Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Civil and Environmental Engineering

Guro Stokseth

Digitalising optimisation of early phase urban stormwater planning

Master's thesis in Bygg- og miljøteknikk
Supervisor: Tone Merete Muthanna and Erle Kristvik
June 2019

NTNU
Norwegian University of
Science and Technology

Guro Stokseth

Digitalising optimisation of early phase urban stormwater planning

Master's thesis in Civil and Environmental Engineering
Supervisor: Tone Merete Muthanna and Erle Kristvik
June 2019

Norwegian University of Science and Technology
Faculty of Engineering
Department of Civil and Environmental Engineering

NTNU
Norwegian University of
Science and Technology

Description of Master Thesis spring 2019

Candidate name: Guro Stokseth

Subject: Stormwater

Title: Digitalising optimisation of early phase urban stormwater planning

Start date: 11th January 2019

Due date: 11th June 2019

Background

Heavy urbanization and precipitation intensities are putting increased strain on existing stormwater systems worldwide. Consequently, many systems need upgrades that counteract these impacts and at the same time account for the uncertainties in future rainfall extremes caused by climate change. In Norway, policies and practice are leaning towards a system design that depend more on open, nature-based solutions (NBS) as alternative to traditional piped systems. Open, nature-based solutions add flexibility to future capacity needs and has positive social-environmental impacts, but they require surface area – a scarce resource in urban areas. One measure to secure area for open, nature-based solutions is to consider stormwater earlier in the planning process than what is usual practice in Norway.

The Norwegian startup Spacemaker (https://spacemaker.ai/) is developing a software based on AI technology for generating and exploring building site proposals, given regulatory and physical constraints and preferences added by the developer. In addition to generating various site proposals, the tool provides more detailed insight of the proposals in the early phases of the planning process than manual methods do today. Adding stormwater as a layer to Spacemaker's framework could help ensuring that stormwater is considered earlier in the planning process and hence facilitate implementation of open, nature-based solutions.

Research questions

The objective of this research is to develop a methodology for assessing placement, size and combinations of SUDS digitally in early- phase urban planning. The master thesis aims to answer the following research questions:

1. To what extent can the proposed methodology address the challenges in traditional approach to stormwater management?
2. Which factors should be optimised for assessing and selecting SUDS configurations?
3. What is the performance of the proposed methodology?

Collaboration partners: Klima2050, BINGO, Spacemaker AI

Location: The master thesis will be conducted at the Department of Civil and Environmental Engineering. The candidate should have regular meetings with advisor(s). The simulations and models will be used with licenses and software available at the Department of Civil and Environmental Engineering

Advisors: Tone Merete Muthanna, Erle Kristvik

Preface

This report is the final product of the course *"TVM4905 Water and wastewater engineering, Master's thesis"* at the Norwegian University of Science and Technology (NTNU), Department of Civil and Environmental Engineering. A preliminary literary research leading up to this master thesis was performed in the fall of 2018 as a specialization project in the course *"TVM4510- Water and wastewater engineering".* The purpose of this thesis is to investigate the possibility of optimising placement, size and configurations of sustainable urban drainage systems (SUDS) through computer programming. Firstly, I would like to extend my sincere gratitude to my two advisors; Professor Tone Merete Muthanna and PhD candidate Erle Kristvik for their guidance throughout the research process. Thank you for guiding, challenging and encouraging me. Your spark for this project has been inspiring. I would also like to thank Simen Braathen at Spacemaker AI for his indispensable guidance and help in the programming part of this research.

The study was made possible by the project Klima2050 and the BINGO project. In addition, there are many helpful souls to whom I would like to extend my gratitude:

- Scientist Jardar Lohne, for guidance in the process of writing a scientific article
- Professor Knut Alfredsen, for help and guidance on hydrological modelling in ArcMap
- Professor Yngve Frøyen, for helping solve Model Builder puzzles in ArcMap
- PhD candidate Ana Juarez, for guidance in the use of ArcMap
- Birgitte Johannesen at the Municipality of Trondheim, for fruitful discussion around the scoring of SUDS as well as providing me with relevant literature
- Norges Kommunaltekniske Forbund (NKF), for bestowing me with a master's scholarship, enabling me to travel and collaborate across city and country boundaries
- Post doc. Santiago Sandoval Arenas, for guiding me in the use of his newly developed software *Urbis* and letting me use it in my research.
- Data Scientist Marie Ameln at Spacemaker AI, for letting me collaborate with Spacemaker, and providing sparks of motivation throughout the research by creating a stormwater channel on Slack.
- Data Scientist Thomas Gjerde, for guidance in the use of Python

Trondheim, June 6, 2019

Guro Stokseth

Guro Stokseth

Thesis structure

This master thesis is presented as a manuscript according to the requirements and structure of a research article. As NTNU's vision "knowledge for a better future", it is the author's wish that this thesis and research be made available for whomever might find it useful. Therefore, an article structure is chosen to facilitate the study's availability for an international audience. The thesis' summary is written both in Norwegian and English. An extended Norwegian summary will also be presented in NKF's (Norsk Kommunalteknisk Forening) journal *Kommunalteknikk*.

This thesis is written in English as a part of the international projects BINGO and Klima2050. The manuscript will be submitted as a research article to the journal *Water Research* and is therefore structured based on the format guidelines provided by this journal. The master thesis is at this date accepted to be presented in a poster presentation at the *Nova Tech*- conference in Lyon, France in July 2019.

The master thesis manuscript intended for censorship is somewhat more extensive than the academic journal manuscript. This choice was made based on the seeming necessity of covering the study's methods to a satisfactory extent. A comprehensive appendix is also included in order to present parts of the study which are not included in the final journal manuscript.

It should be noted that the methodology has been altered quite a bit through the course of the work with this thesis. Initially, the results from the presented programming procedure was to be further modelled and evaluated in a software called Urbis. However, due to computer problems and unsuccessful troubleshooting, the software has not been used, and thus the intended methodology has not been tested in this thesis. However, as the development of a methodology is the objective of this thesis, this initial method is described in chapter 2 *Materials and methods*. The modifications made to the method are briefly explained in chapter 2.5.1 *Modified method*. The study in this thesis has been executed by use of the modified method.

Summary (EN)

Climate change and urbanisation is to a large extent causing the drainage systems to be insufficient which in turn leads to increased flooding in urban areas. The state of the art worldwide today to alleviate such flooding consists in using sustainable urban drainage systems (SUDS). Implementing such solutions proves, however, problematic, since the water management engineers typically enter the building process too late to influence the physical layout of major projects. In this paper, we examine a novel, numerical approach to early inclusion of drainage systems in such projects.

Key factors for the efficiency of SUDS were identified through a literature review. These were used to develop a scoring system based on providing relative proximity to natural conditions. An optimisation routine was then developed with the objective of obtaining the highest possible score. The optimisation routine was scripted in python to obtain the best possible SUDS configurations. Eleven different building proposals for a fictitious development project on a real-life site in Oslo, Norway, were spatially analysed. SUDS were subsequently placed for each building proposal by using the optimising script.

First and foremost, the results showed a significant variation in the potential for SUDS implementations for the different building proposals, ranging from little to considerable flood reduction. This implies that SUDS are highly context dependent. Secondly, the results show great potential to analyse a large number of building proposals and SUDS figuration quite efficiently through a simple script. This implies the applicability of such analysis early in development projects.

The need to include SUDS in early urban planning seems clear. It is paramount in order to ensure that SUDS serve the much-needed resilience they have proved to provide. Through this research, a first step towards ensuring this has been made.

Samandrag (NO)

Klimaendringar og urbanisering fører til at eksisterande dreneringssystem blir utilstrekkelege, noko som vidare leier til ein auka frekvens av urbane flaumar. *State of the art* for handtering av slike utfordringar består verden over i dag av å bruke såkalla berekraftige urbane dreneringssystem, eller *Sustainable Urban Drainage Systems* (SUDS). Implementeringa av slike løysingar har derimot vist seg å vere problematisk ettersom overvann- ingeniørar typisk blir innlemma i byggeprosessen for seint til å ha innverknad på det fysiske oppsettet av tomta. I denne oppgåva ser vi på ei ny, numerisk tilnærming til tidleg inkludering av dreneringssystem i slike byggeprosjekt.

Nøkkelfaktorar for ytingsgrada til SUDS vart identifisert gjennom eit litteraturstudie. Desse faktorane vart så brukt til å utvikle eit skoringssystem basert på eit mål om å oppretthalde naturlege tilstandar. Ei optimaliseringsrutine vart vidare utvikla med mål om å oppnå høgast mogleg skoring. Denne optimaliseringa vart skriven i Python- kode for å oppnå best moglege SUDS- konfigurasjonar. Elleve ulike bygningsforslag for eit fiktivt byggeprosjekt på ei verkeleg tomt i Oslo, Noreg, vart romleg analysert. Deretter vart SUDS plassert for kvart enkelt bygningsforslag gjennom bruk av optimeringsskriptet.

Resultata viser først og fremst ein betydeleg forskjell i SUDS- potensiale for dei ulike bygningsforslaga for tomta, med eit stort spenn i flaumhandteringspotensiale. Dette impliserer at SUDS er svært kontekstavhengige. For det andre viser resultata at med ei enkel kode kan ein på effektivt vis analysere mengder av bygningsforslag og/eller SUDS konfigurasjonar. Dette viser eit stort potensiale for å inkludere desse analysane tidleg i eit byggeprosjekt.

Behovet for å inkludere SUDS tidleg i urban planlegging er tydeleg. Det er avgjerande for å sikre at SUDS yter den sårt trengte robustleiken dei har bevist å kunne sikre. Gjennom denne oppgåva har eit første steg mot denne sikringa vorte tatt.

Table of content

List of figures	I
Abbreviations	II
Abstract	9
Key words	9
1 Introduction	9
2 Material and Methods	11
3 Results	19
4 Discussion	25
5 Conclusion	28
Acknowledgements	28
Bibliography	29
Appendix A – Digital elevation manipulation model	31
Appendix B – Resulting drainage lines for all building proposals	32
Appendix C – Spatial analysis model	33
Appendix D – Python scripts	34
Appendix E – Fixed values from literature for modelling input	51
Appendix F – SUDS placements for all building proposals	53
Appendix G – SUDS placements numbered	55

List of figures and tables

Figure 1 – Overview of initially intended method 11

Figure 2 – Flow conditions for the study area 12

Figure 3 – Flow chart for modified method 19

Figure 4 – Drainage lines for building proposal 2 and 7 21

Figure 5 – SUDS plot for building proposal 2 and 7 22

Table 1 – The SUDS selection aid 20

Table 2 – Resulting numbers for SUDS plots 22

Table 3 – Scoring system 23

Table 4 – Resulting scores for SUDS configurations for all building proposals 24

Abbreviations

SUDS	Sustainable Urban Drainage Systems
GIS	Geographical Information System
IDE	Integrated Development Environment
DEM	Digital Elevation Model

Digitalising optimisation of early phase urban stormwater planning

Guro Stokseth[a]

[a.] Department of civil and environmental engineering, Faculty of engineering, Norwegian University of Science and Technology (NTNU)

ABSTRACT

Climate change and urbanisation is to a large extent causing the drainage systems to be insufficient which in turn leads to increased flooding in urban areas. The state of the art worldwide today to alleviate such flooding consists in using sustainable urban drainage systems (SUDS). Implementing such solutions proves, however, problematic, since the water management engineers typically enter the building process too late to influence the physical layout of major projects. In this paper, we examine a novel, numerical approach to early inclusion of drainage systems in such projects.

Key factors for the efficiency of SUDS were identified through a literature review. These were used to develop a scoring system based on providing relative proximity to natural conditions. An optimisation routine was then developed with the objective of obtaining the highest possible score. The optimisation routine was scripted in python to obtain the best possible SUDS configurations. Eleven different building proposals for a fictitious development project on a real-life site in Oslo, Norway, were spatially analysed. SUDS were subsequently placed for each building proposal by using the optimising script.

First and foremost, the results showed a significant variation in the potential for SUDS implementations for the different building proposals, ranging from little to considerable flood reduction. This implies that SUDS are highly context dependent. Secondly, the results show great potential to analyse a large number of building proposals and SUDS figuration quite efficiently through a simple script. This implies the applicability of such analysis early in development projects.

The need to include SUDS in early urban planning seems clear. It is paramount in order to ensure that SUDS serve the much-needed resilience they have proved to provide. Through this research, a first step towards ensuring this has been made.

KEYWORDS

Stormwater Management, SUDS, Urban Planning, Python, ArcMap

1 INTRODUCTION

Urban watersheds are characterised by high percentage of impervious areas, and only a small change in rainfall intensity can cause severe floods (Eckart et al., 2017). Climate change is inflicting rather severe intensity changes on such urban watersheds, leading to increased flooding in urban areas worldwide. A panel of experts established by the Norwegian government concluded that the costs of damages to the Norwegian society caused directly by stormwater, or by consequences imposed by stormwater, amount to a number between 0,16 to 0,3 billion Euros every year (Hodnesdal, 2018).

The Norwegian Federation for Engineering Consultancy Associations, *Rådgivende Ingeniørers Forening* (RIF), states that the current pipe network in Norway has neither the capacity nor the condition to handle the increased amounts of stormwater imposed by urbanisation and climate change (RIF, 2015).

The main tendency today to alleviate such flooding consists in using surface based sustainable urban drainage systems (SUDS) (Eckart et al., 2018). These solutions add flexibility to future capacity needs and have shown to contribute positively to maintaining the natural hydrological cycle, as well as improving air-quality and eco-systems(Eckart et al., 2018, 2017; Ugarelli et al., 2017; Woods-Ballard et al., 2007). However, these solutions are highly context dependent. Several challenges are involved in using them: Firstly, they typically demand surface area, a scarce resource in urban areas. Secondly, SUDS's performance is highly dependent on their topographic placement. Thirdly, the number of possible SUDS combinations is identified as a challenge (Eckart et al., 2018). In development projects, the current practice in Norway is to consider stormwater management after buildings, parking areas and other elements are considered (Oslo Kommune, 2013). This is limiting the possibility of obtaining optimal placement and sizing of SUDS. Eckart et. al states that a true comprehensive approach to SUDS planning would include concerns regarding water and ecology throughout the planning process (Eckart et al., 2018).

However, through development of data science with the ability to handle, process and analyse big data, different software is emerging, introducing a nearly unlimited analytical capacity. By enabling us to assess thousands of potential SUDS- configurations, data science is introducing the possibility of a paradigm shift in stormwater management. Such an unlimited amount of possible solutions is challenging to evaluate manually. A scoring system could help automate the selection of qualified solutions. The objective of this research has therefor been to evaluate how SUDS can be optimised in terms of placement, size and combinations in early phase development projects, where the physical layout of building mass is still undecided.

Sustainable urban development has in the past decade become the convention, and the amount of research on the subject is abundant. However, a research gap presents itself in terms of scale and timing. On one side, the research is small scale and focused upon optimising the technical components of the solutions (Johannessen et al., 2018, 2017; Paus et al., 2015). On the other side, the research is focused on optimising on a catchment scale, looking at whole districts under one (Kazak et al., 2018; Liu et al., 2016; Zhu et al., 2019). There is little research on optimisation of SUDS for a single site or development project. Both Jia et al. (2013) and Eckart et al. (2018) present optimisation on a site scale. These are, however retrofitting projects and do not assess SUDS prior to the physical layout of the property (Eckart et al., 2018; Jia et al., 2013). Little research is done on optimising SUDS as part of initial physical planning of a site.

The literature outlines that traditional approach to stormwater management presents great challenges for the performance of SUDS. It is clear that stormwater management needs to be assessed earlier in the planning process (Oslo Kommune, 2013). Identified barriers for successful implementation are the complexity of SUDS (Eckart et al., 2017), their context dependency and consequently the failure to assess them early enough in the process to take these important characteristics into account (Eckart et al., 2018). The objective of this research is to develop a methodology for assessing placement, size and combinations of SUDS digitally in early- phase urban planning.

In order to address this inquiry, we pose the following research questions:

1. To what extent can the proposed methodology address the challenges in traditional approach to stormwater management?
2. Which factors should be optimised for assessing and selecting SUDS configurations?
3. What is the performance of the proposed methodology?

2 Material and Methods

The method presented in this chapter is the initially intended method of this thesis. Due to technical computer problems late in the process, this method could not be executed. Nevertheless, it is deemed important to explain the intended methodology, as this has been an objective of the research. The modified method, which was the one executed in this research, is briefly explained in chapter 2.5.1.

In order to answer the research questions, a literary review was performed, laying the basis for the development of a scoring system. Furthermore, a spatial analysis was performed for 11 building proposals for the model site. An optimising script was then made to place, size and combine SUDS for each building proposal. Finally, the rainfall response of the SUDS configurations would be tested through modelling

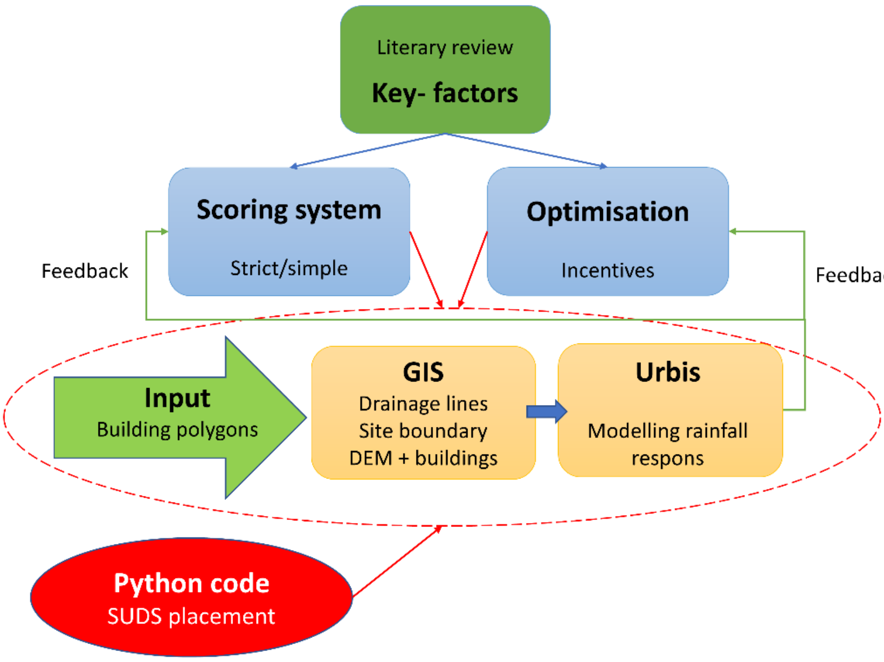


Figure 1- Overview over initially intended method

In Norway, the three- stage approach to stormwater management has been adopted and is frequently used as a guideline. It is based on the principal of local handling of stormwater and refers to three levels of solutions depending on the rainfall intensity and volume. The first stage applies to every-day events for which the objective should be to retain and infiltrate the water. The second stage refers to medium events and the aim is to detain the water delaying the flood peak and subsequent runoff response. The third stage for the large events leading to urban floods in which cases the aim should be to secure safe flood paths (Norsk Vann, 2005). It should be noted that the research reported on in this article, is with this Norwegian convention in mind.

