Maximum Coverage Problems

Theory and Application in Optimal Sensor Selection

Niels Holtgrefe

Bachelor Thesis 2021

TU Delft

Maximum Coverage Problems

Theory and Application in Optimal Sensor Selection

by

Niels Holtgrefe

to obtain the degree of Bachelor of Science in Applied Mathematics
at the Delft University of Technology,
to be defended on Wednesday July 14, 2021 at 15:00.

Student number:	4954181	
Project duration:	April 19, 2021 – July 8, 2021	
Thesis committee:	Dr. ir. T.M.L. Janssen,	TU Delft, supervisor
	Dr. ir. L.E. Meester,	TU Delft
	Dr. J.W. van der Woude,	TU Delft

Maximale Dekkingsproblemen

Theorie en Toepassing in Optimale Sensor Selectie

door

Niels Holtgrefe

ter verkrijging van de graad van Bachelor of Science in Technische Wiskunde
aan de Technische Universiteit Delft,
te verdedigen op woensdag 14 juli 2021 om 15:00 uur.

Student nummer:	4954181	
Project duur:	19 april 2021 – 8 juli 2021	
Beoordelingscommissie:	Dr. ir. T.M.L. Janssen,	TU Delft, begeleider
	Dr. ir. L.E. Meester,	TU Delft
	Dr. J.W. van der Woude,	TU Delft

Acknowledgements

My supervisor, Teun Janssen, was of great help during the duration of this bachelor thesis. The discussions we had in our weekly meetings were very helpful and fun. Although most of our meetings took place online, I was lucky to also meet him in person, which certainly was no guarantee during the Covid-19 pandemic. I especially want to thank him for giving me the freedom to investigate the topics I found interesting, while at the same time making sure the project remained manageable.

Further gratitude goes out towards the other members of my thesis committee: Ludolf Meester and Jacob van der Woude. Thank you for attending my thesis defence and assessing this report.

Jeroen Spandaw and Aniek Temming, the two teachers of the *bachelorcolloquium*, are also thanked. Their course improved my presentation skills and introduced me to the pitfalls of writing a thesis.

I would like to thank the KNMI for making their climate data publicly available and providing me with the geographical coordinates of the Dutch weather stations, upon request. These were extremely useful for the creation of some of the figures.

I owe thanks to my younger brother Tim, for proofreading parts of my thesis and reviewing the many adjustments I made on the figures.

Finally, I would like to thank my parents, other family members and friends. Their emotional support was appreciated even more during the Covid-19 pandemic.

Niels Holtgrefe
The Hague, July 2021

Abstract

In this thesis we analyse the class of *maximum coverage* problems. For all discussed problems, linear programs are formulated. Using the notion of *submodularity*, we prove that for the weighted version of the basic Maximum Coverage problem, where the weights differ per set, a polynomial-time greedy algorithm guarantees a $(1-\frac{1}{e})$-approximation of the optimal solution. This improves the already known bound of $(1-\frac{1}{e}-\varepsilon)$, for all $\varepsilon > 0$. We then show that the same result holds true, if we allow elements to be covered by multiple sets. Furthermore, a completely novel extension is introduced, where weights differ per combination of sets. It is proved that, under the assumption that the weights are submodular and increasing, a greedy algorithm still provides a $(1-\frac{1}{e})$-approximation.

The latter algorithm is tested in the framework of optimal sensor selection. To this end, we consider all official weather stations in the Netherlands as our sensor group. We test the performance of the approximation algorithm, if some of the assumptions do not hold and no theoretical bounds exists. Corresponding weights are calculated, using two approaches: *inverse distance weighting* and *multiple linear regression*. For both approaches the in practice performance of the greedy algorithm is shown to be even higher than $(1-\frac{1}{e})$, even though not all assumptions hold. Finally, the corresponding selection of weather stations is shown.

Lay summary

In this thesis we investigate a class of mathematical problems: the *maximum coverage* problems. These problems are extremely time-consuming to solve, thus we develop approximation algorithms. Such algorithms run faster, with the drawback that the given solution is not optimal. These algorithms are then tested on one of the main uses of the *maximum coverage* problems: modelling sensor selection. We try to find the best placement of weather stations in the Netherlands. Weather stations are essentially sensors and thus fit this thesis perfectly.

This short summary is written towards the general public that is interested in the topic of this thesis. The use of mathematical terms is avoided and not much mathematical knowledge is assumed.

Contents

List of abbreviations

BMC	Budgeted Maximum Coverage
GMC	Generalized Maximum Coverage
IDW	Inverse Distance Weighting
ILP	Integer Linear Program
KNMI	Royal Dutch Meteorological Institute
LP	Linear Program
MC	Maximum Coverage
MLR	Multiple Linear Regression
PDW-MMC	Power Set Dependent Weighted Maximum Multi-Coverage
PDW-MMC*	Power Set Dependent Weighted Maximum Multi-Coverage with restricted weight function
SDW-MC	Set Dependent Weighted Maximum Coverage
SDW-MMC	Set Dependent Weighted Maximum Multi-Coverage
SSR	Sum of Squared Residuals
SST	Total Sum of Squares
WMC	Weighted Maximum Coverage

Chapter 1

Introduction

1.1. Importance of sensors

Our society is slowly becoming more and more data-driven. *Data analysis* and *Big Data* are booming terms and data scientists are highly sought-after. From the social sciences to applied physics, data collection plays an important role in research. While in the humanities most data is collected by means of surveys and interviews, in applied sciences, sensors play a vital role. The Cambridge Dictionary [15] defines a sensor as "a device that is used to record that something is present or that there are changes in something". Whether discussing radio telescopes, traffic sensors or carbon dioxide meters, we are thus essentially talking about sensors.

Most sensors are placed on a subset of all points of interest. It is impossible to equip a home with hundreds of smoke alarms, thus only a few are used to estimate the data in the whole house. Here, an important question arises: "What is the optimal placement of sensors?" In this thesis, we will focus on a class of mathematical optimization problems, used to solve such questions: the *maximum coverage* problems.

1.2. Research objective

The basic Maximum Coverage (MC) problem considers the choosing of c sets of elements, with largest possible union. In a more formal manner:

Maximum Coverage
Instance: $E = \{e_1, \dots, e_n\}$, a set of n elements
$\mathcal{S} = \{S_1, \dots, S_m\}$, a collection of m non-empty subsets of E
c, an integer such that $1 \leq c \leq m$
Objective: Choose a collection of c sets $\mathcal{G} \subseteq \mathcal{S}$, such that the cardinality of the union of the chosen sets is maximal.

We obtain a minimal version of the sensor selection problem, by letting each element e_j correspond to a point of interest and each set S_i represent a possible sensor location. The elements of those sets are then the locations that the sensor would cover. As we maximize the size of the union of chosen sets, we essentially maximize the number of covered locations.

More advanced problems exist, where weights are introduced, corresponding to the quality of information a sensor provides. For example, we could relate the weights to the distance between the sensor and a location. The problem introduced in [8] can be used for such purposes. However, there it is assumed that each point of interest can be covered by at most one sensor. This is quite a restrictive assumption, as it makes a lot more sense to assume that multiple sensors provide us with more information about a point. To this end, we aim to develop a new problem, which allows for points to be covered by multiple sensors. Simultaneously, we explore the possibility of creating separate weights for each combination of sensors that cover a location. We also try to set up a linear program and come up with a polynomial approximation algorithm.

1.3. Thesis outline

The main body of this thesis can roughly be divided into two parts with a different focus, conveniently corresponding to Chapter 2 and 3.

Chapter 2 solely focuses on the mathematical theory behind the maximum coverage problems. Each section in that chapter covers a specific problem, by providing a standalone analysis. The problems are defined and linear programs are formulated. We also provide approximation algorithms and analyse their performance and time complexities. The four problems discussed are all nested, meaning that each is an extension of the previous one. Figure 1.1 provides a visualization of the structure of all discussed models and their relation to some other mentioned problems in this thesis. Whenever the reader feels lost, this figure together with the list of acronyms on page viii, should help to get back on track.

In Chapter 3 the focus is shifted more towards the application of optimal sensor selection. The newly introduced problem from section 2.4 is used to study the selection of weather stations in the Netherlands. Results are discussed and analysed.

This thesis ends with a conclusion and outlook into areas of further research in Chapter 4.

1.4. Notational conventions

We finish this introductory chapter with a short section devoted to the used mathematical notation, as this thesis heavily relies on set-theoretic notions, such as elements and sets, all the way up to sets of sets of sets. To distinguish between these mathematical levels of abstraction and maintain clarity, we will typeset these objects differently. The following conventions will be used throughout this thesis.

- Standard mathematical objects contained in a set, such as elements or indices, are typeset in a lowercase italic font, for example e_i or j.
- Sets of elements are denoted in uppercase italic, such as $E = \{e_1, \dots, e_n\}$.
- Sets of sets will be written down in a calligraphic font, for instance $\mathcal{S} = \{S_1, \dots, S_m\}$ or the powerset of a set $\mathcal{P}(E) = \{E' \subseteq E\}$.
- Sets of sets of sets will be indicated by the use of bold calligraphic letters. An example is the powerset of a set of sets $\boldsymbol{\mathcal{P}}(\mathcal{S})$.

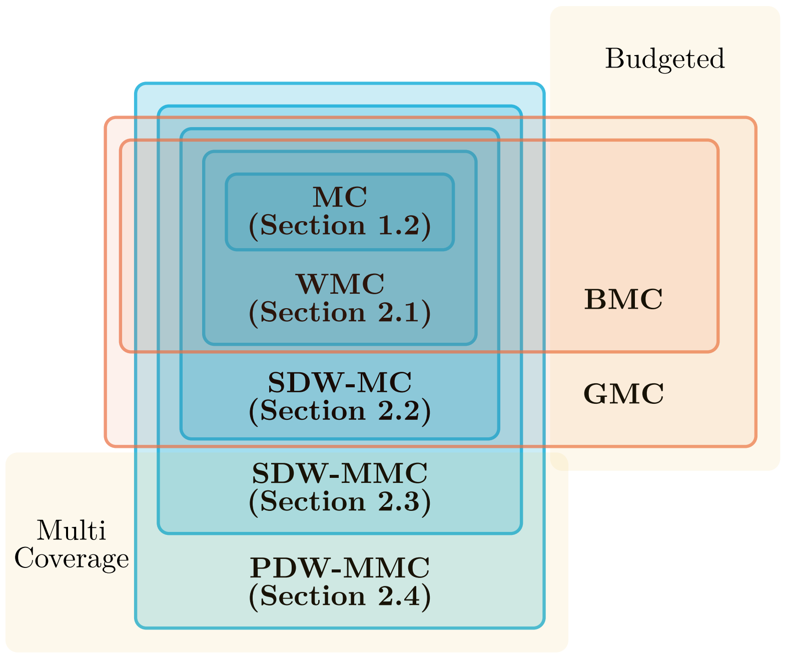


Figure 1.1: Structure of maximum coverage problems. It is also indicated whether a problem is budgeted or whether it allows for elements to be covered by multiple sets. The section which covers the problem is written in parentheses.

Sometimes, we refer to a set of sets as a collection or family of sets. But if ambiguity arises, the font of the object will indicate the mathematical abstraction level at which we operate. Finally, let us note that when a specific optimization problem is discussed, it will be typeset in a small capital font, such as Budgeted Maximum Coverage.

Chapter 2

Maximum Coverage problems

In this chapter we focus on a thorough analysis of different types of maximum coverage problems. Sections 2.1 and 2.2 cover two maximum coverage problems for which existing results are discussed and new results are derived. Section 2.3 contains a derivation of a recently developed extension of this class of problems. A completely new extension, not found in the literature, is introduced and analysed in section 2.4.

2.1. Weighted Maximum Coverage

Weighted Maximum Coverage (WMC) is an extension of the Maximum Coverage problem introduced in section 1.2, where the elements are assigned weights. In subsection 2.1.1 we formally define this problem and set up an integer linear program. A greedy approximation algorithm is discussed in subsection 2.1.2.

2.1.1. Problem description and linear program

WMC is an optimization problem where we have a set of weighted elements and a collection of sets containing these elements. The aim of the problem is to choose a predetermined number of these sets, such that the sum of the weights of all covered elements is maximal. More formally:

Weighted Maximum Coverage

Instance: $E = \{e_1, \ldots, e_n\}$, a set of n elements

$\mathcal{S} = \{S_1, \ldots, S_m\}$, a collection of m non-empty subsets of E

c, an integer such that $1 \leq c \leq m$

$w : E \to \mathbb{R}_{\geq 0}$, a non-negative weight function

Objective: Choose a collection of c sets $\mathcal{G} \subseteq \mathcal{S}$, such that the sum of the weights of the covered elements is maximized. Here, an element is covered if it is in at least one chosen set.

Vohra and Hall [18] introduced WMC as the *maximal covering location problem*, used to model the problem of location selection of facilities, such as fire stations. Hochbaum and Pathria [11] have tried to generalize the use of the problem and analysed it even further. A well-known extension of WMC is Budgeted Maximum Coverage (BMC), discussed in

[13]. This problem introduces a budget constraint and costs for each set. It is trivial to see, that when the budget is set to c and the cost per set to one, BMC reduces to WMC. Figure 1.1 illustrates this nicely. However, in this thesis the focus will lie on the unbudgeted versions of the maximum coverage problems.

A standard method to solve mathematical optimization problems, is by setting up a linear program (LP). This approach only works, if the objective and the constraints can be formulated as linear functions. Shortly, we will see that this is indeed the case. Fast algorithms to solve general LP's exist, such as the *simplexmethod*, explaining the wide use of linear programs. If the decision variables of a linear program are bound to integer values, the LP is often called an integer linear program (ILP). Should the reader not be familiar with linear programs, [1] provides an excellent introduction to this topic. However, we will not be diving too deep into this subject, so the math should not be too hard to follow.

As a start, we first introduce some notation, that will be used in the linear programs in this thesis. Let $I = \{1, \dots, m\}$ be the index set corresponding to the collection of subsets $\mathcal{S}$ and let $J = \{1, \dots, n\}$ be the index set corresponding to the set of elements E. We also define the *indicator function* z_{ij}, which shows whether element e_j is contained in set S_i. That is,

$$z_{ij} = \begin{cases} 1 & \text{if } e_j \in S_i\,, \\ 0 & \text{if } e_j \notin S_i\,. \end{cases} \tag{2.1}$$

We continue by giving an ILP-formulation for WMC. The two decision variables that will be used, are

$$x_i = \begin{cases} 1 & \text{if } S_i \text{ is chosen}\,, \\ 0 & \text{if } S_i \text{ is not chosen}\,, \end{cases} \tag{2.2}$$

$$y_j = \begin{cases} 1 & \text{if } e_j \text{ is covered}, \\ 0 & \text{if } e_j \text{ is not covered}. \end{cases} \tag{2.3}$$

Thus, for each set S_i we have a binary decision variable available, indicating whether the set is chosen or not. Each element e_j also has a decision variable showing whether the element is covered. Equation (2.4) shows the complete ILP-formulation of WMC.

$$\begin{aligned}
\text{maximize} \quad & \sum_{j \in J} y_j \cdot w(e_j), && && \text{(2.4a)}\\
\text{subject to} \quad & \sum_{i \in I} x_i \quad \leq c, && && \text{(2.4b)}\\
& \sum_{i \in I} x_i \cdot z_{ij} \geq y_j, && j \in J, && \text{(2.4c)}\\
& x_i \quad \in \{0,1\}, && i \in I, && \text{(2.4d)}\\
& y_j \quad \in \{0,1\}, && j \in J. && \text{(2.4e)}
\end{aligned}$$

The objective function (2.4a) is quite intuitive. We maximize the weights of all covered elements, as the weights are only counted if an element is covered, implying that $y_j = 1$. Constraint (2.4b) allows for at most c sets to be chosen. Constraint (2.4c) makes sure that an element can only be covered if at least one of the sets that contains the element is chosen. In such a case the left-hand side of the sum is larger or equal than 1 and this allows y_j to also become 1. Finally, constraints (2.4d) and (2.4e) are the integrality constraints of the decision variables.

WMC is known to be NP-hard [18]. Thus, if P = NP, no algorithm can solve WMC in polynomial time. Although algorithms to solve LP's are very fast, this means that it becomes very time-consuming to optimally solve the problem for large instances. This explains the need for an approximation algorithm that runs in polynomial time.

