. Therefore the RIVM uses reference scenarios to handle this uncertainty. The scenario approach was also applied in this research by using a fixed emission of 1e+17 Bq for all input files, i.e. uncertainty in this variable was not accounted for.

[3] C.J.W. Twenhöfel, RIVM, Bilthoven, Personal Communication, November 27, 2007

2.1.4 Simulating possible output plumes

After determining the statistical properties of temporal variation and uncertainty in the prediction of the input variables, the next step is to simulate possible developments of these parameters using random simulation. For the meteorological parameters the following steps were taken:

1. Define temporal variation using the estimated variogram models (see 2.1.3.1).
2. Set the current and six-hour HIRLAM prediction of this parameter.
3. Generate possible six-hour predictions with estimated pdf's of the prediction uncertainty (see 2.1.3.2).
4. Simulate values for time steps between the current and predicted parameter value.

The simulation was done using conditional sequential Gaussian simulation (Deutsch and Journel 1998; Pebesma 2004). The conditional simulation process creates alternative developments of the meteorological variable, honouring the estimated variogram model of the variable. The simulation was "conditional" because the resulting realizations reproduced data values (current and predicted value) at their location, and Gaussian because the variables were represented by continuous data which were assumed normally distributed. Sequential simulation extends the conditioning of new data values by including the original and all previously simulated values (Deutsch and Journel 1998).

Figure 5 visualizes the results of 15 simulations of possible developments of mixing layer height and wind speed at 382m height. These graphs resemble the shape of Figure 4, indicating a successful implementation of the approach to modelling the development of meteorological parameters. The simulations were based on the estimated variogram models and pdfs of the uncertainty in the six-hour HIRLAM prediction.

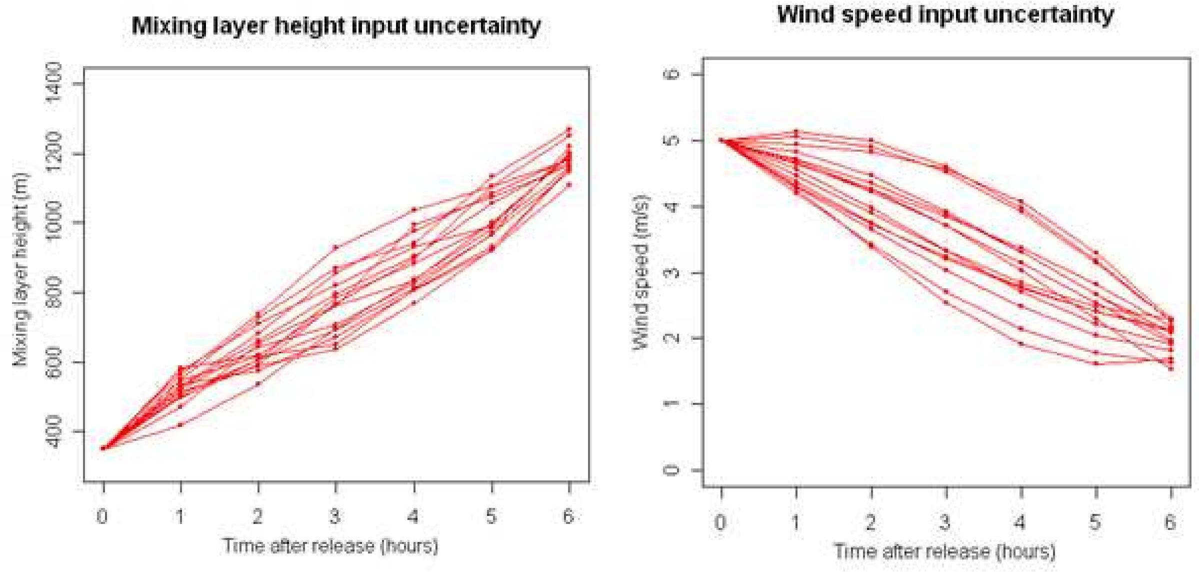


Figure 5: Simulation results of mixing layer height and wind speed at 382m height

2.1.5 Model uncertainty

Unlike input uncertainty, little was known about model uncertainty; therefore the model error was represented by a stochastic residual. We made two assumptions about the model uncertainty. Firstly, it seems reasonable to assume that the NPK-PUFF model was calibrated properly, thus without a systematically over- or underestimate of the true concentration (Heuvelink et al. 2008). Therefore we used a mean of zero for the additive residual. Secondly, we made the assumption that there is a *spatial* correlation in model uncertainty, which means that model errors at nearby locations are similar. This spatial correlation was represented by the following variogram model:

*variogram(model uncertainty) = 0.04 * Spherical(40 km)*

The residual was generated using unconditional sequential Gaussian simulation (Deutsch and Journel 1998; Pebesma 2004). This simulation process was closely related to the simulation of possible developments of meteorological parameters (see 2.1.4). Although this time the simulation is unconditional, as there are no observations of the model error to condition the simulation process. Figure 6 shows two simulation results, which provide an indication of the spatial distribution and values of the model error.

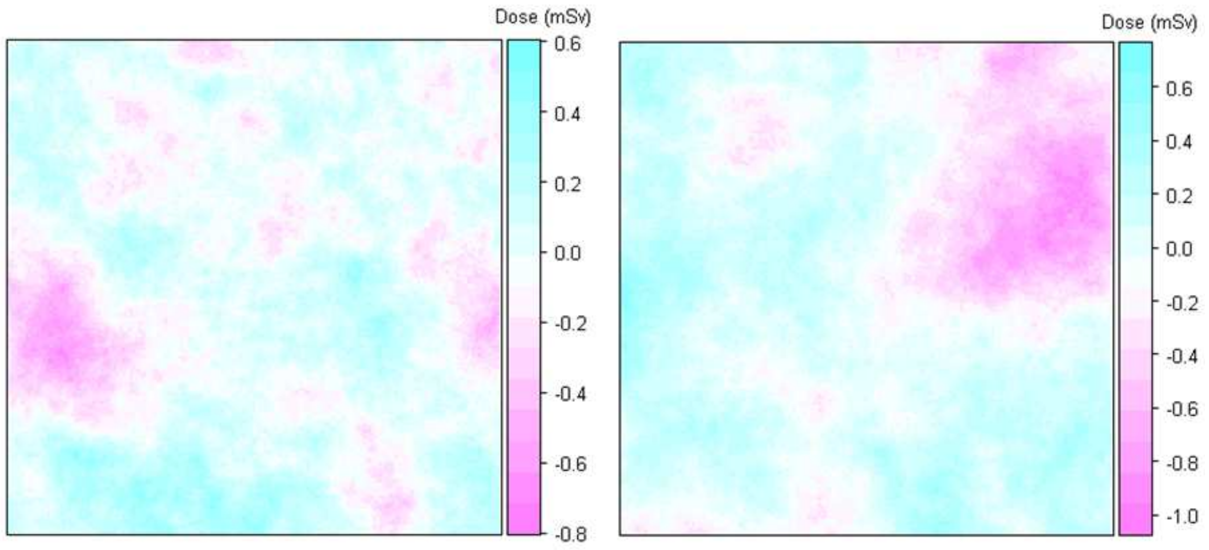


Figure 6: Two model error residuals, created using unconditional simulation

2.2. Use of mobile device measurements to adjust the predicted plume

In case of a nuclear accident, the atmospheric dispersion model NPK-PUFF is used to predict the progression of a radioactive plume. If there is no other information about radioactivity levels available, NPK-PUFF provides the best prediction possible. The output of a single NPK-PUFF model was used as a reference dose map, which had no input uncertainty or model uncertainty. However, NPK-PUFF predictions can be validated by measurements of mobile and static devices (section 2.2.1). These measurements can be used to adjust the NPK-PUFF outcome in order to get the best possible prediction of a radioactive plume. A data assimilation approach was used to combine model predictions with measured data (Heuvelink et al. 2008), as described in section 2.2.2.

2.2.1 Simulating measurements of the devices

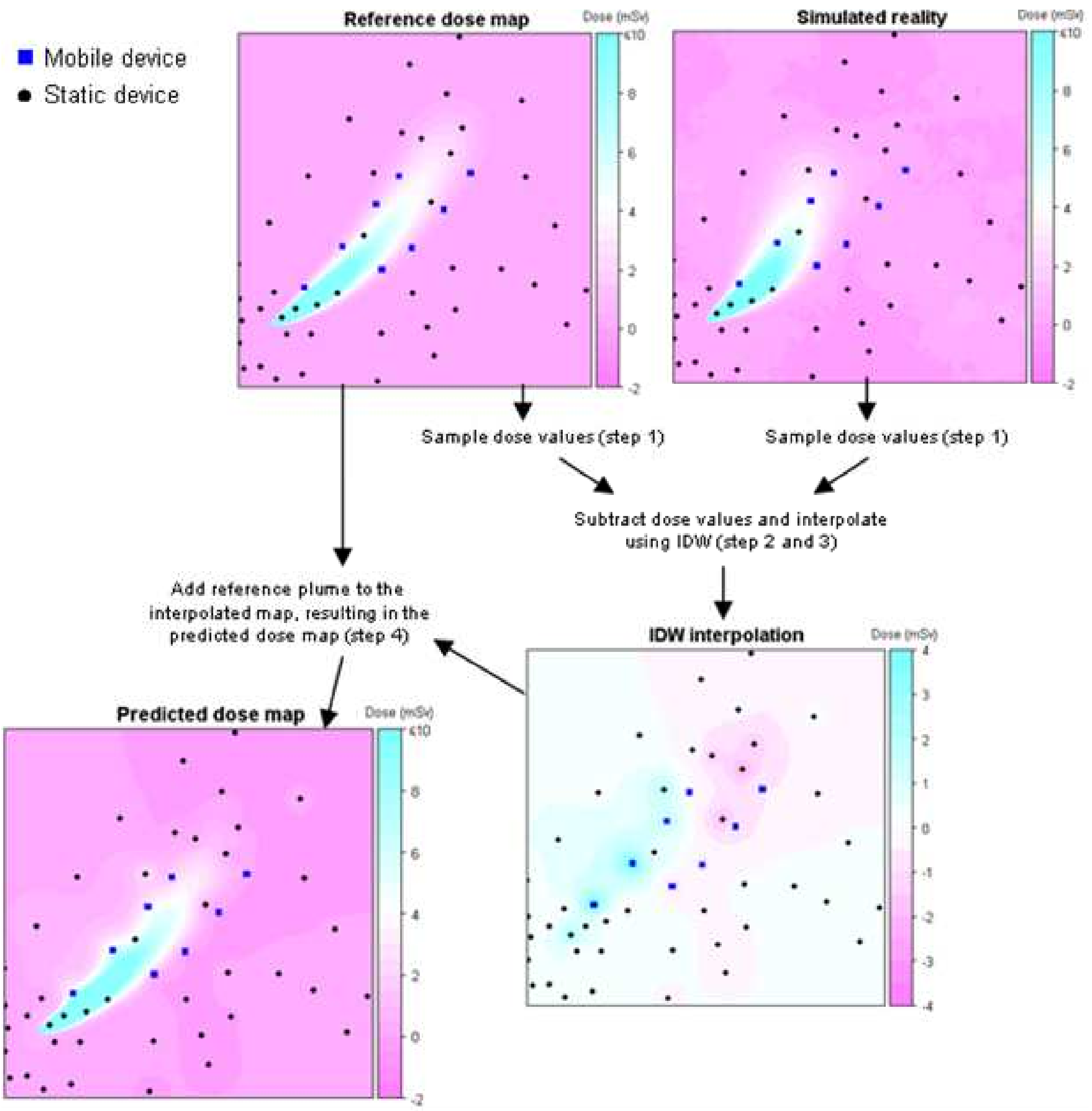
We want to know the values of mobile devices before actually placing them, hence real measurement values of mobile devices are unknown. Real measurement values of the static devices are also unknown, because we are interested in the dose values six hours in the future. Therefore both mobile- and static device measurements had to be simulated. A suitable simulation of measurements of mobile and static devices can be acquired by sampling from the set of simulated realities (see 2.1). This set of “real” dose maps represent possible dose values six hours ahead, thus sampling measurements from points on these maps provides realistic unknown dose values.

2.2.2 Using simulated measurements to adjust the predicted plume

After obtaining (simulated) values of the devices the reference NPK-PUFF dose map was used to improve the quality of the predicted map. This was done by taking the following steps:

1. Sample the dose value at the locations of measuring devices; both at the reference dose map and the simulated reality.
2. Subtract the sampled values of the reference dose map from the sampled values of the simulated reality.
3. Interpolate the subtracted dose values using Inverse Distance Weighting (IDW), resulting in a map representing the deviations from the reference plume.
4. Add the interpolated map to the reference map, resulting in a **predicted dose map**.

Figure 7 illustrates the above steps to create a predicted map from the reference dose map and the simulated reality.



Mobile device
Static device
Reference dose map
Dose (mSv)
Simulated reality
Sample dose values (step 1)
Sample dose values (step 1)
Subtract dose values and interpolate using IDW (step 2 and 3)
IDW interpolation
Add reference plume to the interpolated map, resulting in the predicted dose map (step 4)
Predicted dose map

Figure 7: Methodology to create the predicted dose map

2.3 Optimisation criterion

2.3.1 Introducing the cost function

The optimization of the locations of mobile measurement devices (the sampling design) seeks to minimize uncertainty in the predicted plume. As Jiang (2007) mentioned, the best estimation may not be the result with the lowest spatially averaged error, because the decision-maker could be interested more in minimizing the costs of making a wrong decision due to uncertainty in the prediction. Therefore Jiang (2007) introduced an optimisation criterion based on the idea of *false positive values* (predicted dose value larger than the intervention level while the real value was smaller than the intervention level) and *false negative values* (predicted dose value smaller than the intervention level while the real value was larger than the intervention level). This distinction between an overestimation (false positive) and underestimation (false negative) of the dose was made because these result in different costs. An overestimation could lead to unnecessary actions, like evacuating a town where the dose does not exceed the intervention level, whereas an underestimation could lead to serious dangers for the health of people in an area where the intervention level is unknowingly exceeded. This resulted in the following cost function (Jiang 2007):

$$C(S) = E[\alpha \cdot A_{\text{false positive}} + \beta \cdot A_{\text{false negative}}] \qquad [2]$$

Here, C(S) is the expected costs at a single time instant with sampling design S; E is the mathematical expectation (the expected value of the function); α and β are the cost impact parameters and A represents the area occupied by false predictions.
Figure 8 illustrates this approach.

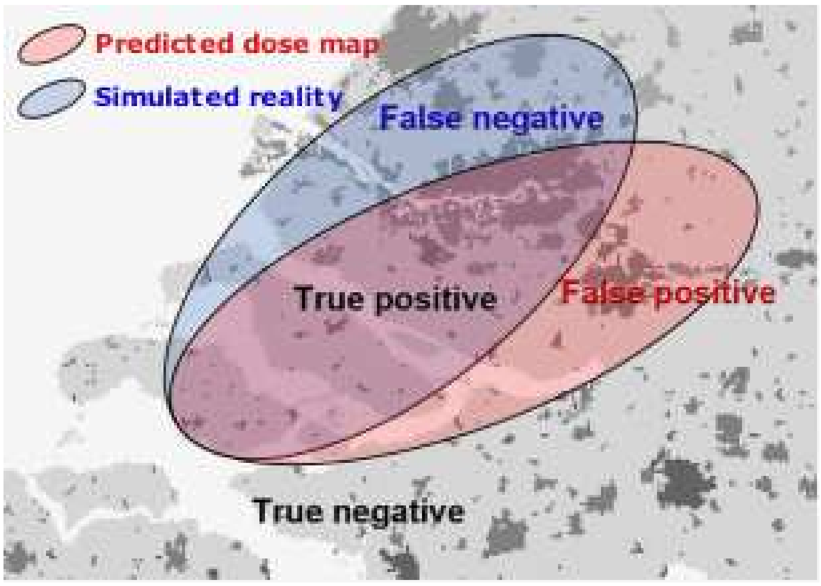


Figure 8: The costs are based on differences between the predicted dose map and the simulated reality

2.3.2 Extending the cost function with a population density map

Equation [2] provided the basis for a more elaborate cost calculation. This extended cost calculation also took into account population density, because the costs of a wrong decision in a densely populated area are higher than in a scarcely populated area. Therefore a map was created, which consisted of weights resembling the population density. This was done by combining information from a table with the total population size of each Dutch municipality, a polygon map with the borders of the municipalities and a polygon map with built-up areas. The following steps were taken to create a suitable, weighted population density map:

1. Add the population information of the municipality to the municipality map.
2. Intersect the built-up polygons with the municipality polygons, resulting in polygons of built-up area containing the name and number of inhabitants of the corresponding municipality.
3. Calculate the total size of built-up area of each municipality.
4. Divide the number of inhabitants of the municipality with the total built-up area of the corresponding municipality, resulting in a population density map.
5. Create a grid of the population density map, using the population density to set the values of the grid cells.
6. Reclassify the population density grid values to create a weighted population density map. Water was given a weight of 0, non-built-up areas weight 1 and built-up areas a weight between 5 and 11, depending on the population density. Figure 9 shows a cut-out of the weighted population density map.