2.1 SITE DESCRIPTION

The site used for the demo project in this research is the urban area of Marienlyst in Oslo, Norway (Figure 2). Specifically, the property of NRK, the Norwegian Broadcasting. This site was chosen because it has already been regulated as a residential area. In addition, the municipality of Oslo has a quite progressive policy for stormwater management, demanding that all rainwater be handled locally (Oslo Kommune, 2017). Therefor it was considered interesting to work with the demands of the municipality of Oslo as an objective for the SUDS configuration. Note should be taken, however, that the building project herein is completely fictive.

Marienlyst lies at around 70 meters above sea level in the north-west of Oslo. The specific site is 41 033,5 m^2 and is sloped at around -7% to the south. Considering the objective of this research being focused on the development of a general methodology, the varying climate of the area has not been considered. The ground water level in the area is at 8m, and thus does not need to be considered for this specific research. The soil conditions in the area is either of low permeability or not registered. However, the municipality of Oslo states that these soil maps should not be given great reliance, as condition may have been greatly altered due to construction in the area and/or effects of trees, roots and other biological mechanisms (Oslo Kommune, 2017).

In this research, 11 building proposals are used as input data to the model site. Each proposal is subject to a spatial analysis and subsequent placement of SUDS. Furthermore, the various proposals are scored based on their ability to facilitate SUDS.

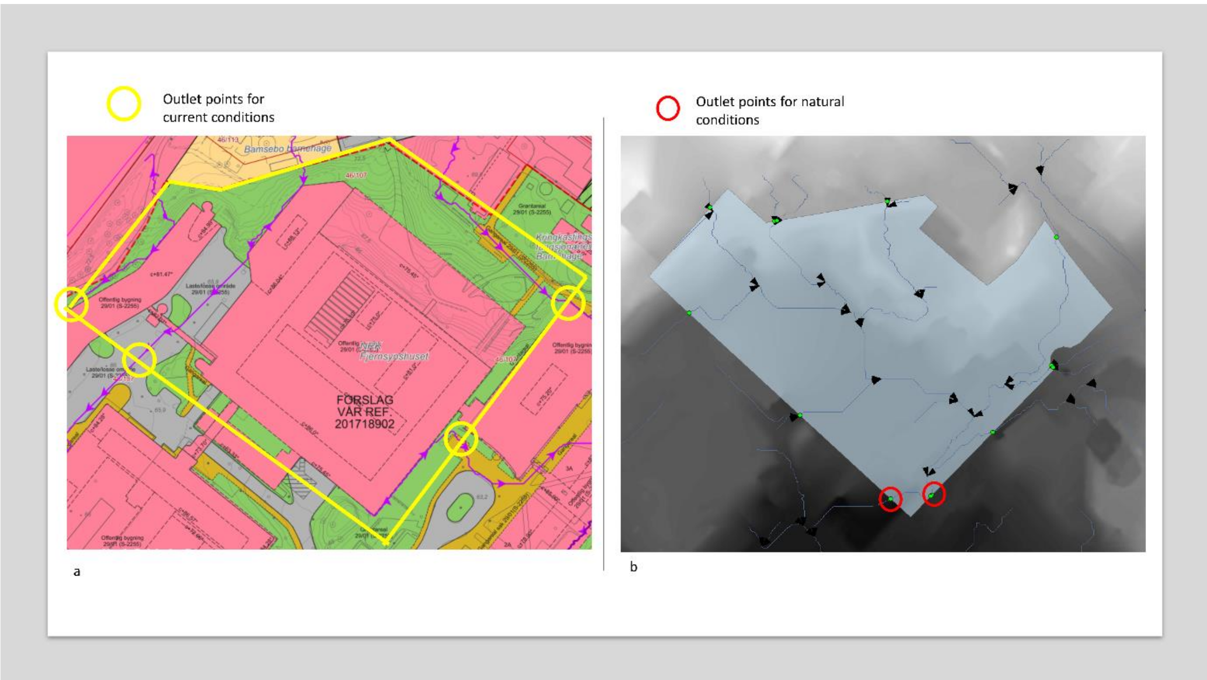


Figure 2- (a) Current flow conditions of the site, (b) flow condition for pre-development conditions

2.2 LITERARY REVIEW

A literary review has been performed in order to get an overview of the state of research for the subject. Furthermore, one specific objective of the literary review was to obtain the key factors affecting the performance, size and placement of SUDS. The literary review consists of two parts. The first part is a comprehensive review performed in the fall of 2018, prior to the actual research. The second part, performed in the spring of 2019, is an extension of the initial review.

The initial literary research of 2018 was performed using the Norwegian search engine *Oria* with the following input keywords; *LID, optimisation, stormwater, SuDS, urban planning, WSUDS, urban flood.* Several academic articles were qualitatively evaluated in order to obtain the most relevant ones for this study. As this study is part of an emerging field in a novel form of technology, the most recent articles were considered most important. Eckart et al. (2017) have reviewed the current state of research considering optimisation, modelling, monitoring and maintenance of SUDS and this was considered a particularly valuable source as it provided relatively fresh information on the state of research. A start set for the literary review was obtained through backwards snowballing, meaning investigation of the bibliography of the most relevant articles (Wohlin, 2014). According to Wohlin (2014), a good start set is diverse, covering several different publishers, years and authors (Wohlin, 2014). The start set for this research consisted of 9 articles and two design guidelines regarding SUDS. The reviewed papers had a publishing time span of 9 years, ranging from 2010 to 2018. These sources also provide a geographical span, which in turn ensures a span in consideration of climate, topography and other factors affecting SUDS.

Through the initial literary review, the following SUDS were chosen for further investigation: *Green roofs, rain gardens, permeable covers, swales* and *open detention basins*. This selection was based on their proven ability to handle water volumes in urbanised areas as well as the amount of documentation and research on their design, construction and use (Woods-Ballard et al., 2007). In addition, it was considered important that the chosen SUDS were well documented in terms of design and performance for different contexts as this can give great variations in the focal points of a study and thus affect the factors considered or mentioned.

The second part of the literary review was executed in the form of a scoping review by following five steps: (1) Identify the research question, (2) Identify relevant studies, (3) Study selection, (4) Charting the data, (5) Collating, summarising and reporting the results (Arksey and O'Malley, 2005). In order to identify relevant studies, Google Scholar was used to perform forward snowballing, meaning identifying new papers through citation (Wohlin, 2014). The 2017 Eckart review was considered particularly relevant and through forward snowballing, 54 additional articles were further evaluated. The evaluation process consisted of three steps of inclusion or exclusion; firstly, looking at the titles, secondly looking at the abstract and thirdly checking the place and context of the citation. The final set of literature consists of 16 articles obtained through the steps described above as well as 9 articles provided by professionals and professors involved in the research.

2.3 SPATIAL ANALYSIS

The objective of this part of the study was to model how drainage lines were affected by the placement of 11 different building proposals. Furthermore, ArcMap was used to model the areas suitable for SUDS placements. The results of this suitability analysis were then used as input for a python code placing raingardens and green roofs on the site. The use of python is deemed rather important in this research for the purpose of facilitating the possibility of assessing thousands of building proposals. A potential for digital SUDS optimisation is evident in the current development of software. By using python, the possibility of utilizing this potential is preserved.

ArcMap 10.6.1 is a geographical information system (GIS) developed for the purpose of creating maps, perform spatial analysis and manage geographic data (esri, n.d.). In this research, ArcMap is used for the purpose of modelling the hydrological response of the catchment to the various building proposals. Though ArcMap demands a license for desktop use, it was deemed appropriate for this study, as it is known to the researcher.

Python was downloaded from www.python.org. By using the integrated development environment (IDE) PyCharm, different virtual environments could be created for different parts of the study. A virtual environment was created to process python codes from ArcMap and was thus using Python 2 which is the python version demanded by ArcMap.

2.3.1 Construction of digital elevation model containing buildings

In order to model the hydrological response to the various building configurations, a digital elevation model (DEM) had to be manipulated to contain the buildings. This was done using Model Builder in ArcMap. The steps for obtaining such a model is visually presented in Appendix A and described in detail below:

1. **Import of DEM:** A digital elevation model for the area was imported from www.hoydedata.no in TIFF format with a solution of 1m in the projection ETRS 1989 UTM Zone 33.
2. **Making a table of building polygons:** The 11 building proposals were imported to ArcMap. In the attribute table of each building proposal, an additional field was added by choosing *Add Field*. This field was given the name *Alt_nr* for all 11 proposals. The entire column was given the number of the corresponding building proposal. All building proposals were initially given as one single polygon, but by use of the *dissolve* tool by *Alt_nr*, each building within the proposal was represented by an individual polygon. All the building proposals were then added to the same list by use of *Append* by *feature class.*
3. **Adding buildings to DEM:** In order to manipulate the DEM to contain buildings, the following procedure was performed for each building proposal by use of the tool *Iterate Feature Selection* in Model Builder:
    a. **Rasterization:** Polygons were converted to a raster dataset by use of the tool *Polygon to Raster.*
    b. **Reclassification:** The tool *Reclassify* was used to assign a value of 0 to the part of the newly made rasters with the initial value of *NoValue*, as this could potentially give problems in the following steps.
    c. **Adding buildings to DEM:** The DEM was manipulated by adding a height of 200m to the DEM within the boundary of each polygon in the building proposal. This was done using the tool *Plus*.
4. **Manipulated DEM:** The model was validated and run, resulting in 11 different manipulated DEMs containing each of the 11 building proposals. Moving forward, these new rasters were used for modelling purposes.

2.3.2 Modelling drainage lines in DEM containing buildings

In order to model the drainage lines for each of the 11 DEMs, the *ArcHydro*- tools were used. Specifically, the 10 steps of terrain pre-processing were executed. The few alterations made to the standard procedure are marked with a star. The procedure was executed in the following order:

Fill sinks → Flow direction → Flow accumulation → Stream definition** → Stream segmentation → Catchment Grid Delineation → Catchment Polygon Processing → Drainage Line Processing → Adjoint Catchment Processing → Drainage Points Processing*

**Fill sinks is done to fill local surface depressions in the DEM to avoid interrupting flow lines when calculating main flow paths. Because the DEMs were manipulated to contain buildings, it was important to hatch the box for Fill Threshold. This was set to 50m to avoid filling the 200m drop in between the buildings, which could be interpreted as depressions.*

***For the given analysis, we were interested in the details of streamlines within the site boundary. Therefore, the number of cells to initiate a stream was set to 1400 cells.*

The result of the hydrological analysis was a set of 11 DEMs representing the hydrological response for the 11 different building proposals. The results can be seen in Appendix B.

2.3.3 Creating a SUDS potential- model

In order to be able to decide the placement of SUDS for the various building proposals, it was necessary to analyse the manipulated DEMs to see where potential for SUDS placement lay. In order to capture water, rain gardens need to lie along the drainage lines of the property. However, not all parts of the drainage lines are potential placements for rain gardens. A set of analyses were performed in ArcMap in order to identify all the points along the drainage lines which could fulfil all the demands for good rain garden placements. This was done by using model builder for one of the building proposals. The order and the complete model is found in Appendix C. The steps of the model are explained in the following:

By using the tool *Intersect,* the intersection points between the drainage lines and the site limit were obtained. These were considered important for the evaluation of how the building proposals affect the drainage lines and thus the flood paths. The number of outlet points from the site also equals the number of directions in which SUDS configurations need to be placed in order to reach the objective of no runoff from the property.

The sub- catchments of the site were obtained through the hydrological analysis described in 3.4.3. In the rain garden potential- model, the catchment raster was converted to polygons by using the tool *Raster to Polygon*. This was done to obtain the area of each sub- catchment, which required a polygon form. Obtaining these areas was considered important in order to calculate the demanded rain garden area within each sub-catchment. Furthermore, the area of these sub-catchments could give information about the size of the area draining to each of the outlet points identified through the process described in 2.4.1.

The drainage lines were cut to the extent of the site limit using the tool *Clip.* In order to be able to analyse the placements along the drainage lines, points were placed with a 2 m distance along the course of all the drainage lines using the tool *Generate Points Along Lines.* Furthermore, these points were given values extracted from the flow accumulation layer using the tool *Extract Values to Points*. The values given to the points were the rastervalue, which was the number of cells draining to the given point.

The site limit polygon was converted to a polyline using the tool *Polygon to polyline*. In that way, a buffer of 2 m could be generated on the inside of the site limit using the tool *Buffer*. This was done to make sure the rain gardens were not placed too close to the site limit. The *Buffer*- tool was also used to generate a buffer around the building polygons. The buffer was given an extent of 2 m to account for the demanded distance between buildings and raingardens (Paus and Braskerud, 2013). The points along the drainage lines that were situated in the buildings buffer zone or the site limit buffer zone were then erased using the *Erase*- tool. The points remaining along the various drainage lines were thus the points available for placement of rain gardens.

The rain garden potential- model, created in ArcMap as described above, was exported to a python script and processed in PyCharm. The code was then looped to run for all 11 building proposals. The result was 11 ArcMap- projects showing only the points available for rain gardens along the drainage lines. The points available for rain garden placements were imported to the python code as “RG_potential”. The total drainage line points- series was also imported to the python code for further analysis.

2.3.4 Script for placement of SUDS

Following the analysis performed in ArcMap and translated to Python code, a new script was created with the objective of placing and dimensioning rain gardens on the site. The script consisted of two steps described below. The complete script can be found in Appendix D.

The purpose of the first step of the script was to identify the drainage line connections and catchment affiliation for each point along the drainage lines:

1. **Identify outlet points from the property:** All points, both available for rain gardens and not, were sorted by descending number of cells draining to the given point. The points were then evaluated based on their “to- and from- nodes”. If the evaluated point was a predecessor of an already evaluated point, in terms of flow direction, it would not be evaluated. If the evaluated point had the highest rastervalue of all the points with the same node pair, it was identified as an outlet point from the propoerty.
2. **Grouping points into drainage line networks:** All points, available for rain gardens or not, were sorted into drainage line networks. This was done by evaluating their to- from node as well as their catchment affiliation. Each point was given the information about who’s successor it was and who was its predecessor. In that way, for each point along a drainage line, one can obtain all its upstream points and associated catchment.

The purpose of the second step of the script was to place and size rain gardens along the drainage lines. In order to handle all the water running off from the site, a script analysing the various drainage line networks from the outlets point moving in the counter flow direction was created. Hence, the following procedure was scripted to analyse all draining line networks for each site. This was done in the following manner:

1. **Calculate demanded raingarden area:** For each outlet point, the demanded raingarden area was defined as 9 % of its upstream catchment area in line with the recommendations found in literature (Magnussen et al., 2015). Each point, moving counter-stream from the outlet point, was then analysed considering the following:
   - **a.** Is the point included in the allowed points- list?
   - **b.** Is the next point included in the allowed points list?

   It was assumed that a raingarden would not be placed unless there were two or more points in a row available, as the distance between points were only 2 m.

2. **Make raingarden polygon:** If two points in a row or more are available, the creation of a polygon was initiated. The polygon was given an extent of 8 m on each side of the drainage line. For each available point along the drainage line this was performed, resulting in a set of coordinates which was then scripted to create a raingarden polygon. In any case where part of the raingarden- polygon crossed a building’s buffer zone or a sub-catchment boundary, the polygon was clipped to the extents of these boundaries. If there was room for the demanded raingarden area, the raingarden was placed. If there was not room for the demanded raingarden area, the largest possible raingarden was placed and defined, and the

analysis proceeded upstream. The demanded are of raingarden was now updated, reduced by the area of the raingarden placed.
When the analysis arrived at a crossroads in the drainage line network, it was scripted to proceed along the line that has the largest associated catchment. It would subsequently go back and analyse the other arm of the crossroads.

3. **Stop when demanded area of rain garden is reached:** The analysis was scripted to break when the demanded area of the rain gardens was reached, or when all points in a drainage line network were analysed.
4. **Placing green roofs:** The amount of green roof was given as an input percentage value. Initially this value was set to 40% and thus 40% of each roof was assigned an extensive green roof. Each roof was assigned a connection to the closest raingarden, so long as the distance was less than 4 meters. This was done as former research has shown these types of treatment trains to be very efficient (Kristvik et al., 2019).

The script gave the following output for each building proposal:

- The number of outlet points from the site. And for each outlet point:
    - It's corresponding draining area
    - Number of raingardens, including individual areas
    - Number of green roofs, including individual areas
    - Which raingardens the various green roofs were connected to

2.4 Modelling rainfall response for SUDS configurations

The newly developed software, Urbis, was to be used for modelling purposes. The software can model rainfall response for stand- alone SUDS as well as for combinations of these. The various SUDS are represented in terms of boxes representing either storage or substrate. The software takes a rainfall as input and outputs the rainfall response and overflow for the given SUDS configuration.

The resulting SUDS- configurations from the ArcMap analysis and Python- script were to be used as input for the Urbis- modelling. The inputs for raingardens and green roofs were given fixed values based on literature, with exception of area. An overview of the fixed valued obtained from literature is found in Appendix E. Hence, the only variables for the modelling procedure were the number and combinations of SUDS, their placement and their areas. The rainfall chosen for modelling was a particularly challenging rainfall event which occurred in Oslo on the 5th of august 2015. As the model site is situated in Oslo, this event was deemed appropriate for the purposes of this research.

The results were to be evaluated with regards to the output and given a score for water quantity control. The results would further be used as feedback to improve both the scoring system and the python script for SUDS placements. The scoring system in question is presented in chapter 2.5.

2.5 Scoring system

In order to make the different SUDS configurations comparable, it was considered necessary to develop a scoring system. The procedure to develop a scoring system presented by Jia et al. (2013) was used as an inspiration. The first step was to develop key criteria categories for which a level of index factors within each category would be selected (Jia et al., 2013). By developing a ranking mechanism that integrated every index factor, we could then obtain a score to compare the various SUDS configuration.

For this research, the following key criteria categories were chosen: (1) *Resilience*, (2) *Water quantity* control and (3) *Other benefits*. All the key criteria categories are given weight points depending on their impact on the performance of SUDS. These impact factors were initially set to 1, in order to better evaluate the result of each factor more clearly. These impact factors can also be altered at a later point in order to put emphasis on whichever criteria might be in focus for the given project. Within each key criteria category, different index factors were given points based on their documented effect or benefit for SUDS performance or other desired qualities.

For resilience, the SUDS and SUDS- configurations were ranked based on their performance documented in the literature (Jia et al., 2013; Kristvik et al., 2019, 2018). They were then given points based on their placement in the ranking in order to give the most beneficial combination or configuration the highest value.