2.1.2. Greedy approximation algorithm

A logical candidate for an explicit algorithm to solve WMC is a *greedy algorithm*, which greedily selects the next set to be added. Algorithm 1 shows the corresponding pseudo code.

Algorithm 1: Greedy approximation algorithm for WMC

```
   Input: an instance of WMC
   Output: a collection $\mathcal{G}$ of sets $S_i$
1  $\mathcal{G} \leftarrow \emptyset$
2  $U \leftarrow \emptyset$
3  while $|\mathcal{G}| < c$ do
4      $\hat{S} \leftarrow \underset{S_i \in \mathcal{S} \setminus \mathcal{G}}{\operatorname{argmax}} \left[\sum_{e_j \in E \setminus U} w(e_j)\right]$
5      $\mathcal{G} \leftarrow \mathcal{G} \cup \{\hat{S}\}$
6      $U \leftarrow U \cup \hat{S}$
7  return $\mathcal{G}$
```

At each stage we add the set S_i that adds the most weight to the intermediate solution, until we have chosen c sets. This is done on line 4. For each S_i that is still available, we check how much weight the set would add. To this end, we keep track of the elements that are already covered and are not to be considered any more, via the set U. Assuming the weights are known, line 4 runs in $O(n \cdot m)$ time. The algorithm runs $c \leq m$ times through the while-loop, resulting in the algorithm having a polynomial time complexity of $O(c \cdot n \cdot m) = O(n \cdot m^2)$.

It can easily be shown that Algorithm 1 does not necessarily find the optimal solution. Consider the set of elements $E = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ with unit weights, in other words, $w(e_j) = 1$ for all $e_j \in E$. Let the corresponding collection of sets be $\mathcal{S} = \{S_1, S_2, S_3\}$ with $S_1 = \{1, 2, 3, 4, 5\}$, $S_2 = \{1, 2, 6, 7\}$ and $S_3 = \{3, 4, 8, 9\}$. If we then set $c = 2$, Algorithm 1 will first choose S_1, as it has the largest total weight. In the second iteration of the algorithm, either S_2 or S_3 is chosen, because both sets improve the total weight by 2, resulting in a final selection with a weight of 7. However, if we choose sets S_2 and S_3, we get a total weight of 8. Thus, Algorithm 1 indeed did not find the optimal solution.

A very useful result from [11] shows that the greedy approach in Algorithm 1 guarantees to find a $(1 - \frac{1}{e})$-approximation[1] of any non-trivial instance of WMC. This bound is tight, indicating that examples can be constructed where exactly this ratio is achieved. An approximation ratio of $(1 - \frac{1}{e})$ means that the total weight of the greedy solution is at least within a factor of $(1 - \frac{1}{e})$ of the optimal solution. Thus a certain degree of suboptimality is always achieved by Algorithm 1. Furthermore, Feige [9] proved in 1998 that an approximation ratio better than $(1 - \frac{1}{e})$ can not be achieved in polynomial time for Maximum Coverage, unless P $\neq$ NP. This result then certainly holds for WMC, as it has Maximum Coverage as a

[1] $1 - \frac{1}{e} \approx 0.632$

special case. Therefore, Algorithm 2 is the best possible polynomial approximation algorithm for WMC, in terms of approximation ratio.

2.2. Set Dependent Weighted Maximum Coverage

Set Dependent Weighted Maximum Coverage (SDW-MC) is a very natural extension of WMC. In subsection 2.2.1 the problem itself is discussed and two versions of a linear program are introduced. In subsection 2.2.2 we turn our attention to an approximation algorithm, for which we analyse its performance guarantee in subsection 2.2.3.

2.2.1. Problem description and linear program

In SDW-MC the objective remains unchanged, in comparison with WMC. However, as the name suggests, the weights differ per set S_i. Thus each element has its own weight function with the set $\mathcal{S}$ as its domain. This means that we need to choose by which set an element is covered, with the constraint that an element can be covered by at most one set. Formally the problem is defined as follows:

Set Dependent Weighted Maximum Coverage

Instance: $E = \{e_1, \dots, e_n\}$, a set of n elements
$\mathcal{S} = \{S_1, \dots, S_m\}$, a collection of m non-empty subsets of E
c, an integer such that $1 \leq c \leq m$
$w_j : \mathcal{S} \to \mathbb{R}_{\geq 0}$, a non-negative weight function per element e_j

Objective: Choose a collection of c sets $\mathcal{G} \subseteq \mathcal{S}$, such that the sum of the weights of the covered elements is maximized. Here, an element can be covered by at most one of the chosen sets and only the corresponding weight is counted. An element can only be covered by a set, if it is contained in that set.

In the literature, no thorough analysis on SDW-MC can be found. However, Cohen and Katzir [8] have intensively researched the Generalized Maximum Coverage (GMC) problem, which is the budgeted version of SDW-MC. In GMC each set and element have a cost. On top of that, a certain budget is available, which bounds the possible selections. By setting the budget equal to c, the cost per set equal to 1 and giving the elements no cost at all, SDW-MC is retrieved. This suggests that SDW-MC is a special case of GMC, as illustrated by Figure 1.1. Therefore, all results derived in [8] should also hold for SDW-MC. It is worth noting that in [8] the problem is formulated in a different, but equivalent way.

Just as in section 2.1, it makes sense to first formulate a linear program of SDW-MC. For WMC we had to specify whether a set was chosen and whether an element was covered. While for SDW-MC we also need to specify by which set an element is covered, as it is important to determine how much weight an element contributes. Therefore, the decision variable y_j in equation (2.3) is subject to a small change, resulting in the following decision variables:

$$x_i = \begin{cases} 1 & \text{if } S_i \text{ is chosen,} \\ 0 & \text{if } S_i \text{ is not chosen,} \end{cases} \tag{2.5}$$

$$y_{ij} = \begin{cases} 1 & \text{if } S_i \text{ covers } e_j\,, \\ 0 & \text{if } S_i \text{ does not cover } e_j\,. \end{cases} \tag{2.6}$$

The corresponding ILP is now depicted in equation (2.7).

$$\begin{aligned}
\text{maximize} \quad & \sum_{i\in I}\sum_{j\in J} y_{ij}\cdot w_j(S_i), && && \text{(2.7a)}\\
\text{subject to} \quad & \sum_{i\in I} x_i && \le c, && && \text{(2.7b)}\\
& \sum_{i\in I} y_{ij} && \le 1, && j\in J, && \text{(2.7c)}\\
& x_i\cdot z_{ij} && \ge y_{ij}, && i\in I, j\in J, && \text{(2.7d)}\\
& x_i && \in\{0,1\}, && i\in I, && \text{(2.7e)}\\
& y_{ij} && \in\{0,1\}, && i\in I, j\in J. && \text{(2.7f)}
\end{aligned}$$

Objective function (2.7a) sums over all possible combinations of sets and elements and only counts the weights when a set covers an element, namely when $y_{ij} = 1$. Constraint (2.7b) still makes sure that at most c sets are chosen, while (2.7c) allows each element to be covered by at most one set S_i. Finally, constraint (2.7d) lets y_{ij} only take the value 1, which it naturally seeks due to the objective of maximization, when both x_i and z_{ij} are equal to 1. Thus an element can only be covered by a set, if the set is chosen and the element is contained in this set. Therefore, for a given element, the weights of all sets that do not contain the element can be arbitrary, as they will never contribute to the objective function. This allows us to make the non-restrictive assumption that $w_j(S_i) = 0$, if $e_j \notin S_i$. With this assumption we can relax constraint (2.7d). It is not necessary anymore for an element to only be covered by a set, if the set contains the element. This is due to the fact that the weight of 0 will now make sure that such a combination of an element and a set will not contribute to the objective function. We now get an equivalent ILP-formulation of SDW-MC in equation (2.8), where constraint (2.8d) differs from (2.7d).

$$\begin{aligned}
\text{maximize} \quad & \sum_{i\in I}\sum_{j\in J} y_{ij}\cdot w_j(S_i), && && \text{(2.8a)}\\
\text{subject to} \quad & \sum_{i\in I} x_i && \le c, && && \text{(2.8b)}\\
& \sum_{i\in I} y_{ij} && \le 1, && j\in J, && \text{(2.8c)}\\
& x_i && \ge y_{ij}, && i\in I, j\in J, && \text{(2.8d)}\\
& x_i && \in\{0,1\}, && i\in I, && \text{(2.8e)}\\
& y_{ij} && \in\{0,1\}, && i\in I, j\in J. && \text{(2.8f)}
\end{aligned}$$

2.2.2. Greedy approximation algorithm

In subsection 2.1.1 it was stated that WMC is NP-hard. As WMC is a special case of SDW-MC, where the weights across different sets are equal, SDW-MC must certainly also be NP-hard. Luckily, Cohen and Katzir [8] came up with a greedy algorithm for GMC, that ensures an approximation ratio of $(1 - \frac{1}{e} - \varepsilon)$, for all $\varepsilon > 0$. However, for the special unit-cost case that is SDW-MC, we will prove in subsection 2.2.3 that we can increase this ratio to

$(1-\frac{1}{e})$, just as for WMC, by using a somewhat different method. Again, this is the best possible bound, as shown in [9]. The idea behind the algorithm is similar to that of Algorithm 1. At each stage we add the set S_i, that gives us the most additional weight. To simplify the notation of the algorithm we will introduce the *truncate-function*.

Definition 2.1. Let x be a real number, then τ is the *truncate-function* that truncates negative values to 0. That is,

$$\tau(x) = \max(x, 0)\,.$$

Using this notation, we now set up the greedy approximation in Algorithm 2.

Algorithm 2: Greedy approximation algorithm for SDW-MC

```
    Input: an instance of SDW-MC
    Output: a collection $\mathcal{G}$ of sets $S_i$
1   $\mathcal{G} \leftarrow \emptyset$
2   while $|\mathcal{G}| < c$ do
3       $\hat{S} \leftarrow \underset{S_i \in \mathcal{S}\setminus\mathcal{G}}{\arg\max} \left[\sum_{j\in J} \tau\left(w_j(S_i) - \max_{S_k \in \mathcal{G}} w_j(S_k)\right)\right]$
4       $\mathcal{G} \leftarrow \mathcal{G} \cup \{\hat{S}\}$
5   return $\mathcal{G}$
```

In this algorithm, we add the c sets one by one, each time choosing the set $\hat{S}$ for which the total increase of weight is the highest. The increase of weight of a set S_i for an element e_j, is the difference between the weight $w_j(S_i)$ and the current maximum of all possible weights. The truncate-function makes sure that if a certain set S_i has a lower weight for an element than the current maximum, the increase is set to 0 and does not become negative. It takes $O(n \cdot m^2)$ time to run over line 3, resulting in a total time complexity of $O(n \cdot m^3)$ for Algorithm 2.

2.2.3. Performance guarantee analysis

For the analysis of the algorithm we rely on a famous article by Nemhauser, Wolsey, and Fisher [14]. As a first step, we define the *total weight* of a selection. This is the largest possible weight of a selection, while still adhering to the conditions of the SDW-MC problem. In other words, for each element e_j we choose the largest weight for the element of all chosen sets.

Definition 2.2. Let $w_j : \mathcal{S} \to \mathbb{R}_{\geq 0}$ be a weight function of an instance of SDW-MC, then the *total weight* $w : \boldsymbol{\mathcal{P}}(\mathcal{S}) \to \mathbb{R}_{\geq 0}$ is

$$w(\mathcal{A}) = \sum_{j\in J}\left(\max_{S_i\in\mathcal{A}} w_j(S_i)\right)\,, \quad \forall \mathcal{A} \in \boldsymbol{\mathcal{P}}(\mathcal{S})\,,$$

with $w(\emptyset) = 0$.

Using this notation, we can essentially write SDW-MC as the problem of finding $\mathcal{G}_{\mathsf{opt}}$, where

$$\mathcal{G}_{\mathsf{opt}} = \underset{\mathcal{A}\subseteq\mathcal{S}}{\arg\max}\{w(\mathcal{A}) : |\mathcal{A}| \leq c\}\,. \tag{2.9}$$

We aim to find a collection of sets $\mathcal{A}$ of size at most c, while maximizing the total weight for this collection.

To derive the approximation ratio of Algorithm 2, we first state what the *residual value* of a set function is.

Definition 2.3. Let $F = \{e_1, \ldots, e_n\}$ be a finite set and let $f : \mathcal{P}(F) \to \mathbb{R}$ be a set function defined on the power set of F, then the *residual value* or *incremental value* of adding e_i to A, is

$$\rho_i(A) = f(A \cup \{e_i\}) - f(A), \quad \forall e_i \in F, \forall A \in \mathcal{P}(F).$$

Thus $\rho_i(A)$ represents the increase in total value, by adding e_i to A.

Furthermore, we introduce what an *increasing* and *submodular* set function is.

Definition 2.4. Let F be a finite set and let $f : \mathcal{P}(F) \to \mathbb{R}$ be a set function defined on the power set of F, then f is *increasing* if

$$f(A) \leq f(B), \quad \forall A, B \in \mathcal{P}(F), \text{ such that } A \subseteq B.$$

Remark. If $-f$ is increasing, then f is *decreasing.*

Definition 2.5. Let F be a finite set and let $f : \mathcal{P}(F) \to \mathbb{R}$ be a set function defined on the power set of F, then f is *submodular* if

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B), \quad \forall A, B \in \mathcal{P}(F).$$

Remark. If $-f$ is submodular, then f is *supermodular.*

An equivalent definition of submodularity, that makes use of the residual value, is given in Proposition 2.6 and was proved by Nemhauser, Wolsey, and Fisher [14].

Proposition 2.6 (Nemhauser, Wolsey & Fisher, 1978)**.** Let F be a finite set and let $f : \mathcal{P}(F) \to \mathbb{R}$ be a set function defined on the power set of F, then the two following statements are equivalent and both define submodular set functions.

(i) $$f(A) + f(B) \geq f(A \cup B) + f(A \cap B), \quad \forall A, B \in \mathcal{P}(F),$$

(ii) $$\rho_i(A) \geq \rho_i(B), \quad \forall A, B \in \mathcal{P}(F), \text{ such that } A \subseteq B \text{ and } \forall e_i \in F \setminus B.$$

We see that submodularity captures the notion of *diminishing (marginal) returns*, saying that the incremental value of a function decreases for larger sets [12].

The strategy to get our approximation ratio now is as follows. First we will prove, with help of a lemma, that the total weight function is submodular. Afterwards, we will show that the total weight function is increasing. Then, we show that Algorithm 2 is based on the residual weight of the total weight function. Finally, we show that all these results together, combined with a theorem from [14], gives us our bound.

The first lemma we present, shows that the maximum is a submodular function.

Lemma 2.7. If A and B are subsets of a finite set F and $f : F \to \mathbb{R}$ is a function defined on their elements, then

$$\max_{x \in A} f(x) + \max_{x \in B} f(x) \geq \max_{x \in A \cup B} f(x) + \max_{x \in A \cap B} f(x).$$

Proof. Let A and B be arbitrary subsets of an arbitrary finite set F and let f be an arbitrary function defined on its elements. Denote

$$x_1 = \arg\max_{x \in A} f(x), \qquad x_2 = \arg\max_{x \in B} f(x),$$

$$x_3 = \arg\max_{x \in A \cup B} f(x)\,, \qquad x_4 = \arg\max_{x \in A \cap B} f(x)\,.$$

It can easily be seen that $x_3 \in A \cup B$ and thus $x_3 \in A$ or $x_3 \in B$ (or both). Without loss of generality, we will now assume that $x_3 \in A$. As x_3 gives the largest value of f of all $x \in A \cup B$, this implies immediately that $x_1 = x_3$. Also note that $f(x_4) \leq f(x_2)$, because $(A \cap B) \subseteq B$. Combining these two facts, we get $f(x_1) + f(x_2) = f(x_3) + f(x_2) \geq f(x_3) + f(x_4)$, which is equivalent to the desired result. □

With help of Lemma 2.7 we can now prove that the total weight function, as defined in Definition 2.2, is also submodular.

Theorem 2.8. If $w : \boldsymbol{\mathcal{P}}(\mathcal{S}) \to \mathbb{R}_{\geq 0}$ is a total weight function of an instance of SDW-MC, then w is a submodular function.

