Using Lampposts to Provide Urban Areas with Multiple Services

Master's Thesis
Econometrics and Management Science
Operations Research and Quantitative Logistics

T.J.C. Vos*

Supervisors:
Dr. W. van den Heuvel & Dr. F. Phillipson

Co-reader:
Dr. D. Huisman

Erasmus School of Economics

August 20, 2016

*Email: vostjc@gmail.com

Abstract

In this thesis the goal is to find an efficient method which is able to assign services to each of the potential lampposts in a city such that all services are distributed adequately (i.e. the coverage requirements are met), against minimal costs. A two-step approach is proposed to find a smart and cost efficient distribution. In the first step the services are distributed over the available lampposts while taking into account the required coverage and the costs for enabling a lamppost to be equipped with services. A formulation of this problem, which is termed the Multi-Service Location Set Covering Problem, is presented. Several heuristic methods are proposed to solve this problem. A heuristic method based on sequentially solving a Set Covering Problem for each of the services is shown to obtain the best results. The quality of this heuristic can in part be attributed to an efficient method which is implemented to solve Set Covering Problems. The selected locations from the first step are then taken as input to the second step, which can be seen as a cost reduction step. In this step lampposts are selected which are enabled to serve as a hub to at most a certain amount of lampposts which are within a specified range. For this problem, termed the Wireless Network Problem a formulation is presented. To solve the Wireless Network Problem several metaheuristics are proposed. It is shown that the Iterated Local Search metaheuristic is able to find the best solutions. The best solution methods are used in the two-step approach to provide cities with multiple services.

Keywords: Set Covering, Facility Location, Capacitated Vertex Cover, Multi-Service, Wireless Network.

Preface

This thesis is the culmination of my education at the Erasmus University Rotterdam and is submitted as a partial fulfillment of the requirements for the degree of Master of Science in Econometrics and Management Science. The inspiration for the subject of this thesis originates from Dr. Frank Phillipson who is affiliated with TNO and is active in the field of ICT/telecom and electricity network planning. This thesis has therefore also been written at TNO under the supervision of Frank and I would like to thank him for his guidance and useful insights.

I also want to thank Dr. Wilco van den Heuvel who has been my supervisor from the Erasmus University Rotterdam. Without exception Wilco's observations and recommendations have found their way into this thesis in some form or another.

Tim Vos
Rotterdam, August 2016

List of Abbreviations

BKS	Best Known Solution
CFLP	Capacitated Facility Location Problem
CVC	Capacitated Vertex Cover
GA	Genetic Algorithm
GRASP	Greedy Randomized Adaptive Search Procedure
ILS	Iterated Local Search
IoT	Internet of Things
LB	Lower Bound
LS	Local Search
LLBP	Lagrangian Lower Bound Problem
LSCP	Location Set Covering Problem
LUBP	Lagrangian Upper Bound Problem
MCLP	Maximal Covering Location Problem
MCSC	Maximal Covering Location Problem with Survivability Constraints
MSLSCP	Multi-Service Location Set Covering Problem
SA	Simulated Annealing
SCP	Set Covering Problem
SSC	Sequential Set Covering
UFLP	Uncapacitated Facility Location Problem
WNP	Wireless Network Problem

Contents

1 Introduction

In this section an introduction will be given to the problem which is considered in this thesis. Some background information is presented which constitutes the inspiration for this research. Subsequently the difficulties inherent to the problem will be discussed and an approach to solve the problem will be proposed. Finally, the goal of this research is summarized and a structure of the thesis will be presented.

1.1 Problem Setting

In our society the information density is condensing more and more. Not only the need for receiving information is increasing, but also the need for processing information gathered by, for example, sensors is increasing. More condensed networks are required to be able to satisfy these increasing needs and to process all this data in an efficient way.

The mobile telecom industry can be taken as an example of the described trend. Currently, there are base stations covering areas of several hundreds or thousands of meters. With the increase in mobile communications the area covered by a base station should decrease, which in effect condenses the network.

A benefit of dense networks is that higher transmission rates can be reached as the distance to a connection point is smaller. This is especially interesting to Internet Service Providers. Nowadays internet connectivity for houses is generally provided through a wired network, which has the drawback of cumbersome maintenance and costly upgrades. A dense wireless network which is able to provide internet connectivity of the same or improved quality might be a cost efficient alternative.

Another development is the maturation of the Internet of Things (IoT). The IoT can be seen as a network of devices which are able to collect and exchange information. The thermostat in a house can be able to measure the current temperature of a room and use this measurement to make a decision on whether to heat the room or not. Jewelry might be equipped with a GPS tracker which exchanges its location allowing it to be retrieved when lost. More advanced implementations of the IoT concern devices which are able to make autonomous decisions based on information received from other devices. A large part of the exchange of information will be wireless, which the infrastructure should be prepared for.

A network should be easily accessible to be useful. For example, for wireless connections to replace wired connections, access points are required at close range. Any network should also be able to offer reliable connections. Additionally the network should cover the locations from where connectivity is requested.

A successful network design meets these requirements while preferably minimizing the costs of setting up the network.