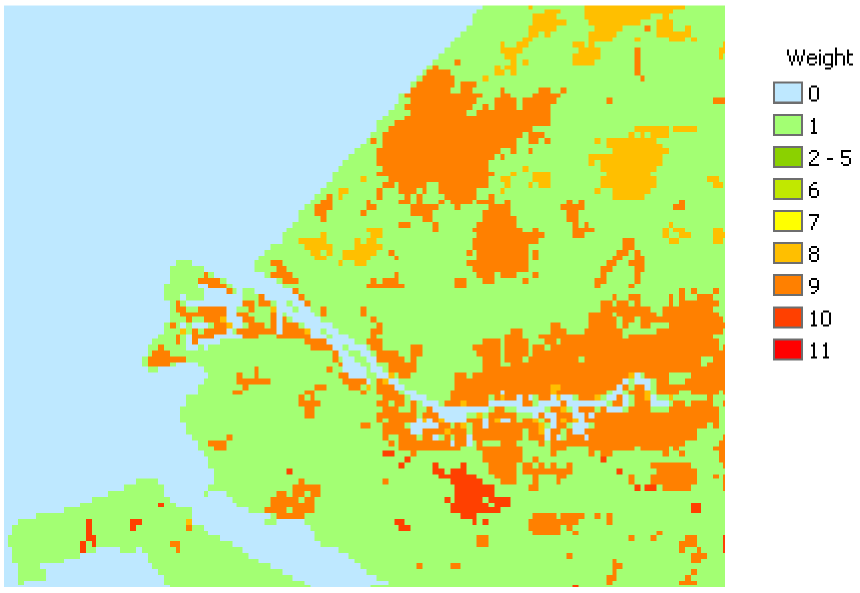


Figure 9: Cut-out of the weighted population density map

2.3.3 Calculating the costs associated with the sampling design

The intervention levels are based on the effective dose, as the effective dose determines the effects of the radioactive plume on the population. The unit of the effective dose is the sievert (Sv). We chose a (fictitious) intervention level of 2.5 mSv, which is in the same order of magnitude as the intervention level for sheltering, 5 mSv (Bader[4], pers. comm). NPK-PUFF is not able to directly determine the effective dose, therefore the effective dose had to be calculated from the NPK-PUFF output. Exact dose calculations are very complicated (Bader[4], pers. comm). However, for this research a rough estimate was good enough, because predicting exact dose values was not the main purpose of our methodology. Therefore we only focused on the inhalation dose - i.e. effective dose from inhalation - which accounts for about 70% to more than 90% of the total effective dose in the first phase after an accident, depending on circumstances (Bader[4], pers. comm). The inhalation dose was calculated with the following equation (ICRP 1995; ICRP 1996):

$$D[Sv] = TIC[Bq.h/m^3] \cdot Q[m^3/h] \cdot DCC[Sv/Bq] \qquad [3]$$

D is the dose, TIC the Time Integrated Concentration at ground level, Q the human breathing rate and DCC the Dose Conversion Coefficient. The reference value for Q is 1 m^3/h. DCC's vary for different nuclides, absorption classes and standards. For ^{131}I a DCC of 7.4e-9 Sv/Bq was selected. This is valid for calculating the adult effective dose from inhalation, according to the EU-base regulations (ICRP 1995; ICRP 1996). The NPK-PUFF model was used to compute the TIC, thus the effective dose was

[4] S. Bader, RIVM, Bilthoven, Personal Communication, January 15, 2008

calculated by multiplying the NPK-PUFF output by Q * DCC (= 1 * 7.4e-9). In this study the values of all NPK-PUFF outputs were converted from time integrated concentration to effective dose using equation 3, enabling the comparison of the plume values with the intervention level.

In order to calculate the average costs of a sampling design, the costs for all N realisations (the combinations of simulated realities and predicted dose maps) had to be calculated. This was done by taking the following steps:

1. For each cell, check if it is false negative or false positive by comparing the predicted map with the simulated reality.
2. If the cell is false negative or false positive, obtain the weight from the population density map and multiply this weight with the cost impact parameter, resulting in the cost of this cell.
3. Add the cell-cost to the total costs of this realisation.

After calculating the costs of all realisations, the average costs could be computed by dividing the total costs by the number of realisations.

2.4 Optimising the locations of mobile devices

Jiang (2007) applied a method to find the sampling design with the least costs by using simulated annealing (Kirkpatrick et al. 1983). Simulated annealing (SA) is an algorithm which was developed to solve non-linear optimization problems. Van Groenigen (1998) extended the SA algorithm to optimise spatial sampling schemes. He describes three core components of the spatial simulated annealing (SSA) algorithm:

- Fitness function; calculates the optimization criterion, defined by the cost-function (2.3).
- Generation mechanism; randomly perturbs a new sampling design by moving one sampling point.
- Probabilistic acceptance criterion; chance of accepting a new sampling design.

First our implementation of SSA and the three core components are discussed in section 2.4.1. Section 2.4.2 deals with the practical issue of creating a map with suitable locations for mobile devices, which limited the movement of sampling points by the generation mechanism.

2.4.1 Spatial Simulated Annealing (SSA)

Our implementation of the SSA algorithm consisted of the following steps:

1. Start with an arbitrary sampling design.
2. Construct a new sampling design by changing the location of one of the mobile devices.
3. Create N predicted dose maps (see 2.2).
4. Calculate the average costs using the optimisation criterion (see 2.3).
5. Compare these costs with the costs of the previous sampling design.
6. Accept the new design if current cost are lower than previous. If the new design is worse then accept it with a probability defined by the probabilistic acceptance criterion.
7. A new sampling design is chosen and loop back to step 1. The SSA-algorithm finishes when the maximum amount of iterations is reached.

Figure 10 shows the total distances over which mobile devices were moved for one SSA process of 1500 iterations. The movement is done by first selecting two random values between a predefined minimum and maximum shift size (one for the x and one for the y dimension) and then adding these shifts to the current location of a randomly selected sample point. As can be seen from this figure the maximum shift size decreases as iterations continue, which follows from the assumption that the sampling design gets closer to an optimum with increasing iterations, and therefore moving sampling points over large distances would not be helpful (Van Groenigen and Stein 1998).

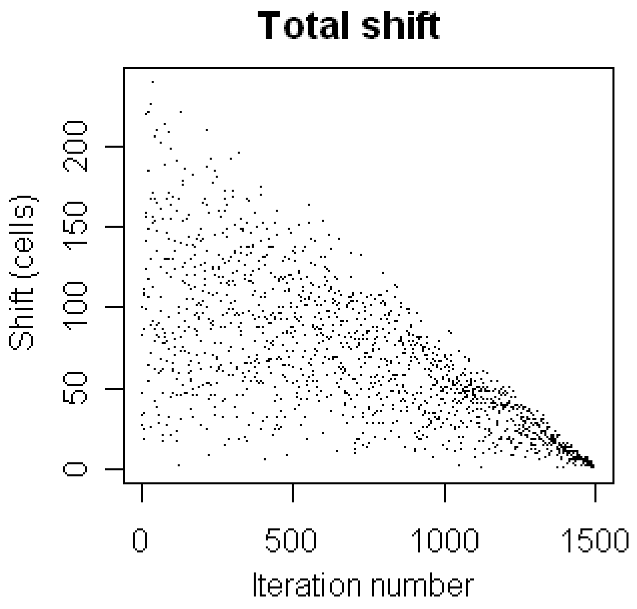


Figure 10: Distances over which mobile devices were moved to create a new sampling design

The cooling schedule denotes the progress of the optimisation. This optimisation takes place by calculating the costs of a newly perturbed sampling design with the fitness function, and accepting this design if the costs are lower. However, the design can also be accepted with a low probability if the costs are higher. The chance of accepting a new sampling design $P_c(S_i \rightarrow S_{i+1})$ was determined by the probabilistic acceptance criterion [4] (adapted from (3.6) from (Van Groenigen and Stein 1998)):

$$P_c(S_i \rightarrow S_{i+1}) = 1, \quad \text{if } C(S_{i+1}) \le C(S_i)$$

$$P_c(S_i \rightarrow S_{i+1}) = 0.2 \times \exp\left(\frac{-10 \times n}{N}\right), \quad \text{if } C(S_{i+1}) > C(S_i) \qquad [4]$$

N represents the total number of iterations, n is the current iteration and C(S_i) the fitness (or cost) function.

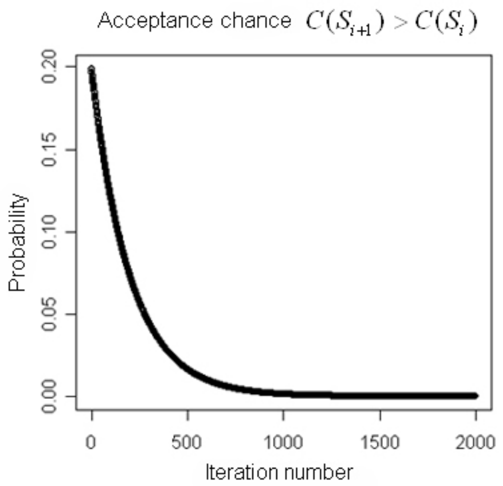


Figure 11: Acceptance probability when costs of the new sampling design are higher

2.4.2 Selecting candidate sampling locations

In SSA, at every iteration a new sampling design was created by moving one of the mobile devices. Not all locations however were suitable for placement of mobile devices. The locations had to be close to a road and on open land. Therefore a suitable locations map was created, which consisted of a grid with the spatial extent of the study area. Each cell had either the value 0 (not-suitable) or 1 (suitable). Two topographic vector data sets (a polygon map with the land cover objects and a line map representing the road network) were used to implement the geographical constraints. The map was made as follows:

1. Select suitable land covers from the polygon land cover map; these were all areas except water and forest.
2. Create a buffer of 300 meter (the maximum distance from a road which a mobile device can be placed) around all paved roads.
3. Intersect the buffer and suitable land covers, creating a polygon map with all suitable areas.
4. Rasterise the polygon map with suitable areas, and set these cell values to 1.
5. Set the remaining cell values to 0.

Figure 12 shows the suitable locations map. The dense Dutch road network made it possible to place the devices on most land areas.

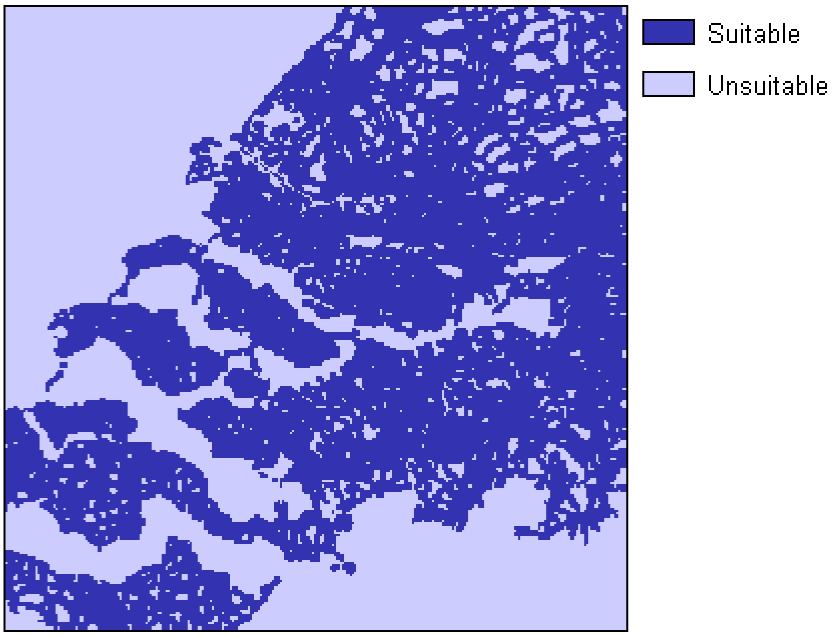


Figure 12: Suitable locations map

3 Results

In order to test the optimisation procedure many SSA-runs with varying settings were done. Appendix 3 shows the cost and settings of all results; interesting comparisons are highlighted with different colours. Varying these settings gave different results and helped to gain insight in the end results. The main parameters that were varied are (in brackets is the abbreviation used to refer to these parameters):

- Number of mobile devices (**nrMob**)
- SSA: number of iterations (**nrIter**)
- SSA: number of simulated realities (**nrSim**)
- SSA: maximum shift size; the maximum distance at the start of the SSA-run that the mobile device could be moved (see 2.4.1).This distance decreased every iteration (**maxShift**)
- IDW: inverse distance power (**IDP**)
- IDW: maximum distance; only devices within this distance from the location were used for prediction (**maxDist**)
- False negative cost impact parameter (see equation [2]): the weight of the areas that were wrongly classified by underestimation (**FNweight**)

This chapter provides an overview of the most important results. First the results of the implemented methodology are given by showing optimised sampling designs in section 3.1. Next the results of SSA-runs with different IDW-parameters are shown in section 3.2. Section 3.3 shows the effects of varying the optimisation criterion.

3.1 Optimisation of the sampling design

Figure 13 shows a graph of the development of the total costs for an SSA-run with 1500 iterations. It is clearly visible that the acceptance criterion ([4]) enables the approval of a sampling design with higher costs, especially in the beginning (see also Figure 11). With increasing iterations, costs are eventually lower than before, which indicates that simulated annealing is able to escape local optima (Van Groenigen and Stein 1998).

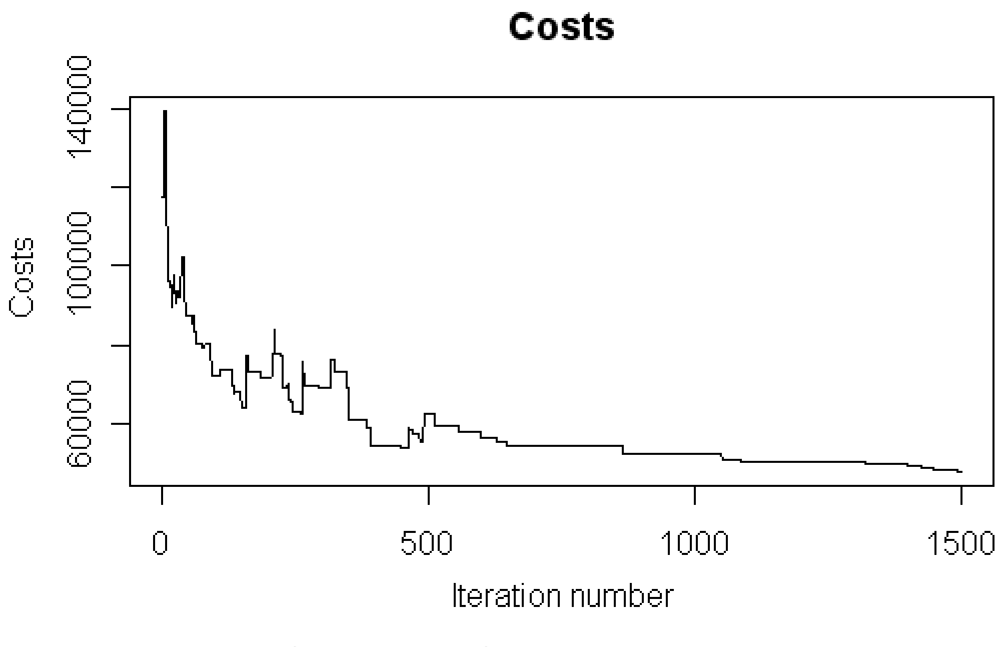


Figure 13: Development of the total costs of an SSA-run

Figure 14 shows the result of an SSA-run. The following settings for this optimisation procedure were used: nrMob = 8; nrIter = 1500; nrSim = 80; maxShift = 170; IDP = 3; maxDist = 20; FNweight = 5. The mobile devices are placed at the edge of the reference map threshold (depicted by the grey line), where the probability of making a wrong decision is large. Furthermore most devices are located in the north, in order to reduce the costs around the populated areas of Rotterdam and surrounding cities. The static devices are also considered in the optimisation procedure; i.e. the mobile devices are placed at gaps in the monitoring network and never near a static device.