Proof. Let $\mathcal{A}$ and $\mathcal{B}$ be arbitrary subsets of $\mathcal{S}$. Then,

$$\begin{aligned}
w(\mathcal{A}) + w(\mathcal{B}) &\overset{\text{Def}}{\underset{2.2}{=}} \sum_{j \in J} \left(\max_{S_i \in \mathcal{A}} w_j(S_i) \right) + \sum_{j \in J} \left(\max_{S_i \in \mathcal{B}} w_j(S_i) \right) \\
&= \sum_{j \in J} \left(\max_{S_i \in \mathcal{A}} w_j(S_i) + \max_{S_i \in \mathcal{B}} w_j(S_i) \right) \\
&\overset{\text{Lem}}{\underset{2.7}{\geq}} \sum_{j \in J} \left(\max_{S_i \in \mathcal{A} \cup \mathcal{B}} w_j(S_i) + \max_{S_i \in \mathcal{A} \cap \mathcal{B}} w_j(S_i) \right) \\
&= \sum_{j \in J} \left(\max_{S_i \in \mathcal{A} \cup \mathcal{B}} w_j(S_i) \right) + \sum_{j \in J} \left(\max_{S_i \in \mathcal{A} \cap \mathcal{B}} w_j(S_i) \right) \\
&\overset{\text{Def}}{\underset{2.2}{=}} w(\mathcal{A} \cup \mathcal{B}) + w(\mathcal{A} \cap \mathcal{B})\,.
\end{aligned}$$

Thus, by Definition 2.5, w is a submodular function. □

The other property the total weight function should have, is that it is increasing. This will be shown in the following proposition.

Proposition 2.9. If $w : \boldsymbol{\mathcal{P}}(\mathcal{S}) \to \mathbb{R}_{\geq 0}$ is a total weight function of an instance of SDW-MC, then w is an increasing function.

Proof. Let $\mathcal{A}$ and $\mathcal{B}$ be arbitrary subsets of $\mathcal{S}$, such that $\mathcal{A} \subseteq \mathcal{B}$. Then,

$$w(\mathcal{A}) \overset{\text{Def}}{\underset{2.2}{=}} \sum_{j \in J} \left(\max_{S_i \in \mathcal{A}} w_j(S_i) \right) \leq \sum_{j \in J} \left(\max_{S_i \in \mathcal{B}} w_j(S_i) \right) \overset{\text{Def}}{\underset{2.2}{=}} w(\mathcal{B})\,.$$

Thus w is an increasing function by Definition 2.4. Here, we used the trivial fact that the maximum of a set is also increasing, as the maximum will never decrease when we add an element to a set. □

Theorem 2.10 now shows that the residual weight of the total weight function, as defined in Definition 2.3, takes on the form on which Algorithm 2 is based. This indicates that Algorithm 2 maximizes the residual weight at each iteration.

Theorem 2.10. Let $w : \boldsymbol{\mathcal{P}}(\mathcal{S}) \to \mathbb{R}_{\geq 0}$ be a total weight function of an instance of SDW-MC, then the residual weight of this function can be written as

$$\rho_i(\mathcal{A}) = \sum_{j \in J} \tau \left(w_j(S_i) - \max_{S_k \in \mathcal{A}} w_j(S_k) \right), \quad \forall S_i \in \mathcal{S}, \forall \mathcal{A} \in \boldsymbol{\mathcal{P}}(\mathcal{S}).$$

Proof. Let $S_i \in \mathcal{S}$ and $\mathcal{A} \in \boldsymbol{\mathcal{P}}(\mathcal{S})$ be arbitrary. Then,

$$\begin{aligned}
\rho_i(\mathcal{A}) &\overset{\text{Def}}{\underset{2.3}{=}} w(\mathcal{A} \cup \{S_i\}) - w(\mathcal{A}) \\
&\overset{\text{Def}}{\underset{2.2}{=}} \sum_{j \in J} \left(\max_{S_k \in \mathcal{A} \cup \{S_i\}} w_j(S_k) \right) - \sum_{j \in J} \left(\max_{S_k \in \mathcal{A}} w_j(S_k) \right) \\
&= \sum_{j \in J} \left(\max \left[w_j(S_i), \max_{S_k \in \mathcal{A}} w_j(S_k) \right] - \max_{S_k \in \mathcal{A}} w_j(S_k) \right) \\
&= \sum_{j \in J} \left(\max \left[w_j(S_i) - \max_{S_k \in \mathcal{A}} w_j(S_k), 0 \right] \right) \\
&\overset{\text{Def}}{\underset{2.1}{=}} \sum_{j \in J} \tau \left(w_j(S_i) - \max_{S_k \in \mathcal{A}} w_j(S_k) \right).
\end{aligned}$$

□

We will continue by stating a result derived by Nemhauser, Wolsey, and Fisher [14]. For the lengthy analysis that lies at the basis of the theorem, we refer to their article.

Theorem 2.11 (Nemhauser, Wolsey & Fisher, 1978)**.** Let F be a finite set and let $f : \mathcal{P}(F) \to \mathbb{R}_{\geq 0}$ be a non-constant, increasing, submodular set function with $f(\emptyset) = 0$. Consider the problem of finding

$$G_{\text{opt}} = \arg\max_{A \subseteq F} \{ f(A) : |A| \leq c \},$$

where $1 \leq c \leq |F|$. Let G be the solution of a greedy algorithm, where G is build up by adding elements one by one, at each stage adding the element of F with largest incremental value. Then,

$$f(G) \geq \left(1 - \left[\frac{c-1}{c} \right]^c \right) \cdot f(G_{\text{opt}}) > \left(1 - \frac{1}{e} \right) \cdot f(G_{\text{opt}}),$$

thus G is a $(1 - \frac{1}{e})$-approximation of G_{opt}.

With help of Proposition 2.9 and Theorems 2.8, 2.10 and 2.11, we can now deduct the performance guarantee of Algorithm 2.

Corollary 2.12. Algorithm 2 guarantees to find a $(1 - \frac{1}{e})$-approximation of any non-trivial instance of SDW-MC.

Proof. This follows directly from Theorem 2.11. As shown earlier in this section, we can write any instance of SDW-MC in the form of equation (2.9). In Theorem 2.8 and Proposition 2.9, we showed that the corresponding function w is increasing and submodular. By excluding trivial cases, w must be non-constant. Furthermore, w was defined to have value 0 for the empty set (Definition 2.2). In Theorem 2.10, we showed that for each iteration, Algorithm 2 indeed adds the set S_i with maximal incremental value. Thus all conditions of Theorem 2.11 are met and we get our approximation bound. $\square$

Interestingly, the minimization counterpart of Theorem 2.11 also holds. The proof of the following theorem also tells us how to rewrite such maximization problems to equivalent minimization problems. To avoid confusion, we would like to emphasize that the function f in the following theorem takes on only non-positive values.

Theorem 2.13. Let F be a finite set and let $f : \mathcal{P}(F) \to \mathbb{R}_{\leq 0}$ be a non-constant, decreasing, supermodular set function with $f(\emptyset) = 0$. Consider the problem of finding

$$G_{\mathsf{opt}} = \underset{A \subseteq F}{\arg\min}\{f(A) : |A| \leq c\}\,,$$

where $1 \leq c \leq |F|$. Let G be the solution of a greedy algorithm, where G is build up by adding elements one by one, at each stage adding the element of F with smallest incremental value. Then,

$$f(G) \leq \left(1 - \left[\frac{c-1}{c}\right]^c\right) \cdot f(G_{\mathsf{opt}}) < \left(1 - \frac{1}{e}\right) \cdot f(G_{\mathsf{opt}})\,,$$

thus G is a $(1 - \frac{1}{e})$-approximation of G_{opt}.

Proof. First let us define the function $g : \mathcal{P}(F) \to \mathbb{R}_{\geq 0}$, as

$$g(A) = -f(A), \quad \forall A \in \mathcal{P}(F)\,.$$

We can rewrite the selection we want to find as follows:

$$G_{\mathsf{opt}} = \underset{A \subseteq F}{\arg\min}\{f(A) : |A| \leq c\} = \underset{A \subseteq F}{\arg\max}\{-f(A) : |A| \leq c\} = \underset{A \subseteq F}{\arg\max}\{g(A) : |A| \leq c\}\,.$$

By Definitions 2.4 and 2.5, $g = -f$ is increasing and submodular. Trivially, g is non-constant, because f was non-constant. Furthermore, we also have that $g(\emptyset) = -f(\emptyset) = 0$. Finally, it can easily be seen that the element of F with smallest incremental value of function f, has the largest incremental value of function g. Thus we are now in the exact situation of Theorem 2.11, so

$$g(G) \geq \left(1 - \left[\frac{c-1}{c}\right]^c\right) \cdot g(G_{\mathsf{opt}}) > \left(1 - \frac{1}{e}\right) \cdot g(G_{\mathsf{opt}})\,.$$

Then by the definition of g,

$$f(G) \leq \left(1 - \left[\frac{c-1}{c}\right]^c\right) \cdot f(G_{\mathsf{opt}}) < \left(1 - \frac{1}{e}\right) \cdot f(G_{\mathsf{opt}})\,.$$

$\square$

2.3. Set Dependent Weighted Maximum Multi-Coverage

In this section we discuss a logical generalization of SDW-MC: Set Dependent Weighted Maximum Multi-Coverage (SDW-MMC), where elements can be covered by multiple sets. This section follows the same structure as section 2.2. In subsection 2.3.1 the problem is defined and a linear program is given. Subsection 2.3.2 discusses an approximation algorithm, which we analyse in subsection 2.3.3.

2.3.1. Problem description and linear program

SDW-MMC allows for an element to be covered by at most $b \leq c$ sets, as opposed to just one set. If this happens, the weights are added accordingly. This results in a slight alteration of the problem definition.

Set Dependent Weighted Maximum Multi-Coverage

Instance: $E = \{e_1, \ldots, e_n\}$, a set of n elements
$\mathcal{S} = \{S_1, \ldots, S_m\}$, a collection of m non-empty subsets of E
c, an integer such that $1 \leq c \leq m$
b, an integer such that $1 \leq b \leq c$
$w_j : \mathcal{S} \rightarrow \mathbb{R}_{\geq 0}$, a non-negative weight function per element e_j

Objective: Choose a collection of c sets $\mathcal{G} \subseteq \mathcal{S}$, such that the sum of the weights of the covered elements is maximized. Here, an element can be covered by at most b of the chosen sets and only the corresponding weights are counted. An element can only be covered by a set, if it is contained in that set.

It is immediately clear, that if we set $b = 1$, we have the exact same problem as SDW-MC, where an element can only be covered once. This implies that SDW-MMC also belongs to the class of NP-hard problems, if P $\neq$ NP.

For the ILP-formulation we need the exact same decision variables as for SDW-MC:

$$x_i = \begin{cases} 1 & \text{if } S_i \text{ is chosen,} \\ 0 & \text{if } S_i \text{ is not chosen,} \end{cases} \tag{2.10}$$

$$y_{ij} = \begin{cases} 1 & \text{if } S_i \text{ covers } e_j\,, \\ 0 & \text{if } S_i \text{ does not cover } e_j\,. \end{cases} \tag{2.11}$$

We will assume that the non-restrictive assumption that $w_j(S_i) = 0$, if $e_j \notin S_i$, still holds. Then, we can reuse the ILP in equation (2.8), by only changing constraint (2.8c), such that an element can be covered by at most b sets. This gives us the ILP in equation (2.12), which can be used to solve the problem optimally.

$$
\begin{aligned}
\text{maximize} \quad & \sum_{i\in I}\sum_{j\in J} y_{ij}\cdot w_j(S_i), && && (2.12a)\\
\text{subject to} \quad & \sum_{i\in I} x_i \le c, && && (2.12b)\\
& \sum_{i\in I} y_{ij} \le b, && j\in J, && (2.12c)\\
& x_i \ge y_{ij}, && i\in I, j\in J, && (2.12d)\\
& x_i \in \{0,1\}, && i\in I, && (2.12e)\\
& y_{ij} \in \{0,1\}, && i\in I, j\in J. && (2.12f)
\end{aligned}
$$

2.3.2. Greedy approximation algorithm

Only very recently, SDW-MMC and its unweighted version, the Maximum Multi Coverage problem, were explored by Barman, Fawzi, Ghoshal, *et al.* [5]. In this article, it was stated that a greedy algorithm designed similarly as Algorithm 2, should also achieve an approximation bound of $(1-\frac{1}{e})$ for Maximum Multi Coverage. However, this direction was not further explored. In the current and next subsection, this idea is worked out and it is proved that the same result holds for the weighted version SDW-MMC.

To simplify our algorithm, let us first introduce some notation for the k^{th}-largest element of a set.

Definition 2.14. Let A be a finite set of numbers, with k a positive integer such that $k \le |A|$, then we denote the k^{th}-*largest element* of A by

$$ {}^{k}\!\max A \quad \text{or} \quad {}^{k}\!\max(A)\,. $$

As we only use non-negative weights throughout this thesis, we set ${}^{k}\!\max(A) = 0$ for $k > |A|$.

To produce the greedy algorithm, we follow the same pattern as in Algorithm 2, but we add an extra while loop. In the first while loop, we add the b sets that have the largest total weight. As each element can be covered by b sets, we do not need to choose by which set an element is covered. In the second while loop, the greedy procedure begins. Instead of comparing the largest current weight of each element, as we did in Algorithm 2, we now compare the b^{th}-largest weight. This is the case, because that is the weight that will possibly drop out, if the new weight is larger. Again, we truncate negative values to 0. The resulting algorithm is Algorithm 3.

Algorithm 3: Greedy approximation algorithm for SDW-MMC

```
    Input: an instance of SDW-MMC
    Output: a collection $\mathcal{G}$ of sets $S_i$
1 $\mathcal{G} \leftarrow \emptyset$
2 while $|\mathcal{G}| < b$ do
3     $\hat{S} \leftarrow \underset{S_i \in \mathcal{S} \setminus \mathcal{G}}{\arg\max} \left[ \sum_{j \in J} w_j(S_i) \right]$
4     $\mathcal{G} \leftarrow \mathcal{G} \cup \{\hat{S}\}$
5 while $|\mathcal{G}| < c$ do
6     $\hat{S} \leftarrow \underset{S_i \in \mathcal{S} \setminus \mathcal{G}}{\arg\max} \left[ \sum_{j \in J} \tau \left( w_j(S_i) - \underset{S_k \in \mathcal{G}}{\max}^{b}(w_j(S_k)) \right) \right]$
7     $\mathcal{G} \leftarrow \mathcal{G} \cup \{\hat{S}\}$
8 return $\mathcal{G}$
```

The time complexity of the algorithm remains unchanged, so it still is $O(n \cdot m^3)$.

2.3.3. Performance guarantee analysis

To analyse the performance of Algorithm 3, we use the same strategy as for Algorithm 2. Thus we first define a new total weight function for SDW-MMC. Instead of adding up the largest weight possible for each element, we add up the b largest weights possible for each element.

Definition 2.15. Let $w_j : \mathcal{S} \to \mathbb{R}_{\geq 0}$ be a weight function of an instance of SDW-MMC, then the *total weight* $w^b : \mathcal{P}(\mathcal{S}) \to \mathbb{R}_{\geq 0}$ is

$$w^b(\mathcal{A}) = \sum_{j \in J} \left(\sum_{k=1}^{b} {}^{k}\!\max_{S_i \in \mathcal{A}} w_j(S_i) \right), \quad \forall \mathcal{A} \in \mathcal{P}(\mathcal{S}),$$

with $w^b(\emptyset) = 0$.

With this new total weight function we can rewrite equation (2.9), which is our optimization target, to

$$\mathcal{G}_{\mathsf{opt}} = \underset{\mathcal{A} \subseteq \mathcal{S}}{\arg\max} \{ w^b(\mathcal{A}) : |A| \leq c \}. \tag{2.13}$$

We continue by proving a generalization of Lemma 2.7, namely that the sum of the b largest elements of a set is submodular.

Lemma 2.16. If A and B are subsets of a finite set F, $f : F \to \mathbb{R}_{\geq 0}$ is a non-negative function defined on their elements and b is a positive integer, then

$$\sum_{k=1}^{b} {}^{k}\!\max_{x \in A} f(x) + \sum_{k=1}^{b} {}^{k}\!\max_{x \in B} f(x) \geq \sum_{k=1}^{b} {}^{k}\!\max_{x \in A \cup B} f(x) + \sum_{k=1}^{b} {}^{k}\!\max_{x \in A \cap B} f(x).$$

Proof. In this proof, when referring to the largest element of a set, the element that has the largest value in terms of the function f is meant.