It is interesting to investigate whether any existing infrastructure can be utilized to save on the costs of setting up a network. Existing infrastructure is useful for this purpose when it is able to meet the requirements of the network design. An interesting infrastructure already in place which is able to provide the required density and which can be found in places where a large part of the population is located are lampposts. Lampposts will be taken as a starting point for existing infrastructure in this research.

Lampposts have several additional properties which make them a useful starting point for this research. For one they are managed by the local or central government who might be interested in opening up the lampposts to additional functionalities, especially when these are in the public interest. Lampposts also are already provided with a power supply and their locations are likely to be well documented. Moreover, everybody will agree on the necessity of lampposts so making them even more useful is worth investigating.

Apart from internet connectivity there might be several other services which might be provided using lampposts as a base location. Mainly sensory equipment is interesting in this case. A lamppost can for example be equipped with a motion detector, such that it can be switched on only when people are within proximity of the lamppost, which saves on energy usage. Lampposts might be able to recognize traffic jams which can be communicated with smart vehicles such that they are rerouted, escaping the traffic jam. It might even be possible to equip lampposts with an alarm for more accurate warnings in case of an emergency. These applications might assist in the transition to ‘smart cities’ for which the ultimate goal is to improve the quality of life in a city.

1.2 Base Problem

The base problem considered in this thesis is coming up with a network design which specifies the way in which the services should be distributed over the lampposts such that coverage is provided against minimal costs.

There are several difficulties involved with the distribution of the services over the lampposts. Not all services have the same coverage area. Some of the services might have a larger coverage area than other services. Also, the points which require coverage can differ per service. This indicates that not every lamppost has to be equipped with all of the services. Rather, a smart distribution is preferred. For each separate service the best distribution can be reached by optimizing the distribution regardless of the distribution of the other services. However, this need not lead to the least costly network design due to an additional complexity.

For all of the services to be provided it is required that there is a connection to an existing network. For example internet connectivity can only be provided when the lamppost has internet connectivity. Also, the measurements from the sensors should be exchanged through such a network.

It is inefficient to connect all lampposts to an existing network as this might turn out to be very costly. It is more efficient to identify locations which should be equipped with services first and to connect these identified locations later. The costs of connecting a lamppost to an existing network are assumed to be fixed, independent of the number of services which a lamppost is equipped with. This directly contradicts the individual distribution of the services as now services are preferably grouped together.

The right balance has to be found between both the goals of an individual distribution of the services over the lampposts and the grouping together of services on the lampposts. To achieve this it is best to make a distribution for all services simultaneously.

1.3 Extension on the Base Problem

In the base problem it is assumed that all lampposts should be connected to an existing network before they can be equipped with services. When solving the base problem a selection from the initial lampposts will be made such that coverage can be provided for the services from the selected lampposts.

Even though not all initial lampposts are to be connected to an existing network it might still turn out to be a very costly endeavor to connect only the selected lampposts. This makes it interesting to investigate whether it is possible to connect only a small part of the lampposts to an existing network and subsequently letting these connected lampposts act as a hub to other lampposts. This way all selected lampposts can be equipped with services such that coverage is provided.

The connection between a lamppost designated as a hub and another lamppost will be realized wirelessly. From the nature of wireless communication it follows that there is a limitation on the distance within which the wireless communication can exist. This limitation should be taken into account when deciding on which lampposts to designate as a hub and which lampposts to connect to the hub.

Connecting lampposts to a hub is no straightforward task. When a lamppost is designated as a hub then this means that not only does the hub have to process its own traffic, but also the traffic of the other lampposts which are connected to the hub. Potential capacity limitations may impose a restriction on the number of lampposts which can be connected to a hub lamppost.

The combined restriction on the maximum distance of wireless communications and the capacity make this problem difficult to solve. Consider for example the situation in which there are ten lampposts within range of a lamppost designated as a hub which due to its capacity can be connected to at most four other lampposts. Then a choice has to made on either connecting zero, one, two, three or four lampposts to the hub as well as which of the lampposts out of the ten these should be. Furthermore, the wireless connection of a lamppost to a hub is assumed to be free, so there is no clear distinction as to which lampposts to connect to the hub.

Now consider doing this for not one hub, but for a much larger amount of hub lampposts. From the selected lampposts a designation of hubs has to be made and additionally a choice on which lampposts to connect to each of the hubs. This all has to be done while taking the restrictions into account and designating hubs in such a way that the costs are minimized.

1.4 Two-Step Approach

It would be best to simultaneously determine a distribution of the services and a selection from the lampposts which are to act as a hub and the respective connections. However, as these problems are already rather difficult to solve on their own, it is chosen to use a two-step approach to arrive at a sufficiently good network design.

In the first step the problem is solved where the services are to be distributed over the lampposts such that coverage is provided and assuming that all lampposts need to be connected to an existing network. This results in a selection from all the available lampposts to only the necessary lampposts to provide coverage. The selected lampposts from the first problem are then taken as input to the second step. In this step it will be attempted to identify lampposts which should act as a hub to other lampposts. Ultimately this will result in a network design where services are distributed efficiently and all locations are connected to an existing network either directly or through another lamppost.