For water quantity control, the SUDS configurations were given scores based on their modelled rainfall response. Following the modelling procedure in Urbis, the result for each SUDS configuration was analysed and compared to each other and to the goal of no overflow, and subsequently ranked and given a score.

The score within the third category is adopted from Jia et al. (2013), where the score is a sum of points given for three sub-categories; *rainwater capture and reuse, ecological benefits* and *aesthetic benefits* (Jia et al., 2013). In this evaluation, raingardens are given a higher score than green roofs both for rainwater capture and aesthetic benefits, whereas the ecological benefits are given the same score for the two SUDS.

2.6 LIMITATIONS

The focal point of this research has been the development of a methodology. The study is therefore limited to optimising the placement, size and combination of two types of SUDS, namely raingardens and green roofs. As the objective of the research is the methodology, practical aspects of implementation and maintenance, as well as aesthetical considerations, are not assessed.

The model site is simplified to a homogenous land cover around the buildings. We do not consider pathways, playgrounds, parking places etc. Furthermore, the model site is sloped less than 15%, which is the demand for the implementation of functional raingardens, and is thus exempting us from considering slope throughout the optimisation (Jia et al., 2013). The soil conditions are not a part of the optimisation as the soil maps are deemed to inaccurately represent the actual conditions in the ground, which may have altered due to construction and biological activity in the ground (Oslo Kommune, 2017).

2.6.1 Modified method

Due to computer related obstacles, the presented method could not be performed, specifically the modelling procedure in Urbis. This has led to certain modifications which were made in order to have results to show for and to discuss. A flow chart for the modified method is shown in figure 3.

As modelling results have not been obtained for the SUDS configurations, the scores given for water quantity control in this research are given qualitatively, based on the results derived through the python code. The score is given based on the individual proposal’s ability to provide enough surface area for raingardens. Provided the assumptions made in the development of a SUDS potential model are correct, the score represents a ranking of the ability to handle water quantity. It should, however, be noted that a modelling procedure of the rainfall response of the various configurations would be valuable in order to confirm this ranking.

Furthermore, the intended feedback from Urbis to the optimisation routine and the scoring system is compromised, which significantly alters the intended methodology. The possibility of looping Urbis results with the python code, and thus optimising SUDS placement is not performed. The python code developed to place, and size SUDS is thus a deterministic one, meaning it will give the same result each time. This is both a result of the lack of a loop with Urbis as well as raingardens being the only SUDS considered for placement on the surface. In order to make a stochastic optimisation routine, that would demand a variable, such as the placement of buildings, a larger number of SUDS or varying preferences as input from a modelling procedure. The term optimisation used in the title should therefore in the following be understood as a mere optimisation of early phase planning rather than the optimisation of SUDS.

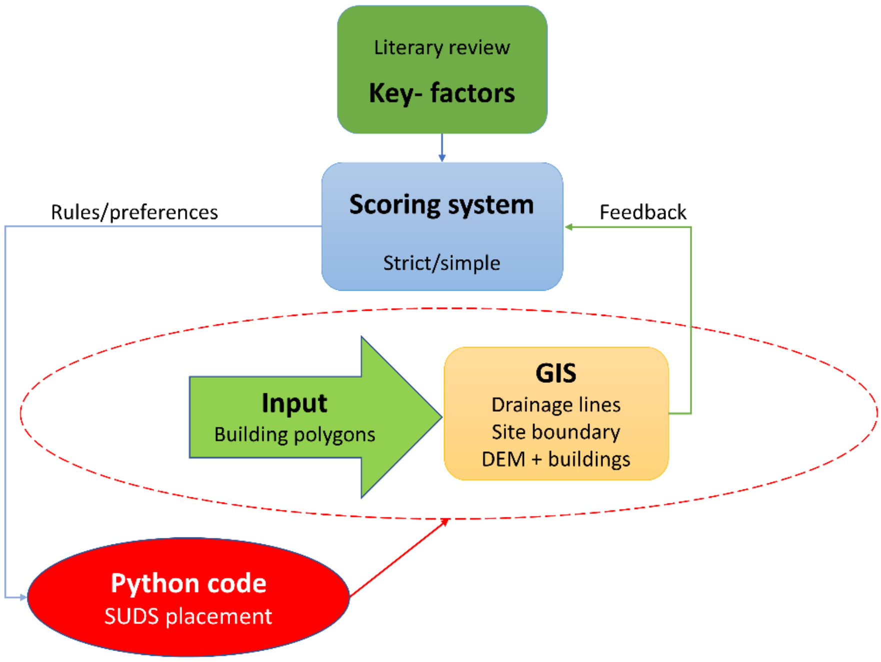


Figure 3- Flow chart for modified method

In the following, all results, discussions and conclusions are based on the modified method.

3 RESULTS

This section is a mere presentation of the results obtained through the methods described in chapter 2 and will be further discussed in chapter 4. It should be noted that the results presented in this chapter are with regards to the assumptions presented throughout the article.

3.1 LITERARY REVIEW

One of the objectives of performing a literary review was to identify factors affecting the performance of the chosen SUDS in terms of their ability to delay flood peaks and handle stormwater volume. Initially, all factors mentioned as important for the performance of SUDS were noted without further evaluation. The next step was to evaluate the identified key factors and further categorise them into groups. It was quickly established that some of the factors gave answers to the question of placement whereas others gave answers to the question of size.

Based on this, the factors were sorted into three main categories according to the discernment of the authors; *placement factors*, *sizing factors* and *other design considerations*. By having all the factors categorised it was easier to get an overview over overlapping terms and these were either clearly separated by distinct terms or combined in one single term, depending on the physical property they were dependent on. The scheme was then completed as an overview of the key factors identified through the literary review for each of the selected SUDS. The scheme was named *the SUDS selection aid.*

Type of consideration	Flood paths	Raingarden	Open detention basin	Green roof	Permeable cover	Swale
Placement factors	Runoff volume, streamlines, topography	Catchment characteristics, depth available, draining area, K_{sat}, soil conditions, topography	Draining area, depth available, soil conditions, streamlines, topography	Climate	Catchment characteristics, K_{sat}, soil conditions, topography	Climate, depth available, K_{sat}, Soil conditions, topography
Sizing factors	Available area, runoff volume	Available area, Catchment characteristics, K_{sat}, evapotranspiration, runoff volume	Available area, interception, runoff volume	Available area, evapotrans piration, design rain, loading capacity of building	Available area, ground stability, K_{sat}, runoff volume, traffic load	Available area, catchment characteristic, evapotranspiratio n, runoff volume
Other design considerations		Accessible for maintenance, distance to building foundation, inflow velocity	Climate	Hight of roof, slope, need to be planned at the time of building design	Avoid large silt loads/vegetation cover on adjacent area, should be downslope from buildings	Interception, land use, difficult in dense urban areas

Table 1- The SUDS selection aid presents the key factors for each type of SUDS derived through a literary study

3.2 Spatial analysis

The spatial analysis consisted of the modelling of drainage lines and the placement of SUDS for each building configuration. The results showed considerable differences between the building proposals, both regarding resulting drainage lines and SUDS potential.

3.2.1 Modelling of drainage lines

The modelling of drainage lines showed a considerable variation in how water flowed through the sites as a response to the various building proposals. In Figure 4, the notable differences of drainage line response is illustrated by displaying the corresponding drainage lines for building proposal 2 and 7. For proposal 2 the buildings are hindering the natural flow to the south, resulting in two outlet points further up on the property, whereas for building proposal 7, all water from the property is crosses the site boundary through a single point in the south. Comparing these results to the flow conditions presented in figure 2 shows that the results for proposal 7 comes close to natural flow conditions whereas the result for building proposal 2 more mirrors the current conditions of the site.

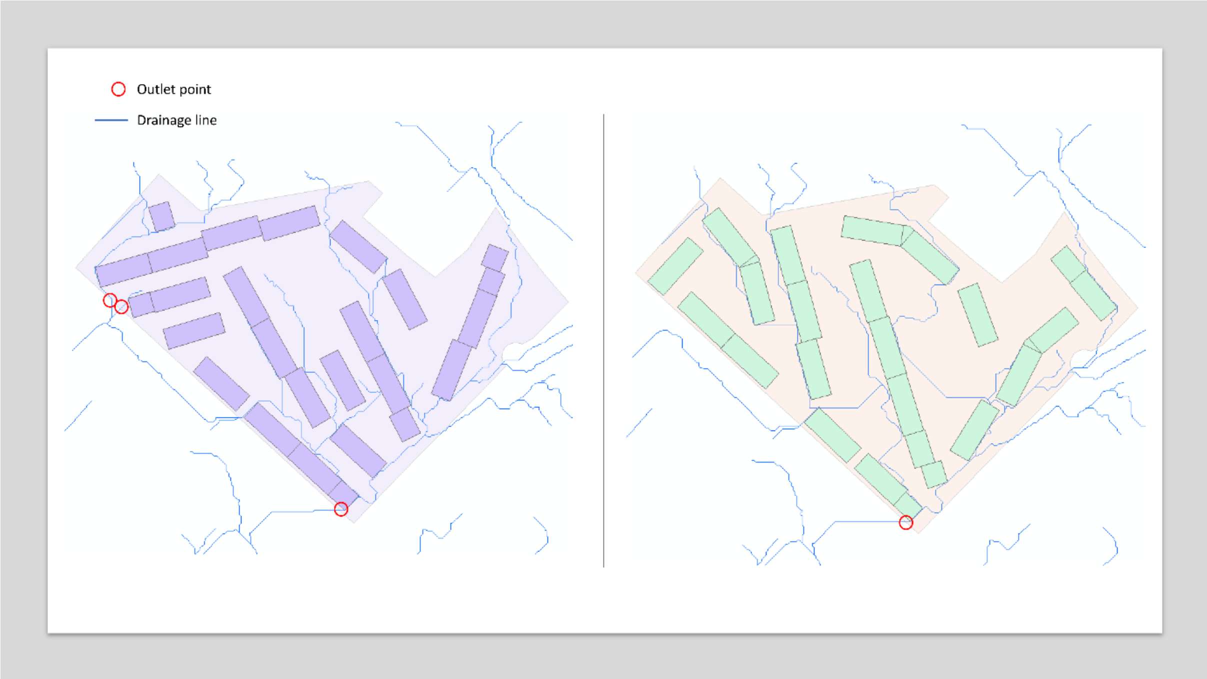


Figure 4- Resulting drainage lines for building proposal 2 (left) and building proposal 7 (right).

The alteration of drainage lines can also be evaluated by looking the resulting number of outlet points from the property for a given proposal. The number of outlet points indicates the number of directions in which SUDS must be placed in order to achieve the goal of no runoff from the property. As can be seen from table 2 the number of outlet points varies from 1 to 4 between the building proposals, which is a considerable difference.

The drainage lines for all building proposals can be found in Appendix B.

3.2.2 Placement of raingardens

The results of the python script for initial placement of rain gardens showed a great variety in the ability to facilitate enough raingarden area as presented in table 2. In line with the demand for a raingarden area equal to 9% of the drainage area, only three building proposals were able to accommodate this demand. In the remaining eight proposals, the raingarden placement potential varied greatly, and the building proposal with the lowest performance considering raingarden placement left 96,5 m^2 of raingarden area unplaced. The difference between the various building proposals is significant and should be noted for further evaluation. Figure 5 shows the resulting plot of alternative 7, which accommodates the demand for raingarden area, and alternative 2, which is the building proposal furthest from meeting the demand.

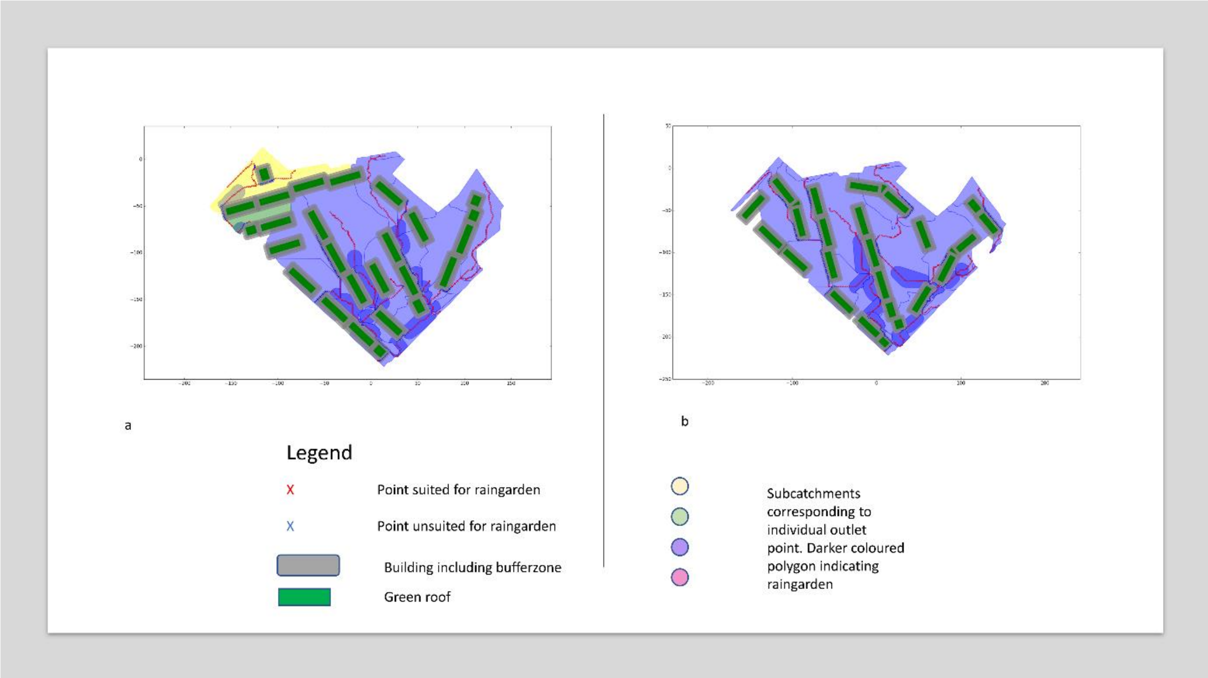


Figure 5- SUDS plot for building proposal 2 (a), which was the situation farthest from facilitating enough surface area for the demanded raingarden area, and building proposal 7 (b), which successfully facilitated enough surface area for raingardens

3.2.3 Placement of green roofs

In this research, all buildings were provided with green roofs. All building proposals were able to accommodate the beneficial connection between green roofs and raingarden, though not in all sub-catchments. However, the amount of connections made varied between 9 and 17 connections, which is a considerable difference.

Building proposal	Number of outlet points	Number of raingardens	Number of green roofs	Number of GR- RG connections	Remaining raingarden area [m^2]
1	4	17	28	13	67,4
2	3	19	26	13	96,5
3	3	17	28	14	89,5
4	2	16	28	16	0
5	4	17	30	15	36,6
6	3	9	27	9	53
7	1	13	28	9	0
8	2	19	23	12	0
9	3	21	27	16	31,2
10	3	16	23	12	92
11	4	17	34	17	51,3

Table 2- An overview of the number of outlet points, raingardens, green roofs and raingarden-green roof connections made. The last column shows the remaining raingarden area that the corresponding building proposal failed to facilitate surface area for.

Plots for SUDS placement of all building proposals can be found in Appendix F. A numbered plot, showing a numbering system of raingardens, green roofs and their connections can be found in Appendix G.

3.3 SCORING SYSTEM

The score for the index factors within each key criteria category is presented below. However, the water quantity control was qualitatively scored and is not given a general score here. The resulting score for each building proposal's SUDS configuration can be found in table 4, chapter 3.3.1. The manner in which the system was developed is described in chapter 2.5 along with the procedure of obtaining the individual key criteria scores.

SUDS/configuration	Resilience	Other benefits	Water quantity control
GR	1	9	N/A
RG	1,5	12	N/A
2 x RG	2	N/A	N/A
2 x RG w/ 2 K_{sat}	2,5	N/A	N/A
Max score, S_{max}	3,5	12	N/A

Table 3- Scoring system showing the score for each index factor within each key criteria category

The resulting score within each key criteria category was normalized using the following equation:

$$f_j = \frac{\sum_{i=1}^{n} Si}{S_{max,j}}$$

Equation 1

Where,

f_j is the score for the j^{th} key criteria category

Si is the score for the i^{th} index factor

$S_{max,j}$ is the highest obtainable score for the j^{th} key criteria category

Each of the SUDS configurations obtained through optimisation could then be given a total score using the following equation:

$$X_j = \sum_{i=1}^{4} e_i * f_{ij}, \mathrm{j} = [1,2,3,4]$$

Equation 2

Where,

X_j is the score for the SUDS configuration connected to outlet point *j*

e_i is the weight factor for the i^{th} key criteria category

f_{ij} is the score within the i^{th} key category for the j^{th} SUDS configuration

It should be noted that the terms *SUDS* configuration is used for the configuration of all SUDS within one sub-catchment of the site. The number of sub- catchments equals the number of outlet points from the site. The final score for the site will be the sum of the scores for each sub-catchment, weighted by the sub-catchment's fraction of the total site area, using the following equation:

$$S_{tot,k} = \frac{\sum_{j=1}^{n} X_j * W_j}{n_j}, k = 1, 2, 3, \dots, 11$$

Equation 3

Where,

$S_{tot,k}$ is the total score for the k^{th} building proposal's SUDS configuration

X_j is the score for the SUDS configuration connected to outlet point *j*

W_j is the j^{th} sub-catchment's fraction of the total site area

n_j is the total number of outlet points

3.3.1 Resulting score of SUDS configurations

Building proposal	Number of outlet points	Remaining raingarden area [m^2]	Resilience score	Other benefits score	Water quantity score	Score
1	4	67,4	0,9	0,9	0,3	2,10
2	3	96,5	0,94	0,94	0	1,88
3	3	89,5	0,93	0,93	0,2	2,06
4	2	0	0,94	0,94	1	2,88
5	4	36,6	0,91	0,90	0,6	2,41
6	3	53	0,94	0,94	0,4	2,28
7	1	0	0,94	0,94	1	2,88
8	2	0	0,92	0,92	1	2,84
9	3	31,2	0,93	0,93	0,7	2,56
10	3	92	0,93	0,93	0,1	1,96
11	4	51,3	0,91	0,91	0,5	2,32

Table 4 - Resulting score for the SUDS configuration of each building proposal as well as scores within each key criteria category

By use of the presented scoring system, the SUDS configuration for each building proposal was given a score. The scores range from 1,88 to 2,88. Assessing the various key criteria categories, it is evident that the category that contributes most to the distinction of the total score is the water quantity score. The scores within resilience and other benefits present a smaller variety.

4 DISCUSSION

In this article, a methodology to automatize the placement and dimensioning of SUDS and SUDS combination has been presented. In this section the results are discussed in light of the research questions.