Let $A \subseteq B \subseteq F$ be arbitrary, let b be an arbitrary positive integer and let $y \in F \setminus B$ be an arbitrary element. We now let $z(A)$ denote the sum of the b largest elements of

A, thus

$$z(A) = \sum_{k=1}^{b} {}^{k}\max_{x\in A} f(x)\,.$$

Then the residual values become

$$\rho_y(A) = \sum_{k=1}^{b} {}^{k}\max_{x\in A\cup\{y\}} f(x) - \sum_{k=1}^{b} {}^{k}\max_{x\in A} f(x)$$

and

$$\rho_y(B) = \sum_{k=1}^{b} {}^{k}\max_{x\in B\cup\{y\}} f(x) - \sum_{k=1}^{b} {}^{k}\max_{x\in B} f(x)\,.$$

Because the sum of the b largest elements of a set always increases or remains the same for larger sets, we have that $\rho_y(A) \geq 0$ and $\rho_y(B) \geq 0$. Note that as f is non-negative, the truncation to 0 for sets that are smaller than b, does not bring us any trouble. We now discuss two cases.

Case 1: $f(y) < {}^{b}\max_{x\in B} f(x)$. In other words, y is not part of the b largest elements of B. In this case we immediately get that:

$$\rho_y(B) = \sum_{k=1}^{b} {}^{k}\max_{x\in B\cup\{y\}} f(x) - \sum_{k=1}^{b} {}^{k}\max_{x\in B} f(x) = \sum_{k=1}^{b} {}^{k}\max_{x\in B} f(x) - \sum_{k=1}^{b} {}^{k}\max_{x\in B} f(x) = 0\,.$$

Since $\rho_y(A) \geq 0$, we get $\rho_y(A) \geq \rho_y(B)$.

Case 2: $f(y) \geq {}^{b}\max_{x\in B} f(x)$. In other words, y gives a larger or equal value, than at least one of the b largest elements in B. This means that

$$\sum_{k=1}^{b} {}^{k}\max_{x\in B\cup\{y\}} f(x) = \sum_{k=1}^{b-1} {}^{k}\max_{x\in B} f(x) + f(y)\,.$$

So $\rho_y(B) = f(y) - {}^{b}\max_{x\in B} f(x)$. Because $A \subseteq B$, we have that ${}^{b}\max_{x\in A} f(x) \leq {}^{b}\max_{x\in B} f(x) \leq f(y)$, giving us the same for A:

$$\sum_{k=1}^{b} {}^{k}\max_{x\in A\cup\{y\}} f(x) = \sum_{k=1}^{b-1} {}^{k}\max_{x\in A} f(x) + f(y)\,.$$

Thus $\rho_y(A) = f(y) - {}^{b}\max_{x\in A} f(x)$. Now $\rho_y(A) - \rho_y(B) = {}^{b}\max_{x\in B} f(x) - {}^{b}\max_{x\in A} f(x) \geq 0$, so we also have $\rho_y(A) \geq \rho_y(B)$.

In both cases it holds that $\rho_y(A) \geq \rho_y(B)$, then by Proposition 2.6 this is equivalent to

$$\sum_{k=1}^{b} {}^{k}\max_{x\in A} f(x) + \sum_{k=1}^{b} {}^{k}\max_{x\in B} f(x) \geq \sum_{k=1}^{b} {}^{k}\max_{x\in A\cup B} f(x) + \sum_{k=1}^{b} {}^{k}\max_{x\in A\cap B} f(x)\,.$$

$\square$

With help of this lemma we can now show that the total weight function from Definition 2.15 is submodular.

Theorem 2.17. If $w^b : \mathcal{P}(\mathcal{S}) \to \mathbb{R}_{\geq 0}$ is a total weight function of an instance of SDW-MMC, then w^b is a submodular function.

Proof. Let $\mathcal{A}$ and $\mathcal{B}$ be arbitrary subsets of $\mathcal{S}$. Then,

$$\begin{aligned} w^b(\mathcal{A}) + w^b(\mathcal{B}) &\overset{\text{Def}}{\underset{2.15}{=}} \sum_{j\in J}\left(\sum_{k=1}^{b} {}^k\!\max_{S_i\in\mathcal{A}} w_j(S_i)\right) + \sum_{j\in J}\left(\sum_{k=1}^{b} {}^k\!\max_{S_i\in\mathcal{B}} w_j(S_i)\right) \\ &= \sum_{j\in J}\left(\sum_{k=1}^{b} {}^k\!\max_{S_i\in\mathcal{A}} w_j(S_i) + \sum_{k=1}^{b} {}^k\!\max_{S_i\in\mathcal{B}} w_j(S_i)\right) \\ &\overset{\text{Lem}}{\underset{2.16}{\geq}} \sum_{j\in J}\left(\sum_{k=1}^{b} {}^k\!\max_{S_i\in\mathcal{A}\cup\mathcal{B}} w_j(S_i) + \sum_{k=1}^{b} {}^k\!\max_{S_i\in\mathcal{A}\cap\mathcal{B}} w_j(S_i)\right) \\ &= \sum_{j\in J}\left(\sum_{k=1}^{b} {}^k\!\max_{S_i\in\mathcal{A}\cup\mathcal{B}} w_j(S_i)\right) + \sum_{j\in J}\left(\sum_{k=1}^{b} {}^k\!\max_{S_i\in\mathcal{A}\cap\mathcal{B}} w_j(S_i)\right) \\ &\overset{\text{Def}}{\underset{2.15}{=}} w^b(\mathcal{A}\cup\mathcal{B}) + w^b(\mathcal{A}\cap\mathcal{B}). \end{aligned}$$

Thus, by Definition 2.5, w^b is a submodular function. □

Quite intuitively, the total weight function from Definition 2.15 is also increasing.

Proposition 2.18. If $w^b : \mathcal{P}(\mathcal{S}) \to \mathbb{R}_{\geq 0}$ is a total weight function of an instance of SDW-MMC, then w^b is an increasing function.

Proof. Let $\mathcal{A}$ and $\mathcal{B}$ be arbitrary subsets of $\mathcal{S}$, such that $\mathcal{A} \subseteq \mathcal{B}$. Then,

$$w^b(\mathcal{A}) \overset{\text{Def}}{\underset{2.15}{=}} \sum_{j\in J}\left(\sum_{k=1}^{b} {}^k\!\max_{S_i\in\mathcal{A}} w_j(S_i)\right) \leq \sum_{j\in J}\left(\sum_{k=1}^{b} {}^k\!\max_{S_i\in\mathcal{B}} w_j(S_i)\right) \overset{\text{Def}}{\underset{2.15}{=}} w^b(\mathcal{B})\,.$$

Thus w is an increasing function by Definition 2.4. Here we used the trivial fact that ${}^k\!\max$ is also increasing, if we only have non-negative weights. □

It remains to show that the greedy procedure in Algorithm 3 adds the set S_i with highest residual value. To this end, the residual value of the total weight function of an instance of SDW-MMC is written out in Theorem 2.19.

Theorem 2.19. Let $w^b : \mathcal{P}(\mathcal{S}) \to \mathbb{R}_{\geq 0}$ be a total weight function of an instance of SDW-MMC, then the residual weight of this function can be written as

$$\rho_i(\mathcal{A}) = \sum_{j\in J} \tau\left(w_j(S_i) - {}^b\!\max_{S_k\in\mathcal{A}} w_j(S_k)\right), \quad \forall S_i \in \mathcal{S}, \forall \mathcal{A} \in \mathcal{P}(\mathcal{S})\,.$$

Remark. If $b > |\mathcal{A}|$, we can simplify this to $\rho_i(\mathcal{A}) = \sum_{j\in J} w_j(S_i)$.

Proof. Let $S_i \in \mathcal{S}$ and $\mathcal{A} \in \mathcal{P}(\mathcal{S})$ be arbitrary. Then,

$$\begin{aligned}
\rho_i(\mathcal{A}) &\overset{\text{Def}}{\underset{2.3}{=}} w^b(\mathcal{A} \cup \{S_i\}) - w^b(\mathcal{A}) \\
&\overset{\text{Def}}{\underset{2.15}{=}} \sum_{j \in J} \left(\sum_{k=1}^{b} {}^k\!\max_{S_k \in \mathcal{A} \cup \{S_i\}} w_j(S_k) \right) - \sum_{j \in J} \left(\sum_{k=1}^{b} {}^k\!\max_{S_k \in \mathcal{A}} w_j(S_k) \right) \\
&= \sum_{j \in J} \left(\max \left[w_j(S_i) + \sum_{k=1}^{b-1} {}^k\!\max_{S_k \in \mathcal{A}} w_j(S_k), \sum_{k=1}^{b} {}^k\!\max_{S_k \in \mathcal{A}} w_j(S_k) \right] - \sum_{k=1}^{b} {}^k\!\max_{S_k \in \mathcal{A}} w_j(S_k) \right) \\
&= \sum_{j \in J} \left(\max \left[w_j(S_i) - {}^b\!\max_{S_k \in \mathcal{A}} w_j(S_k), 0 \right] \right) \\
&\overset{\text{Def}}{\underset{2.1}{=}} \sum_{j \in J} \tau \left(w_j(S_i) - {}^b\!\max_{S_k \in \mathcal{A}} w_j(S_k) \right) .
\end{aligned}$$

Although seemingly complicated, the third equality is quite trivial. It just distinguishes between the case where $w_j(S_i)$ is part of the b largest weights and the case where it is not.

The proof of the remark immediately follows from Definition 2.14, where it was stated that ${}^k\!\max(A) = 0$ for $k > |A|$. □

We are now ready to prove what the performance guarantee of Algorithm 3 is.

Corollary 2.20. Algorithm 3 guarantees to find a $(1 - \frac{1}{e})$-approximation of any non-trivial instance of SDW-MMC.

Proof. This follows directly from Theorem 2.11. As shown earlier in this section, we can write any instance of SDW-MMC in the form of equation (2.13). In Theorem 2.17 and Proposition 2.18 we showed that the corresponding function w^b is increasing and submodular. By excluding trivial cases, w^b must be non-constant. Furthermore, w^b was defined to have value 0 for the empty set (Definition 2.15). In Theorem 2.19 we showed that for each iteration, Algorithm 3 indeed adds the set S_i with maximal incremental value. Note that in the algorithm we used the remark below Theorem 2.19 to simplify the expression for the first b iterations. Together, these results thus show that all conditions of Theorem 2.11 are met and we get our approximation bound. □

2.4. Power Set Dependent Weighted Maximum Multi-Coverage

Although SDW-MMC allows for elements to be covered multiple times, the corresponding weights are always added up. To allow for more interaction between the sets that cover an element, we introduce a completely novel extension: Power Set Dependent Weighted Maximum Multi-Coverage (PDW-MMC). In subsection 2.4.1 the problem is formally introduced and two version of a linear program are given. Subsection 2.4.2 covers a restricted version of PDW-MMC, for which more results can be derived. Finally, subsections 2.4.3 and 2.4.4 cover a greedy algorithm and its performance guarantee.

2.4.1. Problem description and linear program

Whereas SDW-MMC has very rigid weights for each set S_i, for some problems the need for more flexible weight arises, an example will be investigated in Chapter 3. PDW-MMC fills this gap. In the PDW-MMC problem each combination of sets that can cover an element has a possibly different weight. Note that coverage by more sets therefore not necessarily means a higher weight. To formally discuss the problem, we will need to define the *capped power set*.

Definition 2.21. Let S be a set and let $b \leq |S|$ be a natural number. The *capped power set of order* b is

$$\mathcal{P}_b(S) = \{S' \subseteq S : |S'| \leq b\}\,.$$

Remark. Note that $|\mathcal{P}_b(S)| = \sum_{k=0}^{b} \binom{|S|}{k}$ and that $\mathcal{P}_{|S|}(S) = \mathcal{P}(S)$, where $\mathcal{P}(S)$ is the standard power set.

We can now define the PDW-MMC problem.

Power Set Dependent Weighted Maximum Multi-Coverage

Instance:	$E = \{e_1, \dots, e_n\}$,	a set of n elements
	$\mathcal{S} = \{S_1, \dots, S_m\}$,	a collection of m non-empty subsets of E
	c,	an integer such that $1 \leq c \leq m$
	b,	an integer such that $1 \leq b \leq c$
	$w_j : \boldsymbol{\mathcal{P}}_b(\mathcal{S}) \to \mathbb{R}_{\geq 0}$,	a non-negative weight function per element e_j, with $w_j(\emptyset) = 0$
Objective:	Choose a collection of c sets $\mathcal{G} \subseteq \mathcal{S}$, such that the sum of the weights of the covered elements is maximized. Here, an element can be covered by at most b of the chosen sets and only the corresponding weight is counted. An element can only be covered by a set, if it is contained in that set.	

Instead of the set $\mathcal{S}$, the weight function now has the capped powerset $\boldsymbol{\mathcal{P}}_b(\mathcal{S})$ as its domain. So for each element e_j a weight function exists, which depends on the combination of sets that cover the element. The emptyset has weight 0, such that elements that are covered by it (to rephrase it, elements that are not covered) do not contribute to the objective function.

To set up an ILP, we first need some new notation. Let $K_b = \{1, \dots, |\mathcal{P}_b(\mathcal{S})|\}$ be the index set corresponding to the elements of $\boldsymbol{\mathcal{P}}_b(\mathcal{S})$. These elements, which are sets containing different S_i, will be denoted by $\mathcal{T}_k$. Furthermore, we create an index set $I_k = \{i \in I : S_i \in \mathcal{T}_k\}$ for all $k \in K_b$, consisting of all indices of the sets S_i that are contained in $\mathcal{T}_k$.

As a side note, we show that PDW-MMC indeed is a generalization of SDW-MMC. For each instance of SDW-MMC with weight function $w_j : \mathcal{S} \to \mathbb{R}_{\geq 0}$, we can create an instance of PDW-MMC by only changing the weight function to have $\boldsymbol{\mathcal{P}}_b(\mathcal{S})$ as its domain. The new weight function $\tilde{w}_j : \boldsymbol{\mathcal{P}}_b(\mathcal{S}) \to \mathbb{R}_{\geq 0}$ would become:

$$\tilde{w}_j(\mathcal{T}_k) = \begin{cases} \sum_{i \in I_k} w_j(S_i) & \text{if } \mathcal{T}_k \neq \emptyset\,, \\ 0 & \text{if } \mathcal{T}_k = \emptyset\,, \end{cases} \qquad \forall \mathcal{T}_k \in \boldsymbol{\mathcal{P}}_b(\mathcal{S})\,. \tag{2.14}$$

The fact that SDW-MMC is a special case of PDW-MMC, implies that PDW-MMC still belongs to the class of NP-hard problems, if $\mathsf{P} \neq \mathsf{NP}$.

Although PDW-MMC is NP-hard (unless P = NP), an ILP is still usefull. It can be used to solve small instances and allows for a carefull analysis of in practice approximation ratios, as we will see in Chapter 3. For the ILP one of the decision variables is changed, in comparison with SDW-MMC. This results in:

$$x_i = \begin{cases} 1 & \text{if } S_i \text{ is chosen,} \\ 0 & \text{if } S_i \text{ is not chosen,} \end{cases} \tag{2.15}$$

$$y_{kj} = \begin{cases} 1 & \text{if } \mathcal{T}_k \text{ covers } e_j\,, \\ 0 & \text{if } \mathcal{T}_k \text{ does not cover } e_j\,. \end{cases} \tag{2.16}$$

We see that y does not indicate by which set an element is covered, but by which collection of sets it is covered.

With these decision variables, we produce the ILP-formulation in equation (2.17).

$$\begin{aligned}
\text{maximize} \quad & \sum_{j\in J}\sum_{k\in K_b} y_{kj}\cdot w_j(\mathcal{T}_k)\,, && && \text{(2.17a)}\\
\text{subject to} \quad & \sum_{i\in I} x_i \leq c, && && \text{(2.17b)}\\
& \sum_{k\in K_b} y_{kj} \leq 1, && j \in J, && \text{(2.17c)}\\
& x_i\cdot z_{ij} \geq y_{kj}, && j\in J, k\in K_b, i\in I_k, && \text{(2.17d)}\\
& x_i \in \{0,1\}, && i\in I, && \text{(2.17e)}\\
& y_{kj} \in \{0,1\}, && j\in J, k\in K_b\,. && \text{(2.17f)}
\end{aligned}$$

Objective function (2.17a) sums over all elements and all combinations of sets of size at most b. Constraint (2.17b) allows for at most c sets to be chosen, while (2.17c) makes sure that each element is covered by exactly one collection of sets. Therefore, in the objective function, only one weight is counted per element. Finally, constraint (2.17d) handles the relation between decision variables x_i and y_{kj}. An element e_j can only be covered by a collection of sets $\mathcal{T}_k$, if all sets $S_i \in \mathcal{T}_k$ are chosen and if all sets $S_i \in \mathcal{T}_k$ contain the element. Only then, the left-hand side of the inequality becomes 1. This constraint also explains the creation of the indexset I_k, because we only want to consider the x_i for which $S_i \in \mathcal{T}_k$.