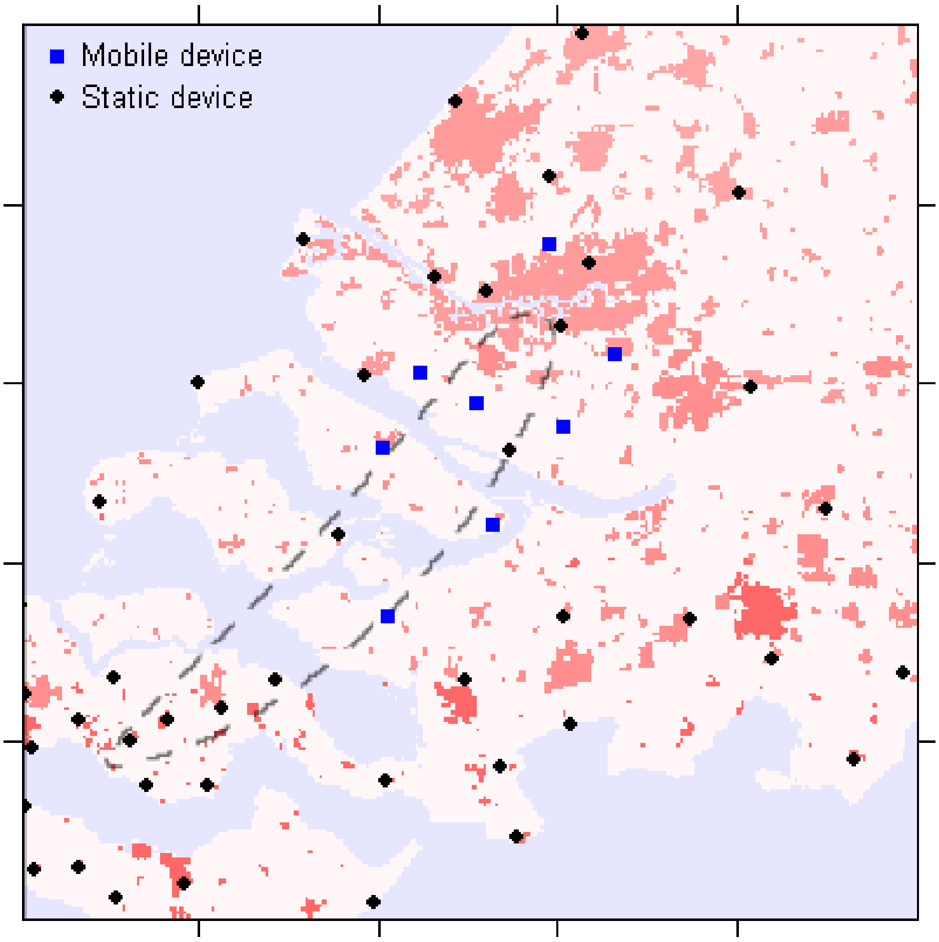


Figure 14: Optimised locations of eight mobile devices

3.2 Varying the IDW-parameters

In order to get the best possible results out of the SSA optimisation procedure, different IDW-settings were tested. Figure 15 shows the start and end cost maps of an SSA-run, with the following settings: nrMob = 8; nrIter = 1500; nrSim = 50; maxShift = 200; IDP = 2; maxDist = unlimited; FNweight = 5. Thus there was no limitation to the distance that sampled dose values affect the estimation of surrounding locations. The costs are shown in red and the blue crosses depict the mobile devices. Be aware that the costs are relative values and dependent on the amount of simulated realities (nrSim), the cost impact parameters and the use of the population density map. On the left a random sampling design prior to optimisation, which illustrates the costs resulting from deviations of the simulated realities and the reference dose map very well. Consequently the shape of the reference plume can easily be distinguished.

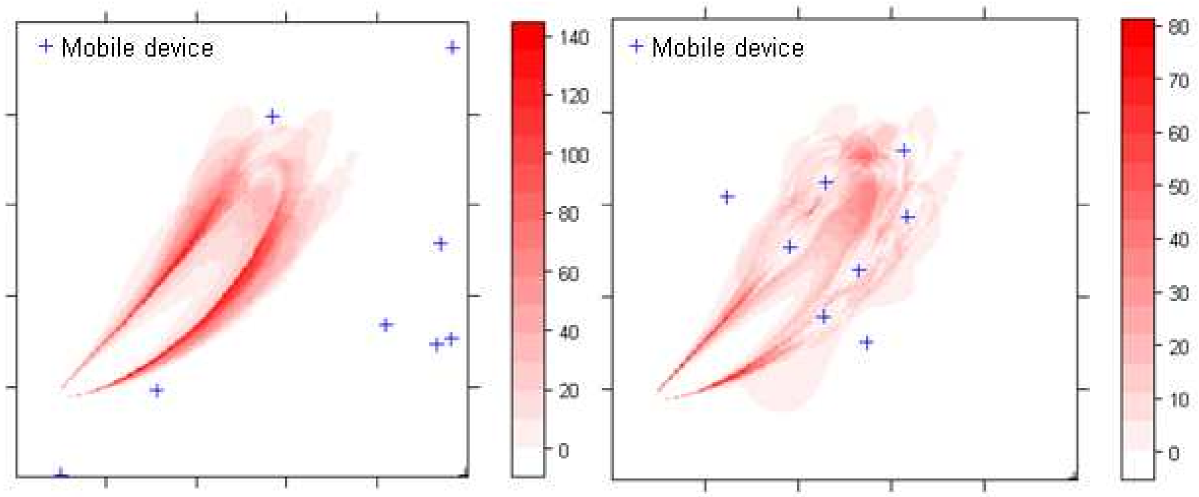


Figure 15: Comparison of the start- and end cost map of an SSA-run with no maxDist; on the left the random start, on the right the final sampling design

Figure 16 shows the final sampling designs with varying maximum distance values used for the IDW-interpolation. All other settings were fixed and the same as used for the SSA-run with no limitations to the maxDist (see Figure 15).

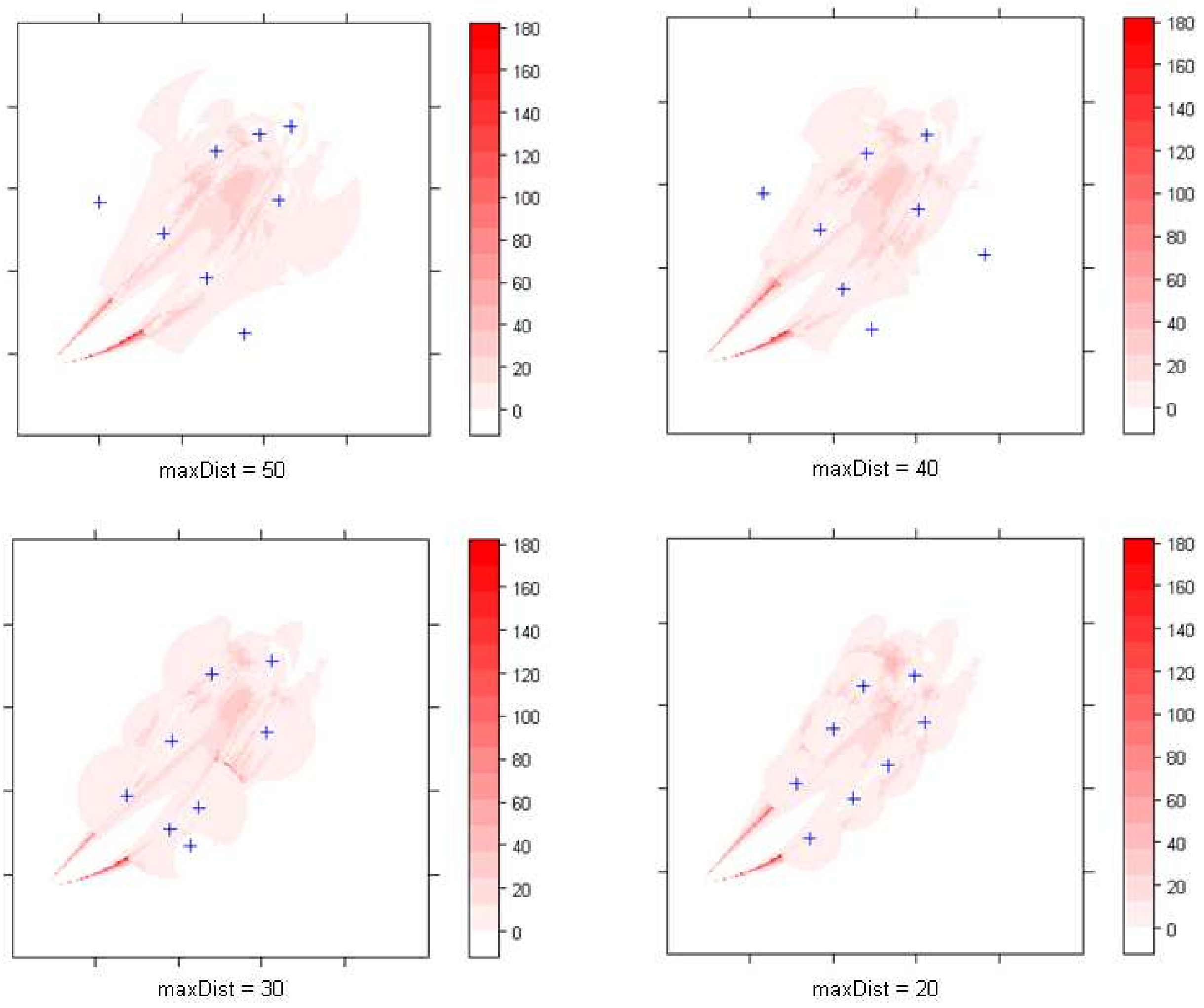


Figure 16: Overview final sampling designs with varying maxDist values and a fixed IDP of 3

The sampling designs show an interesting pattern. It seems that higher maximum distances resulted in mobile devices being placed outside the area of possible plumes. Apparently placing devices in the plume sometimes resulted in higher costs than placing them outside the plume. Only for a maxDist of 30 and 20 the devices were placed more sensible. The resulting average costs were:

no maxDist:	2422
maxDist 50:	2453
maxDist 40:	2381
maxDist 30:	2207
maxDist 20:	2114

Next to the maxDist parameter, the Inverse Distance Power (IDP) could also be varied. In IDW, the influence of a sampling point (i.e. the location of a device) on the estimation of the value at the surrounding locations was determined by its weight. The weights of the sampling points are inversely proportional to the distance, but the weights can be inversely proportional to any power of the distance (Isaaks and Srivastava 1989). The costs of three SSA-runs with different IDPs and the same remaining settings were as follows:

IDP 2:	2160
IDP 3:	2075
IDP 4:	2131

Thus decreasing the maxDist to a size in which there was often only one device responsible for prediction and using a large inverse distance power resulted in the lowest costs. This resembled a nearest neighbour interpolation. It is not what we anticipated, and will be discussed in section 4.2. In order to make fair comparisons, the IDW-settings for all further SSA-runs were an IDP of 3 and a maxDist of 20.

3.3 Varying the cost calculation settings

It was important to analyse the effect of changing the optimisation criterion on the determination of the optimal sampling design. First the results of varying the cost impact parameters are shown, to check whether the mobile designs were actually placed in a way that the probability of a false negative prediction was the smallest (3.2.1). Then the effect of extending the cost calculation by including a population density map is shown (3.2.2).

3.3.1 Comparing different cost impact parameters

In order to provide insight in the costs of a sampling design when varying the cost impact parameters, we needed to visualize both false negative and false positive costs. Therefore two different cost maps of each sampling design were created; one showing the probability (between 0 and 1) of making a false negative decision and a similar map for false positive decisions. Figure 17 shows the resulting sampling designs with the cost impact parameters FNweight = 1 and FNweight = 5. The other settings were as follows: nrMob = 8; nrIter = 1500; nrSim = 50; maxShift = 200; IDP = 3; maxDist = 20;

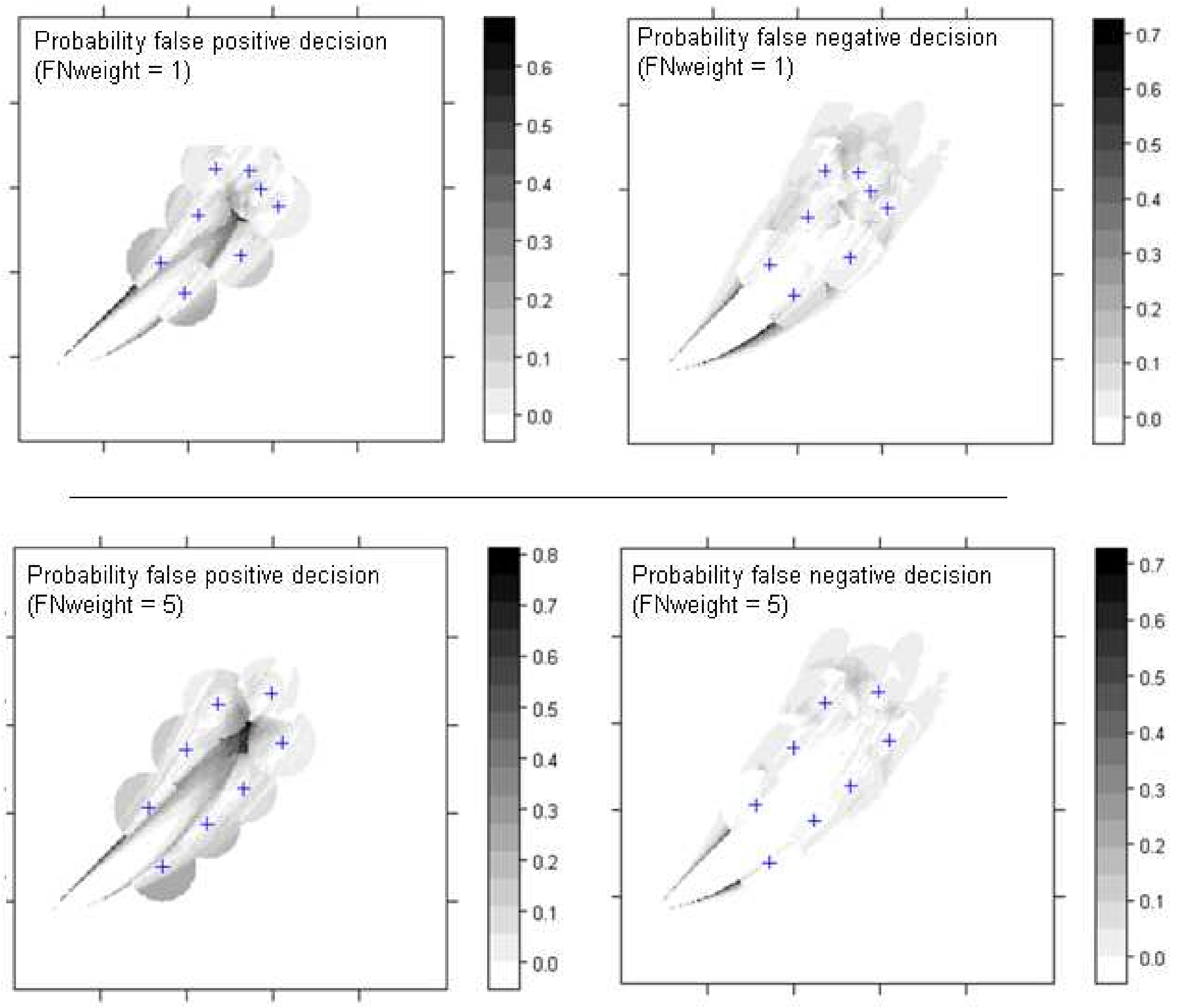

Probability false positive decision
(FNweight = 1)
Probability false negative decision
(FNweight = 1)
Probability false positive decision
(FNweight = 5)
Probability false negative decision
(FNweight = 5)

Figure 17: Probability cost maps for FNweight 1 (above) and FNweight 5 (below)

The resulting probability maps show that the chance of making a false positive decision with an FNweight of 5 is higher than making a false negative decision, especially compared with the probability maps with FNweight 1. Another way of comparing the placement of mobile devices was overlapping both sampling designs (Figure 18), with the intervention level (threshold) of the reference plume on the background. It can be inferred from Figure 18 that, on average, an FNweight of 5 resulted in the placement of mobile devices a bit outside the threshold of the reference plume. For the FNweight of 1 this was not the case. Apparently the probability of making a false negative decision was smaller when the mobile devices were placed just outside of the reference plume threshold. Similar results were found by Jiang (2007).

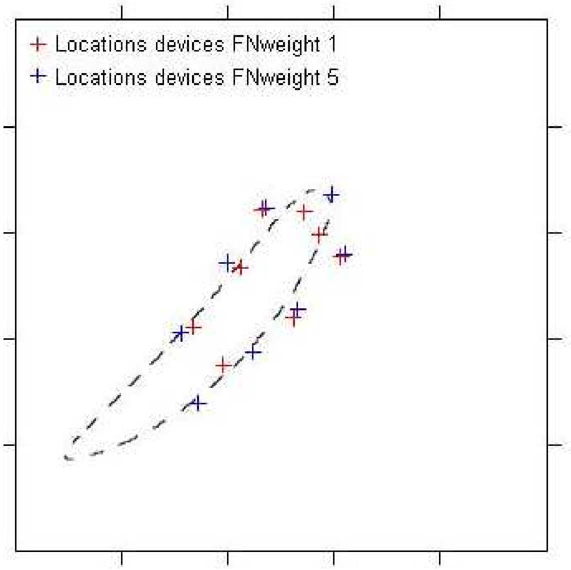


Figure 18: Comparison results SSA-run FNweight 1 (red) and FNweight 5 (blue)

3.3.2 Comparing results with and without population density map

Next to distinguishing between the costs of an over- and underestimation of the dose around the intervention level, a distinction was made between making a wrong decision in a populated and less populated area.