An impression of a final solution resulting from the two-step approach can be sketched as in Figure 1.1. In this figure it is displayed that some lampposts are connected to an existing network, while others are connected through another lamppost. A connection is present for all lampposts which are equipped with services. Additionally it can be seen that not all existing lampposts need necessarily be used in a final solution.

1.5 Goal of this Research

At this point in time the transition to smart cities is still in its infancy. In this research it is intended to prepare for further advances in this area. It is investigated which problems need to be addressed in order to be able to set up a network of services using existing infrastructure. This is done by considering multiple use cases where several services are distributed over the lampposts in a city. The services are distributed in such a way that they provide adequate coverage against the least amount of costs.

The goal of this research is summarized as follows.

Find an efficient method which is able to assign services to each of the potential lampposts in a city such that all services are distributed adequately (i.e. the coverage requirements are met), against minimal costs.

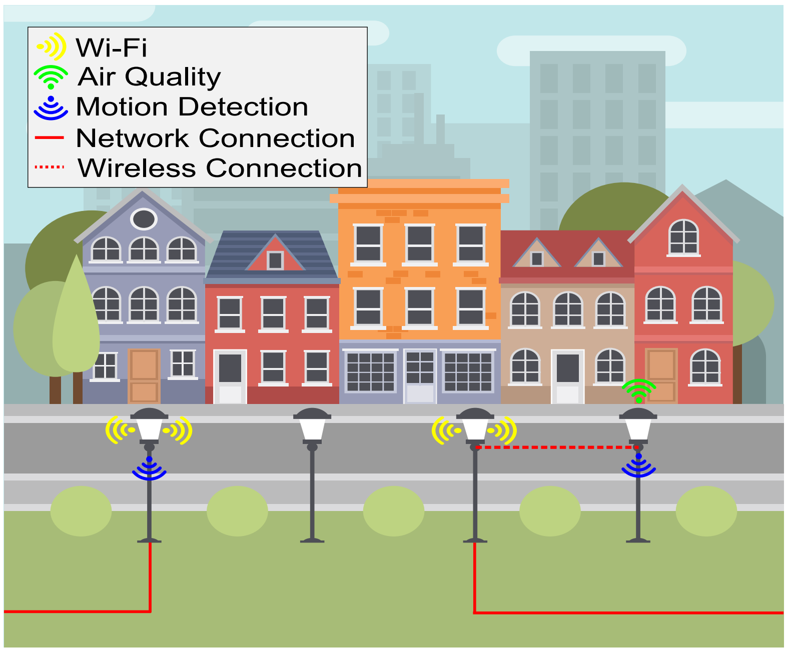

Figure 1.1: Sketch of outcome.

Both problems in the two-step approach will follow a similar definition format. First the problem is mathematically formulated and based on this formulation several solution methods will be devised. These solutions methods will then be compared to see which yields the best results for the specific problems. The models are general descriptions of the problems described and might even be usable for other problems of a similar nature.

1.6 Structure of the Thesis

The remainder of this thesis consists of the following sections. First, in section 2, the mathematical formulation for each of the problems in the two-step approach will be presented. In section 3 a literature review will be given of the Facility Location Problem, the Set Covering Problem and the Capacitated Vertex Cover, as these are the problems related to the formulations in this research. In section 4 several solution methods are proposed specifically designed to solve either the first or the second step in the two-step approach. One of the methods proposed in this section is based on repeatedly solving a Set Covering Problem. To quickly solve Set Covering Problems a method has been implemented which is described in section 5. In this thesis the two-step approach is applied to the problem where a set of services should be distributed over the lampposts in a city such that adequate coverage is provided. The design of the test instances for this problem is discussed in section 6. In section 7 computational results for each of the proposed solution methods on the test instances will be presented. Finally, in section 8 there will be some concluding remarks and suggestions for future research.

2 Problem Formulations

In the two-step approach two distinct problems are considered. First there is the problem where the services should be distributed over the lampposts to provide coverage. Secondly, several lampposts should be designated as a hub to other lampposts to save on the costs required to enable a lamppost to be equipped with services. For both these problems a mathematical formulation is presented in this section.

2.1 The Multi-Service Location Set Covering Problem

The first step in the two-step approach is the distribution of the services over the lampposts such that coverage is provided against minimal costs. The introduced mathematical formulation is termed the Multi-Service Location Set Covering Problem (MSLSCP).