The condition that all sets $S_i \in \mathcal{T}_k$ must contain the element, can be relaxed. Instead of forcing elements to be covered only by sets that contain the element, we can make sure that adding a set that does not contain the element, does not change the weight. This will then never change the value of the objective function and therefore it is a non-restrictive assumption. More formally, if $\mathcal{T}_k \subseteq \mathcal{T}'_k$ and $e_j \notin S_i$, for all $S_i \in \mathcal{T}'_k \setminus \mathcal{T}_k$, then $w_j(\mathcal{T}_k) = w_j(\mathcal{T}'_k)$. One implication of this assumption is that $w_j(\{S_i\}) = 0$, if $e_j \notin S_i$. This can easily be seen, by mentioning that $w_j(\emptyset) = 0$ by the problem definition. This implication is equivalent to the assumption that was made to simplify the LP's for SDW-MC and SDW-MMC.

This new assumption simplifies constraint (2.17d), resulting in the following ILP:

$$
\begin{aligned}
\text{maximize} \quad & \sum_{j \in J} \sum_{k \in K_b} y_{kj} \cdot w_j(\mathcal{T}_k) && && (2.18a)\\
\text{subject to} \quad & \sum_{i \in I} x_i \ \leq c, && && (2.18b)\\
& \sum_{k \in K_b} y_{kj} \leq 1, && j \in J, && (2.18c)\\
& x_i \ \geq y_{kj}, && j \in J, k \in K_b, i \in I_k, && (2.18d)\\
& x_i \ \in \{0,1\}, && i \in I, && (2.18e)\\
& y_{kj} \in \{0,1\}, && j \in J, k \in K_b\,. && (2.18f)
\end{aligned}
$$

As shown above, this problem is very hard to solve. The number of constraints is even higher than the previously discussed problems, due to the exponential size of the set of weights. Thus there is a strong need for a polynomial approximation algorithm. Unfortunately, it is not possible to construct an algorithm that solves PDW-MMC within a certain ratio. There are no real assumptions on the weight function and therefore an optimum can lie at any possible combination of chosen sets. Any arbitrary set could have a weight close to ∞. It is thus not possible to find this combination in a general way, without checking all possibilities. However, if we assume a certain structure for the weights, some nice results are still possible. In the next subsection we will introduce these assumptions.

2.4.2. Restricted version of the problem

We introduce a restricted version of PDW-MMC which we will call PDW-MMC*. The imposed restrictions will allow us to set up a greedy algorithm and find its approximation ratio. We make three assumptions:

1. Elements can be covered by all chosen sets, thus $b = \infty$.
2. The weights are increasing. By Definition 2.4 we then have,
$$w_j(\mathcal{A}) \leq w_j(\mathcal{B}), \quad \forall \mathcal{A}, \mathcal{B} \in \boldsymbol{\mathcal{P}}(\mathcal{S}), \text{ such that } \mathcal{A} \subseteq \mathcal{B}\,. \tag{2.19}$$
3. The weights are submodular. By Definition 2.5 we then have,
$$w_j(\mathcal{A}) + w_j(\mathcal{B}) \geq w_j(\mathcal{A} \cap \mathcal{B}) + w_j(\mathcal{A} \cup \mathcal{B}), \quad \forall \mathcal{A}, \mathcal{B} \in \boldsymbol{\mathcal{P}}(\mathcal{S})\,. \tag{2.20}$$

Assumptions 1 and 2 are enough to change the ILP. As the weights are increasing and elements can be covered by all chosen sets, only the collections of size c should be taken into account. If we now define a new index set $K' = \{|\mathcal{P}_{c-1}(\mathcal{S})| + 1, \dots, |\mathcal{P}_c(\mathcal{S})|\}$ corresponding to all sets of size c and we change I_k to I'_k accordingly, we get the following ILP:

$$\begin{aligned}
\text{maximize} \quad & \sum_{j \in J} \sum_{k \in K'} y_{kj} \cdot w_j(\mathcal{T}_k)\,, && (2.21a)\\
\text{subject to} \quad & \sum_{i \in I} x_i \ \leq c, && (2.21b)\\
& \sum_{k \in K'} y_{kj} \leq 1, \qquad j \in J, && (2.21c)\\
& x_i \ \geq y_{kj}, \qquad j \in J, k \in K', i \in I'_k, && (2.21d)\\
& x_i \ \in \{0,1\}, \quad i \in I, && (2.21e)\\
& y_{kj} \in \{0,1\}, \quad j \in J, k \in K'\,. && (2.21f)
\end{aligned}$$

Assumption 3 is necessary for the performance guarantee of the approximation algorithm that will be presented in the next subsection.

2.4.3. Greedy approximation algorithm

Due to the increasing nature and submodularity of the weights in PDW-MMC*, a logical candidate to solve this problem greedily would be Algorithm 4.

Algorithm 4: Greedy approximation algorithm for PDW-MMC*

```
    Input: an instance of PDW-MMC*
    Output: a collection $\mathcal{G}$ of sets $S_i$
1   $\mathcal{G} \leftarrow \emptyset$
2   while $|\mathcal{G}| < c$ do
3       $\hat{S} \leftarrow \operatorname{argmax}_{S_i \in \mathcal{S} \setminus \mathcal{G}} \sum_{j \in J} (w_j(\mathcal{G} \cup \{S_i\}) - w_j(\mathcal{G}))$
4       $\mathcal{G} \leftarrow \mathcal{G} \cup \{\hat{S}\}$
5   return $\mathcal{G}$
```

At every iteration, we include the set that adds the most additional weights. Because $b = \infty$, we can simply compare the difference in weights for each possible new set S_i. The time complexity of the algorithm, which is $O(n \cdot m^2)$, makes this a polynomial algorithm.

2.4.4. Performance guarantee analysis

Of course it is of great interest to know how well Algorithm 4 approximates the optimal solution. To this end, we first define the total weight of an instance of PDW-MMC*. Because the weights are increasing and $b = \infty$, we can just add up the weight of each element for the current selection.

Definition 2.22. Let $w_j : \mathcal{P}(\mathcal{S}) \to \mathbb{R}_{\geq 0}$ be a weight function of an instance of PDW-MMC*, then the *total weight* $w : \mathcal{P}(\mathcal{S}) \to \mathbb{R}_{\geq 0}$ is

$$w(\mathcal{A}) = \sum_{j \in J} w_j(\mathcal{A}), \quad \forall \mathcal{A} \subseteq \mathcal{S}\,.$$

Just as in the previous sections, we are trying to find a subset of $\mathcal{S}$ of size at most c, while maximizing the total weight of this subset. Therefore, we can write PDW-MMC* as the problem of finding $\mathcal{G}_{\mathsf{opt}}$, where

$$\mathcal{G}_{\mathsf{opt}} = \arg\max_{\mathcal{A}\subseteq\mathcal{S}}\{w(\mathcal{A}) : |A| \leq c\}\,. \tag{2.22}$$

The next step in our analysis, is to show that on line 3 of Algorithm 4, we add the set with largest incremental weight.

Theorem 2.23. Let $w : \boldsymbol{\mathcal{P}}(\mathcal{S}) \to \mathbb{R}_{\geq 0}$ be a total weight function of an instance of PDW-MMC*, then the residual weight of this function can be written as

$$\rho_i(\mathcal{A}) = \sum_{j\in J}(w_j(\mathcal{A}\cup\{S_i\}) - w_j(\mathcal{A}))\,, \quad \forall S_i \in \mathcal{S}, \forall \mathcal{A} \in \boldsymbol{\mathcal{P}}(\mathcal{S})\,.$$

Proof. Let $S_i \in \mathcal{S}$ and $\mathcal{A} \in \boldsymbol{\mathcal{P}}(\mathcal{S})$ be arbitrary. Then,

$$\begin{aligned}\rho_i(\mathcal{A}) \overset{\text{Def}}{\underset{2.3}{=}} w(\mathcal{A}\cup\{S_i\}) - w(\mathcal{A}) &\overset{\text{Def}}{\underset{2.22}{=}} \sum_{j\in J} w_j(\mathcal{A}\cup\{S_i\}) - \sum_{j\in J} w_j(\mathcal{A}) \\ &= \sum_{j\in J}(w_j(\mathcal{A}\cup\{S_i\}) - w_j(\mathcal{A}))\,.\end{aligned}$$

□

We are now ready to prove the main result of this subsection: the performance guarantee of Algorithm 4. To derive an approximation bound, we again turn our attention to Theorem 2.11 and show that its conditions hold for PDW-MMC* and Algorithm 4.

Corollary 2.24. Algorithm 4 guarantees to find a $(1-\frac{1}{e})$-approximation of any non-trivial instance of PDW-MMC*.

Proof. This follows directly from Theorem 2.11. As shown earlier in this subsection, we can write any instance of PDW-MMC* in the form of equation (2.22). For an instance of PDW-MMC*, the weights w_j are increasing and submodular by the assumptions made in equations (2.19) and (2.20). This immediately implies that the total weight w (as defined in Definition 2.22) is increasing and submodular, as we just sum over all elements. By excluding trivial cases, w must be non-constant. Furthermore, w_j was defined to have value 0 for the empty set (see the problem description of PDW-MMC) and thus w has this same property. In Theorem 2.23, we showed that for each iteration, Algorithm 4 indeed adds the set S_i with maximal incremental value. Thus, all conditions of Theorem 2.11 are met and we get that Algorithm 4 provides a $(1-\frac{1}{e})$-approximation for PDW-MMC*. □

Chapter 3

Selection of weather stations in the Netherlands

One application of the class of maximum coverage problems is the selection of the optimal collection of sensors, as argued by Aggarwal, Bar-Noy, and Shamoun in [2] and [3]. Specifically, we will use the PDW-MMC problem to model the selection of weather stations in the Netherlands. In section 3.1 the problem is described. Afterwards, two different approaches to calculate weights are investigated in section 3.2 and 3.3.

3.1. Problem description

The Netherlands has a total of 46 official weather stations monitored by the Royal Dutch Meteorological Institute (KNMI). These stations, or sensors, gather useful data, such as temperature, radiation and wind speed. We will try to find a subset of these stations of a certain size, that gives us the maximal information. Only the stations itself are considered as our points of interest. We focus on three physical quantities:

- Temperature: the daily mean temperature (in degrees Celsius);
- Precipitation: the daily amount of rainfall (in millimeters);
- Humidity: the daily mean relative atmospheric humidity (in percentages).

Figure 3.1 shows the 34 sensors that capture these quantities. The stations indicated by white squares are used to test the performance of the greedy algorithm. To model this problem as a PDW-MMC problem, we first need to define what the elements e_j and the sets S_i are. As our stations are also our locations of interest, both the elements and the sets correspond to a possible sensor location. Each set S_i contains all elements e_j, as every station always gives some information about other stations. Furthermore, we set $b = \infty$, as we want every weather station available to help with the estimations. It should be noted that we are implicitly implying that the set $\mathcal{S}$ now becomes a multiset, where duplicate elements are allowed, as each S_i contains the same elements. However, this has no influence on the analysis conducted in this thesis, as we never used the characteristic that the S_i are unique. In [8], GMC was even defined in such a way that it allows multisets.

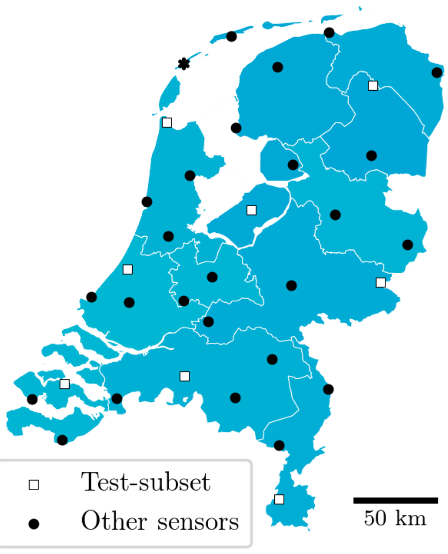


Figure 3.1: Locations of the 34 official KNMI-monitored weather stations in the Netherlands, that measure temperature, humidity and precipitation. The stations used for testing purposes are indicated by white squares. (The station indicated by $*$ only measures temperature and humidity and no precipitation.)

It remains to determine weights w_j per location for each combination of sensors. Then we can maximize the total weight to obtain a subset of c sensors. Two approaches, based on two of the most used interpolation methods in geospatial analysis [17], are tried to determine weights: an interpolation technique called *inverse distance weighting* and *multiple linear regression*. We refer to these different weights as IDW-weights and MLR-weights. For both approaches we estimate the values at a certain location for each day that we have data available, using only the data of the selected sensors. As a side note, let us mention that it is important to make a distinction between interpolation and forecasting. We aim to *interpolate* the data, meaning that we predict certain values based on values at another location. This is unalike *forecasting*, were one seeks to predict future data.

After the interpolation is conducted, we calculate the weights by setting them equal to the R^2 of the estimate. The *coefficient of determination* or R^2 is a measure of the fit of a model, independent of the scale of the variables [10]. The R^2 can be calculated as

$$R^2 = 1 - \frac{SSR}{SST} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}\,. \tag{3.1}$$

Here y_i denotes the real data, $\hat{y}_i$ denotes the estimated data and $\bar{y}$ is the mean over all y_i. SSR is the sum of squared residuals, in other words the sum of squared errors. SST is the total sum of squares, which gives an indication of the variation of the data with respect to its mean. It is quite clear that $R^2 \leq 1$, as the fraction never becomes negative due to the squares. The R^2 increases when the fit gets better and is thus a good indicator of the importance of a selection. However, a small adjustment must be made. If we do not estimate y with regression, the R^2 can become negative and give us a worse model, than just taking a constant. Thus, we will truncate the weights to 0, when the R^2 is negative. In such cases we assume that a constant model will be used, as in real life we always have the option to not use parts of the data. The objective of PDW-MMC is to maximize the sum of weights, thus we are maximizing the sum of R^2 for each weather station. In essence, we can reformulate this as the maximization of the average R^2 of all locations, as we have a fixed number of weather stations.

Our two approaches differ in the way the data is estimated. While the first method has a more meaningful physical interpretation, the second method gives us nicer mathematical results.

3.2. Weights with inverse distance weighting

The interpolation that we will be using first, is *inverse distance weighting* (IDW), introduced in 1968 by Shepard [16]. It is a deterministic method, not dependent on any statistical assumptions and therefore widely applicable. Furthermore, it is not too computationally heavy. The core of the interpolation is that it is distance-dependent: points further away of the point of interest contribute less to the interpolation. This has a very nice physical interpretation, as weather data of points nearby are likely to be similar. In subsection 3.2.1 this technique will be explained in more detail. Subsection 3.2.2 analyses the in practice performance of a greedy algorithm, while subsection 3.2.3 shows the results of the algorithm.

3.2.1. Inverse distance weighting

IDW uses all available data to interpolate missing data-points. A formal definition, adapted from [16], is given in Definition 3.1.