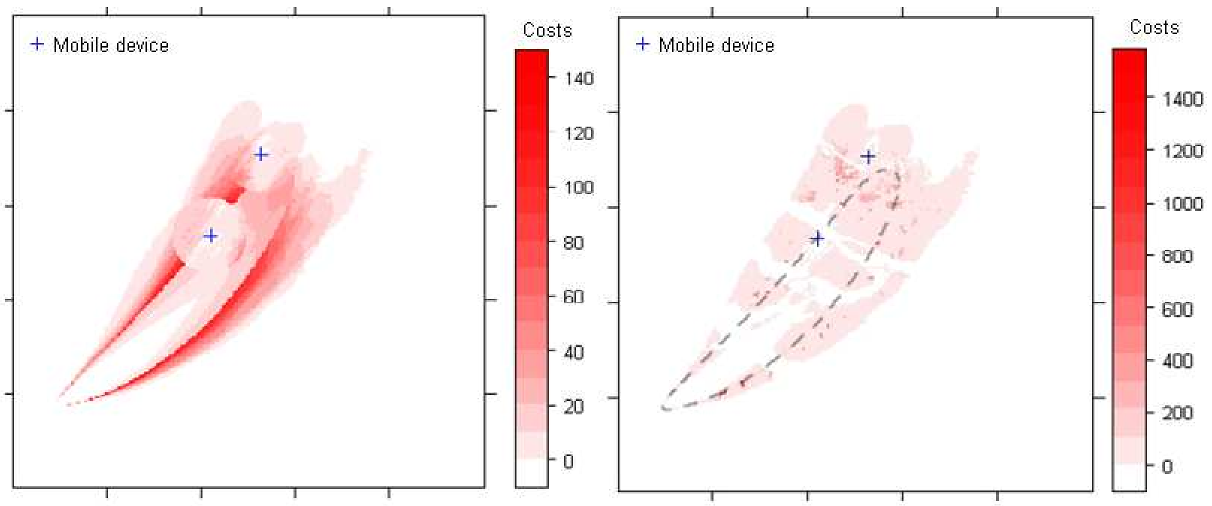


Figure 19: Cost maps without (left) and with (right) population density taken into account

Figure 19 shows the effects of considering population density on the cost calculation. To better visualise the costs, only two devices were used in this example. On the left cost map population density is not taken into account, resulting in smooth cost areas.

The right cost map shows the effect of the use of a population density map in the cost calculation. The grey line is the threshold of the reference dose map and shown for orientation. The non-inhabited areas (sea and rivers) in the areas around the intervention level can easily be derived, as they did not have any costs. These are the white areas between the red areas, whereas the highly populated areas can be distinguished by the deep red colour.

Figure 20 shows the effect of adding population density to the cost-function on the optimisation of the sampling design. The settings for both SSA-runs were:
nrMob = 8; nrIter = 1500; nrSim = 80; maxShift = 170; IDP = 3; maxDist = 20; FNweight = 5.
When population density was not taken into account (left cost map), the devices were placed in a uniform way around the threshold of the reference plume. However, when population density did affect

the optimisation (right cost map), the devices were placed more to the North, in the highly populated area of Rotterdam and its surrounding cities.

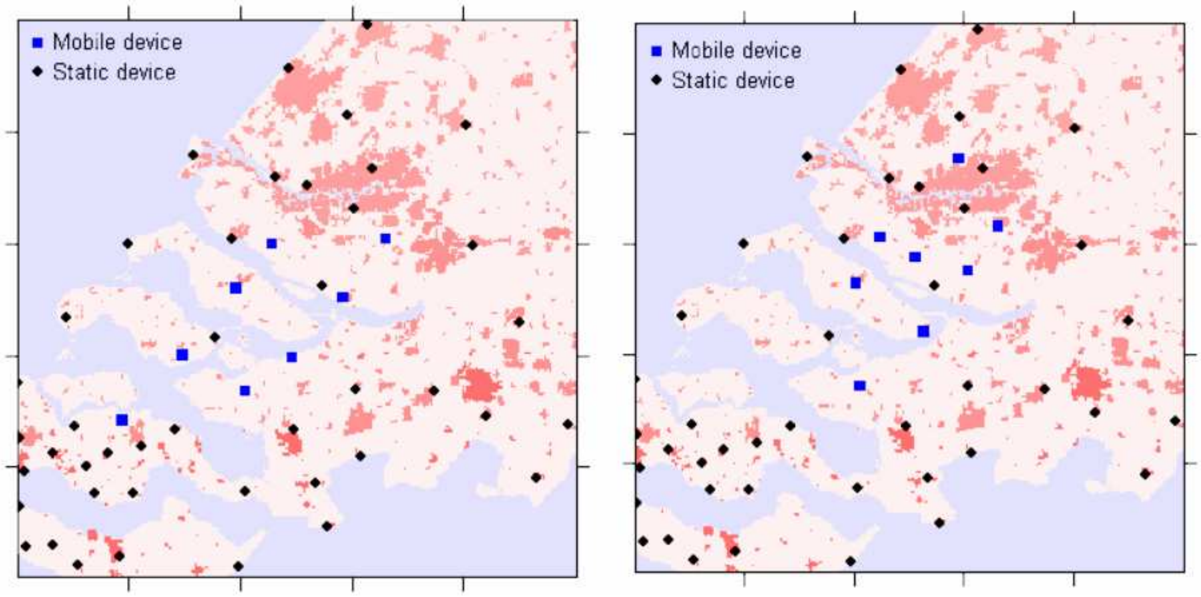


Figure 20: Results SSA-run without (left) and with (right) population density taken into account

3.4 Computation time

In case of an emergency, the locations of the mobile devices have to be predicted within half an hour (Heuvelink et al. 2008). Within this time the following computations have to be made:

1. Create input files for the NPK-PUFF model with the HIRLAM data of the current and predicted meteorological circumstances (five minutes)
2. Run the NPK-PUFF model with all these input files (four hours).
3. Create the simulated realities by loading NPK-PUFF output files and adding the simulated model error residual (one hour).
4. Execute the SSA-algorithm with 1500 iterations (42 hours).

The time mentioned in brackets gives an estimation of the current computation time for 100 simulated realities. The calculations were executed on a PC with an Intel Dual Core 2 CPU, 1.68 Ghz and 3 GB of RAM, using Windows XP as operating system. In total the whole procedure took about two days. Obviously this is much too long for operational use. Although the creation of a practical implementation was not the main purpose of the thesis, this is a serious problem which should be addressed and taken into account for future research.

4 Discussion of results

4.1 Optimisation of the sampling design

The optimisation procedure of the sampling design is dependent on the settings of the SSA-parameters, i.e. the number of simulated realities, the amount of iterations, the maximum shift (the maximum distance a mobile device can be moved to generate a new sampling design, see 2.4.1) and the probabilistic acceptance criterion ([4]). Fine tuning these parameters in order to get the best results was a delicate process, because some parameters affect each other.

The maximum shift size was chosen in a way that all mobile devices will be placed in an area where there are costs. If the maximum shift is too small, then there is a probability that one or more mobile devices get stuck in an area without costs and thus are of no use (e.g. when placed on an island with no suitable locations nearby). This seems unlikely with for example 1500 iterations, but considering that there are about eight mobile devices, only one device is moved every iteration and the shift size decreases each iteration, it becomes more probable (see Figure 10 for absolute values of the shift sizes of one SSA-run). However, lowering the maximum shift size in general results in a better sampling design as there are more iterations for fine-tuning the device-locations. Thus a balance had to be found between a large enough maximum shift size to locate all mobile devices in the right area, but small enough to optimise the location. Furthermore, the decrease in maximum shift size is dependent on the number of iterations: increasing the maximum number of iterations could decrease the maximum shift size, because there are more iterations with large enough values to avoid getting stuck in no-cost areas.

Another questionable component of SSA is the probabilistic acceptance criterion. What should be the probability of accepting a worse design and how should this probability decrease in time? We chose for an exponential function [4], which resulted in an increasingly slower decreasing probability of accepting a worse design (see Figure 11). This acceptance criterion function was judged by looking at the development of the costs (one example is Figure 13). The resulting graphs show that, especially at the start, the costs regularly increase and thus a worse design is accepted. But eventually the costs are always lower than at the start, which indicates the evasion of possible local optima. However, this did not prove that this acceptance criterion was the most suitable and the question remains open for discussion. Similarly to the maximum shift size, increasing the number of iterations enables decreasing the chance of accepting a worse design, as more iterations result in more possible evasions of local optima.

4.2 Varying the IDW-parameters

Before adjusting the IDW-parameters, the outputs of SSA-runs gave seemingly erroneous results. Mobile devices were placed far outside potential cost areas (= areas where dose values of simulated realities are near the intervention level and deviate from the reference plume). How could this be? If the SSA worked properly, this meant that the costs had to be lower when one or more devices were placed outside cost-areas. After testing several sampling designs it indeed seemed the case that a mobile device could lower the costs of a sampling design more when placed outside of the plume, instead of being placed at a logical location in the high cost areas. Moreover, the costs could even be higher when one or multiple extra mobile devices were placed in a high cost area. Figure 21 shows the effect of adding two devices (pointed out by the arrows) in a cost area. When looking closely at the redness around the devices it can be seen that the costs in the immediate vicinity of the added mobile devices are lowered, but further away are sometimes higher. This could mean that the total costs increased.

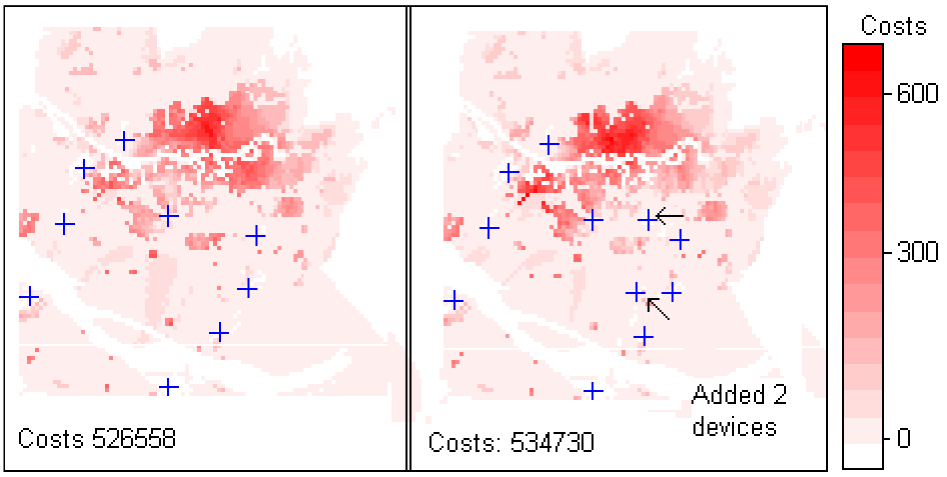


Figure 21: Addition of two devices could result in higher costs by affecting areas further away from the devices

Clearly, this was not anticipated and resulted in peculiar sampling designs which should not be used in reality. In order to find the reason for these results the process of calculating the costs was explored. In short, costs are the result of a wrong prediction of the dose value around the intervention level, calculated by comparing the predicted dose maps with the simulated realities. The better the prediction, the lower the costs. Could it be that the predicted map became worse after adding a device? Figure 22 provides insight in this question. It shows two cut-outs of the absolute difference of the dose values of a predicted map and a simulated reality. On the left the initial situation is shown, on the right the effect of the addition of a mobile device in an area where the prediction could be improved. Upon close inspection it can be seen that the addition of one device (indicated by the black arrow) improves the prediction in the immediate neighbourhood, but makes the prediction further away (in this case east of the added device) worse. This can lead to higher costs if the prediction further away results in an over- or underestimation of the intervention level.

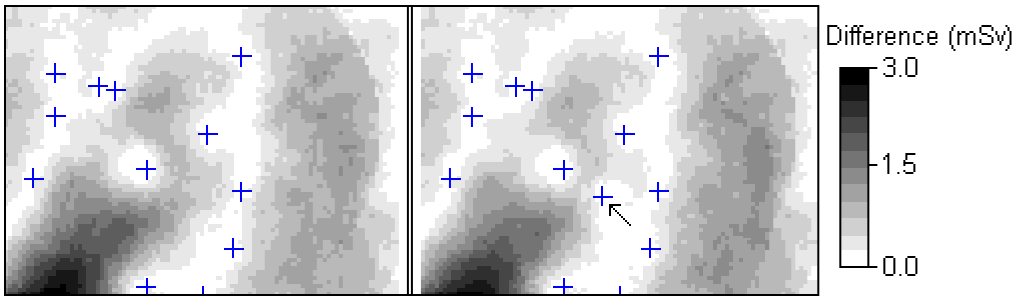


Figure 22: Effect of adding one device on absolute difference between predicted map and simulated reality

Apparently the IDW interpolation has it drawbacks. The cause of the problem is the influence of devices relatively far away from the estimated location. In short, a device that is closest to a high-cost area is not the only device that influences the prediction there, also a device with a possibly very different dose a bit further away influences the prediction. Increasing the Inverse Distance Power (IDP) parameter improves the situation, as this gives higher weights to devices most nearby. Another way to overcome possible wrong influences on the prediction is to set the maximum distance parameter. This ensures that in a sparsely sampled area (e.g. near Belgium) one device cannot have influence on the prediction at the edge of the plume far away.

This explains the results from section 3.2, where setting the IDW-parameters maxDist and IDP in a way that the IDW-interpolation resembles nearest neighbour interpolation resulted in the lowest costs and the most logical sampling designs. This approach greatly decreased the influence of other measurements further away from the plume.

An explanation for these difficulties in interpolating the measurements is the spatial distribution of dose values, i.e. the shape of the plume. There are high dose values in the centre of the plume, but a few cells away (especially nearby the source) the values drop rapidly. Thus there is much variation at small distances, which makes it troublesome to adjust dose values based on observations far away.

4.3 Varying the cost calculation settings

The costs are the result of differences in dose values around the intervention level between the predicted dose map and the simulated reality, thus the more the reference dose map is adjusted to the simulated reality, the better the prediction. Around the devices the costs are lower, as the reference dose map can be adjusted with information of the “true” dose from the simulated reality obtained by the mobile measuring device.

Improving the prediction however does not necessarily result in lower costs, as we are purely interested in making the right decision; whether or not the dose value is above or below the intervention level. The prediction can be exceedingly wrong, but if both the predicted- and the simulated reality dose value are below or above the intervention level there are no costs. This explains that there are no costs at the centre of and far away from the reference plume, but only at the edges of the reference plume where dose values are close to the intervention level. Therefore the devices are most effective when placed in these areas.

The cost impact parameter influences the placement of devices. When the false negative weight is five and the false positive weight is one, the mobile devices are placed just outside of the reference plume threshold (see Figure 18). This can be explained as follows: when sampling dose values just outside the threshold of the reference map, the sampled reference dose value is smaller than the intervention level but the simulated reality value could be larger than the intervention level. When the value of the simulated reality is larger than the threshold, the adjustment of the reference dose map results in a predicted map with values around the device which are larger than the threshold. In case the simulated reality values further away from the device are larger than the threshold this resulted in an overestimation (false positive).

Similarly, it can also be the case that the sampled simulated reality value just outside the threshold of the reference map is smaller than the intervention level. Then the adjustment of the reference map results in lowering the values around the device. In this case, both the values of the predicted and the simulated reality map are larger than the intervention level and there are no costs. Thus the probability that there is an underestimation (false negative) is smaller than the probability of an overestimation (false positive).

5 Conclusion and recommendations

Research objective
The overall objective of this study was similar to the work of Jiang (2007): to find a methodology to optimise the locations of mobile measuring devices (sampling designs) in order to minimize the risk of making a wrong decision. This study also used spatial simulated annealing to determine the best sampling design; i.e. the sampling design with the lowest costs according to the optimisation criterion. The research questions are discussed below.

What are the sources of uncertainty in the prediction of the radioactive plume?
Several extensions of the work of Jiang were successfully implemented. A major refinement was extending and applying knowledge about the sources that contribute to uncertainty in the prediction of the radioactive plume. This was achieved by modelling pdfs of uncertainty in the input parameters of the NPK-PUFF model. Using these pdfs many possible developments of meteorological circumstances and source conditions were simulated. Subsequently the NPK-PUFF model was run, resulting in a set of model outputs. To account for the uncertainty in the model itself, a stochastic residual was generated and added to the NPK-PUFF model output. This resulted in a set of simulated realities which played a considerable role in the optimisation process.

Many simplifications were made in determining the probability distribution functions of the source uncertainty variables. The most important simplification was leaving out uncertainty in the spatial distribution of meteorological parameters, which was necessary due to the limited input-possibilities of the NPK-PUFF model. This was a serious drawback as for example wind speed and wind direction are not the same over the whole study area. Therefore we still added a fairly large residual to the NPK-PUFF outputs in order to create the simulated realities (equation [1]). It is likely that in the near future the spatial resolution of the meteorological input data of the NPK-PUFF model will be increased (Kok 2005), which is a first step towards a possible implementation of spatial variability.

Another simplification was omitting the correlation between different meteorological variables. There are many complicated relationships and interactions between these variables. For example, mixing layer height has a decisive influence on the temperature profile in the atmosphere, which affects the relationship between wind direction at different heights (Bijvoet et al. 1979). Another example is the influence of atmospheric stability on the development of mixing layer height (Wieringa and Rijkoort 1983). Modelling correlations between meteorological variables is a complicated task and should preferably be done by an expert in the fields of meteorology and spatial statistics.