In the MSLSCP the problem is to distribute a set of services $\mathcal{F} = \{1, \dots, F\}$ over a set of locations $\mathcal{L} = \{1, \dots, L\}$ such that the demand points in $\mathcal{G}^u = \{g^u, \dots, G^u\}$ for each service $u \in \mathcal{F}$ are covered against minimal costs. There are fixed costs $f_j > 0$ associated with enabling a location $j \in \mathcal{L}$ to be equipped with services. For the problem considered in this research this refers to connecting a lamppost to an existing network. Consequently, there are costs $c_j^u > 0$ associated with equipping the enabled location $j \in \mathcal{L}$ with service $u \in \mathcal{F}$. The information on whether a demand point $i \in \mathcal{G}^u$ can be covered from location $j \in \mathcal{L}$ for service $u \in \mathcal{F}$ is stored in the binary a_{ij}^u matrix. In this matrix $a_{ij}^u = 1$ when demand point $i \in \mathcal{G}^u$ can be covered from location $j \in \mathcal{L}$ for service $u \in \mathcal{F}$ and $a_{ij}^u = 0$ otherwise. The mathematical formulation is as follows.

$$\text{Min} \sum_{j \in \mathcal{L}} \sum_{u \in \mathcal{F}} c_j^u x_j^u + \sum_{j \in \mathcal{L}} f_j y_j \tag{2.1}$$

$$\text{s.t.} \sum_{j \in \mathcal{L}} a_{ij}^u x_j^u \geq 1 \qquad \forall i \in \mathcal{G}^u, u \in \mathcal{F} \tag{2.2}$$

$$x_j^u \leq y_j \qquad \forall j \in \mathcal{L}, u \in \mathcal{F} \tag{2.3}$$

$$x_j^u \in \{0, 1\} \qquad \forall j \in \mathcal{L}, u \in \mathcal{F} \tag{2.4}$$

$$y_j \in \{0, 1\} \qquad \forall j \in \mathcal{L} \tag{2.5}$$

Costs f_j are incurred when the decision is made to enable location $j \in \mathcal{L}$ to be equipped with services. These are so-called ‘connection costs’. Subsequently costs c_j^u are incurred by determining with which services $u \in \mathcal{F}$ location $j \in \mathcal{L}$

should be equipped, so-called ‘equipment costs’. The goal is to minimize the total costs (2.1), represented as the sum of equipment costs and connection costs. The objective function is minimized while satisfying all constraints. Coverage requirements are represented by constraint set (2.2). A demand point $i \in \mathcal{G}^u$ should be covered by at least one location for service $u \in \mathcal{F}$. Constraint set (2.3) represents the restriction that a location can only be equipped with services when it is enabled to be equipped with services. Finally, the decision variables are binary. In the formulation $y_j = 1$ if it chosen to enable location $j \in \mathcal{L}$ to be equipped with services and $y_j = 0$ otherwise. When an enabled location $j \in \mathcal{L}$ is equipped with service $u \in \mathcal{F}$, $x_j^u = 1$ and $x_j^u = 0$ otherwise.

The formulation reduces to a Set Covering Problem when the number of services which is considered is equal to one. This is true as no location will be enabled to be equipped with services when it is not equipped with the sole service which is considered. From this follows that the decision for y_j and x_j^u is the same. Hence, a valid formulation when considering only one service is taking the costs of a location to be $c_j^u + f_j$ and finding a subset of locations such that the total costs are minimized while all demand points are covered.

This leads to the conclusion that the MSLSCP is at least as hard as the Set Covering Problem. The complexity of the Set Covering Problem is known to be NP-hard from Bernhard et al. [9] and thus the complexity of the MSLSCP is also NP-hard.

2.2 The Wireless Network Problem

In the second step of the two-step approach the goal is to identify lampposts which are to act as a hub for other lampposts. In this way each of the lampposts has internet connectivity, either directly or through another lamppost. This allows all selected lampposts to be equipped with services while not all have to be connected to an existing network.

Some constraints should be taken into account when implementing this extension. Namely, lampposts are unlikely to be able to communicate with one another when the range between them is too large. In this case a wireless connection between the two lampposts can not exist. Another constraint is that a lamppost which services as a hub for other lampposts is subject to bandwidth restrictions. This can be expressed by a maximum number of lampposts which are able to communicate with the lamppost acting as a hub.

Again, let f_j be the costs of connecting lamppost $j \in \mathcal{L}$ to an existing network. Each lamppost $j \in \mathcal{L}$ which is connected to an existing network can service as a hub for at most k_j other lampposts, the so-called capacity. Between each lamppost $i \in \mathcal{L}$ and lamppost $j \in \mathcal{L}$ a distance can be calculated. When the calculated distance is less than the range of wireless communication, a wireless connection can exist between the lampposts. This information is represented by the parameter a_{ij} for which $a_{ij} = 1$ when lamppost $i \in \mathcal{L}$ is within range of lamppost $j \in \mathcal{L}$ and $a_{ij} = 0$ otherwise. When each lamppost has the same range, the a_{ij} matrix is symmetric.