Definition 3.1 (Inverse distance weighting)**.** Let $D \subset \mathbb{R}^m$ be a set of n points $\mathbf{x}_i$, for which we know function values $f(\mathbf{x}_i) : D \to \mathbb{R}$. Let $p \geq 0$ be a real number and let $d(\mathbf{x}_1, \mathbf{x}_2)$ denote the distance (of a certain metric) between points $\mathbf{x}_1$ and $\mathbf{x}_2$. Then the *IDW-interpolation* of a point $\mathbf{y} \in \mathbb{R}^n$ is

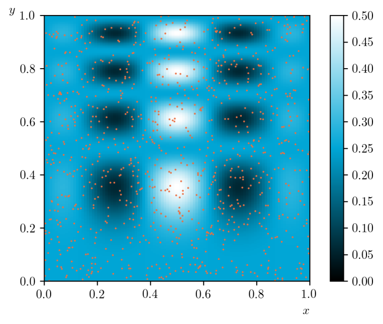
$$u(\mathbf{y}) = \begin{cases} \dfrac{\sum_{i=1}^n v_i(\mathbf{y}) f(\mathbf{x}_i)}{\sum_{i=1}^n v_i(\mathbf{y})} & \text{if } d(\mathbf{x}_i, \mathbf{y}) \neq 0 \text{ for all } i\,, \\ f(\mathbf{x}_i) & \text{if } d(\mathbf{x}_i, \mathbf{y}) = 0 \text{ for some } i\,, \end{cases}$$

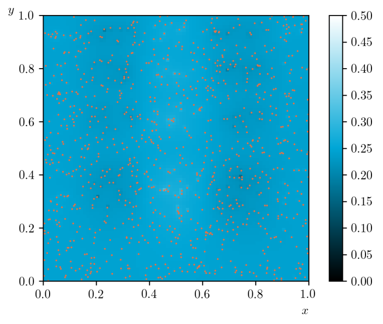
with weights

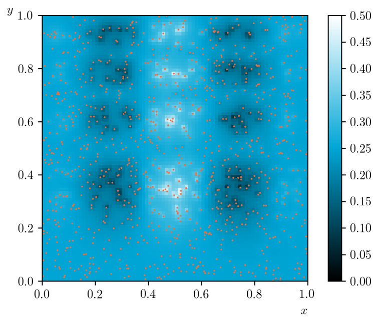
$$v_i(\mathbf{y}) = \frac{1}{d(\mathbf{x}_i, \mathbf{y})^p}\,.$$

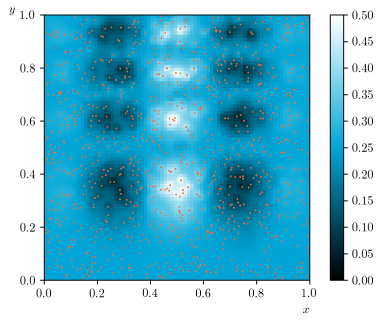
We see that IDW-interpolation returns a weighted average of all available points. The weights v_i (not to be confused with the weights of the maximum coverage models) are inversely proportional with the distance to the available points. As a result, points further away have a smaller weight than points close by. With the *power parameter* p we can tune this relation. For $p = 0$, all available points carry the same weight and thus each IDW-interpolation gives back the average of all points. For $p = 1$, the relationship is exactly inversely proportional. By letting p grow, points that are further away get less and less influence. Finally, if $p \to \infty$ we reach the so-called *nearest neighbor* interpolation. In this form of interpolation, a point takes on the value of its closest neighbor for which we know the function value. For a two-dimensional function this results in the plot being a Voronoi diagram.

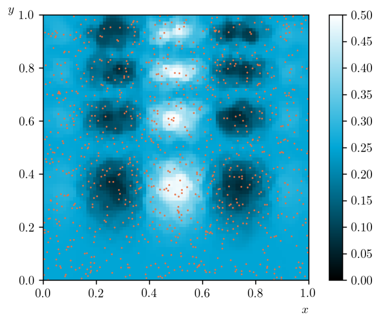
To illustrate the influence of the power parameter, an IDW-interpolation was executed in Figure 3.2 for five different values of p. Figure 3.2a shows the arbitrarily chosen function $f(x, y) = x \cdot (1 - x) \cdot \cos(4\pi x) \cdot \sin^2(4\pi y^2) + \frac{1}{4}$ and the 1000 random points used for the


(a) Original function


(b) IDW with $p = 1$


(c) IDW with $p = 2$


(d) IDW with $p = 3$


(e) IDW with $p = 5$

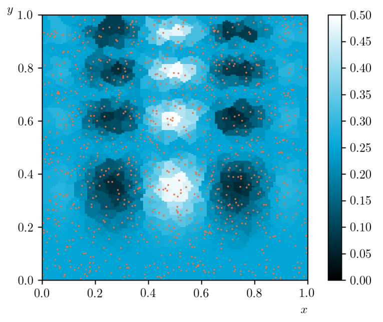

(f) IDW with $p \to \infty$ (Nearest neighbour interpolation)

Figure 3.2: IDW-interpolation using 1000 random samples from the function $f(x, y) = x \cdot (1 - x) \cdot \cos(4\pi x) \cdot \sin^2(4\pi y^2) + \frac{1}{4}$ for five different values of p. The orange dots denote the points in the dataset D.

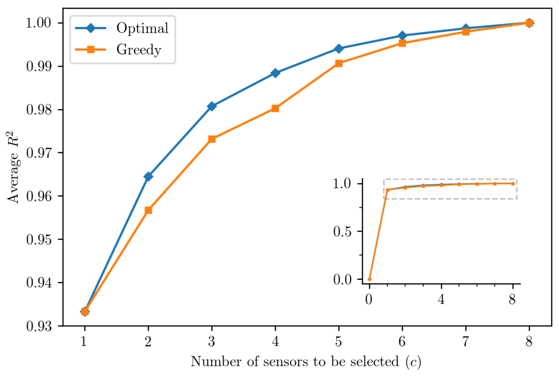
interpolation. From Figures 3.2b–e we can clearly see that for larger p the interpolated function values are less influenced by points further away. This also results in harder cut-offs and a less smooth surface. Figure 3.2f indeed shows that for $p \to \infty$ we get a Voronoi diagram. It seems that for the function f, the IDW-interpolation with $p = 5$ gives us the best result, although one could argue that $p = 3$ is better due to its smoothness. However, this p is highly dependent on the underlying function and the number of used datapoints [7]. It is quite easy to come up with functions for which other values of p are preferable. So, one must be careful when choosing the parameter p. In geospatial analysis a choice of $p = 2$ is widely used. Higher values were also tried in the research leading to this thesis, but results did not change significantly. Therefore, we will adhere to the convention of $p = 2$ in this thesis.

Unfortunately, when using IDW to estimate data, no structure exists for the corresponding R^2. It is not necessarily increasing nor submodular, as it is perfectly imaginable that a certain weather station has a highly negative influence on the estimates. For meteorological reasons, a station close by might have severely different data and thus make the estimates worse. Even though without these characteristics, no approximation guarantees hold, we are still interested in the in practice performance of Algorithm 4 for different values of c.

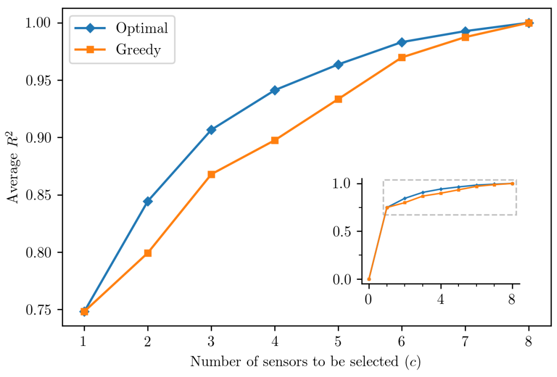
3.2.2. In practice performance analysis

As mentioned in the previous subsection, a greedy algorithm has no performance guarantee, when IDW-weights are used. However, we would still like to know how our algorithm performs in practice. To obtain a measure for the performance, we would need to know the optimal solution to the problem. This allows us to compare the approximate solutions. Here, a problem arises. As explained in subsection 2.4.1, PDW-MMC is a NP-hard problem. It is thus impossible to solve the problem optimally for the whole dataset, as this would require an enormous amount of computation time. Therefore, we first turn our focus to the subset of eight sensors shown in Figure 3.1.

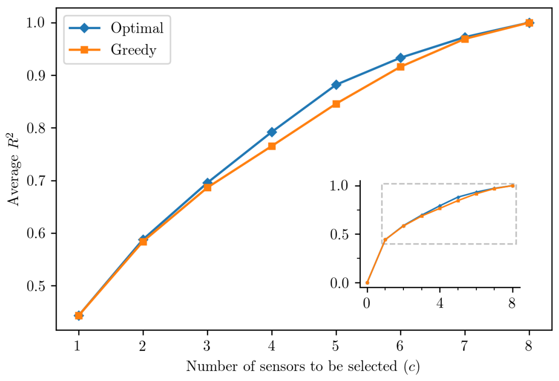
With help of Python, all estimates necessary to find the weights, are calculated using IDW. Then, the corresponding R^2 of the estimates is calculated, such that we can acquire our weights for the PDW-MMC problem. We then have an instance of PDW-MMC, which we can solve optimally with the ILP given in equation (2.18). We will be using the efficient commercial LP-solver Gurobi[1], on an academic license, to fulfill this task. Afterwards, Algorithm 4 is used to give an approximation of these solutions.



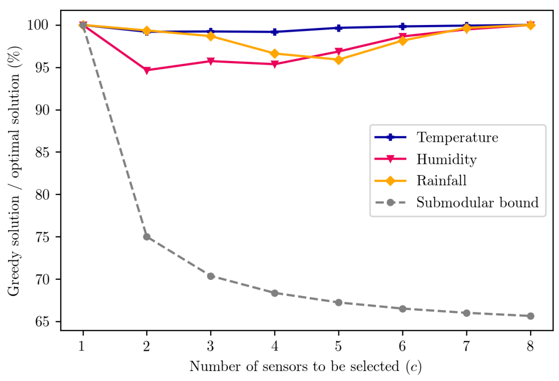
(a) Temperature



(b) Humidity



(c) Rainfall



(d) Approximation ratio of greedy and optimal

Figure 3.3: Total objective value of the optimal solution and greedy solution found by Algorithm 4 of the selection of weather stations from the test-subset. Results for different numbers of sensors to be selected are shown and all three quantities of interest are depicted. Weights are calculated using IDW. The corresponding approximation ratios are also plotted.

[1] https://www.gurobi.com/

For each quantity of interest both the optimal and the approximate solution are calculated for all possible values of c. This gives us objective values, which we can compare. The results for our three quantities of interest are depicted in Figures 3.3a–c. Instead of using the sum of R^2, the average R^2 is used, as it has a more intuitive interpretation. Note that the three graphs are all scaled to fit the window and start at $c = 1$. The smaller graphs inside the figures show the complete graphs. Figure 3.3d shows the corresponding approximation ratios, in comparison with the theoretical bound we get under the assumptions of subsection 2.4.2.

Although theoretically these weights are not increasing, Figures 3.3a–c clearly show us that the objective value of the greedy algorithm do increase. Figure 3.3d even shows that the ratio between the greedy solution and the optimal solution stays well above the bound that is known for submodular weights, even though this assumption does not hold. Thus, in practice, Algorithm 4 seems to give us good approximations.

Another phenomenon worth discussing, is that the average R^2 for $c = 1$ differs quite a bit among the three quantities. When looking at temperature (Figure 3.3a), one sensor gives us an average R^2 larger than 0.93. This is easily explained by the fact that the temperature does not vary largely across the Netherlands and one sensor already gives quite good estimates. The daily rainfall starts at a much smaller R^2 (Figure 3.3c). Probably, because rain is a very local phenomenon that has large variations. The third quantity, humidity, lies in between temperature and rainfall, with a starting R^2 of approximately 0.75.

3.2.3. Greedy results

Although theoretically no bound on the approximation can be derived, the previous subsection shows that in practice our greedy solution gives us good results. As explained before, it is impossible to solve the problem optimally for the whole dataset, so we only show the greedy solutions. In Figure 3.4 the color of the stations indicate at which stage a sensor was added. The darker the color the earlier the sensor was added and thus the ‘more important’ a sensor is. For all three quantities there is good spread of dark and light sensors, indicating that the greedy algorithm does not choose clusters of sensors. This is obviously very intuitive. If viewed digitally, Figures A.1a–c in Appendix A provide animations corresponding to Figure 3.4.

Figure 3.5 shows how the objective function behaves. We see the same kind of behavior as for the test subset of eight stations, with the difference that the increase in R^2 is a bit slower. This is quite logical, because the average is now taken over more sensors and thus there is more room for mistakes.

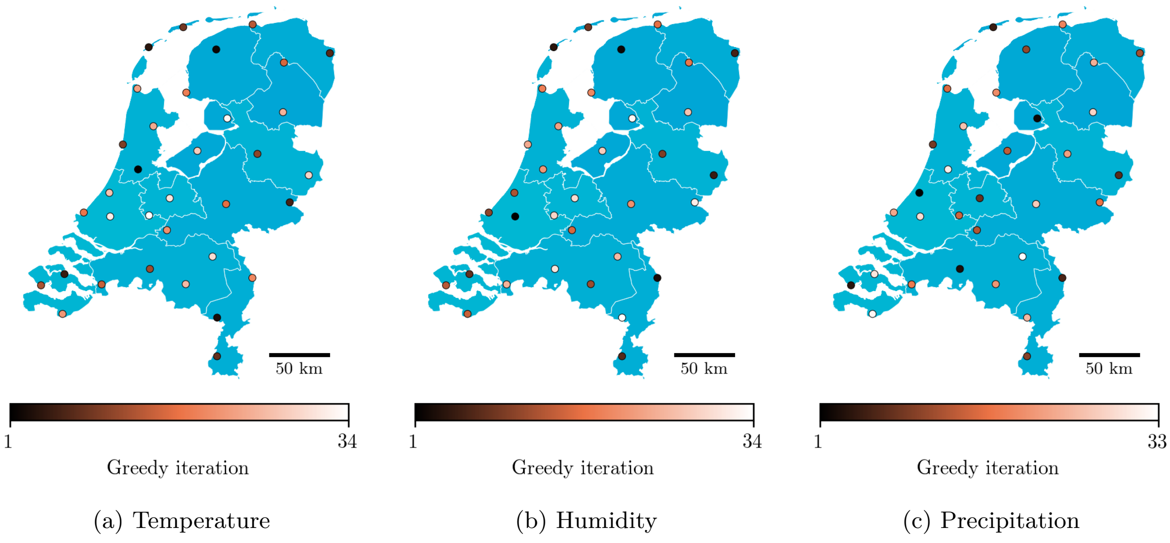


Figure 3.4: Selection of weather stations by Algorithm 4 with IDW-weights. All three quantities of interest are depicted. The color indicates at which iteration a station is chosen.

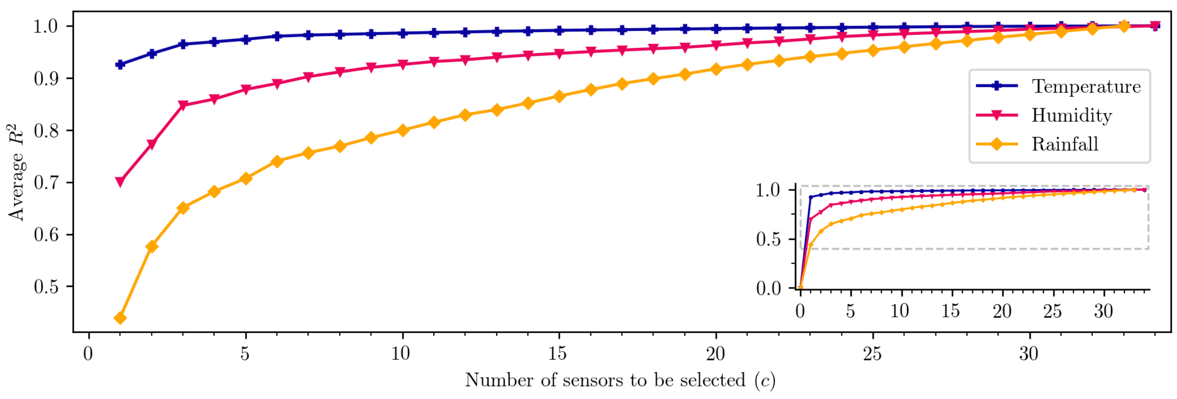


Figure 3.5: Objective value for each iteration of Algorithm 4, applied to the selection of weather stations in the Netherlands based on three different quantities. Weights are calculated with IDW.

3.3. Weights with multiple linear regression

A more sophisticated method to make estimations is by using *multiple linear regression* (MLR), also known as *multiple regression*. In contrast to IDW, it is a stochastic method dependent on certain statistical assumptions. Although no physical reasoning lies behind the estimation, one major advantage of this method is that the weights will always be increasing. The more regressors we add, the better the estimate will be and thus the higher the weights. In the next subsection we prove this result and give a short overview of MLR. In subsections 3.3.2 and 3.3.3 we discuss the performance of the greedy algorithm and the corresponding results.