4.1 COMPUTER PROGRAMMING POTENTIAL

The need for SUDS is clearly stated in the literature (Eckart et al., 2018; Ugarelli et al., 2017; Woods-Ballard et al., 2007) and is now also a demand in the municipality of Oslo (Oslo Kommune, 2017). In Norway, SUDS are traditionally considered late in the planning process, but clear guidelines now state that they should be considered earlier (Oslo Kommune, 2013). However, the complexity of SUDS has been identified as a barrier for implementation of such solutions (Eckart et al., 2017). Furthermore, it has been questioned if it is even feasible to analyse the many possible configurations of SUDS for a site (Eckart et al., 2018). Given this complexity, simple trial- and- error approaches are deemed inappropriate for the purpose of SUDS planning (Zhang and Chui, 2018). However, the complexity that computer programming can handle seems to surpass that of SUDS, according to the research reported on in this article. In this research, we have been able to create a general script, applicable for other sites and situations, with the ability to calculate the need for raingarden area as well as placing both raingardens and green roofs on the site. For this specific research, only 11 building proposals were assessed, it should however be noted that the script could have been run for a much higher number of building proposals. This would more clearly illustrate the time saving potential of the methodology.

Though the methodology developed in this research is not a comprehensive one in the sense that it does not include all types of SUDS, it clearly shows the potential for digitalisation of stormwater planning. By simply assessing the impact of various building proposals on the drainage lines, we can say something about a building proposals suitability for SUDS. Through a spatial analysis, the alternation of drainage lines has been illustrated thereby offering a way to improve the traditional approach to stormwater management; building proposals with a negative impact on the flood situation can be rejected at an early stage, thus saving both time and money.

The methodology presented in this article is a simple one, demanding little input, but is still providing valuable information about placement of buildings and SUDS. SUDS are highly context dependent, meaning that correct placement and construction is paramount in order to secure their function. By use of this methodology, we can ensure that areas suitable for SUDS are secured at an early stage when assessing all their demands is an actual possibility. In that way we can help ensure that SUDS perform the much-needed resilience they have proved to provide.

Assessing the SUDS selection aid obtained through the literary review in this research, it seems evident that there are many rules for the implementation of SUDS, and that many key factors coincide for researches across continent boundaries. There are both clear rules, guidelines and desires for placement, dimensioning and combinations of SUDS. None of which are too complex to assess in a script. Translating the planning and dimensioning of SUDS to a script has proved challenging, but is, however, possible. Such a script will only execute the concrete assignments it has been given, meaning desires and guidelines need to be scripted in a way that holds for a general situation. This is a time demanding task but is nevertheless feasible according to the research conducted.

As this research only concerns two types of SUDS, assessing all SUDS would, no doubt, increase the complexity of the scrip considerably. On the other hand, making a general script for the optimisation of SUDS is a one-time effort which in turn exempts us from having to face the complexity of SUDS each time stormwater management is assessed. For each time such a script is used, it can be evaluated and updated and thus continuously improve.

4.2 Key factors for early assessment of SUDS

In the development of the SUDS selection aid, three main categories for key factors were obtained: *Placement factors*, *Sizing factors* and *Other design considerations*. Some terms were overlapping for two or more categories as they had an impact on multiple aspects of the SUDS. Initially, the factors for placement and size were considered the most important ones. However, the key factors included in *Other design considerations* turned out to be very valuable for the purpose of this research as they gave more information about SUDS relations to surrounding assets, such as buildings.

Computer programming can handle an enormous detail level. It can be discussed, however, whether assessing all possible factors is necessary. Considering the significant variations in potential for SUDS placement obtained through the relatively simple script created in this research, the improvements that can be made through only a few steps seem notable. In the development of a script for SUDS placement, we were not able to take all key factors into account. However, by assess in only a few factors, we are able to give some information about which building proposals are more suitable than others. The results presented in table 2 shows that the building proposals resulting in the lowest numbers of outlet points, are the proposals that best facilitates raingarden area. We may not be able to say that the proposed SUDS configurations for the successful proposals are sufficient, but we can, however, say something about which proposals are likely to have a negative impact on the drainage lines and the SUDS potential. In other words, in order to facilitate an improvement of stormwater management, only the consideration of a few factors may be enough.

The three steps in the three- stage approach to stormwater management are presented, quite intuitively, based on the severity of the rainfall events. The first step concerns the management of everyday rainfall events while the third step concerns securing safe flood paths. This implies a way of thinking concerning stormwater managements. Reviewing the results in this research, however, it could be argued that a reversion of this three- stage approach would be more desirable in terms of stormwater planning. The SUDS selection aid shows that securing safe flood paths does not depend upon many factors. Additionally, safe flood paths are related to the nature of the drainage lines, which is shown in this research to be strongly affected by the physical layout of a development project. Given their relatively simple nature, the potential to assess some aspect of drainage lines before the physical layout of a major project is decided, seems clear. More important than simplicity and potential is the importance of securing safe flood paths for a flood event in urban areas.

Furthermore, regarding SUDS implementation, it is evident through the results of this research that the potential flood reducing effect of this early assessment is significant. In this research where only 11 building proposals are assessed, the variation in SUDS potential reflected in the ability to facilitate raingarden area is considerable. Maybe the complete design and dimensioning of SUDS at an early stage in the process is a little bit down the road, however, a simple assessment early on could give very valuable information and save both time and money. This indicates that with some rules or incentives to where buildings should be placed with regards to drainage line, could strongly enhance the current practice and thus the flood safety of a development project. Accepting the cost of such an early assessment should be easy to accept as damage to property and ecosystems as a result of urban floods often has proved to exceed the cost of stormwater management (Eckart et al., 2018).

4.3 Performance of the proposed methodology

The concrete results of this research, being the SUDS placement, size and combinations for Marienlyst is not as important as what these results imply. The results clearly imply that the physical layout of a property has severe influence on the drainage lines and flood paths as well as potential for SUDS implementation. More importantly, the results imply that it actually is possible to assess this in a simple way at an early stage. The results from the placement of rain gardens for 11 different building proposals showed considerable difference in the ability to facilitate enough surface area for raingardens. This indicates the importance of assessing SUDS potential before the physical layout of the site is determined. In urban areas where the damage potential in a flood event is large, securing the stormwater handling ability of a site is of grave importance.

An important result is that most of the building proposals are able to accommodate the beneficial configurations of green roof and raingarden, resulting in a rather small variation of score for this key criteria category. However, by adding the score for water quantity control, the image is quite another as the distinction between the building proposals is much clearer. A development of the scoring system to mirror the site’s actual ability to handle stormwater would be beneficial. The weight factors of the key criteria categories could be altered in order to achieve this. Based on the results and objective of this research, the water quantity control key factor criteria should be weighted heavier than resilience and other benefits. The reliability of the resulting score would however have been higher if the score was given based on modelling results. Nevertheless, a scoring system is deemed useful in order to optimise SUDS configurations.

The qualitative score of water quantity control is limited in the way that it is only assessed based on the ability to accommodate 9% of the drained area for raingarden area. It can be expected that the introduction of green roofs will reduce the need for raingarden area. This would be beneficial to illustrate in a modelling study. However, limited or not, this research does clearly illustrate that the building proposals have a great impact on a site’s ability to provide sufficient surface area for stormwater management.

Due to time limitation, cost has not been a part of the optimisation and scoring system in this research. It should however be noted, that cost should also be a part of the optimisation. It is of the author’s opinion important that such a score should account both for structural and maintenance costs but should not fail to assess the costs saved due to avoided flood incidents.

The performance of the methodology presented in this research may be evaluated in terms of the concrete results, it should however be noted that the implications brought forth through these results are of a much higher value. The research has shown that scripting placement and size of raingardens, green roofs and their interconnection is possible. The suggested methodology can clearly be improved. However, the general script created in this research resulted in quite telling variation of SUDS placement and flood security performance. Providing the assumptions made in the development of the script is correct, the importance of assessing SUDS early in a development project is clearly shown through the greatly varying results in SUDS potential for the eleven proposals.

5 Conclusion

The results obtained through this research shows both the potential that lies in early assessment of SUDS as well as the negative consequences that failing to do so might lead to. A simple assessment of drainage lines and building placement appears to have a considerable impact on the SUDS potential for a development project, and consequently the ability to handle stormwater sufficiently, avoid floods and save money.

Through this research, a change of mindset is also implied, as the complexity of SUDS has been proved to be manageable through computer programming. The development of new software is certainly providing a possibility for digitalising stormwater management and optimisation of SUDS. Furthermore, a call to change of mindset has been suggested through the reversion of the three-stage approach to stormwater management. The most severe flood incidents, which are the most damaging ones with regards to property and human health, seem to be the least complex ones to assess, and should therefore be at the front of the line when planning for stormwater management.

The scoring system developed through this research is limited to the two SUDS assessed. Future work should seek to develop a comprehensive scoring system, providing a score that can more accurately mirror the performance of SUDS configurations with regards to water quantity control. This could be of assistance in an optimisation routine where the objective could be to obtain the highest possible score.

For future work, the presented script could be developed and improved by use of genetic algorithm. In genetic algorithms, good solutions are identified in a population of solutions and used to make new, better solutions, whereas bad solutions are eliminated. In that way, computer learning can be used to improve such scripts at a high rate (Deb, 1999). This type of algorithm could be used for designing software to optimise SUDS and could also be utilized to obtain guidelines for developers in situations where the use of such a software is not an option.

For processes where the assessment of multiple building proposals is not a possibility, guidelines should be put forth for the placement of buildings with regards to drainage lines and SUDS placement. These guidelines could then be used by architects, landscapers or others with an impact on the physical layout of a major project. This would improve the current practice and ensure a better approach to stormwater management at an early stage.

A small step towards optimising SUDS configurations has been made through this research. The potential in developing this methodology is clearly stated. Any attempts to further develop or use the results in this research are more than welcome.

Acknowledgements

This research was supported by Klima 2050 (http://www.klima2050.no/), BINGO- a better future under climate change (http://www.projectbingo.eu/) and NKF (Norsk Kommunalteknisk Forening).

BIBLIOGRAPHY

Arksey, H., O'Malley, L., 2005. Scoping studies: Towards a methodological framework. Int. J. Soc. Res. Methodol. Theory Pract. 8, 19–32. https://doi.org/10.1080/1364557032000119616

Deb, K., 1999. An introduction to genetic algorithms. Sadhana 24, 293–315.

Eckart, K., McPhee, Z., Bolisetti, T., 2018. Multiobjective optimization of low impact development stormwater controls. J. Hydrol. https://doi.org/10.1016/j.jhydrol.2018.04.068

Eckart, K., McPhee, Z., Bolisetti, T., 2017. Performance and implementation of low impact development – A review. Sci. Total Environ. https://doi.org/10.1016/j.scitotenv.2017.06.254

Hodnesdal, H., 2018. Manglende håndtering av overvann er et betydelig problem [WWW Document]. Finans Norge. URL https://www.finansnorge.no/aktuelt/nyheter/2018/10/manglende-handtering-av-overvann-er-et-betydelig-problem/

Jia, H., Yao, H., Tang, Y., Yu, S.L., Zhen, J.X., Lu, Y., 2013. Development of a multi-criteria index ranking system for urban runoff best management practices (BMPs) selection. Environ. Monit. Assess. 185, 7915–7933. https://doi.org/10.1007/s10661-013-3144-0

Johannessen, B.G., Hanslin, H.M., Muthanna, T.M., 2017. Green roof performance potential in cold and wet regions. Ecol. Eng. 106, 436–447. https://doi.org/10.1016/j.ecoleng.2017.06.011

Johannessen, B.G., Muthanna, T.M., Braskerud, B.C., 2018. Detention and retention behavior of four extensive green roofs in three Nordic climate zones. Water (Switzerland) 10, 1–23. https://doi.org/10.3390/w10060671

Kazak, J.K., Chruściński, J., Szewrański, S., 2018. The development of a novel decision support system for the location of green infrastructure for stormwater management. Sustain. 10. https://doi.org/10.3390/su10124388

Kristvik, E., Johannessen, B., Muthanna, T., Kristvik, E., Johannessen, B.G., Muthanna, T.M., 2019. Temporal Downscaling of IDF Curves Applied to Future Performance of Local Stormwater Measures. Sustainability 11, 1231. https://doi.org/10.3390/su11051231

Kristvik, E., Kleiven, G.H., Lohne, J., Muthanna, T.M., 2018. Assessing the robustness of raingardens under climate change using SDSM and temporal downscaling. Water Sci. Technol. 77, 1640–1650. https://doi.org/10.2166/wst.2018.043

Liu, Y., Theller, L.O., Pijanowski, B.C., Engel, B.A., 2016. Optimal selection and placement of green infrastructure to reduce impacts of land use change and climate change on hydrology and water quality: An application to the Trail Creek Watershed, Indiana. Sci. Total Environ. 553, 149–163. https://doi.org/10.1016/j.scitotenv.2016.02.116

Magnussen, K., Wingstedt, A., Rasmussen, I., Reinvang, R., 2015. Kostnader og nytte ved overvannstiltak.

Norsk Vann, 2005. Veileder i overvannshåndtering, rapport 144/2005.

Oslo Kommune, 2017. Overvannshåndtering- En veileder for utbygger.

Oslo Kommune, 2013. Strategi for overvannshåndtering i Oslo.

Paus, K.H., Braskerud, B.C., 2013. Forslag til dimensjonering og utforming av regnbed for norske forhold. Vann.

Paus, K.H., Muthanna, T.M., Braskerud, B.C., 2015. Uncorrected Proof implications for design. Hydrol.

Res. 1–14. https://doi.org/10.2166/nh.2015.084

RIF, 2015. Norges tilstand 2015 State of the nation.

Ugarelli, R., Hidalgo Martínez, C., Ahmadi, M., Raspati, G., Sivertsen, E., 2017. ASSET MANAGEMENT OF NATURE-BASED SOLUTIONS: WHAT INFORMATION TO COLLECT FOR MAINTENANCE MANAGEMENT-APPLICATION IN TRONDHEIM, NORWAY. LESAM Conf.

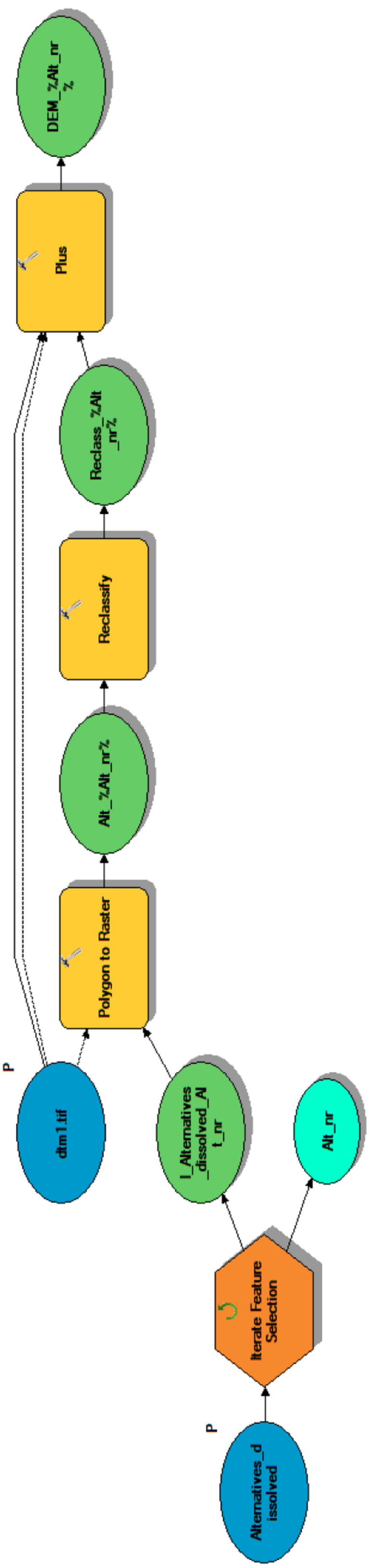
Wohlin, C. (Blekinge I. of T., 2014. Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering, in: 18th International Conference on Evaluation and Assessment in Software Engineering.

Woods-Ballard, B., Kellagher, R., Woods Ballard, B., Construction Industry Research and Information Association, Great Britain, Department of Trade and Industry, Environment Agency, 2007. The SUDS manual, Ciria, ….