In the mathematical formulation, termed the Wireless Network Problem (WNP), a choice should be made on which lampposts to use as a hub for other lampposts and which lampposts are then connected to the hub. The choice to connect a lamppost to an existing network is indicated by the decision variable y_j which is set to 1 when lamppost $j \in \mathcal{L}$ is connected to an existing network and to 0 otherwise. The choice to have lamppost $j \in \mathcal{L}$ act as a hub for lamppost $i \in \mathcal{L}$ is made by setting $x_{ij} = 1$ and $x_{ij} = 0$, otherwise.

$$\begin{aligned}
\text{Min} \quad & \sum_{j \in \mathcal{L}} f_j y_j & & \qquad (2.6)\\
\text{s.t.} \quad & \sum_{i \in \mathcal{L}} x_{ji} + y_j = 1 & \forall j \in \mathcal{L} & \qquad (2.7)\\
& \sum_{i \in \mathcal{L}} x_{ij} \leq k_j y_j & \forall j \in \mathcal{L} & \qquad (2.8)\\
& x_{ij} \leq a_{ij} y_j & \forall i, j \in \mathcal{L} & \qquad (2.9)\\
& x_{ij} \in \{0,1\} & \forall i, j \in \mathcal{L} & \qquad (2.10)\\
& y_j \in \{0,1\} & j \in \mathcal{L} & \qquad (2.11)
\end{aligned}$$

The goal is to minimize the total costs of connecting lampposts to an existing network (2.6) while taking the following into account. First, each lamppost should be either connected to a lamppost which is connected to an existing network, or it should be connected to an existing network directly (2.7). Secondly, constraint set (2.8) states that a connected lamppost can service as a hub to at most a pre-specified number of other lampposts. The constraint set (2.9) represents the requirement that a connection between lampposts can only exist when they are in range of one another. Finally, the decision variables are binary.

3 Literature Review

For the problem formulations which are proposed in this research there are several problems in the literature which should be discussed. For the MSLSCP these are the Facility Location Problems and the Set Covering Problem. For the WNP the interesting problems in the literature are the Capacitated Vertex Cover and the Single-Source Capacitated Facility Location Problem. In this section a literature review on these problems will be presented and the similarities as well as the differences between the problems and the models in this research will be discussed.

3.1 Facility Location Problems

Facility Location Problems are typically applied when a decision has to be made on where to locate a set of facilities, such as industry plants or warehouses, such that the demand for a certain commodity can be satisfied against minimal costs. Generally, fixed costs are associated with placing a facility at a certain location and transportation costs are incurred when servicing a client from a facility. Usually, the transportation costs are assumed to be metric, which means they are symmetric and satisfy the triangle inequality.

The Facility Location Problem is an extensively studied problem for which mainly two versions are distinguished. There is the Uncapacitated Facility Location Problem (UFLP) [23] and the Capacitated Facility Location Problem (CFLP) [49]. The CFLP is defined as follows.

Given is a set $\mathcal{D} = \{1, \dots, D\}$ of demand locations which need to be serviced and a set $\mathcal{F} = \{1, \dots, F\}$ of locations such that opening a facility at location $j \in \mathcal{F}$ incurs cost $f_j > 0$. Servicing customer $i \in \mathcal{D}$ from the opened facility $j \in \mathcal{F}$ incurs a transportation cost of $c_{ij} > 0$. The objective is to open a subset of facilities in $\mathcal{F}$ and satisfy all demand from this subset of facilities such that the total costs are minimized and all demand is satisfied. The mathematical formulation of this problem is as follows.

$$\text{Min} \sum_{i\in\mathcal{D}}\sum_{j\in\mathcal{F}} c_{ij}x_{ij} + \sum_{j\in\mathcal{F}} f_j y_j \tag{3.1}$$

$$\text{s.t.} \sum_{j\in\mathcal{F}} x_{ij} = 1 \qquad \forall i \in \mathcal{D} \tag{3.2}$$

$$\sum_{i\in\mathcal{D}} d_i x_{ij} \leq s_j y_j \qquad \forall j \in \mathcal{F} \tag{3.3}$$

$$0 \leq x_{ij} \leq 1 \qquad \forall i \in \mathcal{D}, j \in \mathcal{F} \tag{3.4}$$

$$y_j \in \{0,1\} \qquad \forall j \in \mathcal{F} \tag{3.5}$$

In this formulation the demand of customer $i \in \mathcal{D}$ is represented by d_i and the capacity of facility $j \in \mathcal{F}$, represented by s_j, may not be exceeded. The difference between the UFLP and the CFLP is the fact that for the UFLP there is no restriction on the throughput of a facility. The mathematical formulation of the UFL can easily be derived from the mathematical formulation of CFLP by setting the capacity s_j in constraint set (3.3) sufficiently large. Both the UFLP and the CFLP have been found to be NP-hard by Rosenwein [50].

For the MSLSCP there also is a fixed cost associated with enabling a location to be equipped with services. This makes literature on Facility Location Problems interesting as these might inspire solution methods.

Due to the NP-hardness of the Facility Location Problems, a lot of work has been done to come up with good approximation algorithms. An approximation algorithm is an algorithm which returns in polynomial time a solution which is at most a known factor larger than the optimal solution. Shmoys et al. [51] were the first to arrive at a constant factor approximation algorithm for the UFLP. Recently, an approximation algorithm was found of factor 1.448 by Li [42]. In between these two results many other approximation algorithms have been proposed. One of those is presented in the paper by Mahdian et al. [44] in which the authors come up with an approximation algorithm for both the UFLP and CFLP. They propose an algorithm which is used to achieve a 1.52-approximation for the UFLP and a 2-approximation for the CFLP. An overview of other solution methods for the CFLP is presented by Sridharan [52].