3.3.1. Multiple linear regression

This short recap on multiple linear regression is loosely based on [10]. For more details, we refer to that study book.

Let

$$y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}, \quad X = \begin{pmatrix} 1 & x_{11} & \dots & x_{1k} \\ \vdots & \vdots & & \vdots \\ 1 & x_{n1} & \dots & x_{nk} \end{pmatrix}, \quad \beta = \begin{pmatrix} \beta_0 \\ \vdots \\ \beta_k \end{pmatrix}, \quad \varepsilon = \begin{pmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_n \end{pmatrix}. \tag{3.2}$$

Here y is the vector of dependent variables which we seek to estimate. Each of the n variables y_i has k corresponding *regressors* or *explanatory variables.* The matrix X contains all these regressors and a column of ones, corresponding to the addition of a constant in the model. In other words, X represents the complete dataset, used to estimate y. In MLR, it is assumed that each y_i is a linear combination of the k regressors in X, plus a random error ε. If we denote the corresponding parameters by β, we get,

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_k x_{ik} + \varepsilon_i, \quad i = 1, \dots, n. \tag{3.3}$$

Using the notation in equation (3.2) we can write these n equations as,

$$y = X\beta + \varepsilon. \tag{3.4}$$

It is our goal to find an estimate $\hat{\beta}$ of the vector β, to get estimates $\hat{y}$ for y. The criterion used in MLR, is to find the vector b which minimizes the sum of squared residuals, given by

$$SSR = \sum_{i=1}^{n} (y_i - b_0 - b_1 x_{i1} - \cdots - b_k x_{ik})^2. \tag{3.5}$$

The *ordinary least squares* (OLS) estimator does exactly this job and can be calculated as,

$$\hat{\beta}_{\mathsf{OLS}} = (X'X)^{-1}X'y, \tag{3.6}$$

where X' denotes the transpose of X. Thus, MLR gives the following estimate for the vector y:

$$\hat{y} = X\hat{\beta}_{\mathsf{OLS}} = X(X'X)^{-1}X'y. \tag{3.7}$$

Because MLR minimizes the SSR and the R^2 is based on the SSR (see equation (3.1)), we can prove the following useful result.

Theorem 3.2. When using multiple linear regression to estimate data, the R^2 is increasing, if we view R^2 as a set-function with the set of regressors as its domain.

Proof. Let A, B both be arbitrary sets of regressors, such that $A \subseteq B$. On page 137 in [10] the very intuitive result that the SSR is decreasing, is proved. With slight abuse of notation, we thus have that $SSR(A) \geq SSR(B)$. Furthermore, the total sum of squares SST is not dependent on the regressors and so $SST(A) = SST(B)$. Then by the definition of R^2, given in equation (3.1), we get

$$R^2(A) = 1 - \frac{SSR(A)}{SST(A)} \leq 1 - \frac{SSR(B)}{SST(B)} = R^2(B).$$

So, R^2 is increasing by Definition 2.4. $\square$

Although we now have that the R^2 is increasing, the second property we would appreciate, does not hold. In Theorem 3.3 we state that the R^2 is not submodular (nor supermodular). The proof of the theorem also supplies us with a counterexample.

Theorem 3.3. When using multiple linear regression to estimate data, the R^2 is neither submodular nor supermodular, if we view R^2 as a set-function with the set of regressors as its domain.

Proof. We will prove this by giving two easy counterexamples. Let,

$$y = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}, \quad X = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 1 & 2 \\ 1 & 1 & 2 & 3 \end{pmatrix}.$$

Each of the columns of X correspond to a regressor. Now let $A = \{1,2\}$ and $B = \{1,3\}$ both be sets of these regressors. Then, with slight abuse of notation, we find $R^2(A) = R^2(\{1,2\}) = 0$, $R^2(B) = R^2(\{1,3\}) = 0.75$, $R^2(A \cup B) = R^2(\{1,2,3\}) = 1$ and $R^2(A \cap B) = R^2(\{1\}) = 0$. Thus we get that

$$R^2(A) + R^2(B) \ngeq R^2(A \cup B) + R^2(A \cap B)\,.$$

Then, by Definition 2.5, R^2 is not submodular.

Now, let $C = \{1,4\}$. It can easily be verified that $R^2(C) = R^2(\{1,4\}) = 1$, $R^2(A \cup C) = R^2(\{1,2,4\}) = 1$ and $R^2(A \cap C) = R^2(\{1\}) = 0$. Thus,

$$R^2(A) + R^2(C) \nleq R^2(A \cup C) + R^2(A \cap C)\,.$$

Then, by Definition 2.5, R^2 is also not supermodular. $\square$

3.3.2. In practice performance analysis

In comparison with IDW-weights, MLR-weights have more structure: they are increasing. However, even with increasing weights, the selection of weather stations can still not be modelled as a PDW-MMC* problem. As shown in Theorem 3.3, the assumption of submodularity does not hold. Therefore, we once more turn our attention to a small subset of the problem to investigate the in practice approximation ratio of a greedy algorithm.

Just as for IDW-weights, the greedy algorithm that is tested is Algorithm 4. The ILP that is used to obtain the optimal solutions can be simplified a bit. As explained in subsection 2.4.2, only the assumptions that $b = \infty$ and that the weights are increasing are enough to use the ILP from equation (2.18), which has fewer options to consider. This ILP will also be solved with `Gurobi`.

Figures 3.6a–c show the results of this analysis. When compared to Figures 3.3a–c, which show the results for IDW-weights, a lot of similarities appear. Nevertheless, one difference can be seen: the average R^2 in Figures 3.6a–c is larger. This is very logical, as MLR minimizes the sum of squared residuals and thus maximizes the R^2, whereas IDW is not specifically tailored towards this maximization goal. What attracts attention, is that the difference in R^2 between IDW and MLR is minimal. This indicates that IDW already delivered very good estimates.

From Figure 3.3d we see that the approximation is even better than before. Thus we again conclude that de facto our algorithm seems to give us good approximations. The rest of the trends in the figures remain unchanged. Temperature still starts with the highest R^2, while rainfall begins with a low R^2.

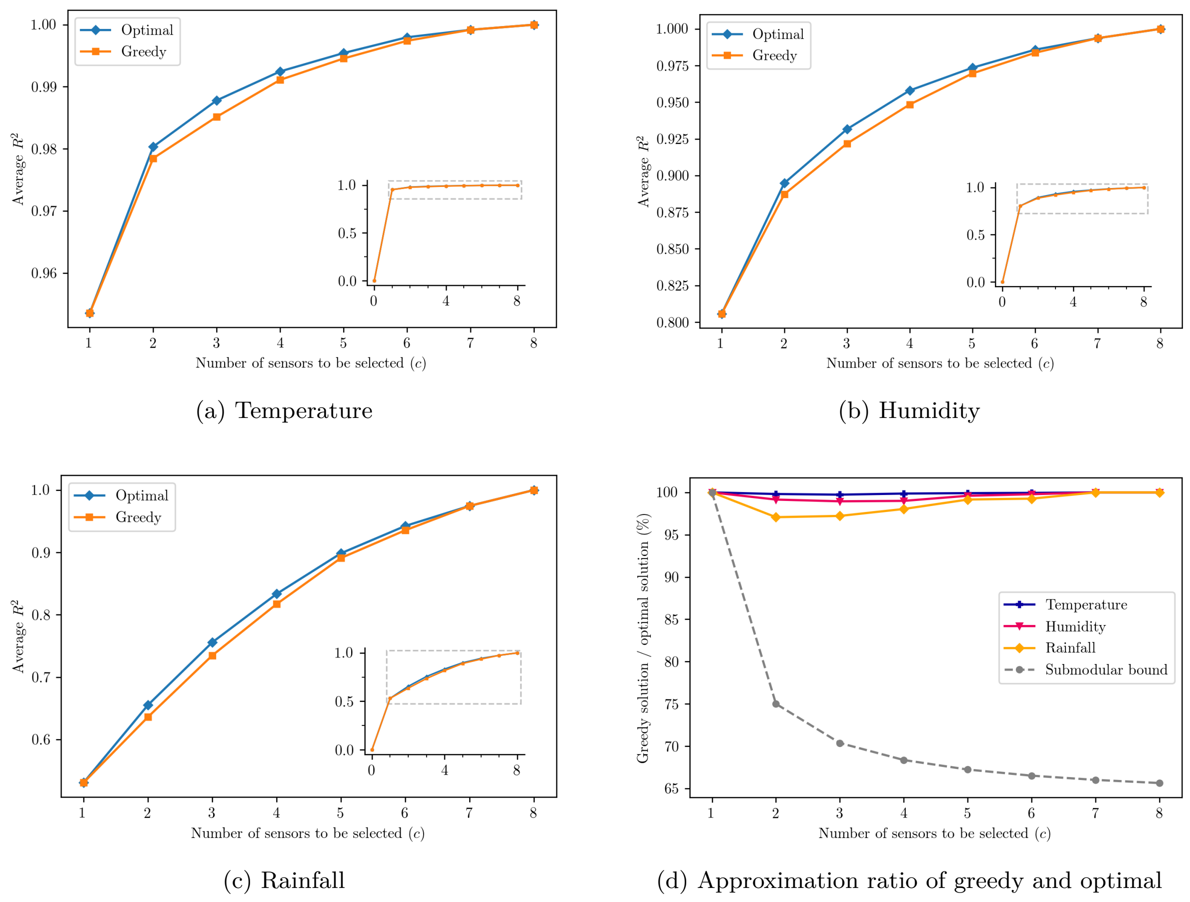


Figure 3.6: Total objective value of the optimal solution and greedy solution found by Algorithm 4 of the selection of weather stations from the test-subset. Results for different numbers of sensors to be selected are shown and all three quantities of interest are depicted. Weights are calculated using MLR. The corresponding approximation ratios are also plotted.

3.3.3. Greedy results

From the previous subsection we have learned that Algorithm 4 has a very good in practice performance for MLR-weights, even better than for IDW-weights. Therefore, we can now turn our focus on the whole group of weather stations.

Figure 3.7 depicts at what stage Algorithm 4 chooses a certain weather station. Again a good spread of the lighter and darker coloured stations can be seen. It is also quite apparent that the sensors are chosen in different orders for the different quantities of interest. Figures A.1d–f in Appendix A again provide visually more pleasing animations of Figure 3.7, which can only be viewed digitally.

Furthermore, Figure 3.8 shows the objective values corresponding to Figure 3.7. Just as for IDW-weights, when we want to measure temperature or humidity, even a few sensors gives

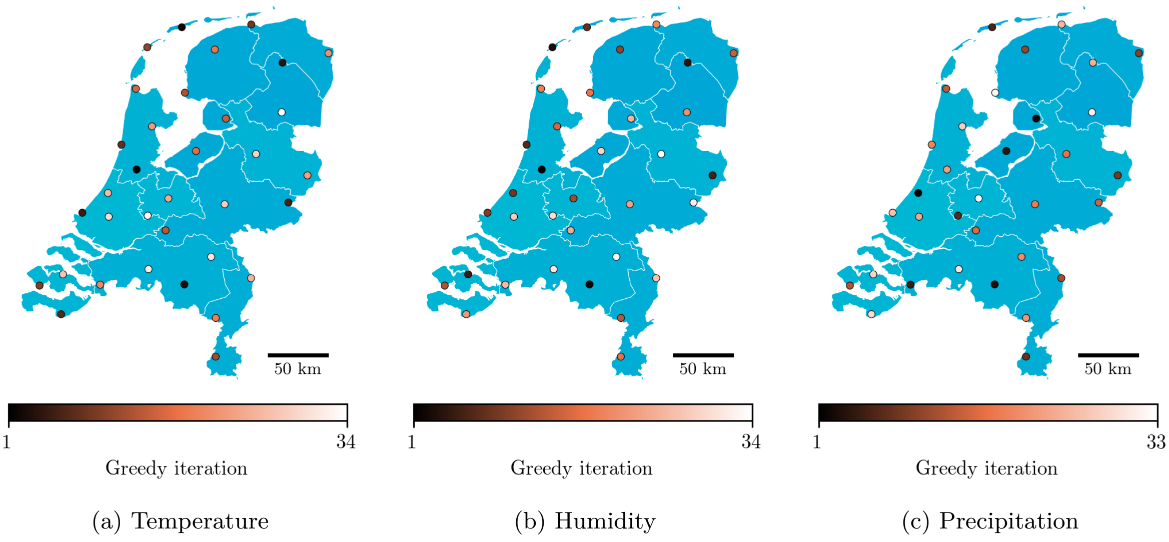


Figure 3.7: Selection of weather stations by Algorithm 4 with MLR-weights. All three quantities of interest are depicted. The color indicates at which iteration a station is chosen.

us a large average R^2. As discussed before, rainfall is a more local phenomenon, demanding more sensors to get the same average R^2.

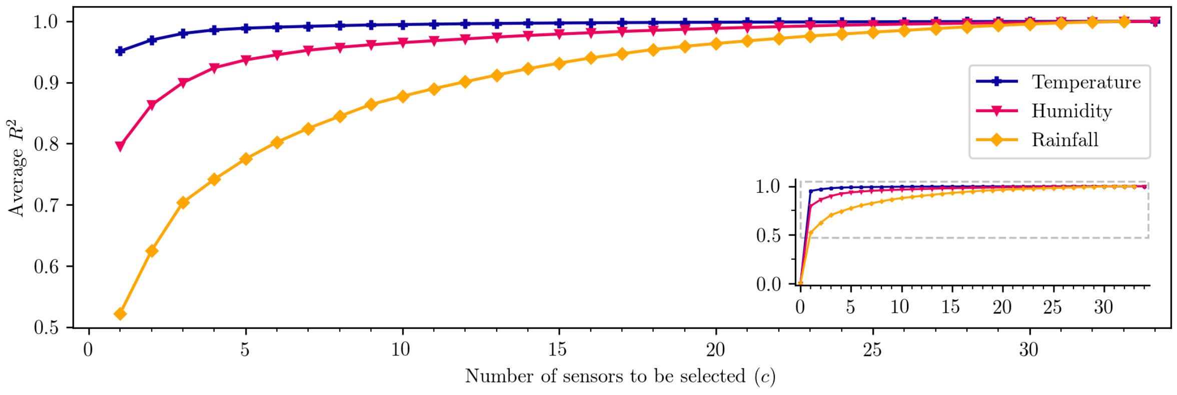


Figure 3.8: Objective value for each iteration of Algorithm 4, applied to the selection of weather stations in the Netherlands based on three different quantities. Weights are calculated with MLR.

Chapter 4
Conclusion and outlook

4.1. Conclusion

We have discussed several maximum coverage problems. First, some already known results for WMC were revised. Afterwards, we defined the SDW-MC problem and formulated an ILP. We discussed some results from [14], which we used to prove that a greedy algorithm gives a $(1 - \frac{1}{e})$-approximation of the optimal solution. Then, we extended the problem by allowing elements to be covered by at most b sets, as opposed to just one. This resulted in the SDW-MMC problem. A similar analysis as for SDW-MC was conducted, to show that a greedy algorithm also guarantees a $(1 - \frac{1}{e})$-approximation for SDW-MMC. To allow for more flexibility, we let every combination of sets have a possibly different weight. We set up an ILP-formulation to solve this PDW-MMC problem optimally. Although a greedy algorithm has no theoretical performance guarantee for this problem, under the assumptions that the weights are increasing and submodular, and each element can be covered by all sets, we again derived a performance guarantee of $(1 - \frac{1}{e})$.

In the subsequent chapter, the PDW-MMC problem was tested. In the framework of sensor selection, we modelled the optimal selection of weather stations in the Netherlands. We used a publicly available dataset of climate data from the KNMI to determine our weights. For each combination of sensors we estimated the data of all sensors. From these estimations we could calculate the R^2 for each sensor, which is a measure of the fit of our estimations. These R^2 were then used as weights. The estimations were done by two different methods. First, we tried inverse distance weighting, an interpolation technique which returns a weighted average of the data from other sensors, based on the distance they are apart. Afterwards, multiple linear regression was used.

Although neither of the methods fulfilled the requirement of submodular weights, the in practice performance of the greedy algorithm, tested on a subset of the problem, was very high. We eventually found an order in which the sensors are added by the greedy algorithm for three different quantities of interest: temperature, rainfall and humidity. As a side result, we found out that even one sensor gives an average R^2 larger than 0.93, when estimating temperature. For humidity and rainfall these values were approximately 0.70 and 0.41, respectively. Therefore, the quality of estimation varies a lot, due to the physical properties of the quantities. The amount of rainfall has large variations, even across a small country, such as the Netherlands. Meanwhile, the temperature does not differ a lot between parts of the

Netherlands. This explains the relatively high R^2.