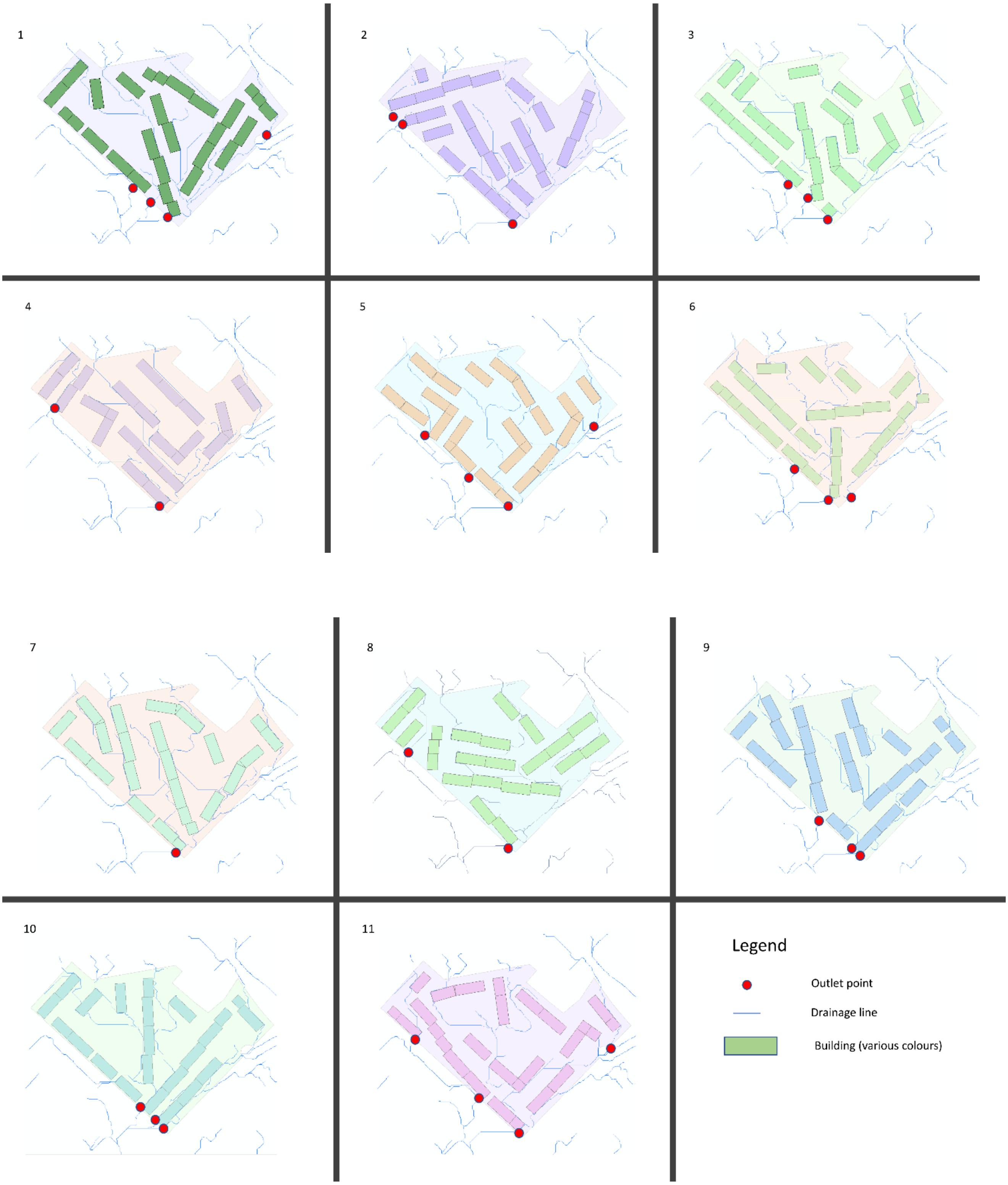
Zhang, K., Chui, T.F.M., 2018. A comprehensive review of spatial allocation of LID-BMP-GI practices: Strategies and optimization tools. Sci. Total Environ. 621, 915–929. https://doi.org/10.1016/j.scitotenv.2017.11.281

Zhu, Z., Chen, Z., Chen, X., Yu, G., 2019. An assessment of the hydrologic effectiveness of low impact development (LID) practices for managing runoff with different objectives. J. Environ. Manage. 231, 504–514. https://doi.org/10.1016/j.jenvman.2018.10.046

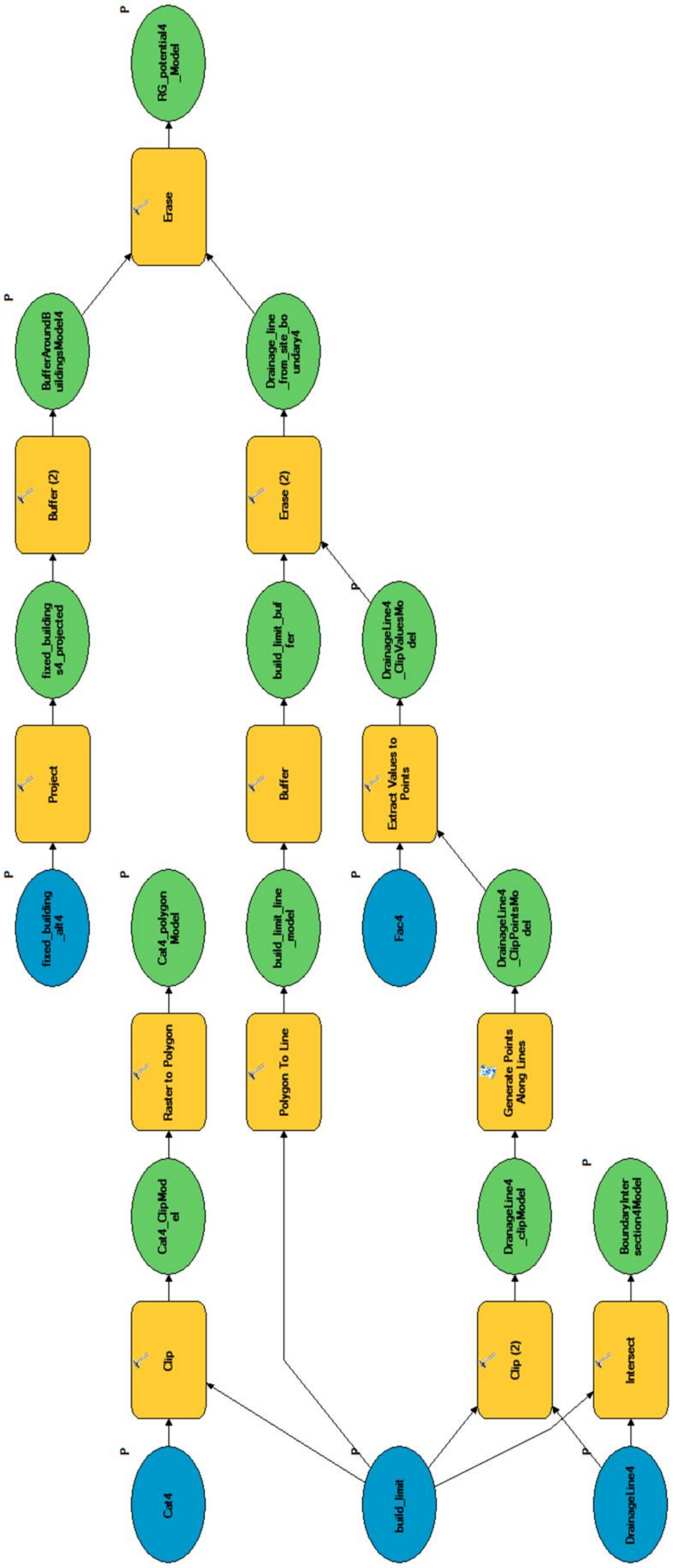
Appendix A- DEM manipulation model



Appendix B- Resulting drainage lines for all building proposals



Appendix C – Spatial analysis model



Appendix D – Python scripts

arcgis_analysis.py

```
# -*- coding: utf-8 -*-
# ---------------------------------------------------------------------------
--
# arcgis_analysis.py
# Created on: 2019-05-14 10:03:23.00000
#   (generated by ArcGIS/ModelBuilder)
# Usage: arcgis_analysis-2 <Cat4> <build_limit> <DrainageLine4> <Fac4>
<fixed_building_alt4> <Cat4_polygonModel> <BoundaryIntersection4Model>
<DrainageLine4_ClipValuesModel> <RG_potential4_Model>
<BufferAroundBuildingsModel4>
# Description:
# This is a model performing all the analysis we need, after the
hydrological analysis/modelling is performed
# ---------------------------------------------------------------------------
--

# Set the necessary product code
# import arcinfo


# Import arcpy module
import arcpy
arcpy.env.workspace = "C:\\Users\\guros\\OneDrive - NTNU\\Master vår
2019\\"
arcpy.env.overwriteOutput=True


for ii in range(4, 15):
    i = str(ii)

    # Script arguments
    print("Starting with {}\nSetting parameter values...\n".format(i))
    Cat = "ArcGIS_faktisk\\Layers\\cat{}".format(i)

    build_limit = "ArcGIS_faktisk\\build_limit\\build_limit.shp"

    DrainageLine =
"ArcGIS_faktisk\\avrenningslinjer_garra_original.gdb\\DrainageLine{}".forma
t(i)

    Fac = "ArcGIS_faktisk\\Layers\\fac{}".format(i)

    fixed_building_alt =
"ArcGIS_faktisk\\alternatives\\alt{}\\fixed_building_alt{}.shp".format(i,
i)

    Cat_polygonModel =
"ArcGIS_faktisk\\Resultat.gdb\\Cat{}_polygonModel".format(i)

    BoundaryIntersectionModel =
"ArcGIS_faktisk\\Resultat.gdb\\BoundaryIntersection{}Model".format(i)

    DrainageLine_ClipValuesModel =
"ArcGIS_faktisk\\Resultat.gdb\\DrainageLine{}_ClipValuesModel".format(i)
```

```
    RG_potential_Model =
"ArcGIS_faktisk\\Resultat.gdb\\RG_potential{}_Model".format(i)

    BufferAroundBuildingsModel =
"ArcGIS_faktisk\\Resultat.gdb\\BufferAroundBuildingsModel{}".format(i)

    # Local variables:
    Cat_ClipModel =
"ArcGIS_faktisk\\Resultat.gdb\\Cat{}_ClipModel".format(i)
    DrainageLine_clipModel =
"ArcGIS_faktisk\\Resultat.gdb\\DranageLine{}_clipModel".format(i)
    DrainageLine_ClipPointsModel =
"ArcGIS_faktisk\\Resultat.gdb\\DrainageLine{}_ClipPointsModel".format(i)
    build_limit_line_model =
"ArcGIS_faktisk\\Resultat.gdb\\build_limit_line_model"
    build_limit_buffer = "ArcGIS_faktisk\\Resultat.gdb\\build_limit_buffer"
    Drainage_line_from_site_boundary =
"ArcGIS_faktisk\\Resultat.gdb\\Drainage_line_from_site_boundary{}".format(i
)
    fixed_buildings_projected =
"ArcGIS_faktisk\\Resultat.gdb\\fixed_buildings{}_projected".format(i)

    # Process: Intersect
    print("Intersection between DrainageLine and build_limit")
    arcpy.Intersect_analysis("{} #;{} #".format(DrainageLine, build_limit),
BoundaryIntersectionModel, "ALL", "", "POINT")

    # Process: Clip
    print("Clipping Cat to build_limit")
    arcpy.Clip_management(Cat, "260797,731117598 6651867,76572595
261153,825933687 6652129,89853315", Cat_ClipModel, build_limit, "128",
"ClippingGeometry", "NO_MAINTAIN_EXTENT")

    # Process: Raster to Polygon
    print("RasterToPolygon for clipped catchment")
    arcpy.RasterToPolygon_conversion(Cat_ClipModel, Cat_polygonModel,
"SIMPLIFY", "Value", "SINGLE_OUTER_PART", "")

    # Process: Clip (2)
    print("Clipping DrainageLine within build_limit")
    arcpy.Clip_analysis(DrainageLine, build_limit, DrainageLine_clipModel,
"")

    # Process: Generate Points Along Lines
    print("Generating points along DrainageLine within build_limit")
    arcpy.GeneratePointsAlongLines_management(DrainageLine_clipModel,
DrainageLine_ClipPointsModel, "DISTANCE", "2 Meters", "", "")

    # Process: Extract Values to Points
    print("Extracting values from Fac to DrainageLine_ClipPointsModel")
    arcpy.CheckOutExtension("Spatial")
    arcpy.gp.ExtractValuesToPoints_sa(DrainageLine_ClipPointsModel, Fac,
DrainageLine_ClipValuesModel, "NONE", "VALUE_ONLY")
    arcpy.CheckInExtension("Spatial")

    # Process: Polygon To Line
    print("Converting build_limit to line around polygon")
    arcpy.PolygonToLine_management(build_limit, build_limit_line_model,
"IDENTIFY_NEIGHBORS")

    # Process: Buffer
```

```
    print("Creating buffer around site boundary")
    arcpy.Buffer_analysis(build_limit_line_model, build_limit_buffer, "3 
Meters", "FULL", "ROUND", "NONE", "", "PLANAR")

    # Process: Erase (2)
    print("Erasing drainage line points within site limit buffer zone")
    arcpy.Erase_analysis(DrainageLine_ClipValuesModel, build_limit_buffer, 
Drainage_line_from_site_boundary, "")

    # Process: Project
    print("Projecting building polygons")
    arcpy.Project_management(fixed_building_alt, fixed_buildings_projected, 

"PROJCS['ETRS_1989_UTM_Zone_33N',GEOGCS['GCS_ETRS_1989',DATUM['D_ETRS_1989'
,SPHEROID['GRS_1980',6378137.0,298.257222101]],PRIMEM['Greenwich',0.0],UNIT
['Degree',0.0174532925199433]],PROJECTION['Transverse_Mercator'],PARAMETER[
'False_Easting',500000.0],PARAMETER['False_Northing',0.0],PARAMETER['Centra
l_Meridian',15.0],PARAMETER['Scale_Factor',0.9996],PARAMETER['Latitude_Of_O
rigin',0.0],UNIT['Meter',1.0]]",
                             "ETRS_1989_To_WGS_1984",

"PROJCS['WGS_1984_UTM_Zone_32N',GEOGCS['GCS_WGS_1984',DATUM['D_WGS_1984',SP
HEROID['WGS_1984',6378137.0,298.257223563]],PRIMEM['Greenwich',0.0],UNIT['D
egree',0.0174532925199433]],PROJECTION['Transverse_Mercator'],PARAMETER['Fa
lse_Easting',500000.0],PARAMETER['False_Northing',0.0],PARAMETER['Central_M
eridian',9.0],PARAMETER['Scale_Factor',0.9996],PARAMETER['Latitude_Of_Origi
n',0.0],UNIT['Meter',1.0]]",
                             "NO_PRESERVE_SHAPE", "", "NO_VERTICAL")

    # Process: Buffer (2)
    print("Creating buffer zone around buildings")
    arcpy.Buffer_analysis(fixed_buildings_projected, 
BufferAroundBuildingsModel, "2 Meters", "FULL", "ROUND", "NONE", "", 
"PLANAR")

    # Process: Erase
    print("Erasing DrainageLine_ClipValuesModel within buffer zone around 
buildings")
    arcpy.Erase_analysis(DrainageLine_ClipValuesModel, 
BufferAroundBuildingsModel, RG_potential_Model, "")

    print("Finished with {}\n\n".format(i))
```

extract_features_from_layer.py

```
import arcpy
from shapely.geometry import Polygon as shp_poly


class Point:
    def __init__(self, x, y, from_node, to_node, arc_id, n_draining_cells):
        self.x = x
        self.y = y
        self.from_node = from_node
        self.to_node = to_node
        self.arc_id = arc_id
        self.n_draining_cells = n_draining_cells

        self.catchment_area = None
        self.predecessors = []
```

```
        self.successors = []

    def get_tree(self):
        return [self] + [point for list_of_points in [p.get_tree() for p in
self.predecessors] for point in list_of_points]

    def get_upstream_draining_area(self):
        tree = self.get_tree()
        all_catchments = set([point.catchment_area for point in tree])

        return sum([shp_poly(catchment.coordinates).area for catchment in
all_catchments])


class ArcgisPolygon:
    def __init__(self, coordinates):
        self.coordinates = coordinates

        self.points_within = []


class GreenRoof:
    def __init__(self, coordinates, rain_garden_connection):
        self.coordinates = coordinates
        self.rain_garden_connection = rain_garden_connection


def extract_polygons(infc, ref_point=None):
    if not ref_point:
        ref_point = [0., 0.]

    polygons = []
    for row in arcpy.da.SearchCursor(infc, ["SHAPE@"]):
        for part in row[0]:
            coordinates = []

            for pnt in part:
                if pnt:
                    coordinates.append([pnt.X - ref_point[0], pnt.Y -
ref_point[1]])
                else:
                    # If pnt is None, this represents an interior ring
                    print("Interior Ring.\nNo polygon added.")
            polygons.append(ArcgisPolygon(coordinates))
    return polygons


def extract_points(infc, ref_point=None):
    if not ref_point:
        ref_point = [0., 0.]

    points = []
    for row in arcpy.da.SearchCursor(infc, ["SHAPE@", "from_node",
"to_node", "arcid", "RASTERVALU"]):
        for pnt in row[0]:
            points.append(Point(pnt.X - ref_point[0], pnt.Y - ref_point[1],
row[1], row[2], row[3], row[4]))

    return points
```

plot_features_from_layer.py

```python
# -*- coding: utf-8 -*-

import arcpy
from extract_features_from_layer import extract_polygons, extract_points
from plot_helper import plot_polygons_lines_and_points as plot


ref_point = [260985.0, 6652104.0]
arcpy.env.workspace = "C:\\Users\\guros\\OneDrive - NTNU\\Master vår 
2019\\"


def test_buildings_plot():
    infc = "ArcGIS_faktisk\\Resultat.gdb\\buildings12_projected"

    buildings = extract_polygons(infc, ref_point)
    plot(blue_buildings=[buildings_polygons.buildings_coordinates for 
buildings_polygons in buildings])


def test_polygon_plot():
    infc = "ArcGIS_faktisk\\Resultat.gdb\\Cat12_polygonModel"

    catchments = extract_polygons(infc, ref_point)
    plot(blue_polygons=[polygon.coordinates for polygon in catchments])


def test_point_plot():
    infc_blue = "ArcGIS_faktisk\\Resultat.gdb\\RG_potential12_Model"
    infc_red = 
"ArcGIS_faktisk\\Resultat.gdb\\DrainageLine12_ClipValuesModel"

    blue_points = extract_points(infc_blue, ref_point)
    red_points = extract_points(infc_red, ref_point)
    plot(blue_points=[[point.x, point.y] for point in blue_points], 
red_points=[[point.x, point.y] for point in red_points])


def test_polygon_and_line_plot():
    infc = "ArcGIS_faktisk\\Resultat.gdb\\Cat12_polygonModel"
    infc_blue = "ArcGIS_faktisk\\Resultat.gdb\\RG_potential12_Model"
    infc_red = 
"ArcGIS_faktisk\\Resultat.gdb\\DrainageLine12_ClipValuesModel"
    infc_yellow = "ArcGIS_faktisk\\Resultat.gdb\\buildings12_projected"

    catchments = extract_polygons(infc, ref_point)
    blue_points = extract_points(infc_blue, ref_point)
    red_points = extract_points(infc_red, ref_point)
    buildings = extract_polygons(infc_yellow, ref_point)

    # plot(yellow_polygons=[buildings_polygons.coordinates_buildings for 
buildings_polygons in buildings],
    #      green_polygons=[polygon.coordinates for polygon in catchments],
    #      blue_points=[[point.x, point.y] for point in blue_points],
    #      red_points=[[point.x, point.y] for point in red_points],
    #      )


test_polygon_and_line_plot()
```