For many applications of Facility Location Problems in general, the reader is referred to the paper by Hamacher and Drezner [34].

A problem showing resemblance to the WNP is the SSCFLP. The SSCFLP is an extension of the CFLP. In the SSCFLP the variable x_{ij} is restricted to be binary, whereas in the CFLP it can be any number in the interval [0,1]. This means that in the SSCFLP all demand of a customer has to be satisfied from exactly one facility. To the formulation of the SSCFLP sometimes the constraint (3.6) is added. From Yang et al. [56] it follows that this results in a tighter linear relaxation.

$$x_{ij} \leq y_j \qquad \forall i \in \mathcal{D}, j \in \mathcal{L} \tag{3.6}$$

When x_{ij} is chosen freely in the interval [0,1] this means that, given a set of opened facilities, assigning customers to these facilities in the CFLP results in a LP-formulation. This is easily solved to optimality using the simplex method. Given a set of opened facilities in the SSCFLP, where x_{ij} is restricted to be binary, assigning customers to a set of opened facilities results in the NP-hard Generalized Assignment Problem (see Fisher et al. [28]). This makes the SS-CFLP significantly harder to solve.

The SSCFLP can be linked to the WNP as follows. Let the set of customers and locations be the same set $\mathcal{L}$. Let f_j be the costs of connecting location $j \in \mathcal{L}$ to an existing network. The parameter c_{ij} represents the costs of wirelessly connecting location $i \in \mathcal{L}$ to location $j \in \mathcal{L}$. These costs can either be $c_{ij} = 0$ when location $i \in \mathcal{L}$ is within range of location $j \in \mathcal{L}$ or $c_{ij} = f_i + M$ otherwise. Here M is a sufficiently large number. This ensures that, in an optimal solution, it is always more efficient to open a new location than to connect a location which is out of range, which should not occur.

As the SSCFLP is a NP-hard problem most solution methods have focused on finding good approximations of an optimal solution. Among the heuristics Lagrangian relaxation is used most often. The applications of Lagrangian relaxation to the SSCFLP differ in which constraint is relaxed. A choice can be made to either relax the assignment constraint (3.2), the capacity constraint (3.3) or both. Additionally, Lagrangian relaxation methods differ in the way a feasible solution is generated from the solution to the relaxed problem.

One of the first to propose a Lagrangian relaxation were Klincewicz and Luss [40]. In their approach they relax the capacity constraint. The resulting problem now is a UFLP, which they solve using a dual ascent algorithm. The solution to the relaxed problem is then made feasible by using a so-called add heuristic. A final adjustment heuristic is used to improve the best solution generated.

An early Lagrangian relaxation of the assignment constraint is presented by Barceló and Casanovas [3]. Relaxing this constraint results in several knapsack subproblems which are solved to obtain feasible solutions. Hindi and Pieńkosz [35] also relax the assignment constraint. They combine a greedy heuristic and restricted neighborhood search to obtain feasibility.

A Lagrangian relaxation of both the assignment and capacity constraints is presented by Beasley [7]. In the paper extensive computational results can be found for the Lagrangian relaxations.

Chen and Ting [16] implement a Multiple Ant Colony System as well as a hybrid algorithm of both Lagrangian relaxation and an Ant Colony System to solve the SSCFLP.

Recently several heuristic methods have been proposed which use methods other than Lagrangian relaxation to obtain solutions to the SSCFLP. In Ahuja et al. [1] a Very Largescale Neighborhood Search is applied to the SSCFLP. Contreras and Díaz [20] use scatter search to solve the SSCFLP. Ho [36] presents a simple iterated tabu search algorithm, which is shown to result in high quality solutions while requiring little computational effort. Guastaroba and Speranza [32] extend the Kernel Search heuristic to general Binary Integer Linear Programming problems. They show the effectiveness of the extended Kernel Search heuristic by applying it to the SSCFLP.

Several exact solution approaches have been proposed as well. Holmberg et al. [38] propose a branch and bound method where in each iteration they calculate a lower bound using a Lagrangian heuristic and an upper bound using a strong primal heuristic. Diaz and Fernández [22] implement a column generation procedure in a branch and price framework. The column generation procedure is used to find upper and lower bounds. The most recent exact solution approach is proposed by Yang et al. [56]. They present a cut-and-solve based algorithm. In each iteration of the algorithm two problems are considered. A sparse problem with a limited solution space is solved using a commercial MIP solver to obtain an upper bound. In the dense problem a linear relaxation is solved to obtain a lower bound. The obtained lower bound is subsequently strengthened using a cutting plane method.

When the WNP is formulated as a SSCFLP only for optimal solutions a guarantee of feasibility to the WNP can be given. However, as the SSCFLP is known to be an NP-hard problem it might be particularly difficult to obtain an optimal solution. Hence, in some cases no guarantee of feasibility to the WNP can be given.