Furthermore the method used to simulate temporal variation in atmospheric stability was not ideal, as Pasquill values are on an ordinal scale and we treated them as if it was a continuous variable. Perhaps a better way to model atmospheric stability would be to use the Monin-Obukhov length, which is a more recent and suitable method to indicate the stability (Kok 2005). Another solution could be to use a more appropriate way to simulate temporal variation of Pasquill values, e.g. by using Markov-chains to model the probability of transitions between different Pasquill-classes (Heuvelink[5], *pers. comm.*).

Next to the meteorological parameters the release strength was difficult to model, as we were unable to determine its probability distribution function. Therefore a fixed value of 1e+17 Bq was chosen. Furthermore we focussed on the release of only one nuclide (^{131}I), which was not realistic but a logical first step in modelling the source uncertainty.

Another point for consideration is uncertainty in the model error. Because our knowledge of model uncertainty was limited, the model error was created by unconditional Gaussian simulation using a variogram model. However, both the simulation and the variogram model were mere speculations about the actual model error. Researching the parameters of the model error was problematic as no real disaster took place to test the NPK-PUFF model. Kok (2005) did a sensitivity analysis of the NPK-PUFF model using concentration data from the Kincaid-experiment, a controlled "accident" to research dispersion of a tracer gas. However, the results of his sensitivity analysis did not provide insight in the size and spatial distribution of the difference between the NPK-PUFF outcome and the true measured values. Therefore this analysis was not suitable to infer useful information about the model error.

[5] G.B.M. Heuvelink, Wageningen University and Research Centre, Wageningen, November 16, 2007

How can measurements of mobile devices be used to locate the predicted plume more accurately?
Difficulties in creating predicted dose maps resulted in deficient sampling designs. The predicted map is created by adjusting the reference dose map with information from measurements obtained from simulated realities. This is done by first subtracting the sampled dose value (at the locations of the devices) of the simulated reality from the values of the reference dose map, then these subtracted values are interpolated with Inverse Distance Weighting (IDW) and finally this map is added to the reference dose map. Unfortunately the Inverse Distance Weighting interpolation did not always improve the overall prediction. The reason for the inadequacy of IDW is the spatial distribution of dose values of the plume; i.e. the variability and spatial continuity are dependent on the shape of the plume, especially in cost areas at the edge of the plume. Here the mobile devices have maximum effect. IDW takes the values of surrounding measurements into account, which can decrease the quality of the prediction in the cost areas. Therefore adapting the IDW settings in a way that only the nearest measurement influences the prediction (i.e. gets a weight of 1) results in the lowest costs, and thus in the best sampling design. We highly recommend to improve the interpolation method, as the IDW did not gave satisfactory results.

How can the costs of wrong decisions associated with the sampling design be calculated?
The optimisation criterion was successfully enhanced by taking population density into account. This criterion was used to assess the fitness of the sampling design and to make comparisons between different sampling designs possible. Extending the criterion with population density resulted in a preference for placing mobile devices in densely populated areas.

Another extension to the work of Jiang (2007) was the implementation of dose calculations in order to comply with more realistic intervention levels used by the RIVM. This calculation is fairly straightforward; the dose is calculated by multiplying time integrated concentration (directly calculated by NPK-PUFF) with a Dose Conversion Coefficient (equation [3]). This study only focused on the dispersion of ^{131}I, however in reality multiple nuclides will be released in case of a nuclear accident. Furthermore the calculation of the effective dose was greatly simplified. In order to adapt the methodology for practical purposes, both aspects should be considered.

What is a suitable methodology to optimise the locations of mobile devices?
Another important extension to the work of Jiang (2007) was the addition of the static device network in the optimisation procedure, which greatly improved the realism of the whole procedure. Results clearly showed that mobile devices were placed as an addition to the static device network, which is exactly their purpose. The optimisation of the sampling design by SSA worked well. The optimisation criterion, the perturbation of new sampling designs and the acceptance criterion were successfully implemented. The resulting sampling designs were not always satisfactory, this however was mainly because of the mentioned difficulties in updating the predicted dose map with IDW. The SSA-algorithm tries to find the sampling design with lowest costs, and if a sampling design with a mobile device placed outside the plume results in low costs the SSA-algorithm is not the reason for faulty sampling designs. However, there is room for improvement of the SSA-parameters by fine-tuning the generation of new sampling designs and the probability of accepting a worse sampling design. Increasing the number of simulated realities and iterations will also provide better results, but brings us to a drawback of the SSA-algorithm: the computation time. With a relatively small amount of 100 simulated realities and 1500 iterations it took about 40 hours to finish the SSA-procedure. However, there are many methods to speed up the computation which could dramatically lower the total time to calculate the optimal sampling design. Examples are the use of a faster computer, implement parallel computing (Heuvelink et al. 2008), write a more efficient algorithm and using a coarser grid for computationally demanding tasks.

The two major recommendations are to further explore uncertainty in the meteorological predictions and NPK-PUFF model, and to improve the update of the predicted map with point observations. In spite of the mentioned problems the research objective has been reached. This methodology to optimise locations of mobile measurement devices provided promising and interesting results. It has the potential to be used in reality; not only for radioactive monitoring networks, but for various situations in which measurements of mobile devices could be used to improve model predictions.

6 References

About HIRLAM programme (n.d.). Retrieved December 20, 2007 from http://www.hirlam.org

Bijvoet, H. C., Frantzen, A. J., Schmidt, F. H., van Ulden, A. P., Velds, C. A. and Wisse, J. A. (1979). Luchtverontreiniging en weer. de Bilt, Staatsuitgeverij 's-Gravenhage.

Deutsch, C. V. and Journel, A. G. (1998). GSLIB: Geostatistical Software Library and User's Guide. Oxford, Oxford University Press.

Eleveld, H., Kok, Y. S. and Twenhöfel, C. J. W. (2007). "Data assimilation, sensitivity and uncertainty analyses in the Dutch nuclear emergency management system: a pilot study." International journal Emergency Management **4**(3): 551-563.

Heuvelink, G. B. M., Brown, J. D. and Van Loon, E. E. (2007). "A probabilistic framework for representing and simulating uncertain environmental variables." International journal of geographical information science **21**(5): 497-513.

Heuvelink, G. B. M., Jiang, Z., de Bruin, S. and Twenhöfel, C. J. W. (2008). "Optimization of Mobile Radioactivity Monitoring Networks." International Journal of Geographical Information Science **(Submitted)**.

ICRP (1995). "Age-dependent Doses to Members of the Public from Intake of Radionuclides: Part 4, Inhalation Dose Coefficients." Annals of the ICRP 25 (3/4) **ICRP Publication 71**.

ICRP (1996). "Age-dependent Doses to Members of the Public from Intake of Radionuclides: Part 5, Compilation of Ingestion and Inhalation Dose Coefficients." Annals of the ICRP 26 (1) **ICRP Publication 72**.

Isaaks, E. H. and Srivastava, M. R. (1989). An introduction to Applied Geostatistics, Oxford University Press.

Jiang, Z. (2007). Optimization of Mobile Radioactivity Sampling Designs. MGI. Wageningen, Wageningen University.

Karssenberg, D. and De Jong, K. (2005). "Dynamic environmental modelling in GIS: 2. Modelling error propagation." International journal of geographical information science **19**(6): 623.

Kirkpatrick, S., Gelatt, C. D. and Vecchi, M. P. (1983). "OPTIMIZATION BY SIMULATED ANNEALING." Science **220**(4598): 671-680.

Kok, Y. S. (2005). Analyses van gevoeligheden en onzekerheden van het luchtverspreidingsmodel NPK-Puff, LSO/RIVM**:** 62.

Palmer, M. (n.d). "Environmental Variables in Constrained Ordination (e.g. CCA, RDA, DCCA)". Retrieved December 11, 2007 from http://ordination.okstate.edu/envvar.htm

Pebesma, E. J. (2004). "Multivariable geostatistics in S: the gstat package." Computers & Geosciences **30**(7): 683-691.

Smetsers, R. C. G. (2005). Radiologische basiskennis kernongevallenbestrijding, RIVM.

Smetsers, R. C. G. and van Lunenburg, A. P. P. A. (1994). "Evaluation of the Dutch radioactivity monitoring network for nuclear emergencies over the period 1990-1993." Radiation Protection Dosimetry **55**(3): 165.

Twenhöfel, C. J. W., de Hoog van Beynen, C., van Lunenburg, A. P. P. A., Slagt, G. J. E., Tax, R. B., van Westerlaak, P. J. M. and Aldenkamp, F. J. (2005). "Operation of the Dutch 3rd Generation National Radioactivity Monitoring Network." Spatial interpolation comparison 2004 report: 19-34.

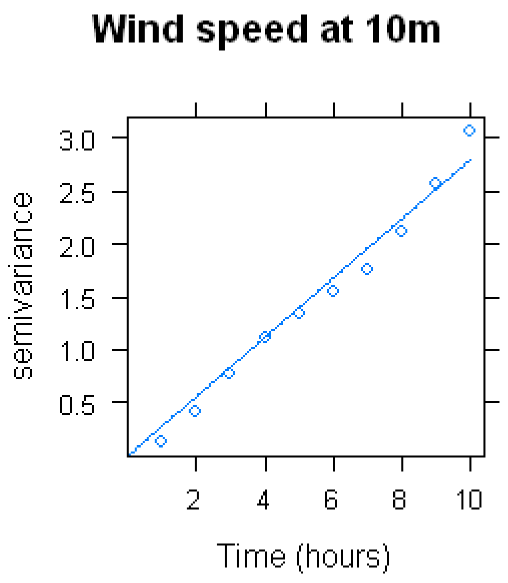
Van Groenigen, J. W. and Stein, A. (1998). "Constrained optimization of spatial sampling using continuous simulated annealing." Journal of Environmental Quality **27**(5): 1078-1086.

van Pul, W. A. J. (1992). Technical description of the RIVM/KNMI PUFF dispersion model. Bilthoven, National Institute for Public Health and Environment.

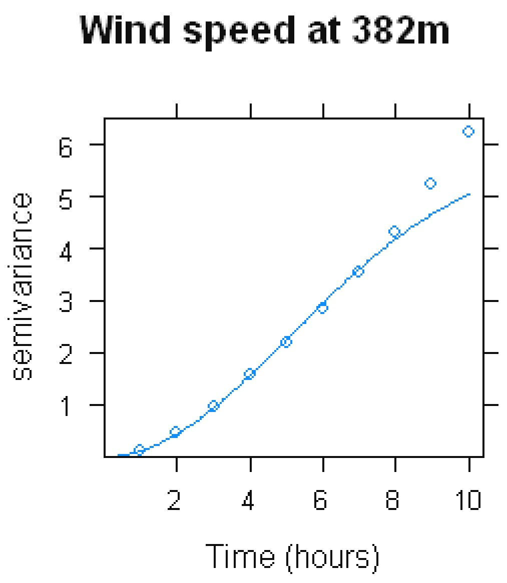
Verver, G. H. L., de Leeuw, F. A. A. M. and van Rheineck-Leyssius, H. J. (1990). Description of the RIVM/KNMI puff dispersion model. Bilthoven, de Bilt, RIVM, KNMI**:** 53.

Wieringa, J. and Rijkoort, P. J. (1983). Windklimaat van Nederland, Staatsuitgeverij, Den Haag.

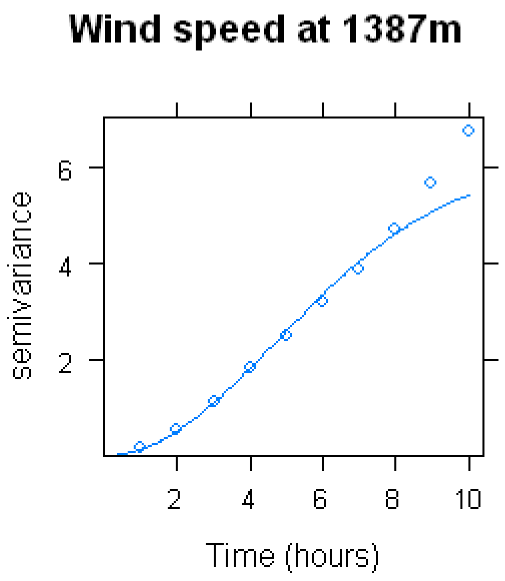
Appendix I: Variogram models of the meteorological parameters



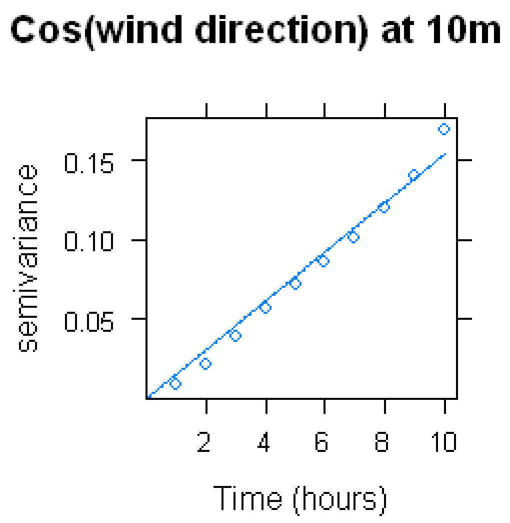
Variogram model = 2.8Linear(10)



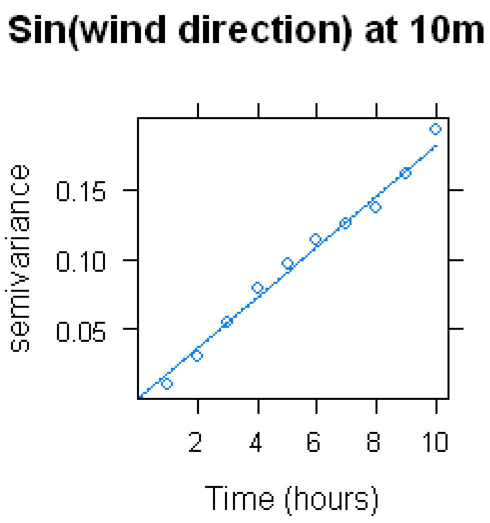
Variogram model = 5.9Gaussian(7.2)



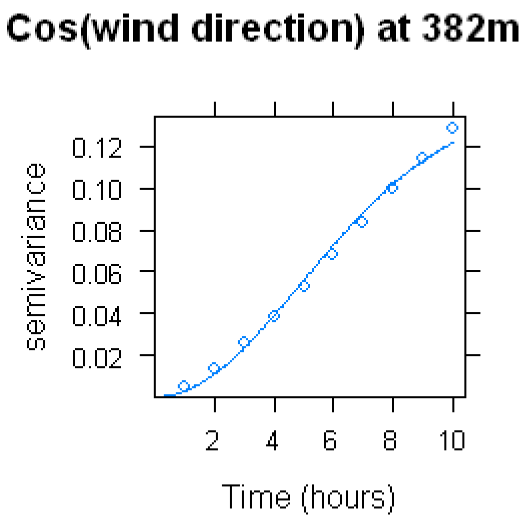
Variogram model = 6.1Gaussian(6.7)



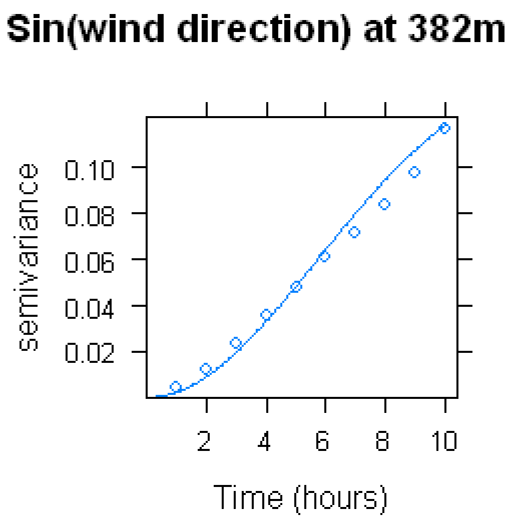
Variogram model = 0.2Linear(13)



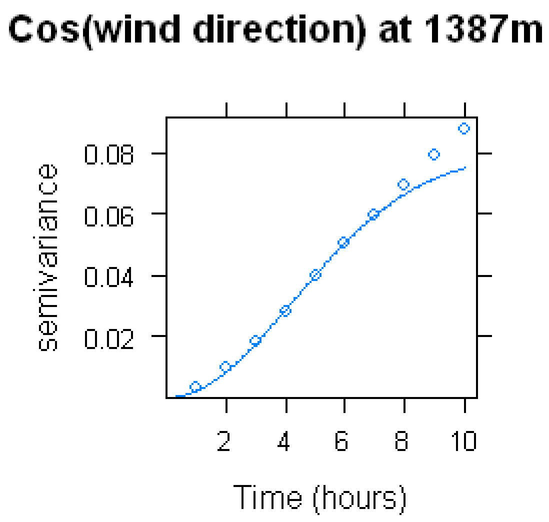
Variogram model = 0.2Linear(11)



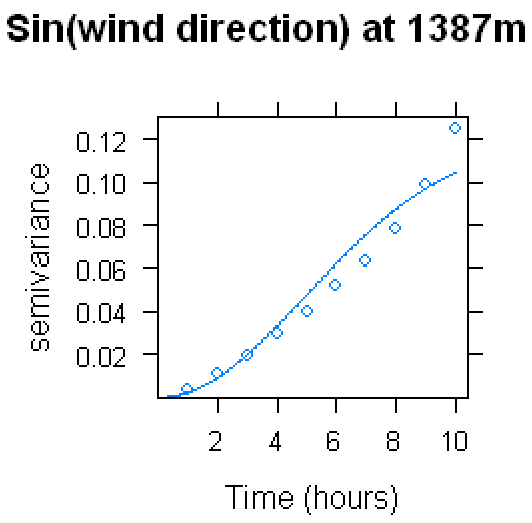
Variogram model = 0.14Gaussian(7)