We should address that the way we modelled the selection of weather stations in the Netherlands is a mere simplification of reality. We only assumed each weather station to be a point of interest, as no climate data is available for the entirety of the Netherlands. This results in the modelling becoming somewhat artificial, as the greedy algorithm is drawn towards areas with a high density of weather stations. Those areas contribute the most to the total objective function. Thus, if we want to interpret the acquired results, we should do so with caution. A logical and safe interpretation is that the order in which the stations are added, resembles the order of importance of the stations. Therefore, it can be used to choose which stations should be kept, if the authorities were to remove some of them in the future.

4.2. Further research

One could think of numerous directions of further research. Either, in areas where our approaches have their limitations or by diving deeper in some of the matter discussed in this thesis. The most notable examples will be listed below.

We first return to the problem of the algorithm being drawn towards areas with a lot of weather stations. One way to circumvent this issue, would be by dividing the research area in sections. If we then multiply each station with an importance factor, inversely proportional to the number of stations in each section, we make sure that all areas of the Netherlands get some coverage from the KNMI. In this way, lone weather stations contribute more to the total objective function.

A very obvious extension of some of the models discussed in this thesis is adding budget constraints. Instead of a cardinality constraint, where a predetermined number of sets are to be chosen, we then assign costs to each set (and possibly to each element). The total cost must then adhere to the given budget. As discussed in section 2.1 and 2.2, both WMC and SDW-MC already have known budgeted versions: BMC and GMC, respectively. However, SDW-MMC and PDW-MMC do not. This is nicely illustrated in Figure 1.1. By including a budget, we improve the applicability of both models, as it not only allows us to model true financial costs, but also costs in terms of power consumption or resources [3].

In subsection 2.3.3 it was proved that Algorithm 3 gives a $(1 - \frac{1}{e})$-approximation of any instance of SDW-MMC. In [5] no explicit algorithm was found that could beat this bound, but an implicit algorithm was developed with a higher approximation ratio of $(1 - b^b e^{-b}/b!)$, taking advantage of the fact that elements can be covered multiple times. This algorithm first solved the *LP-relaxation* of the problem and then made use of a rounding method called *pipage rounding*. In this thesis, our focus lies on explicit algorithms and so this was of no immediate use. However, the result from [5] suggests their algorithm might have an explicit counterpart, which would improve Algorithm 3.

The PDW-MMC problem from subsection 2.4 is an extension of SDW-MMC. In a very recent, not yet published article by Barman, Fawzi, and Fermé [4], a different extension was introduced. Instead of simply adding up the weights if an element is covered multiple times, the weights are multiplied by a concave function, depending on how many times the element is covered. Although this problem not immediately applies to the setting of weather stations, it might work better than our proposed PDW-MMC for other problems. Therefore, this article must certainly be explored further.

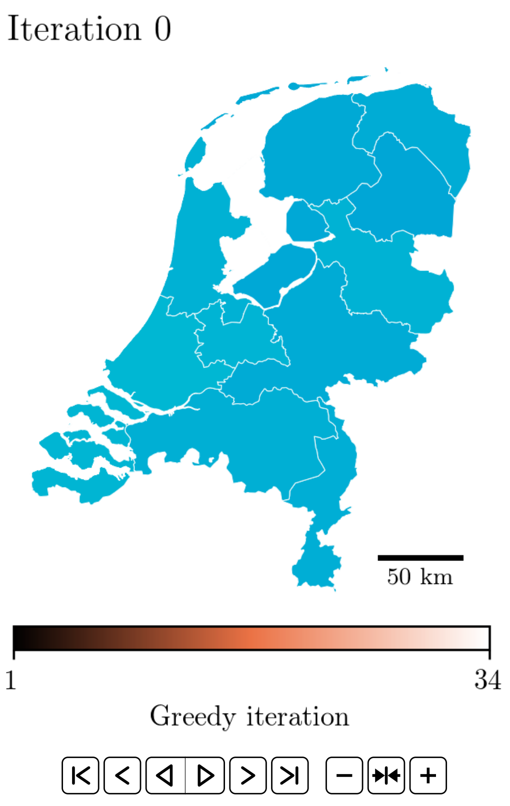
A negative property of PDW-MMC is that its greedy algorithm only has a performance guarantee under the assumptions that the weights are increasing and submodular. In a recent article by Chamon, Pappas, and Ribeiro [6], results were derived for *approximately submodular* functions. A function is approximately submodular, if it violates true submodularity only up to an additive or multiplicative constant. Although in general this does not influence our approximation bounds, for specific weight functions these notions might provide us with bounds, even if the assumption of submodularity does not hold. This certainly is an area worth investigating.

We now turn our attention to the calculation of weights in Chapter 3. We used the R^2 as a measure of fit and proved that it is not submodular. In [12] it is shown that under certain assumptions a different objective, the logarithm of the determinant of the covariance matrix, is supermodular. Chamon, Pappas, and Ribeiro [6] described this so-called logdet as a “supermodular surrogate” of the mean squared error, implying that it also is a measure of fit. Due to the supermodularity, this might be an interesting area of research.

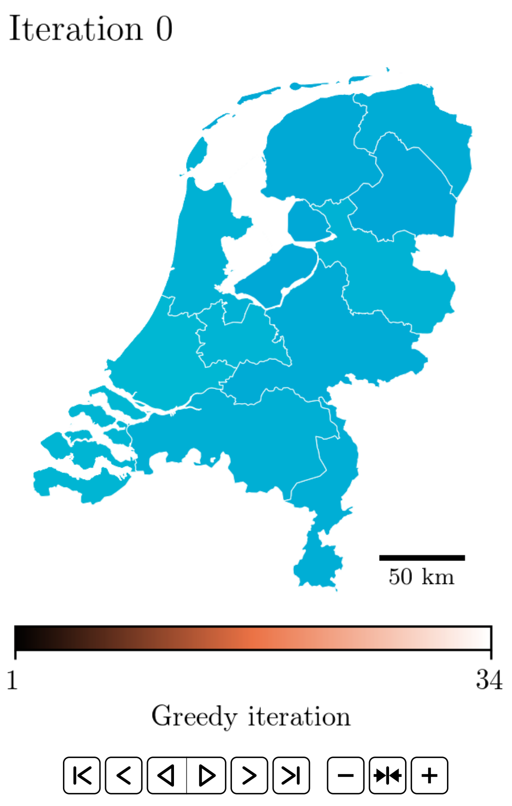
Appendix A

Animations of greedy selection

This appendix contains the same results as in Figures 3.4 and 3.7, which showed the selection process of Algorithm 4 applied to weather stations in the Netherlands. To better visualise this process, animated versions of these figures are depicted in Figure A.1. Figures A.1a–c correspond to Figure 3.4, which showed the results for the IDW-weights. Figures A.1d–f show the results for the MLR-weights and correspond to Figure 3.7. These animations can only be viewed digitally in `Adobe Acrobat Reader`[1] or `Foxit Reader`[2].



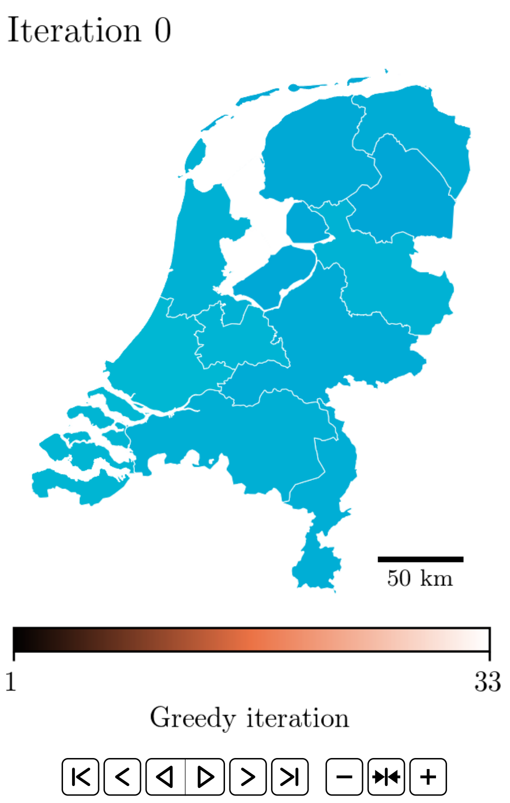
(a) IDW-weights – Temperature



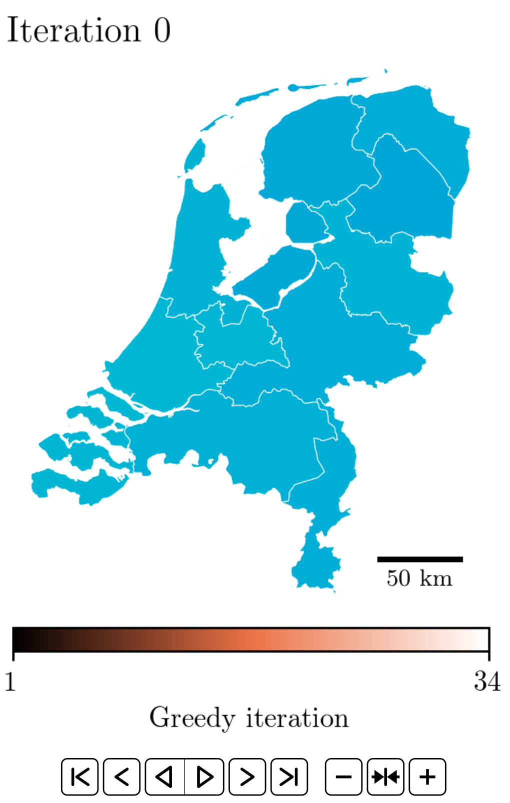
(b) IDW-weights – Humidity

Figure A.1: Animation of selection of weather stations by Algorithm 4 with MLR- and IDW-weights. All three quantities of interest are depicted.

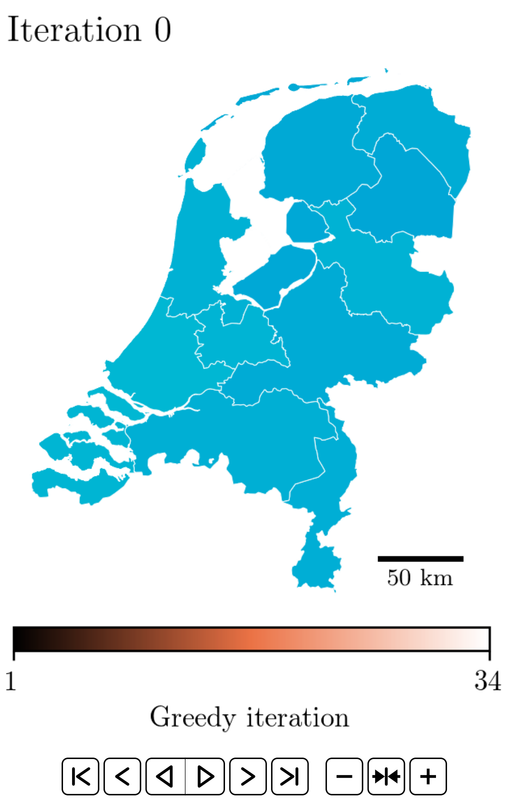
[1] `https://get.adobe.com/nl/reader/`
[2] `https://www.foxit.com/pdf-reader/`



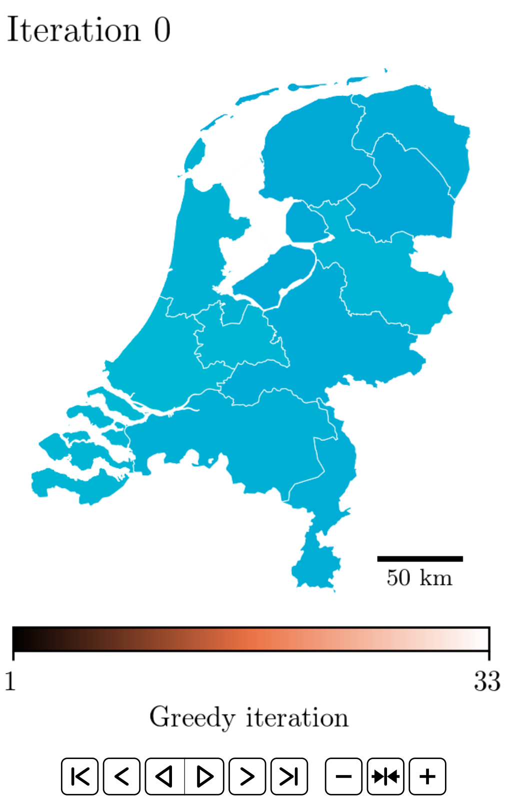
(c) IDW-weights – Precipitation



(d) MLR-weights – Temperature



(e) MLR-weights – Humidity



(f) MLR-weights – Precipitation

Figure A.1: Animation of selection of weather stations for different quantities by Algorithm 4 with IDW- and MLR-weights. All three quantities of interest are depicted.

References

[1] K. Aardal, L. Van Iersel, and R. Janssen, *Lecture notes am2020 optimization*, Lecture Notes, 2020.

[2] C. C. Aggarwal, A. Bar-Noy, and S. Shamoun, "On sensor selection in linked information networks," in *2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)*, IEEE, 2011, pp. 1–8.

[3] C. C. Aggarwal, A. Bar-Noy, and S. Shamoun, "On sensor selection in linked information networks," *Computer Networks*, vol. 126, pp. 100–113, 2017.

[4] S. Barman, O. Fawzi, and P. Fermé, "Tight approximation guarantees for concave coverage problems," *arXiv preprint arXiv:2010.00970*, 2020.

[5] S. Barman, O. Fawzi, S. Ghoshal, and E. Gürpınar, "Tight approximation bounds for maximum multi-coverage," in *International Conference on Integer Programming and Combinatorial Optimization*, Springer, 2020, pp. 66–77.

[6] L. F. Chamon, G. J. Pappas, and A. Ribeiro, "Approximate supermodularity of kalman filter sensor selection," *IEEE Transactions on Automatic Control*, vol. 66, no. 1, pp. 49–63, 2020.

[7] F.-W. Chen and C.-W. Liu, "Estimation of the spatial rainfall distribution using inverse distance weighting (idw) in the middle of taiwan," *Paddy and Water Environment*, vol. 10, no. 3, pp. 209–222, 2012.

[8] R. Cohen and L. Katzir, "The generalized maximum coverage problem," *Information Processing Letters*, vol. 108, no. 1, pp. 15–22, 2008.

[9] U. Feige, "A threshold of ln n for approximating set cover," *Journal of the ACM (JACM)*, vol. 45, no. 4, pp. 634–652, 1998.

[10] C. Heij, P. De Boer, P. Franses, T. Kloek, and H. Van Dijk, *Econometric Methods with Applications in Business and Economics*. Oxford University Press, Jan. 2004.

[11] D. S. Hochbaum and A. Pathria, "Analysis of the greedy approach in problems of maximum k-coverage," *Naval Research Logistics (NRL)*, vol. 45, no. 6, pp. 615–627, 1998.

[12] S. T. Jawaid and S. L. Smith, "Submodularity and greedy algorithms in sensor scheduling for linear dynamical systems," *Automatica*, vol. 61, pp. 282–288, 2015.

[13] S. Khuller, A. Moss, and J. S. Naor, "The budgeted maximum coverage problem," *Information processing letters*, vol. 70, no. 1, pp. 39–45, 1999.

[14] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions—i," *Mathematical programming*, vol. 14, no. 1, pp. 265–294, 1978.

[15] "Sensor," in *Cambridge Dictionary*, Cambridge University Press, 2021. [Online]. Available: https://dictionary.cambridge.org/dictionary/english/sensor (visited on 07/05/2021).

[16] D. Shepard, “A two-dimensional interpolation function for irregularly-spaced data,” in *Proceedings of the 1968 23rd ACM national conference*, 1968, pp. 517–524.

[17] R. Sluiter, “Interpolation methods for climate data: Literature review,” *KNMI intern rapport, Royal Netherlands Meteorological Institute, De Bilt*, 2009.

[18] R. V. Vohra and N. G. Hall, “A probabilistic analysis of the maximal covering location problem,” *Discrete Applied Mathematics*, vol. 43, no. 2, pp. 175–183, 1993.