Point_catchment_family.py

```
# -*- coding: utf-8 -*-
from extract_features_from_layer import extract_polygons, extract_points,
GreenRoof
from shapely.geometry import Point as shp_point, Polygon as shp_polygon, \
    MultiPolygon as shp_multi, \
    LineString as shp_line, mapping

from plot_helper import plot_polygons_lines_and_points


def representative_point(polygon):
    shapley_representative_point =
shp_polygon(polygon).representative_point()
    x = shapley_representative_point.x
    y = shapley_representative_point.y

    return [x, y]


def calculate_area_of_rain_garden(draining_area, size_percentage=0.09):
    return draining_area*size_percentage


def sort_points_by_number_of_draining_cells(points):
    sorted_points = sorted(points, key=lambda p: p.n_draining_cells,
reverse=True)
    return sorted_points


def find_outlet_points(points):
    outlet_points = []
    covered_from_to_pairs = set()

    sorted_points = sort_points_by_number_of_draining_cells(points)

    for point in sorted_points:
        from_to_pair = (point.from_node, point.to_node)

        already_covered = from_to_pair in covered_from_to_pairs
        if already_covered:
            continue

        covered_from_to_pairs.add(from_to_pair)

        predecessor_of_already_covered = any([pair[0] == point.to_node for
pair in covered_from_to_pairs])
        if predecessor_of_already_covered:
            continue

        outlet_points.append(point)

    return outlet_points


def update_to_node_for_outlet_point_pair(new_to_node_value, outlet_point,
points):
    from_to_pair_for_outlet_point = (outlet_point.from_node,
outlet_point.to_node)
    for point in points:
```

```
        from_to_pair_for_point = (point.from_node, point.to_node)

        if from_to_pair_for_point == from_to_pair_for_outlet_point:
            point.to_node = new_to_node_value


def update_point_network(to_node, points, successor=None):
    points_with_correct_to_node = [point for point in points if
point.to_node == to_node]
    from_to_pairs = list(set([(point.from_node, point.to_node) for point in
points_with_correct_to_node]))

    for from_to_pair in from_to_pairs:
        points_between_same_nodes = [point for point in points
                                     if point.from_node == from_to_pair[0]
                                     and point.to_node == from_to_pair[1]]

        sorted_points_between_same_nodes =
sorted(points_between_same_nodes,
                                                  key=lambda p:
p.n_draining_cells,
                                                  reverse=True)

        if successor:

sorted_points_between_same_nodes[0].successors.append(successor)

successor.predecessors.append(sorted_points_between_same_nodes[0])

        for point_index, point in
enumerate(sorted_points_between_same_nodes[:-1]):

point.predecessors.append(sorted_points_between_same_nodes[point_index +
1])
            sorted_points_between_same_nodes[point_index +
1].successors.append(point)

        update_point_network(from_to_pair[0],
                             points,
                             successor=sorted_points_between_same_nodes[-
1])


def pair_points_and_catchments(points, catchments):
    for point in points:
        for catchment in catchments:
            if
shp_polygon(catchment.coordinates).contains(shp_point([point.x, point.y])):
                point.catchment_area = catchment
                catchment.points_within.append(point)
                break


def update_to_nodes_for_points_related_to_outlets(outlet_points,
list_of_list_of_points_to_update):
    for outlet_point_index, outlet_point in enumerate(outlet_points):
        for list_of_points in list_of_list_of_points_to_update:
            update_to_node_for_outlet_point_pair(-outlet_point_index - 1,
outlet_point, list_of_points)
```

```python
def find_upstream_rain_gardens(start_point, required_raingarden_area,
allowed_points, building_buffer_polygons):
    rain_gardens = []
    current_catchment = start_point.catchment_area

    current_point = start_point
    next_point = current_point.predecessors[0]

    while (required_raingarden_area > 0
           and len(current_point.predecessors) == 1
           and next_point.catchment_area == current_catchment):
        current_point_available = any([(p.x, p.y) == (current_point.x,
current_point.y) for p in allowed_points])

        if not current_point_available:
            current_point = current_point.predecessors[0]
            next_point = current_point.predecessors[0] if
len(current_point.predecessors) > 0 else None
            continue

        subsequent_allowable_points = [current_point]
        while len(next_point.predecessors) == 1 \
                and next_point.predecessors[0].catchment_area ==
current_catchment \
                and any([(p.x, p.y) == (next_point.x, next_point.y) for p
in allowed_points]):
            subsequent_allowable_points.append(next_point)
            next_point = next_point.predecessors[0]

        if len(subsequent_allowable_points) > 1:
            # find polygon of raingarden
            rain_garden_polygon =
find_raingarden_polygon(subsequent_allowable_points,

required_raingarden_area,

building_buffer_polygons,
                                                          rain_gardens)
            rain_gardens.append(rain_garden_polygon)
            required_raingarden_area -=
shp_polygon(rain_garden_polygon).area

        current_point = next_point

        if not len(next_point.predecessors) > 0:
            break

        next_point = current_point.predecessors[0]

    if required_raingarden_area <= 0:
        return rain_gardens, required_raingarden_area

    if len(current_point.predecessors) < 1:
        return rain_gardens, required_raingarden_area

    if len(current_point.predecessors) > 1:
        rain_gardens_from_predecessors = []
        remaining_area_from_predecessors = []
        for predecessor in current_point.predecessors:
            if len(predecessor.predecessors) < 1:
                continue
```

```
            total_draining_area = predecessor.get_upstream_draining_area()
            required_area_of_rain_garden =
min(calculate_area_of_rain_garden(total_draining_area),
required_raingarden_area)

            rain_gardens_from_predecessor, remaining_area_to_place =
find_upstream_rain_gardens(
                predecessor,
                required_area_of_rain_garden,
                allowed_points,
                building_buffer_polygons
            )

            area_placed = required_area_of_rain_garden -
remaining_area_to_place
            required_raingarden_area -= area_placed


rain_gardens_from_predecessors.append(rain_gardens_from_predecessor)

remaining_area_from_predecessors.append(remaining_area_to_place)

        rain_gardens += [raingardens for predecessor_rgs in
rain_gardens_from_predecessors
                         for raingardens in predecessor_rgs]
    elif not next_point.catchment_area == current_point.catchment_area\
            and len(next_point.predecessors) > 0:
        total_draining_area = next_point.get_upstream_draining_area()
        required_area_of_rain_garden =
min(calculate_area_of_rain_garden(total_draining_area),
required_raingarden_area)

        rain_gardens_from_predecessor, remaining_area_to_place =
find_upstream_rain_gardens(
            next_point,
            required_area_of_rain_garden,
            allowed_points,
            building_buffer_polygons
        )

        area_placed = required_area_of_rain_garden -
remaining_area_to_place
        required_raingarden_area -= area_placed

        rain_gardens += rain_gardens_from_predecessor

    return rain_gardens, required_raingarden_area


def find_raingarden_polygon(subsequent_allowable_points,
                            required_raingarden_area,
                            building_buffer_polygons,
                            previous_rain_gardens):
    max_distance = 8.0
    lower_catchment_area = 100.0

    catchment = subsequent_allowable_points[0].catchment_area

    if shp_polygon(catchment.coordinates).area < lower_catchment_area:
        return []
```

```
    line_segment = shp_line([(p.x, p.y) for p in
subsequent_allowable_points])
    raingarden_polygon = line_segment.buffer(max_distance).convex_hull
    raingarden_polygon =
raingarden_polygon.intersection(shp_polygon(catchment.coordinates))

    for preoccupied_polygons in [bbp.coordinates for bbp in
building_buffer_polygons] + previous_rain_gardens:
        if
raingarden_polygon.intersects(shp_polygon(preoccupied_polygons)):
            raingarden_polygon =
raingarden_polygon.difference(shp_polygon(preoccupied_polygons))

    if isinstance(raingarden_polygon, shp_multi):
        possible_raingarden_polygons = list(raingarden_polygon)
        for poly in possible_raingarden_polygons:
            if any([poly.contains(shp_point([p.x, p.y])) for p in
subsequent_allowable_points]):
                raingarden_polygon = poly
                break
        if isinstance(raingarden_polygon, shp_multi):
            return []

    if raingarden_polygon.area > 1.1*required_raingarden_area and
len(subsequent_allowable_points) > 2:
        polygon_based_on_one_point_less =
find_raingarden_polygon(subsequent_allowable_points[:-1],
                                                            required_raingarden_area,
                                                            building_buffer_polygons,
                                                            previous_rain_gardens)
        if shp_polygon(polygon_based_on_one_point_less).area >=
required_raingarden_area:
            return polygon_based_on_one_point_less

    return mapping(raingarden_polygon)["coordinates"][0]


def place_rain_gardens_for_outlet_point(outlet_points,
allowed_rain_garden_points, building_buffer_polygons):
    all_rain_gardens = []
    all_remaining_area = []
    for outlet_point in outlet_points:
        total_draining_area = outlet_point.get_upstream_draining_area()
        required_area_of_rain_garden =
calculate_area_of_rain_garden(total_draining_area)

        rain_gardens, missing_area =
find_upstream_rain_gardens(outlet_point,

required_area_of_rain_garden,

allowed_rain_garden_points,

building_buffer_polygons)

        all_rain_gardens.append([rg for rg in rain_gardens if rg])
        all_remaining_area.append(missing_area)

    return all_rain_gardens, all_remaining_area
```

```
def find_required_drainage_area_for_point(point):
    all_points = point.get_tree()
    all_catchments = set([p.catchment_area for p in all_points])
    area_of_all_catchments = sum([shp_polygon(catchment.coordinates).area
for catchment in all_catchments])
    required_drainage_area =
calculate_area_of_rain_garden(area_of_all_catchments)

    return required_drainage_area


def place_green_roofs(buildings, raingardens, required_green_roof_area):
    green_roofs = []

    for building in buildings:
        roof_polygon = building.coordinates
        area_of_roof = shp_polygon(roof_polygon).area
        green_roof_polygon = roof_polygon

        while shp_polygon(green_roof_polygon).area >
required_green_roof_area*area_of_roof + 5.0:
            green_roof_polygon =
mapping(shp_polygon(green_roof_polygon).buffer(-0.1))["coordinates"][0]

        nearest_raingarden = sorted(raingardens, key=lambda rg:
shp_polygon(roof_polygon).distance(shp_polygon(rg)))[0]
        distance_to_nearest_raingarden =
shp_polygon(roof_polygon).distance(shp_polygon(nearest_raingarden))

        if distance_to_nearest_raingarden > 4.0:
            nearest_raingarden = None

        green_roofs.append(GreenRoof(green_roof_polygon,
nearest_raingarden))

    return green_roofs


def update_green_roofs_with_name_of_rain_garden_connections(green_roofs,
outlet_points, rain_gardens):
    rain_gardens_with_names = [(rg, "{}-{}".format(rg_list_i+1, rg_i+1))
                               for rg_list_i, rg_list in
enumerate(rain_gardens)
                               for rg_i, rg in enumerate(rg_list)]
    catchments_with_names = []
    for op_i, op in enumerate(outlet_points):
        op_name = str(op_i+1)
        catchments_for_op = list(set([p.catchment_area for p in
op.get_tree()]))
        for catchment in catchments_for_op:
            catchments_with_names.append((catchment.coordinates, op_name))

    for green_roof in green_roofs:
        if green_roof.rain_garden_connection is not None:
            for rain_garden, name in rain_gardens_with_names:
                if rain_garden == green_roof.rain_garden_connection:
                    green_roof.rain_garden_connection = name
                    break
        else:
            name_of_nearest_catchment = sorted(catchments_with_names,
```

```
                                                    key=lambda
catchment_with_name:
shp_polygon(green_roof.coordinates).distance(shp_polygon(catchment_with_nam
e[0])))[0][1]
            green_roof.rain_garden_connection = name_of_nearest_catchment


def place_rain_gardens_on_site(input_path_to_rain_garden_points,
                               input_path_to_drainage_line_points,
                               input_path_to_catchment,
                               input_path_to_buildings,
                               input_path_to_buildings_with_buffer,
                               ref_point,
                               required_green_roof_ratio):
    points_rg = extract_points(input_path_to_rain_garden_points, ref_point)
    points_dl = extract_points(input_path_to_drainage_line_points,
ref_point)

    catchments = extract_polygons(input_path_to_catchment, ref_point)
    buildings = extract_polygons(input_path_to_buildings, ref_point)
    building_buffer_polygons =
extract_polygons(input_path_to_buildings_with_buffer, ref_point)

    pair_points_and_catchments(points_dl + points_rg, catchments)
    points_dl = [p for p in points_dl if p.catchment_area]
    points_rg = [p for p in points_rg if p.catchment_area]

    outlet_points = find_outlet_points(points_dl)

    update_to_nodes_for_points_related_to_outlets(outlet_points,
[points_dl, points_rg])

    total_catchment_area = sum([shp_polygon(catchment.coordinates).area for
catchment in catchments])
    required_area = calculate_area_of_rain_garden(total_catchment_area)

    for outlet_point in outlet_points:
        update_point_network(outlet_point.to_node, points_dl,
successor=None)

    all_rain_gardens, all_remaining_area =
place_rain_gardens_for_outlet_point(outlet_points,

points_rg,

building_buffer_polygons)

    all_rain_gardens_flat = [rg for rg_list in all_rain_gardens for rg in
rg_list]

    green_roofs = place_green_roofs(buildings,
                                    all_rain_gardens_flat,
                                    required_green_roof_ratio)
    update_green_roofs_with_name_of_rain_garden_connections(green_roofs,
outlet_points, all_rain_gardens)

    blue_polygons = [catchment.coordinates for catchment in
set([p.catchment_area for p in outlet_points[0].get_tree()])]
    yellow_polygons = [catchment.coordinates for catchment in
set([p.catchment_area for p in outlet_points[1].get_tree()])] if
len(outlet_points) > 1 else []
```

```
    green_polygons = [catchment.coordinates for catchment in
set([p.catchment_area for p in outlet_points[2].get_tree()])] if
len(outlet_points) > 2 else []
    red_polygons = [catchment.coordinates for catchment in
set([p.catchment_area for p in outlet_points[3].get_tree()])] if
len(outlet_points) > 3 else []

    plot_polygons_lines_and_points(red_points=5*[(p.x, p.y) for p in
outlet_points] + [(p.x, p.y) for p in points_rg],
                                   blue_points=[(p.x, p.y) for p in
points_dl],
                                   blue_polygons=blue_polygons + [rg for
rg_list in all_rain_gardens for rg in rg_list],
                                   yellow_polygons=yellow_polygons,
                                   green_polygons=green_polygons,
                                   red_polygons=red_polygons,
                                   gray_polygons=[b.coordinates for b in
buildings + building_buffer_polygons] * 2,
                                   white_green_polygons=[gr.coordinates for
gr in green_roofs],
                                   points_with_text=[((p.x - 5, p.y - 5),
p_i+1) for p_i, p in enumerate(outlet_points)] +\

[(representative_point(gr.coordinates), gr.rain_garden_connection) for gr
in green_roofs] +\

[(representative_point(rg), "{}-{}".format(rgl_i + 1, rg_i + 1)) for rgl_i,
rgl in enumerate(all_rain_gardens) for rg_i, rg in enumerate(rgl)]
                                    )

    return outlet_points, all_rain_gardens, required_area,
all_remaining_area, green_roofs
```

plot-helper.py

```
import matplotlib
import matplotlib.patches
import matplotlib.pyplot as plt
from matplotlib.lines import Line2D


def plot_rain_garden_points(all_rain_gardens, catchments, points_dl,
buildings, buffered_buildings):
    all_rain_gardens_flattened = [rg for list_of_rg in all_rain_gardens for
rg in list_of_rg]

    plot_polygons_lines_and_points(green_polygons=[c.coordinates for c in
catchments],
                                   blue_polygons=[p for p in
all_rain_gardens_flattened if p],
                                   red_points=[(p.x, p.y) for p in
points_dl],
                                   red_polygons=[b.coordinates for b in
buildings + buffered_buildings])


def plot_polygons_lines_and_points(
   blue_polygons=None,
   blue_lines=None,
   blue_points=None,
```

```
    yellow_polygons=None,
    yellow_lines=None,
    red_polygons=None,
    red_lines=None,
    red_points=None,
    green_polygons=None,
    green_lines=None,
    gray_polygons=None,
    white_green_polygons=None,
    additional_polygons=None,
    points_with_text=None,
):
    lines_2d = []
    patches = [] if additional_polygons is None else additional_polygons
    if blue_polygons:
        for p in blue_polygons:
            polygon = matplotlib.patches.Polygon(p, True, alpha=0.4,
color="blue")
            patches.append(polygon)
    if red_polygons:
        for p in red_polygons:
            polygon = matplotlib.patches.Polygon(p, True, alpha=0.4,
color="red")
            patches.append(polygon)
    if yellow_polygons:
        for p in yellow_polygons:
            polygon = matplotlib.patches.Polygon(p, True, alpha=0.4,
color="yellow")
            patches.append(polygon)
    if yellow_lines:
        for line in yellow_lines:
            line_2d = Line2D(
                [p[0] for p in line], [p[1] for p in line], color="yellow",
linewidth=1
            )
            lines_2d.append(line_2d)
    if green_polygons:
        for p in green_polygons:
            polygon = matplotlib.patches.Polygon(p, True, alpha=0.4,
color="green")
            patches.append(polygon)
    if green_lines:
        for line in green_lines:
            line_2d = Line2D(
                [p[0] for p in line], [p[1] for p in line], color="green",
linewidth=1
            )
            lines_2d.append(line_2d)
    if gray_polygons:
        for p in gray_polygons:
            polygon = matplotlib.patches.Polygon(p, True, alpha=0.4,
color="gray")
            patches.append(polygon)
    if white_green_polygons:
        for p in white_green_polygons:
            polygon = matplotlib.patches.Polygon(p, True, color="green")
            patches.append(polygon)
    if blue_lines:
        for line in blue_lines:
            line_2d = Line2D(
                [p[0] for p in line], [p[1] for p in line], color="blue",
```

```
linewidth=1
            )
            lines_2d.append(line_2d)
    if blue_points:
        for px, py in blue_points:
            lines_2d.append(
                Line2D(
                    [px - 0.5, px + 0.5],
                    [py - 0.5, py + 0.5],
                    color="blue",
                    linewidth=1,
                )
            )
            lines_2d.append(
                Line2D(
                    [px - 0.5, px + 0.5],
                    [py + 0.5, py - 0.5],
                    color="blue",
                    linewidth=1,
                )
            )
    if red_lines:
        for line in red_lines:
            line_2d = Line2D(
                [p[0] for p in line], [p[1] for p in line], color="red",
linewidth=1
            )
            lines_2d.append(line_2d)
    if red_points:
        for px, py in red_points:
            lines_2d.append(
                Line2D(
                    [px - 0.5, px + 0.5], [py - 0.5, py + 0.5], color="red",
linewidth=1
                )
            )
            lines_2d.append(
                Line2D(
                    [px - 0.5, px + 0.5], [py + 0.5, py - 0.5], color="red",
linewidth=1
                )
            )

    fig = plt.figure()
    ax = fig.add_subplot(111)
    ax.axis("auto")

    if points_with_text:
        for p, text in points_with_text:
            ax.text(p[0], p[1], text)

    for patch in patches:
        ax.add_patch(patch)

    for line in lines_2d:
        ax.add_line(line)

    plt.axis("equal")

    plt.show()
```

main.py

```
# -*- coding: utf-8 -*-

import arcpy
import json

from point_catchment_family import place_rain_gardens_on_site,
find_required_drainage_area_for_point
from shapely.geometry import Polygon as shp_polygon

ref_point = [260985.0, 6652104.0]
arcpy.env.workspace = "C:\\Users\\guros\\OneDrive - NTNU\\Master vår
2019\\"


results = {}

for building_alternative in range(4, 15):
    print("Starting with {}...".format(building_alternative))
    infc_rg =
"ArcGIS_faktisk\\Resultat.gdb\\RG_potential{}_Model".format(building_altern
ative)
    infc_dl =
"ArcGIS_faktisk\\Resultat.gdb\\DrainageLine{}_ClipValuesModel".format(build
ing_alternative)
    infc_cat =
"ArcGIS_faktisk\\Resultat.gdb\\Cat{}_polygonModel".format(building_alternat
ive)
    infc_buildings =
"ArcGIS_faktisk\\Resultat.gdb\\buildings{}_projected".format(building_alter
native)
    infc_building_with_buffer =
"ArcGIS_faktisk\\Resultat.gdb\\BufferAroundBuildingsModel{}".format(buildin
g_alternative)

    output_results = place_rain_gardens_on_site(infc_rg,
                                                infc_dl,
                                                infc_cat,
                                                infc_buildings,
                                                infc_building_with_buffer,
                                                ref_point,
                                                0.4)

    outlet_points, rain_gardens, required_area_total, remaining_area,
green_roofs = output_results

    remaining_area = [round(area, 1) for area in remaining_area]
    required_area_pr_outlet_point =
[round(find_required_drainage_area_for_point(op), 1) for op in
outlet_points]

    results["Run {}".format(building_alternative)] =
{"required_total_raingarden_area": round(required_area_total, 1),

"sum_of_required_raingarden_area_pr_outlet_point":
sum(required_area_pr_outlet_point),

"sum_of_remaining_area": sum([area for area in remaining_area if area >=
0]),
```