3.2 Set Covering Problems

Set Covering Problems arise in various practical applications, among which are crew scheduling, vehicle routing and location problems. The Set Covering Problem (SCP) has been around for a long time [54].

Formally, the SCP is defined as follows. Let a_{ij} be a $M \times N$ binary matrix. Let $\mathcal{N} = \{1, \ldots, N\}$ be the set of columns and $\mathcal{M} = \{1, \ldots, M\}$ the set of rows. A column $j \in \mathcal{N}$ is said to cover row $i \in \mathcal{M}$ if $a_{ij} = 1$. For the SCP to be defined properly it is assumed that each row $i \in \mathcal{M}$ can be covered by at least one column $j \in \mathcal{N}$. Let $c_j > 0$ be the costs of column $j \in \mathcal{N}$. The goal of the SCP is to find a subset of columns $\mathcal{S} \subseteq \mathcal{N}$, such that the columns in $\mathcal{S}$ cover all rows in $\mathcal{M}$ against minimal costs. The mathematical formulation of the SCP is as follows.

$$\text{Min} \sum_{j \in \mathcal{N}} c_j x_j \tag{3.7}$$

$$\text{s.t.} \sum_{j \in \mathcal{N}} a_{ij} x_j \geq 1 \qquad \forall i \in \mathcal{M} \tag{3.8}$$

$$x_j \in \{0, 1\} \qquad \forall j \in \mathcal{N} \tag{3.9}$$

As has been noted the MSLSCP reduces to a SCP when there is only one service considered. Because of this literature on SCP and methods to solve them might inspire several solution methods.

The SCP is a NP-hard problem as can be seen from Garey and Johnson [31]. An early greedy heuristic for the SCP was presented by Chvatal [19]. Since then, many more solution methods have been presented.

Caprara et al. [13] compare the six most effective heuristic methods up to the release date of the paper. Each of the heuristics is tested on the test-bed instances of Beasley's OR Library [6]. Because each of the heuristics is tested on the same data sets a fair comparison can be made, but also the results can be compared to some known optimal solutions. In the article it is shown that the method by Caprara et al. [12] returns optimal, or the best known solutions, while requiring the least computational effort. In the paper problems were considered with up to 5,500 rows and 1,100,000 columns. Also some exact solution approaches are considered. However, on the smaller problems with up to 400 rows and 4,000 columns these take 50 times as long as the heuristics to come up with a solution.

There is also a more recent comparison of methods, found in Umetani and Yagiura [55]. The added value of this paper is the fact that it summarizes improvements on the methods used in Caprara et al. [13]. However, the authors present results from previous papers in which different computers were used. This means that the comparison of the results is not as fair as in Caprara et al. [13].

The Set Covering Problem has been found to have applications in many fields. For this research however, the application to location problems is most interesting. This set of problems is also referred to as the Location Set Covering Problem (LSCP). An early introduction in this field has been made by Toregas et al. [54]. In this paper, the authors used the SCP formulation to determine the best location for emergency service facilities. Facilities were positioned in such a way that all demand points were within a specified distance of a facility while minimizing the total costs of opening the facilities.

The original formulation of the LSCP has been around for decades and continues to show a lasting applicability to practical problems. Some recent applications found in the literature are optimizing watchtower locations for fire monitoring purposes [2], the planning of street lighting [47], location optimization of strategic alert sites for homeland defense [8] and locating collection areas for urban waste management [4].

Following the taxonomy Daskin [21] introduced for location models the LSCP can be classified as a covering-based, discrete, set covering model. For discrete location models there is a finite set of candidate locations and demand arises at known demand points. Covering-based refers to using an underlying distance metric to determine whether a demand point is within a specified critical distance and the set covering classification requires a minimal number of facilities such that all demand points are covered.

An extension to the LSCP has been given by Church and Velle [17]. They introduced the Maximal Covering Location Problem (MCLP). In the MCLP the goal is to cover as many demand points as possible with a fixed number of facilities.

In Farahani et al. [24] multiple reviews on covering problems are given, as well as a comprehensive review of models, solutions and applications. The authors divided the paper in a section on the SCP and the MCLP. For each of these problems a basic formulation, as well as many extensions are given. Also an overview is given of covering models which are not easily related to the SCP or the MCLP, even though they are somehow related to covering problems. The paper gives the most extensive overview of covering models introduced between 1997 and 2011. However, none of the models presented seem to cover the problem which is proposed in this research.

Lee and Murray [41] apply a covering model to a real life case study in which they want to supply the city of Dublin, Ohio with a citywide wireless broadband connection. In this paper an extension of the MCLP is proposed, namely the Maximal Covering Location Problem with Survivability Constraints (MCSC). The MCSC is used to cover as much of the population of the city as possible while ensuring a reliable network. Interesting insights can be obtained from the highlighted trade-off between coverage and costs.

Another case study has been performed by Bell et al. [8]. In this case study the authors determine aircraft alert sites for the defense of important national areas in the USA. First a LSCP is used to determine how many facilities are required and secondly the minimum aggregate network distance is found for this number of facilities. This is done by solving a p-median problem. Finally, the authors check the robustness of their solutions.