Variogram model = 0.15Gaussian(8)

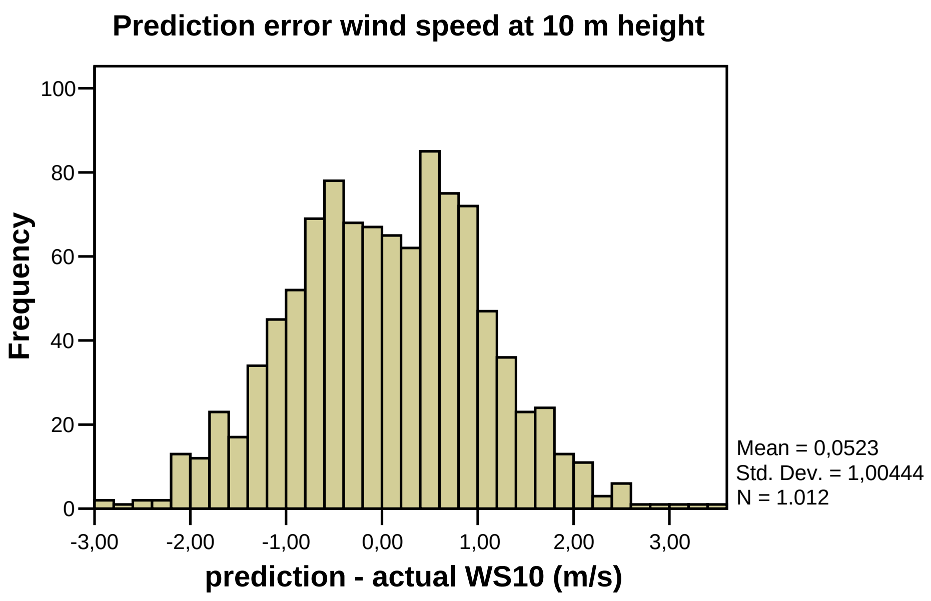


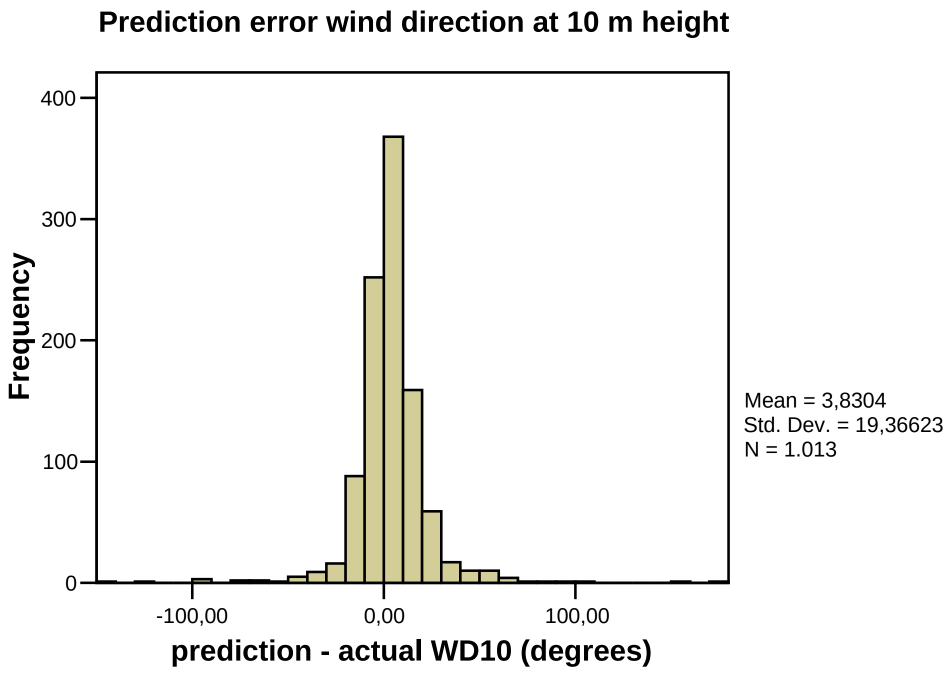
Variogram model = 0.08Gaussian(6)



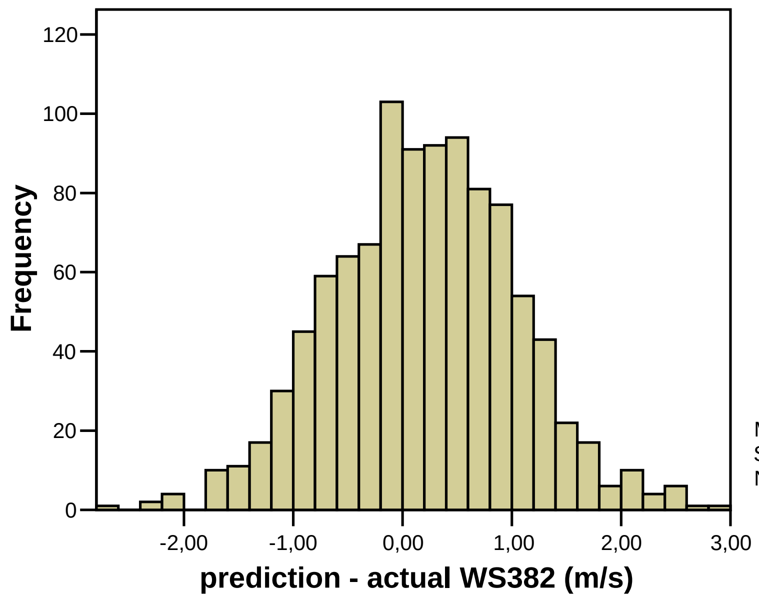
Variogram model = 0.12Gaussian(7)

Appendix II: HIRLAM prediction uncertainty

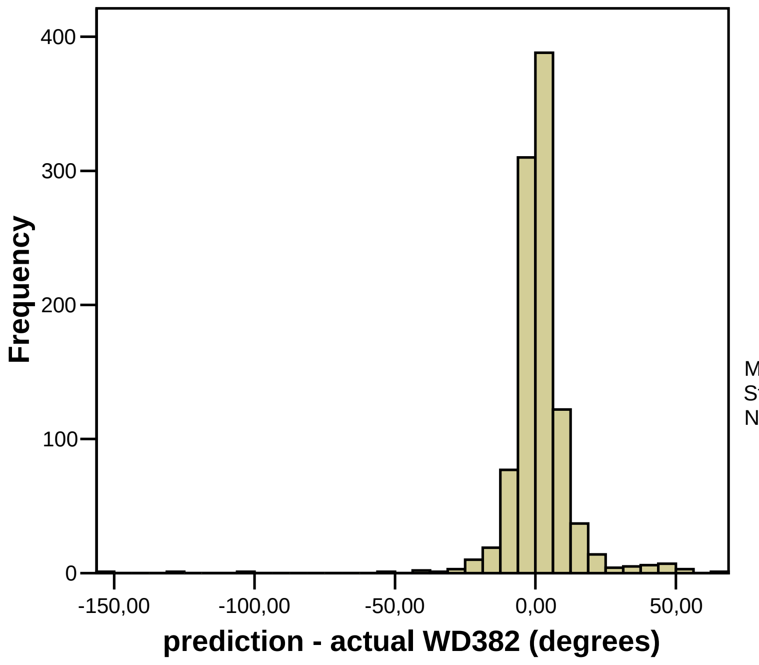




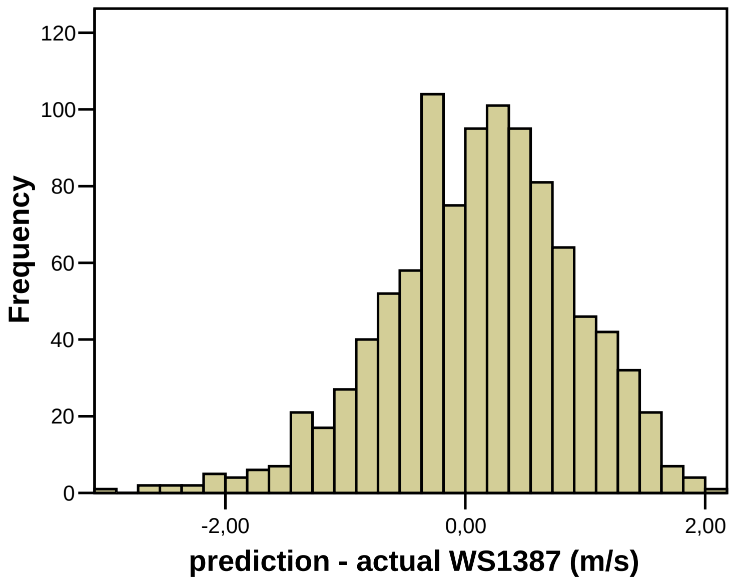
Prediction error wind speed at 382 m height



Prediction error wind direction at 382 m height

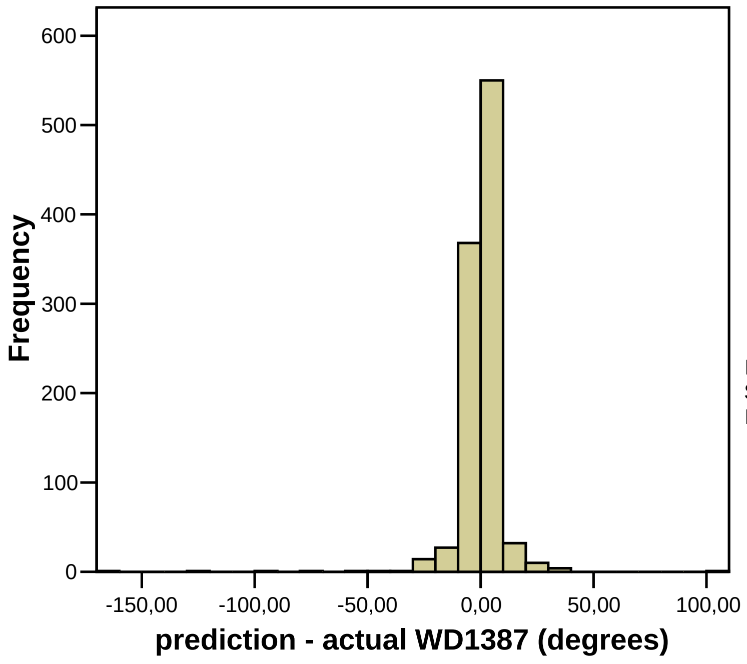


Prediction error wind speed at 1387 m height

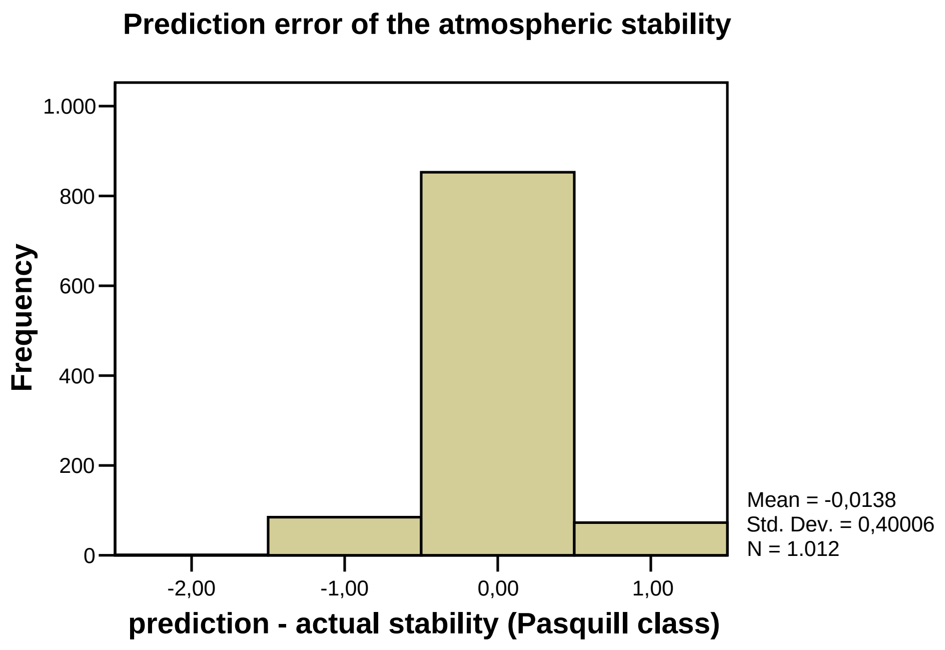


Mean = 0,1147
Std. Dev. = 0,77908
N = 1.012

Prediction error wind direction at 1387 m height



Mean = 0,4381
Std. Dev. = 11,00592
N = 1.013


Prediction error of the atmospheric stability
Frequency
1.000
800
600
400
200
0
-2,00
-1,00
0,00
1,00
prediction - actual stability (Pasquill class)
Mean = -0,0138
Std. Dev. = 0,40006
N = 1.012

Appendix III: Overview results SSA-runs

Static included?	Population Density	nrMob.	FNweight	nrSim.	nrIter.	MaxDist	IDP	maxShift	Avg. Costs
No	No	16	5	70	2500	Unlimited	10	250	1734
no	no	16	5	70	2500	Unlimited	4	250	1882
yes	No	10	5	60	1500	Unlimited	2	250	2977
No	no	8	5	50	1500	20	3	200	2114
no	no	8	5	50	1500	30	3	200	2207
no	no	8	5	50	1500	40	3	200	2381
No	No	8	5	50	1500	50	3	200	2453
No	No	8	5	50	1500	Unlimited	3	200	2422
yes	No	8	5	50	1500	20	2	200	2160
yes	no	8	5	50	1500	20	3	200	2075
yes	no	8	5	50	1500	20	4	200	2131
No	No	8	5	50	1500	20	2	200	2143
No	No	8	5	50	1500	20	4	200	2202
No	No	8	1	50	1500	20	3	200	963
No	No	8	5	50	3000	20	3	200	2234
No	No	8	5	100	1500	20	3	200	2184
No	yes	10	5	70	1500	20	3	170	3444
no	yes	10	5	70	1500	20	3	170	3681
No	yes	10	5	70	1500	20	3	170	3428
No	yes	8	5	100	2000	20	3	170	4023
No	yes	12	5	100	2000	20	3	170	3439
yes	yes	8	5	70	1500	20	3	170	3483
yes	yes	8	5	80	1500	20	3	170	3439
yes	No	8	5	80	1500	20	3	170	2131

Explanation colour scheme:

🟩	Comparison different maximum distances
🟨	Comparison different Inverse Distance Powers (with static devices)
🟪	Comparison different Inverse Distance Powers (no static devices)
🟦	Comparison costs final sampling design with same settings
🟧	Comparison different amount of mobile devices