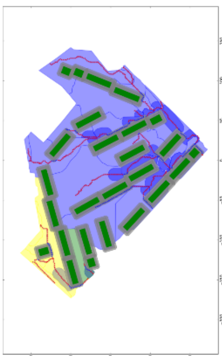
```
"required_area_pr_outlet_point": required_area_pr_outlet_point,

"remaining_area_pr_outlet_point": remaining_area,

"rain_gardens_with_areas": ["{}-{}, area: {}".format(
                                                          rg_for_op_i+1,
rg_i+1, round(shp_polygon(rg).area, 1))
                                                          for rg_for_op_i,
rg_for_op in enumerate(rain_gardens)
                                                          for rg_i, rg in
enumerate(rg_for_op)],

"green_roof_areas_with_connection": ["{}, area: {}".format(

gr.rain_garden_connection, round(shp_polygon(gr.coordinates).area), 1)
                                                          for gr in
green_roofs]}
    print("Sum of remaining area: {}".format(sum([area for area in
remaining_area if area > 0])))
    print("Finished!\n\n")

print("_____\nFinal results")
print(json.dumps(results))
```

Appendix E – Fixed values from literature for modelling input

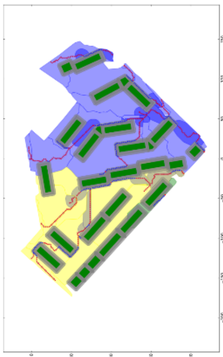
SUDS	Variables	Description	Other design considerations	Numbers	Source
Raingarden	**A_raingarden**	Surface area of raingarden [m2]	Distance to buildings > 1,5 m	Available area	Paus, K.H., Braskerud, B.C., 2013. Forslag til dimensjonering og utforming av regnbed for norske forhold. Vann.
	A_subcat	Area of sub-catchment [m2]		Slope of raingarden (5%, <20%)	Paus, K.H., Braskerud, B.C., 2013. Forslag til dimensjonering og utforming av regnbed for norske forhold. Vann.
	c	Average runoff coefficient for the catchment [-]	Buffer distance to roads < 30m, buffer distance to stram >30, to buildings >3m		Jia, H., Yao, H., Tang, Y., Yu, S.L., Zhen, J.X., Lu, Y., 2013. Development of a multi-criteria index ranking system for urban runoff best management practices (BMPs) selection. Environ. Monit. Assess. 185, 7915–7933. https://doi.org/10.1007/s10661-013-3144-0
	P	Dimensioning precipitation (input for modelling)		Precip input	
	h_max	Height of water table when it goes to overflow [m]		0,15-0,30 m	Paus, K.H., Braskerud, B.C., 2013. Forslag til dimensjonering og utforming av regnbed for norske forhold. Vann.
	K_sat	Hydraulic conductivity of filter media [m/t]		(40 cm tjukt) K > 0,1 m/h	Paus, K.H., Braskerud, B.C., 2013. Forslag til dimensjonering og utforming av regnbed for norske forhold. Vann.
	t_r	Dimensioning duration of runoff into the raingarden [t] (time of concentration?)			
Green roof			ET = 4mm/ day in Oslo		Johannessen, B.G., Hanslin, H.M., Muthanna, T.M., 2017. Green roof performance potential in cold and wet regions. Ecol. Eng. 106, 436–447. https://doi.org/10.1016/j.ecoleng.2017.06.011
	ET			hydraulic cond > 0,6-70mm/min to avoid ponding	Johannessen, B.G., Muthanna, T.M., Braskerud, B.C., 2018. Detention and retention behavior of four extensive green roofs in three

					Nordic climate zones. Water (Switzerland) 10, 1–23. https://doi.org/10.3390/w10060671
	A_roof	Area available for green roof [m2]	Slope < 10 %		Woods-Ballard, B., Kellagher, R., Woods Ballard, B., Construction Industry Research and Information Association, Great Britain, Department of Trade and Industry, Environment Agency, 2007. The SUDS manual, Ciria, ….
	Substrate depth	Height of substrate [m]		0,10 m	Johannessen, B.G., Hanslin, H.M., Muthanna, T.M., 2017. Green roof performance potential in cold and wet regions. Ecol. Eng. 106, 436–447. https://doi.org/10.1016/j.ecoleng.2017.06.011
	Field capacity			Storage = min 0,05 m	Johannessen, B.G., Hanslin, H.M., Muthanna, T.M., 2017. Green roof performance potential in cold and wet regions. Ecol. Eng. 106, 436–447. https://doi.org/10.1016/j.ecoleng.2017.06.011
	Wilting point				
Permeable cover	**A_surface**	Surface area of permeable cover [m2]			
	Permeability			Runoff coeff 0,40	Woods-Ballard, B., Kellagher, R., Woods Ballard, B., Construction Industry Research and Information Association, Great Britain, Department of Trade and Industry, Environment Agency, 2007. The SUDS manual, Ciria, ….

Appendix F – SUDS placement for all building proposals

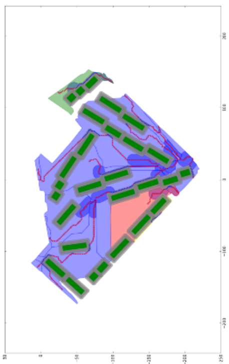


1

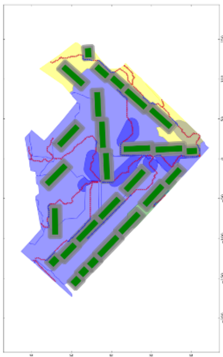


2

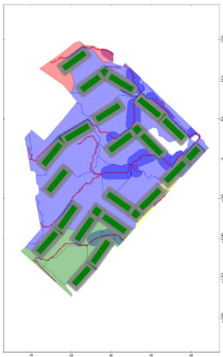
3



4

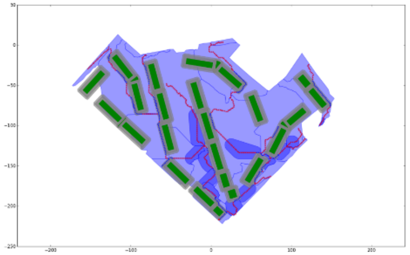


5

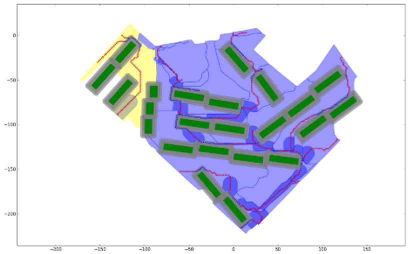


6

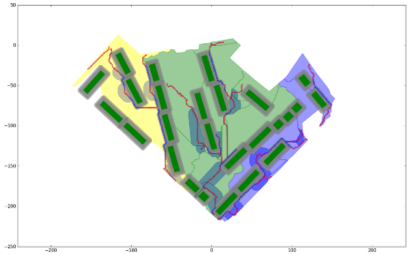
7



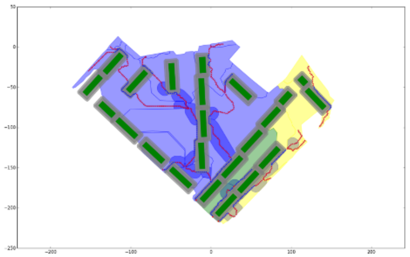
8



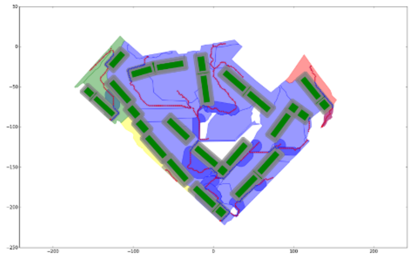
9



10



11



Legend

X Point suited for raingarden

X Point unsuited for raingarden

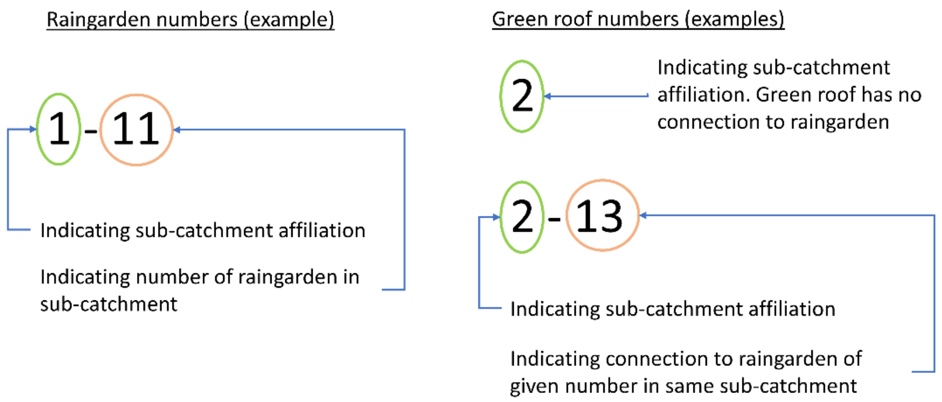
Subcatchments corresponding to individual outlet point. Darker coloured polygon indicating raingarden

Building including bufferzone

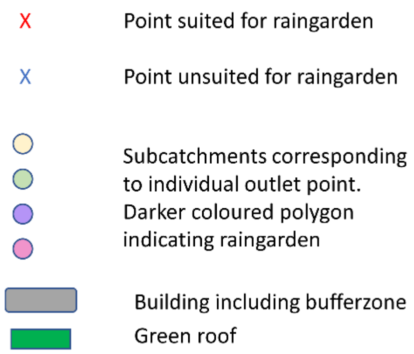
Green roof

Appendix G – SUDS placements, numbered

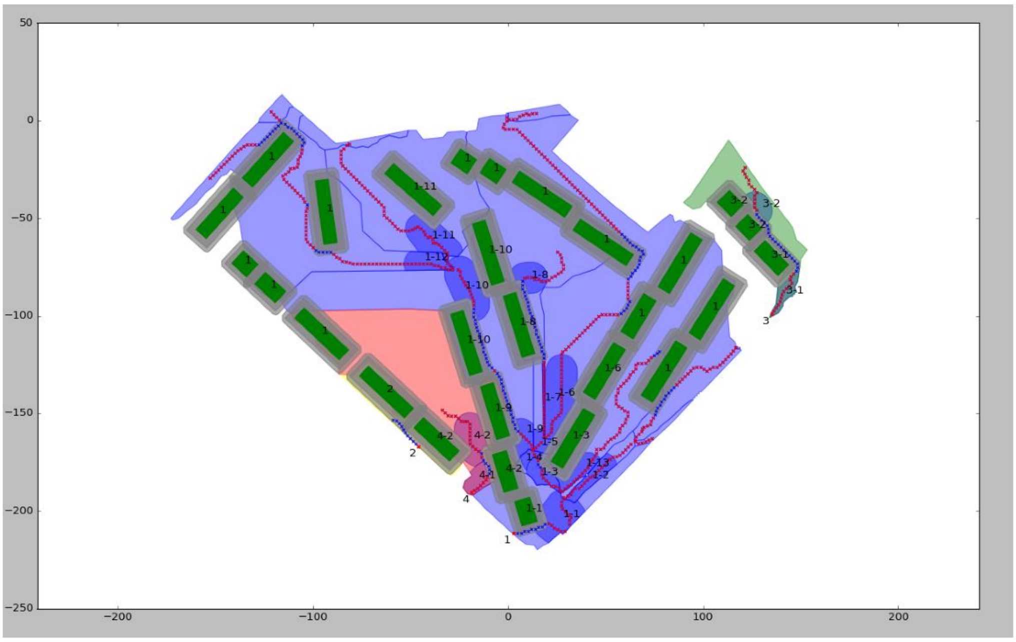
Explanation of numbers



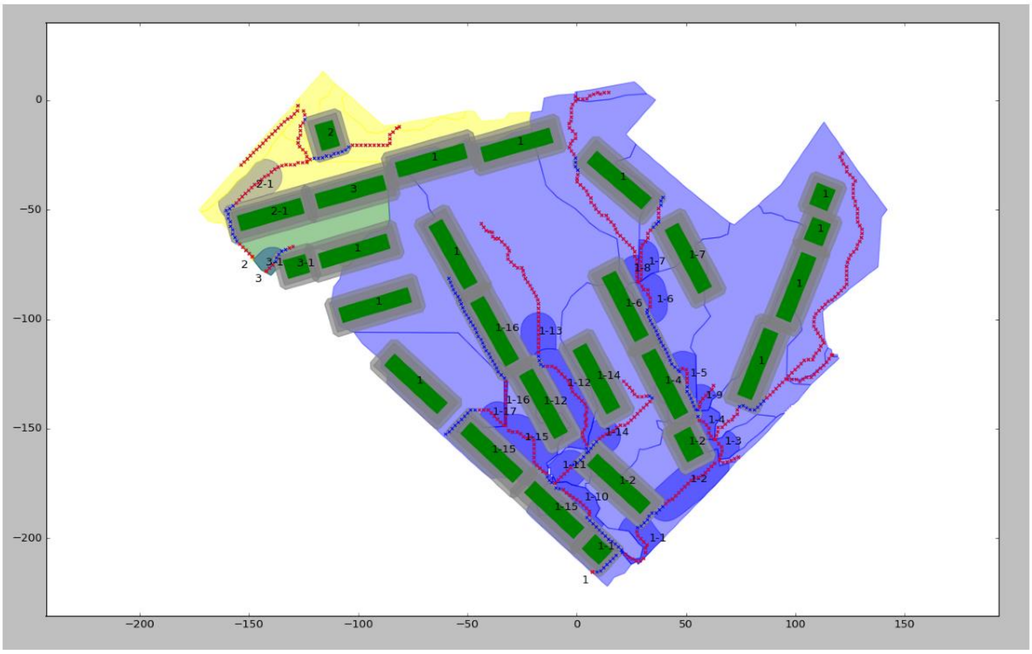
Legend



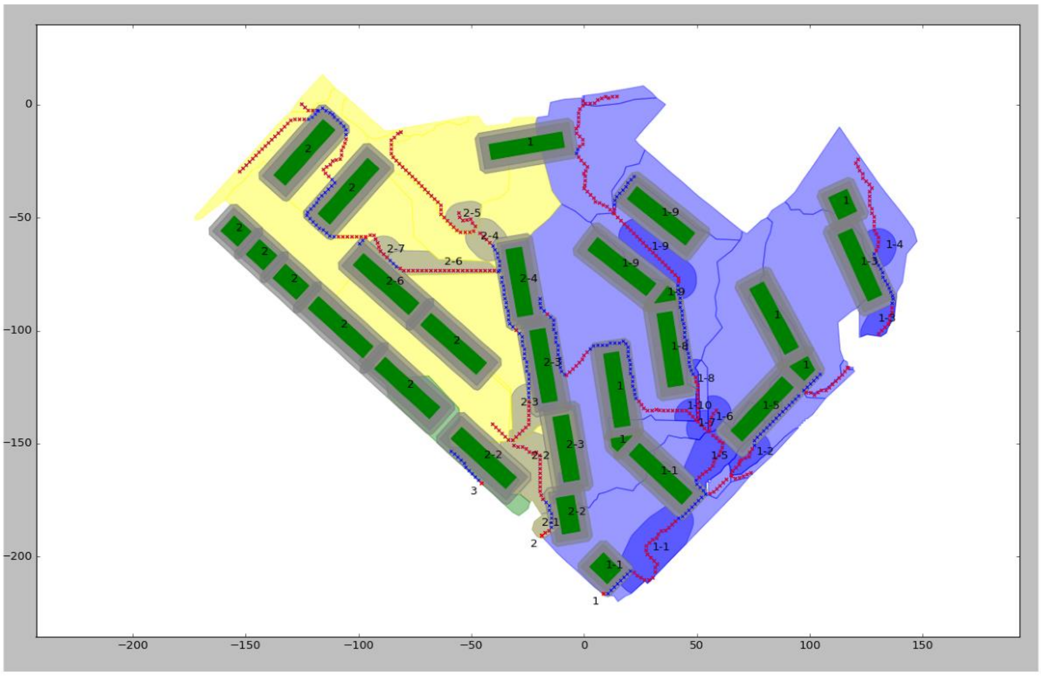
1


50
0
-50
-100
-150
-200
-250
-200
-100
0
100
200

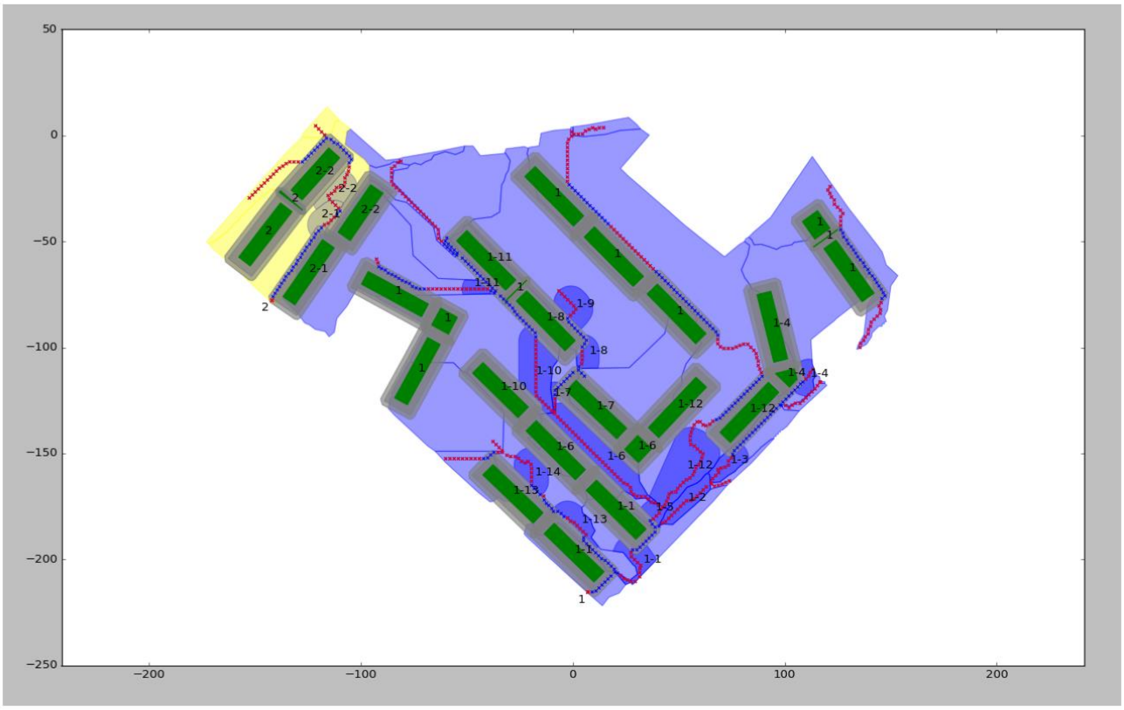
2


0
-50
-100
-150
-200
-200
-150
-100
-50
0
50
100
150

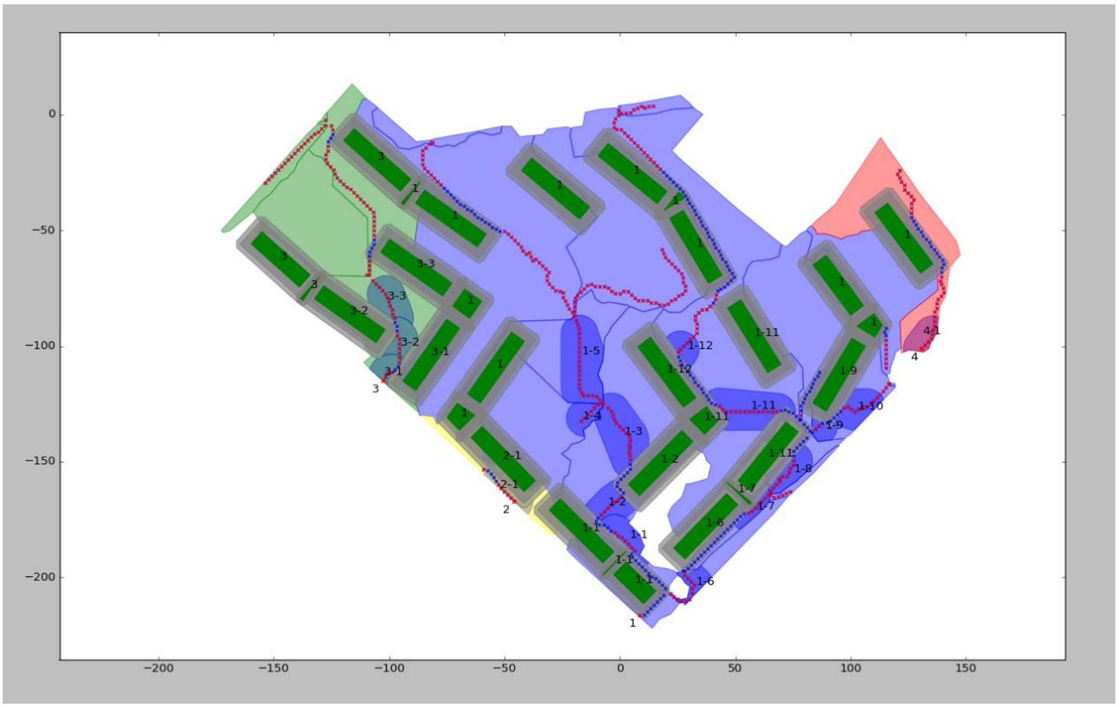
3


0
−50
−100
−150
−200
−200
−150
−100
−50
0
50
100
150

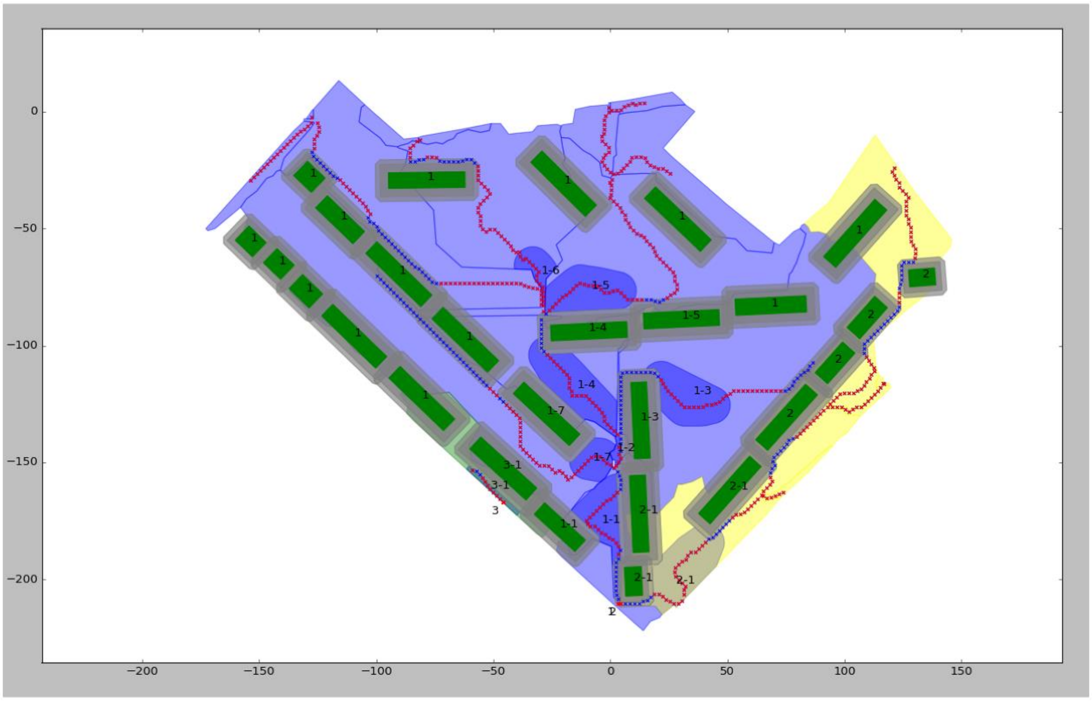
4


50
0
−50
−100
−150
−200
−250
−200
−100
0
100
200

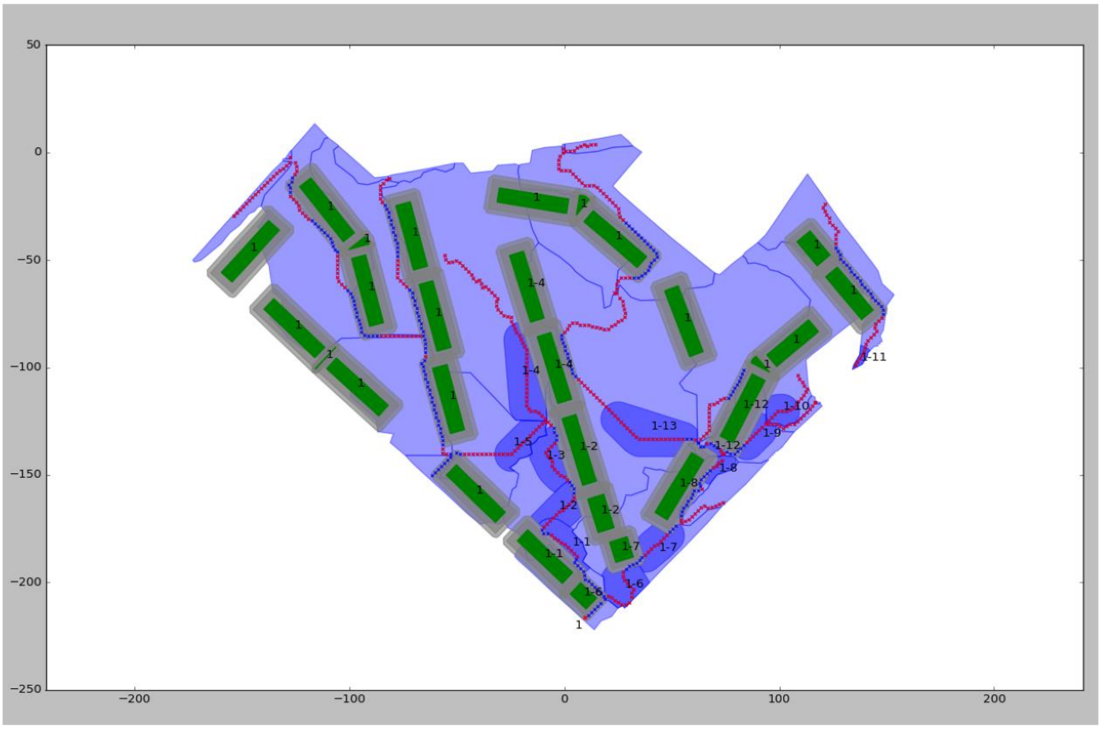
5



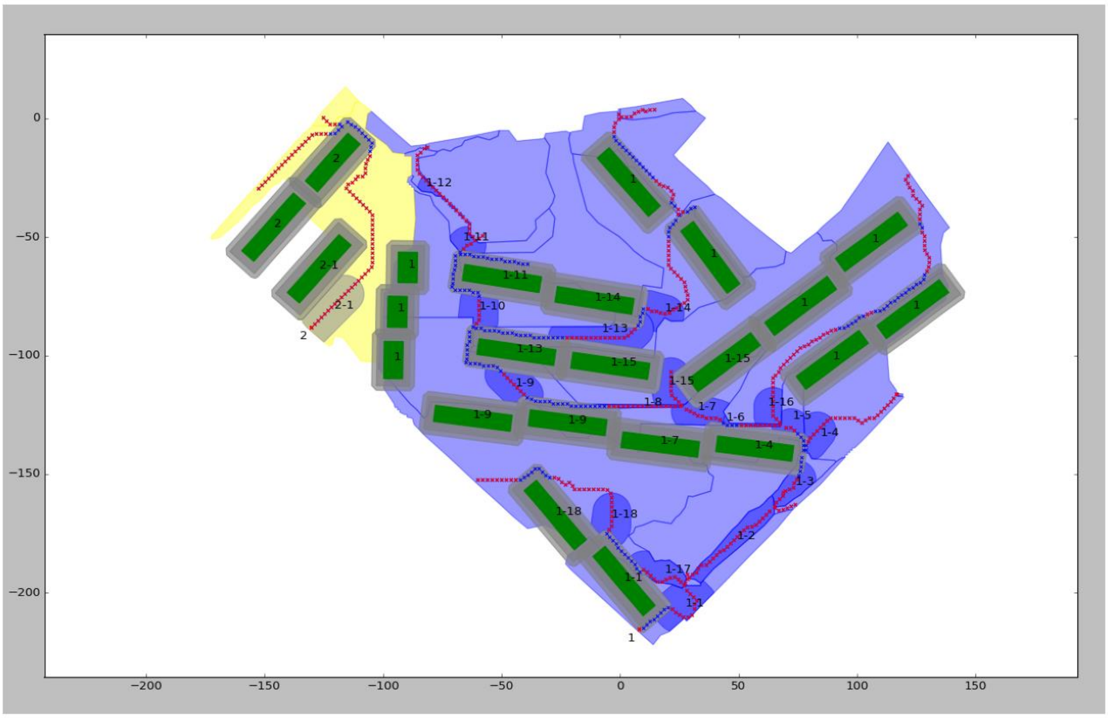
6



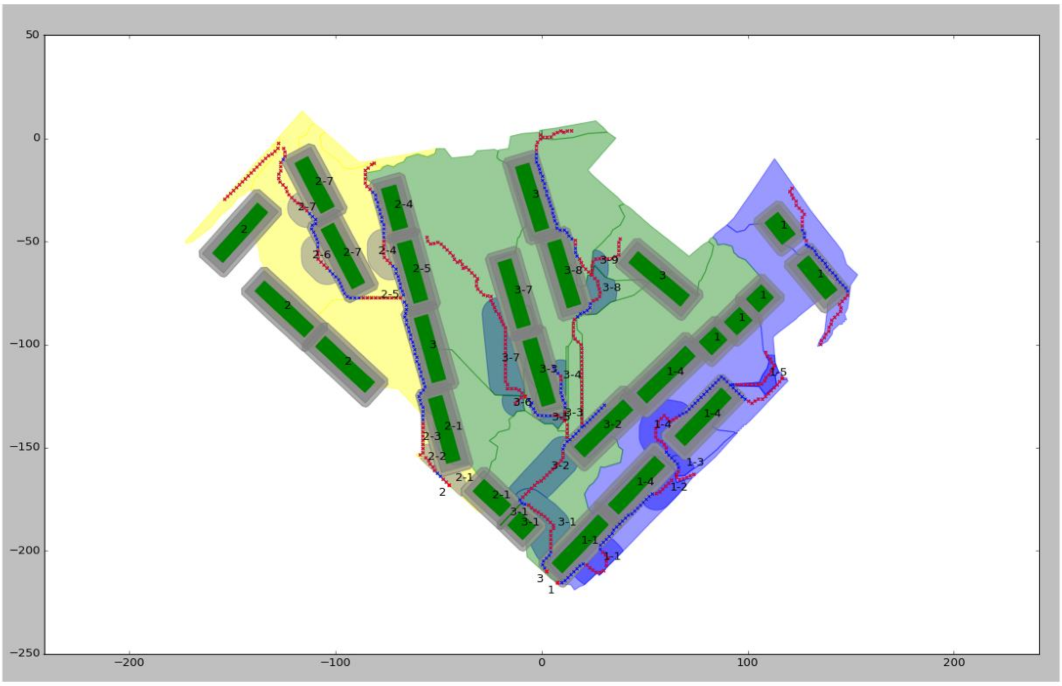
7



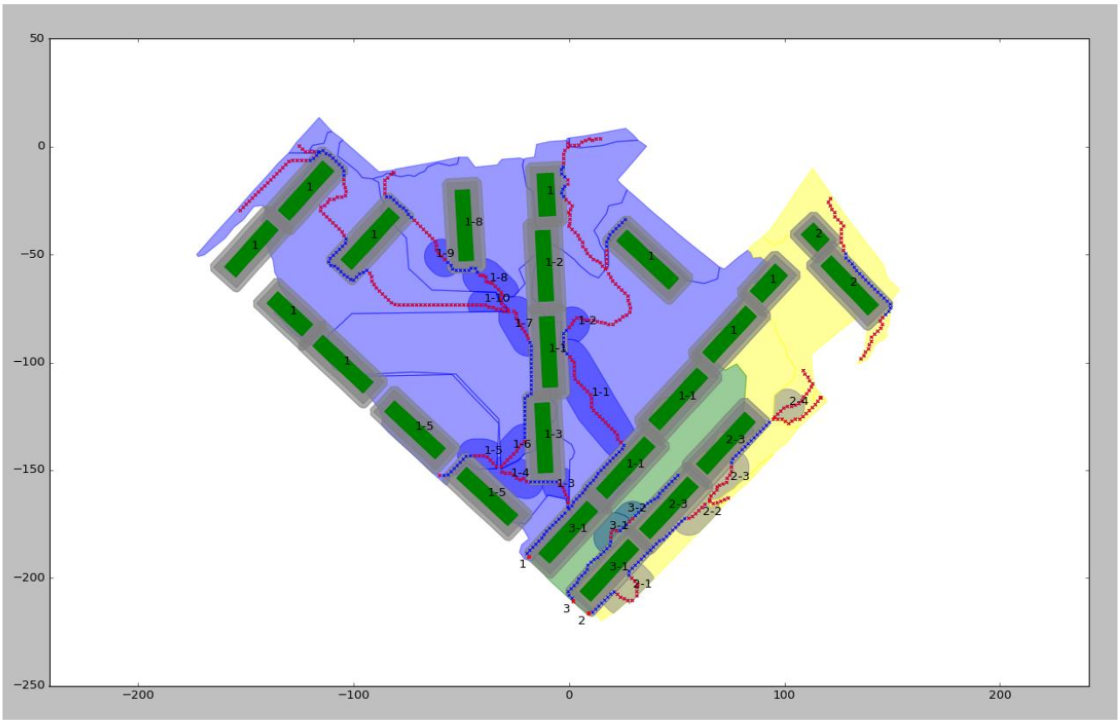
8



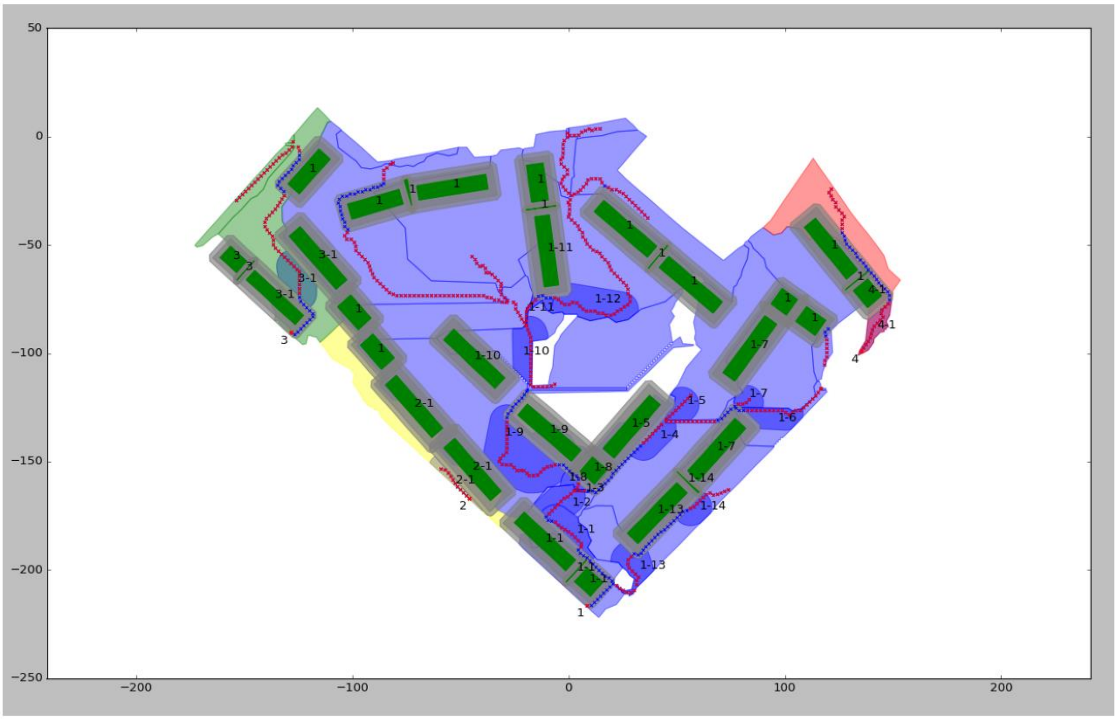
9



10



11



NTNU
Norwegian University of
Science and Technology