Appendix IV: Commented R-script for creating NPK-PUFF input files

```
#load libraries
library(gstat)
library(circular)

nrSim = 100;

# time parameters
timeStart = 1; timeAnalysis = 1; timePrediction = 7
timeEnd = 7; nrTimeSteps = timePrediction

# date & time of disaster
startYear = 7; startMonth = 4; startDay = 18; startHour = 12

# grid to simulate time series with a spatial component
timeGrid = expand.grid(x = 1:nrTimeSteps, y = 1:2)
gridded(timeGrid) = ~x + y

#------------------ FUNCTIONS --------------------------------#

checkDegreeInRange = function(obj)
{
  # checks if obj is between 0 and 360 degrees; if not then transform obj

  if(obj < 0)
    obj = obj + 360
  if(obj > 360)
    obj = obj - 360

  return(obj)
}

simulateData = function(currentSim, predSim, vgm)
{
  # simulates values of time steps between current and predicted value

  # create matrix to store results
  simData = matrix(1,nrow = nrSim, ncol = nrTimeSteps)

  for(i in 1:nrSim)
  {
    # create spatial points data frame with current and predicted values
    tempData = data.frame(z=c(currentSim[i],predSim[i]),
    x=c(timeAnalysis,timePrediction), y=c(1.0001,1.0002))
    coordinates(tempData)=~x+y

    # use conditional Gaussian simulation to simulate time step values
    x = krige(z~1, tempData, timeGrid, model = vgm, nsim=1)

    simData [i,] = x$sim1[1:nrTimeSteps]
  }

  return(simData)
}

simulateWDData=function(currentSimX,currentSimY,predSimX,predSimY,cosVgm,sinVgm)
{
  # simulates the values of the time steps between the current and predicted -
  # sine and cosine value, afterwards calculate angle using arctan function

  wdData = matrix(1,nrow = nrSim, ncol=nrTimeSteps)  # create matrix to store data

  for(i in 1:nrSim)
  {
    # create spatial points data frame with current and predicted values
    tempData = data.frame(z=c(currentSimX[i],predSimX[i]),
      x=c(timeAnalysis,timePrediction), y=c(1.0001,1.0002))
       coordinates(tempData)=~x+y

    # use conditional Gaussian simulation to simulate time step values
    dataX = krige(z~1, tempData, timeGrid, model = cosVgm, nsim=1)

    tempData = data.frame(z=c(currentSimY[i],predSimY[i]),
      x=c(timeAnalysis,timePrediction), y=c(1.0001,1.0002))
    coordinates(tempData)=~x+y
    dataY = krige(z~1, tempData, timeGrid, model = sinVgm, nsim=1)

    for(j in 1:nrTimeSteps)
    {
      # calculate angles from the cos and sin values
      angleTemp = deg(atan2(dataY$sim1[j], dataX$sim1[j]))

      if(angleTemp < 0)
        angleTemp = angleTemp + 360

       wdData [i,j] = angleTemp
     }
   }

   return(wdData)
}

#---------------- SOURCE PARAMETERS -------------------------#

# source emission strength: no uncertainty because of scenario-approach
emStrength = 1e17

# if no scenario is used the commented code below can be used
#emStrengthSim = rnorm(nrSim, emStrength, 1e16) # test value
#emStrengthData = as.array(emStrengthSim)    # convert to array

# emission height: set a low or high source
emHighSource = FALSE

if(emHighSource == FALSE)
  emHeightSim = runif(nrSim, 10, 30)

if(emHighSource == TRUE)
  emHeightSim = runif(nrSim, 200, 600)

emHeightData = as.array(emHeightSim)


#----------- METEOROLOGY PARAMETERS -----------------------#

# precipitation
rain = 0.11

#--------------- WINDSPEED ----------------------#

ws10Current = 3; ws10Pred = 4;
sillWs10 = 2.8; rangeWs10 = 10
ws10Vgm = vgm(sillWs10,"Lin", rangeWs10)                # variogram model wind speed
ws10CurrentSim = rnorm(nrSim, ws10Current, 0)
ws10PredSim = rnorm(nrSim, ws10Pred-0.0523, 1.004)
ws10Data = simulateData(ws10CurrentSim, ws10PredSim, ws10Vgm)

# Wind speed 382 meter
ws382Current = 4; ws382Pred = 4;
sillWs382 = 5.9; rangeWs382 = 7.2
ws382Vgm = vgm(sillWs382,"Gau",rangeWs382)              # variogram model wind speed
ws382CurrentSim = rnorm(nrSim, ws382Current, 0)
ws382PredSim = rnorm(nrSim, ws382Pred-0.195, 0.841)
```

```r
ws382Data = simulateData(ws382CurrentSim, ws382PredSim, ws382Vgm)

# Wind speed 1387 meter
ws1387Current = 5; ws1387Pred = 5;
sillWs1387 = 6.1; rangeWs1387 = 6.7
ws1387Vgm = vgm(sillWs1387,"Gau",rangeWs1387)   # variogram model wind speed
ws1387CurrentSim = rnorm(nrSim, ws1387Current, 0)
ws1387PredSim = rnorm(nrSim, ws1387Pred-0.115, 0.779)
ws1387Data = simulateData(ws1387CurrentSim, ws1387PredSim, ws1387Vgm)


#---------------- WIND DIRECTION ---------------#

# Wind direction 10 meter
wd10Current = 250; wd10Pred = 190
wd10CosVgm = vgm(0.2, "Lin", 13)           # variogram model wind direction
wd10SinVgm = vgm(0.2, "Lin", 11)           # variogram model wind direction

wd10CurrentSim = rnorm(nrSim, wd10Current, 0)
wd10PredSim = rnorm(nrSim, wd10Pred-3.83, 19.4)

wd10CurrentSimX = array(1, dim=nrSim); wd10CurrentSimY = array(1, dim=nrSim)
wd10PredSimX = array(1, dim=nrSim); wd10PredSimY = array(1, dim=nrSim)

for(i in 1:nrSim)    # calculate the sin and cos of the current and pred wds
{
  wd10CurrentSimX[i] = cos(rad(wd10CurrentSim[i]))
  wd10CurrentSimY[i] = sin(rad(wd10CurrentSim[i]))
  wd10PredSim[i] = checkDegreeInRange(wd10PredSim[i])
  wd10PredSimX[i] = cos(rad(wd10PredSim[i]))
  wd10PredSimY[i] = sin(rad(wd10PredSim[i]))
}

wd10Data = simulateWDData(wd10CurrentSimX, wd10CurrentSimY, wd10PredSimX,
  wd10PredSimY, wd10CosVgm, wd10SinVgm)

# Wind direction 382 meter
wd382Current = 240; wd382Pred = 200
wd382CosVgm = vgm(0.14, "Gau", 7)
wd382SinVgm = vgm(0.15, "Gau", 8)

wd382CurrentSim = rnorm(nrSim, wd382Current, 0)
wd382PredSim = rnorm(nrSim, wd382Pred-1.54, 12.2)

wd382CurrentSimX = array(1, dim=nrSim); wd382CurrentSimY = array(1, dim=nrSim)
wd382PredSimX = array(1, dim=nrSim); wd382PredSimY = array(1, dim=nrSim)

for(i in 1:nrSim)    # calculate the sin and cos of the current and pred wds
{
  wd382CurrentSimX[i] = cos(rad(wd382CurrentSim[i]))
  wd382CurrentSimY[i] = sin(rad(wd382CurrentSim[i]))
  wd382PredSim[i] = checkDegreeInRange(wd382PredSim[i])
  wd382PredSimX[i] = cos(rad(wd382PredSim[i]))
  wd382PredSimY[i] = sin(rad(wd382PredSim[i]))
}

wd382Data = simulateWDData(wd382CurrentSimX, wd382CurrentSimY, wd382PredSimX,
 wd382PredSimY, wd382CosVgm, wd382SinVgm)

# Wind direction 1387 meter
wd1387Current = 230; wd1387Pred = 210
wd1387CosVgm = vgm(0.08, "Gau", 6)                # variogram model wind direction
wd1387SinVgm = vgm(0.12, "Gau", 7)                # variogram model wind direction

wd1387CurrentSim = rnorm(nrSim, wd1387Current, 0)
wd1387PredSim = rnorm(nrSim, wd1387Pred-0.44, 11)

wd1387CurrentSimX = array(1, dim=nrSim); wd1387CurrentSimY = array(1, dim=nrSim)
```

```
wd1387PredSimX = array(1, dim=nrSim); wd1387PredSimY = array(1, dim=nrSim)

for(i in 1:nrSim)    # calculate the sin and cos of the current and pred wds
{
  wd1387CurrentSimX[i] = cos(rad(wd1387CurrentSim[i]))
  wd1387CurrentSimY[i] = sin(rad(wd1387CurrentSim[i]))
  wd1387PredSim[i] = checkDegreeInRange(wd1387PredSim[i])
  wd1387PredSimX[i] = cos(rad(wd1387PredSim[i]))
  wd1387PredSimY[i] = sin(rad(wd1387PredSim[i]))
}

wd1387Data = simulateWDData(wd1387CurrentSimX, wd1387CurrentSimY,
  wd1387PredSimX, wd1387PredSimY, wd1387CosVgm, wd1387SinVgm)


#------------------ STABILITY  (Monin Obukhov length) -------------------#

stabCurrent = 4; stabPred = 4;
sillStab = 0.62; rangeStab = 10
stabVgm = vgm(sillStab,"Lin", rangeStab)               # variogram model stability
stabCurrentSim = rnorm(nrSim, stabCurrent, 0)
stabPredSim = rnorm(nrSim, stabPred, 0.4)
stabData = simulateData(stabCurrentSim, stabPredSim, stabVgm)

for(i in 1:nrSim)
{
  for(k in 1:nrTimeSteps)
  {
     stabData[i,k] = as.integer(stabData[i,k]+0.5)

     if(stabData[i,k] > 4)
            stabData[i,k] = 4

     # convert to Monin-Obukhov length
     if(stabData[i,k] == 1)
        stabData[i,k] = -6.7

     if(stabData[i,k] == 2)
        stabData[i,k] = -25

     if(stabData[i,k] == 3)
        stabData[i,k] = -100

     if(stabData[i,k] == 4)
        stabData[i,k] = 1000

     if(stabData[i,k] == 5)
        stabData[i,k] = 100

     if(stabData[i,k] == 6)
        stabData[i,k] = 13.3
      }
}

#-------- MIXING LAYER HEIGHT --------------------#

mlhCurrent = 300; mlhPred = 1200
sillMlh = 10000; rangeMlh = 6
mlhVgm = vgm(sillMlh, "Exp", rangeMlh)
mlhCurrentSim = rnorm(nrSim, mlhCurrent, 0)
mlhPredSim = rnorm(nrSim, mlhPred, 50)
mlhData = interpolateData(mlhCurrentSim, mlhPredSim, mlhVgm)


#-------- WRITE OUTPUT/NPK-PUFF INPUT FILES -------------------#

yearStr = "";  monthStr = "";  dayStr = "";  hourStr = "";  endhourStr = ""
```

```r
# make sure the date variables are in the right format; always 2 digits
if(startYear<10) yearStr=sprintf("0%i",startYear) else yearStr=toString(startYear)
if(startMonth<10)monthStr=sprintf("0%i",startMonth)else
monthStr=toString(startMonth)
if(startDay<10) dayStr=sprintf("0%i",startDay) else dayStr=toString(startDay)

for(i in 1:nrSim)
{
  # use a base file to copy input information that isn't changed to output file
  conRead <- file("C:\\Johan\\PuffInputFiles\\puffBase.txt", "r",
    blocking = FALSE)

  # open the output file to write all the data to
  outputFileName = sprintf("C:\\Johan\\runs\\test4\\Inputs\\input%i.txt", i)
  conWrite <- file(outputFileName, "w", blocking = FALSE)

  if(startHour<10) hourStr=sprintf("0%i",startHour) else
    hourStr=toString(startHour)
  startTime = sprintf("%s %s %s %s", yearStr, monthStr, dayStr, hourStr)
  writeLines(startTime, conWrite, sep = "\n")

  endHour = startHour + nrTimeSteps-1
  if(endHour<10)endhourStr=sprintf("0%i",endHour)else endhourStr=toString(endHour)
  endTime = sprintf("%s %s %s %s", yearStr, monthStr, dayStr, endhourStr)
  writeLines(endTime, conWrite, sep = "\n")

  # write 6 lines of base file
  tempData = readLines(conRead, 6)
  writeLines(tempData, conWrite, sep = "\n")

  # write the source height
  tempData = sprintf("%.0f          ! SIMULATED: stack height", emHeightData[i])
  writeLines(tempData, conWrite, sep = "\n")

  # write 2 lines of base file
  tempData = readLines(conRead, 2)
  writeLines(tempData, conWrite, sep = "\n")

  # write the emission strength
  tempData = sprintf("%G     ! SIMULATED: emission first hour", emStrength)
  writeLines(tempData, conWrite, sep = "\n")

  # write 11 lines
  tempData = readLines(conRead, 11)
  writeLines(tempData, conWrite, sep = "\n")

  # write number of time steps
  tempData = sprintf("%i            ! NR. TIME STEPS", nrTimeSteps)
  writeLines(tempData, conWrite, sep = "\n")

  # write meteo data
  for(j in 1:nrTimeSteps)
  {
    hour = startHour+j-1

    hourStr = ""
    if(hour < 10) hourStr = sprintf("0%i", hour) else hourStr = toString(hour)

    tempData=sprintf("%s %s %s %s %.1f %.1f %.1f %.1f %.1f %.1f %.2f %.0f %.0f",
    yearStr, monthStr, dayStr, hourStr, wd10Data[i,j], ws10Data[i,j],
    wd382Data[i,j], ws382Data[i,j], wd1387Data[i,j], ws1387Data[i,j], rain,
     mlhData[i,j], stabData[i,j])
    writeLines(tempData, conWrite, sep = "\n")
  }

  close(conRead)
  close(conWrite)
```

```
    # convert to unix format with FROMDOS program
    shell(paste("C:\\Johan\\PuffInputFiles\\FROMDOS.EXE ", outputFileName))

  } # end for loop

  # write IBterm file (necessary for NPK-Puff to run, has no real information)
  outputFileName = sprintf("C:\\Johan\\PuffInputFiles\\ibterm.txt")
  conWriteIbterm <- file(outputFileName, "w", blocking = FALSE)

  for(j in 1:nrTimeSteps+1)
  {
    hour = startHour+j-1

    hourStr = ""
    if(hour < 10) hourStr = sprintf("0%i", hour) else hourStr = toString(hour)

    tempData = sprintf("%s,%s,%s,%s,3.72088,51.4286,10,0.000,0.01, 0,0, 0,0, 0,0,
    0,0, 0,0, 0,0, 0,0 ", yearStr, monthStr, dayStr, hourStr)
    writeLines(tempData, conWriteIbterm, sep = "\n")
  }

  close(conWriteIbterm)

  # convert to unix format with FROMDOS program
  shell(paste("C:\\Johan\\PuffInputFiles\\FROMDOS.EXE ", outputFileName))
```

Appendix V: Commented R-implementation of SSA-algorithm

Part 1: SSA-algorithm

```r
rm(list=ls()) # remove all

library(gstat)
library(sp)
library(lattice)

source("M:\\Thesis\\Algorithm\\optimizationFunctions.r")

#------------------ CONSTANTS -------------------------------------#

criticalDose = 2.5;     # set the critical dose / intervention level
nrSim = 100;            # number of simulated true dose maps
nrMobDev = 8;           # number of mobile measuring devices
useFixedDev = TRUE;     # set use of fixed devices

idpValue = 3;           # Inverse Distance Power used for IDW
useMaxDist = TRUE;      # use max distance parameter
maxDist = 20;           # size of max distance parameter

fpWeight = 1;           # weight of false positive pixel
fnWeight = 5;           # weight of false negative pixel

#  settings for SSA
maxShift=170;      # maximum nr of gridcells point can be shifted
k=0;               # number of iteration
maxIterations=1500;
oldCosts = 99999999999;
newCosts = 99999999999;

trueDoseMap = list();
predDoseMap = list();

plumeColors = grey((256:0 / 256)^2);
greyColors = grey(256:0 / 256);
costColors = rgb(1,((255:0)/255), ((255:0)/255));
mapColors = rgb(1,((255:0)/255)^0.7, ((255:0)/255)^0.7);
mapColors[1]="#CCCCFF";   # set no costs to blue (water)

# create path for output info
outputDate = format(Sys.time(), "%d%H%M");
outputPath = sprintf("M:\\Thesis\\Algorithm\\SSA\\OutputRuns\\
  It%i_Si%i_Mob%i_Fn%i_IDP%i_%s\\",
  maxIterations, nrSim, nrMobDev, fnWeight, idpValue, outputDate);
dir.create(outputPath);

#--------- LOAD COST MAP, SUITABLE LOCATION MAP AND FIXED DEVICES --------#

# create a 250 x 250 point-grid (1 gridcell = 400 by 400 metre)
mapGridPoints <- expand.grid(1:250, 1:250);
names(mapGridPoints) <- c("x","y");

# create a true grid-class from these points
mapGrid = mapGridPoints;
coordinates(mapGrid) =  c("x", "y");
gridded(mapGrid) = TRUE;

# load cost map
costMapData = scan("M:\\Thesis\\Algorithm\\SSA\\InputData\\costMap.txt");
costMapData = invertY(costMapData, 250);   # invert Y
costMap = SpatialPixelsDataFrame(points=mapGridPoints,data
=data.frame(costMapData));
```

```
# load map with suitable locations
suitLocMapFile = read.csv("M:\\SSA\\InputData\\possLocTotal400.csv",
 header=FALSE);
suitLocMap = SpatialPixelsDataFrame(points = mapGridPoints, data = suitLocMapFile);

# read locations of the fixed devices and retrieve amount of fixed devices
if(useFixedDev == TRUE)
    locFixed = read.csv("M:\\SSA\\InputData\\locFixedGrid250.csv", header=TRUE);
if(useFixedDev == FALSE)
    locFixed = read.csv("M:\\SSA\\InputData\\locFixedGridNONE.csv", header=TRUE);

nrFixed = length(locFixed$ID);

# store the locations of fixed devices in a point structure for viewing
locFixedSpatialPoints = locFixed;
coordinates(locFixedSpatialPoints) = ~Xgrid+Ygrid;
ptsFixedDev = list("sp.points", locFixedSpatialPoints, pch = 3, col = "black",
       alpha = 1);


#------------------ LOAD TRUE DOSE AND REFERENCE MAPS  -------------------------#

doseMapPath = sprintf("M:\\Thesis\\Algorithm\\SSA\\DoseMapsSmallME\\");

# read Puff reference output
fileName = paste(doseMapPath,"referenceDoseMap.txt");
z = scan(fileName);
referenceDoseMap = SpatialPixelsDataFrame(points
=mapGridPoints,data=data.frame(z));

for(i in 1:nrSim)
{
  fileName = paste(doseMapPath,sprintf("trueDoseMap%i.txt", i));
  z = scan(fileName);
  trueDoseMap[[i]] =
SpatialPixelsDataFrame(points=mapGridPoints,data=data.frame(z));
}

#---------- CREATE RANDOM SAMPLING DESIGN ---------------------#

# set locations mobile devices and retrieve amount of mob. devices
locMob = matrix(1, nrow=nrMobDev, ncol=2);

for(i in 1:nrMobDev)
{
  goodLoc = FALSE;

  while(goodLoc == FALSE)
  {
    locMob[i,1] = as.integer(runif(1,1,251)); # [,1] = x-coord
    locMob[i,2] = as.integer(runif(1,1,251)); # [,2] = y-coord
    goodLoc = checkGoodLoc(locMob[i,1],locMob[i,2], i-1);
  }
}

#------------- SSA ALGORITHM -------------------------------------#

startPAccept=0.2;   # probabilistic acceptance criterion
pAcceptList = array(0, maxIterations);  # keep track of P

costList = array(0, maxIterations);  # keep track of costs
shiftListX = array(0, maxIterations); # keep track of shift size in X-dimension
shiftListY = array(0, maxIterations); # keep track of shift size in Y-dimension
locList = list();


# show cost map of start sampling design
startSamplingDesign = locMob;
```

```
locMobSpatialPoints=data.frame(Xgrid=locMob[1:nrMobDev,1],Ygrid
=locMob[1:nrMobDev,2]);
coordinates(locMobSpatialPoints) = ~Xgrid+Ygrid;
ptsMobDev = list("sp.points", locMobSpatialPoints, pch = 3, col = "blue",alpha=1);

newSamplingDesign = startSamplingDesign;

for(i in 1:nrSim)
  predDoseMap[[i]] = createPredMapAll(i);

startCosts = calculateCostsOutput(paste(outputPath,sep="","sdStart.bmp"));
startDate = date();  # store start date

for(k in 1:maxIterations)
{
    #--------- CONSTRUCT NEW SAMPLING DESIGN ------------------#

    # random select a mobile device to be moved
    selectedDev = as.integer(runif(1,0,nrMobDev)+1);
    oldLoc = c(locMob[selectedDev,1], locMob[selectedDev,2]);

    goodLocCheck = FALSE;
    while(goodLocCheck == FALSE)
    {
      newLoc = moveDevice(selectedDev);
      goodLocCheck = checkGoodLoc(newLoc[1],newLoc[2],nrMobDev);
    }

    # one device is moved, store new location in new sampling design
    newSamplingDesign = locMob;
    newSamplingDesign[selectedDev,]=newLoc;

    #-------- CREATE SPATIAL POINTS OF MOB DEV TO USE FOR VISUALIZATION -------#

    locMobSpatialPoints = data.frame(Xgrid = newSamplingDesign[1:nrMobDev,1],
      Ygrid = newSamplingDesign[1:nrMobDev,2]);
    coordinates(locMobSpatialPoints) = ~Xgrid+Ygrid;
    ptsMobDev = list("sp.points", locMobSpatialPoints, pch = 3,col="blue",alpha=1);


    #--------- CREATE PREDICTED MAPS -----------------------#

    for(i in 1:nrSim)
    {
      predDoseMap[[i]] = createPredMapAll(i);
     }

    #--------- CALCULATE THE COSTS -------------------------#

    newCosts = calculateCostsFast();

    newPAccept=runif(1,0,1);                    # get random number for acceptance
    pAccept = startPAccept*exp(-10*k/maxIterations); # calculate acceptance chance

    # check if this sampling design is better
    if(newCosts <= oldCosts)
    {
      oldCosts = newCosts;
      locMob = newSamplingDesign;
    }
    else  # new costs higher, chance to still accept with acceptance criterion
    {
      if(newPAccept < pAccept)
      {
        oldCosts = newCosts;
        locMob = newSamplingDesign;
        print("HIGHER COSTS!! BUT Sampling design still accepted!");
      }
    }

    costList[k] = oldCosts;     # store the costs
    pAcceptList[k] = pAccept;   # store chance of acceptance
    locList[[(k+1)]]= locMob;   # store locations sampling design
    shiftListX[k] = abs(oldLoc[1] - newLoc[1]);
    shiftListY[k] = abs(oldLoc[2] - newLoc[2]);
}

# save the location list and the cost list
save(costList, file = paste(outputPath, sep="","costdata.Rdata"))
save(locList, file = paste(outputPath, sep="","locdata.Rdata"))
save(shiftListX, file = paste(outputPath, sep="","shiftXdata.Rdata"))
save(shiftListY, file = paste(outputPath, sep="","shiftYdata.Rdata"))

#---------------------- CREATE BITMAPS OF RESULTS ----------------------------#

endCosts = calculateCostsOutput(paste(outputPath,sep="","sdFinal.bmp"));

# create the false positive and false negative chance maps
calculateFpFnCosts()

f=bmp(paste(outputPath,sep="","endMap.bmp"));
print(spplot(costMap, sp.layout =
list(ptsMobDev,ptsFixedDev),col.regions=mapColors,
  main = "End result"))
dev.off()

f=bmp(paste(outputPath,sep="","suitLocMap.bmp"));
print(spplot(suitLocMap, sp.layout = list(ptsMobDev, ptsFixedDev),
  main = "Suitable location map"))
dev.off()

f=bmp(paste(outputPath,sep="","costGraph.bmp"));
plot(costList, type = "l", main = "Costs", xlab = "Iteration number", ylab =
"Costs");
dev.off();

f=bmp(paste(outputPath,sep="","shiftLocX.bmp"));
plot(shiftListX, pch=".", main = "Shift X-dimension", xlab = "Iteration number",
 ylab = "X-Shift (cells)");
dev.off();

f=bmp(paste(outputPath,sep="","shiftLocY.bmp"));
plot(shiftListY, pch=".", main = "Shift Y-dimension", xlab = "Iteration number",
ylab = "Y-Shift (cells)");
dev.off();

#------------- WRITE OUTPUT FILE --------------------------------#

# open an output file connection
outputFile = file(paste(outputPath,sep="","output.asc"), "w");
cat("start costs: ", startCosts, " ", "End costs: ", endCosts, " ",
   "Max shift: ", maxShift, " ", "Nr mob dev: ", nrMobDev, " ",
   "Inverse Distance Power: ", idpValue, " ", "Maxdist set: ", useMaxDist, " ",
   "Max IDW distance: ", maxDist, " ", "Start date: ", startDate, " ",
   "End date: ", date(), " ", file = outputFile, sep = "\n");

# calculate locations mobile devices cells
write("Locations mobile devices (grid cells)", outputFile);

for(i in 1:nrMobDev)
{
  locMobDev = sprintf("Mobile device %i: (%i, %i)",i,locList[[maxIterations]][i,1],
     locList[[maxIterations]][i,2]);
   write(locMobDev, outputFile);
}
```

```
# calculate RD locations mobile devices
locList[[maxIterations]][,1] = locList[[maxIterations]][,1]*400-200+30000;
locList[[maxIterations]][,2] = locList[[maxIterations]][,2]*400-200+366000;

write("\nLocations mobile devices (Rijksdriehoekstelsel)", outputFile);
for(i in 1:nrMobDev)
{
  locMobDev = sprintf("Mobile device %i: (%i, %i)",i,locList[[maxIterations]][i,1],
     locList[[maxIterations]][i,2]);
   write(locMobDev, outputFile);
}

close(outputFile);
```

Part 2: Functions

```
invertY = function(mapData, n)
{
  # this function mirrors the values of the Y-axis; thus switching the upper
  # row with the lower row and moving to the next row. This is necessary
  # because Puff output is read upside down.

  tempData = mapData
  for(i in 1:n)
    mapData[((i-1)*n+1):(i*n)] = tempData[((n-i)*n+1):(((n+1)-i)*n)]

  return(mapData)
}

checkGoodLoc = function(possLocX, possLocY, nrPlacedMobDev)
{
  goodLoc = TRUE

  # check if the location is within the boundaries of the study area
  if(possLocX < 1 || possLocX > 250 || possLocY < 1 || possLocY > 250)
    goodLoc = FALSE

  # check if placed in same loc as a fixed device
  for(i in 1:nrFixed)
  {
    if((locFixed$Xgrid[i] == possLocX) && (locFixed$Ygrid[i] == possLocY))
      goodLoc = FALSE
  }

  # check if placed in same loc as an already placed mobile device
  if(nrPlacedMobDev>0 && goodLoc == TRUE)
  {
    for(i in 1:nrPlacedMobDev)
    {
      if(locMob[i,1] == possLocX && locMob[i,2] == possLocY)
        goodLoc = FALSE
    }
  }

  # check if the mob device can be placed in reality with suitable locations map
  if(goodLoc == TRUE)
    if(suitLocMap$V1[(possLocX) + (possLocY-1)*250] == 0)
      goodLoc = FALSE

   return(goodLoc)

} # end checkGoodLoc

moveDevice = function(selectedDev)
{
    # calculate the size of the shift; depends on number of iteration
    # minimum size is 2 ((maxShift - 1 * (maxShift - 2) = 2)
    shift=as.integer(maxShift-((k/maxIterations) * (maxShift - 2)));

    print("Shift size: ")
    print(shift)

    shiftScale=seq(shift*(-1),shift,1)

    xShift=sample(shiftScale,1)
    yShift=sample(shiftScale,1)

    newLoc = c(locMob[selectedDev,1]+xShift,locMob[selectedDev,2]+yShift)
    return (newLoc)
}

createPredMapAll = function(n)
{
    #------------- OBTAIN DOSE AT MEASURING DEVICES ------------------#

    #read dose at locations fixed devices of reference and true dose map
    fixedDoseSim = array(0, nrFixed)
    fixedDoseSim[1:nrFixed] = trueDoseMap[[n]]$z[locFixed$Xgrid[1:nrFixed]+
      (250*locFixed$Ygrid[1:nrFixed]-250)]
    fixedDoseRef = array(0, nrFixed)
    fixedDoseRef[1:nrFixed] = referenceDoseMap$z[locFixed$Xgrid[1:nrFixed]+
      (250*locFixed$Ygrid[1:nrFixed]-250)]

    # subtract reference plume from true plume
    fixedSubtractedDose = array(0, nrFixed)
    fixedSubtractedDose[1:nrFixed] = fixedDoseSim[1:nrFixed] -
fixedDoseRef[1:nrFixed]

    # read dose at locations mobile devices of reference and true dose map
    mobDoseSim = array(0, nrMobDev)
    mobDoseSim[1:nrMobDev] = trueDoseMap[[n]]$z[newSamplingDesign[1:nrMobDev,1]
      +(250*newSamplingDesign[1:nrMobDev,2]-250)]
    mobDoseRef = array(0, nrMobDev)
    mobDoseRef[1:nrMobDev] = referenceDoseMap$z[newSamplingDesign[1:nrMobDev,1]
      +(250*newSamplingDesign[1:nrMobDev,2]-250)]

    # subtract reference plume from true plume
    mobSubtractedDose = array(0, nrMobDev)
    mobSubtractedDose[1:nrMobDev] = mobDoseSim[1:nrMobDev] - mobDoseRef[1:nrMobDev]

    allLocSpatial = data.frame(allSubtractedDose = c(fixedSubtractedDose,
    mobSubtractedDose), allLocX = c(locFixed$Xgrid,
newSamplingDesign[1:nrMobDev,1]),
    allLocY = c(locFixed$Ygrid, newSamplingDesign[1:nrMobDev,2]))

    coordinates(allLocSpatial) = ~allLocX+allLocY

    #---------- USE IDW TO INTERPOLATE DOSE VALUES AND CREATE PREDICTED MAP ------#

    if(useMaxDist == TRUE)
    {
      subtractedMap = idw(allSubtractedDose ~ 1, allLocSpatial, mapGrid,
        maxdist = maxDist, idp = idpValue)

      # remove missing values
      subtractedMap$var1.pred[is.na(subtractedMap$var1.pred)] = 0;
    }
    else
       subtractedMap = idw(allSubtractedDose ~ 1, allLocSpatial, mapGrid,
       idp = idpValue)

    # add IDW with reference plume
    predMap = referenceDoseMap
    predMap$z = predMap$z + subtractedMap$var1.pred
    return(predMap)

}  # end create Predicted map

calculateCostsFast = function()
{
  costs = 0

  for(i in 1:nrSim)
  {
    # calculate the difference between the true dose and the critical dose
    diffCriticalTrue = trueDoseMap[[i]]$z - criticalDose

    # calculate the difference between the predicted dose and the critical dose
    diffCriticalPred = predDoseMap[[i]]$z - criticalDose

    # for every pixel on the map, check if it is false positive or false negative,
    # and if yes then calculate costs of this pixel and add it to total costs
    for(j in 1:62500)
    {
        # check for false positive
        if(diffCriticalPred[j] > 0 && diffCriticalTrue[j] < 0)
            costs = costs + fpWeight * costMapData[j]

        # check for false negative
        if(diffCriticalPred[j] < 0 && diffCriticalTrue[j] > 0)
           costs = costs + fnWeight * costMapData[j]
    } # end for
  } # end for 1:nrSim

  return(costs)

} # end calculateCostsFast

calculateCostsOutput = function(fileName)
{
  costs = 0

  # store the costs of every pixel in order to visualize costs
  testCosts = array(0,62500)

  for(i in 1:nrSim)
  {
     diffCriticalTrue = trueDoseMap[[i]]$z - criticalDose
     diffCriticalPred = predDoseMap[[i]]$z - criticalDose

     for(j in 1:62500)
     {
        # check for false positive
        if(diffCriticalPred[j] > 0 && diffCriticalTrue[j] < 0)
        {
            costs = costs + fpWeight * costMapData[j]
            testCosts[j] = testCosts[j] + fpWeight * costMapData[j]
        }

        # check for false negative
        if(diffCriticalPred[j] < 0 && diffCriticalTrue[j] > 0)
        {
            costs = costs + fnWeight * costMapData[j]
            testCosts[j] = testCosts[j] + fnWeight * costMapData[j]
        }
    } # end for

  } # end for 1:nrSim

  testCostsMap = SpatialPixelsDataFrame(points = mapGridPoints,
    data = data.frame(testCosts))
  print(spplot(testCostsMap, sp.layout = list(ptsMobDev, ptsFixedDev),
    col.regions=costColors, main = "Costs total", scales=list(draw=T)));

  # create output image of costmap
  f=bmp(fileName)
  print(spplot(testCostsMap, sp.layout = list(ptsMobDev, ptsFixedDev),
    col.regions=costColors, main = "Costs total", scales=list(draw=T)))
  dev.off()

  return(costs)

} # end calculateCostsOutput


calculateFpFnCosts = function()
{
    # store the number of fn and fp in every pixel
    fpData = array(0,62500)
    fnData = array(0,62500)

    for(i in 1:nrSim)
    {
       diffCriticalTrue = trueDoseMap[[i]]$z - criticalDose
       diffCriticalPred = predDoseMap[[i]]$z - criticalDose

       for(j in 1:62500)
       {
          # check for false positive
          if(diffCriticalPred[j] > 0 && diffCriticalTrue[j] < 0)
              fpData[j] = fpData[j] + 1

          # check for false negative
          if(diffCriticalPred[j] < 0 && diffCriticalTrue[j] > 0)
              fnData[j] = fnData[j] + 1
        } # end for all pixels of the map

    } # end for 1:nrSim

    # calculate the percentage
    fpData = fpData / nrSim
    fnData = fnData / nrSim

    fpMap = SpatialPixelsDataFrame(points = mapGridPoints, data =
data.frame(fpData))
    fnMap = SpatialPixelsDataFrame(points = mapGridPoints, data =
data.frame(fnData))

    print(spplot(fpMap, sp.layout = list(ptsMobDev, ptsFixedDev), col.regions=
     greyColors, Main = "Prob. false positive classification",
scales=list(draw=T)));

    print(spplot(fnMap, sp.layout = list(ptsMobDev, ptsFixedDev), col.regions=
    greyColors, main = "Prob. false negative classification",
scales=list(draw=T)));

    # create output image of fp and fn map
    f=bmp(paste(outputPath,sep="","fpMap.bmp"))
    print(spplot(fpMap, sp.layout = list(ptsMobDev, ptsFixedDev),
      col.regions=greyColors, main = "Prob. false positive classification",
      scales=list(draw=T)));
    dev.off()

    f=bmp(paste(outputPath,sep="","fnMap.bmp"))
    print(spplot(fnMap, sp.layout = list(ptsMobDev, ptsFixedDev),
      col.regions=greyColors, main = "Prob. false negative classification",
      scales=list(draw=T)));
    dev.off()
```

```
} # end calculateFnFpCosts


#-------------- CREATE A THRESHOLD MAP -------------------------------------#

calculateThresholdMap = function(values)
{
  # this function determines the pixels which are close to the threshold
#(intervention level) and draws them on the screen

  thresholdRange = 0.05

  thresholdValues = values

  for(j in 1:length(thresholdValues))
  {
     if(thresholdValues[j] > (criticalDose - thresholdRange) &&
      thresholdValues[j] < (criticalDose + thresholdRange))
         thresholdValues[j] = 1
     else if(thresholdValues[j] <= (criticalDose - thresholdRange))
           thresholdValues[j] = 0
     else if(thresholdValues[j] >= (criticalDose + thresholdRange))
         thresholdValues[j] = 2
  }

  thresholdMap = trueDoseMap[[1]]
  thresholdMap$z = thresholdValues
  print(spplot(thresholdMap, main = "Thresholdmap", scales=list(draw=T)))

  return(thresholdMap)
}
```