Also worth mentioning is the paper by Murray et al. [48]. The authors argue that the mathematical representation of a study region influences the outcomes of covering models. Historically, the demand points have been modeled as irregular data points corresponding to neighborhoods, towns, et cetera. Related to this is using a regular pattern of points to model a study region. The authors researched the possibility of coverage modeling as regions instead of points. In the paper it is shown that each mathematical representation of a study region has its pros and cons. Using data points inherently leads to coverage gaps, however it is computationally more efficient. Using regions to model a study region usually results in too many facilities being sited and a higher computational complexity, however it does result in better coverage. In the paper some theoretical results are given and additionally the authors show practical results using the real-world data set of Dublin, Ohio as an example.

3.3 Capacitated Vertex Cover

The Capacitated Vertex Cover (CVC) has first been proposed by Guha et al. [33]. A more general name for this problem is the Set Covering Problem with Hard Constraints. The Capacitated Vertex Cover is a generalization of the well-known Vertex Cover problem.

Given is a graph $G = (V, E)$. An edge $e = \{u, v\} \in E$ should be covered either by vertex $u \in V$ or $v \in V$. To each vertex $v \in V$ is assigned a weight $w(v)$, i.e. costs, a capacity $k(v)$, i.e. a maximum number of edges which can be covered, and $b(v)$, a maximum number of duplications. The objective is to cover all edges $e = \{u, v\} \in E$ using a multi-set $C \subseteq V$ of minimal weight. In a multi-set the same element can occur multiple times, which might occur due to possible duplication of a vertex. Duplicating a vertex $v \in V$ l times increases the capacity $k(v)$ by a factor l.

Let $x(v) \in \{0, \dots, b(v)\}$ be the number of times vertex $v \in V$ is added to the cover. For each $e = \{u, v\} \in E$ let $y(e, v) = 1$ when edge $e \in E$ is covered by vertex $v \in V$ and $y(e, v) = 0$ otherwise. The set $\delta(v)$ contains all edges $e \in E$ adjacent to vertex $v \in V$. The mathematical formulation as presented in the literature is as follows.

$$\min \sum_{v \in V} w(v)x(v) \tag{3.10}$$

$$\text{s.t. } y(e, u) + y(e, v) = 1 \qquad \forall e = \{u, v\} \in E \tag{3.11}$$

$$y(e, v) \leq x(v) \qquad \forall v \in e \in E \tag{3.12}$$

$$k(v)x(v) - \sum_{e \in \delta(v)} y(e, v) \geq 0 \qquad \forall v \in V \tag{3.13}$$

$$y(e, v) \in \{0, 1\} \qquad \forall v \in e \in E \tag{3.14}$$

$$x(v) \in \{0, \dots, b(v)\} \qquad \forall v \in V \tag{3.15}$$

The goal is to find a minimum weight vertex cover (3.10) subject to several constraints. Constraint set (3.11) indicates that each edge should be covered from exactly one vertex. An edge can only be covered from a selected vertex due to constraint set (3.12). No vertex can have more connections than its capacity allows due to constraint set (3.13). Finally, there is the binary decision variable $y(e, v)$ in constraint set (3.14) and the integer decision variable $x(v)$ in constraint set (3.15).

When the capacity $k(v) = |V| - 1$ for each vertex $v \in V$ and the decision variable $x(v) \in \{0, 1\}$ the CVC reduces to the Minimum Weight Vertex Cover. The Minimum Weight Vertex Cover problem is known to be NP-hard from Michael and David [45] and thus the CVC is also at least NP-hard.

For the CVC there are mainly two versions, namely the one with 'soft constraints' and the one with 'hard constraints'. Referring back to the formulation for the CVC the difference is in the unbounded $x(v) \in \mathbb{N}_0$ for the soft constrained version and the bounded $x(v) \in \{0, 1, \dots, b(v)\}$ for the version with the hard constraints.

The CVC has been introduced by Guha et al. [33] in which they applied the CVC with soft constraints and arbitrary weights to a problem arising in the field of glycobiology. For the resulting problem a 2-approximation algorithm using a primal-dual approach and a 4-approximation algorithm based on LP-rounding are given. Gandhi et al. [29] give a 2-approximation algorithm for the same problem based on LP-rounding.

For the CVC with hard constraints, where the value $b(v)$ is bounded, the problem becomes significantly harder. Chuzhoy and Naor [18] show that the CVC with hard constraints is at least as hard to approximate as the Set Covering

Problem when arbitrary weights are used. In their paper, Chuzhoy and Naor [18] give a 3-approximation algorithm for the unweighted case, where $b(v)$ is bounded to one, based on randomized rounding of an LP-relaxation followed by an alteration step.

In Gandhi et al. [30] the algorithm by Chuzhoy and Naor [18] is improved in two crucial ways to obtain a 2-approximation for the unweighted case of the CVC with hard constraints. Namely, Gandhi et al. [30] add a pre-processing step and they modify the alteration step from Chuzhoy and Naor [18].

The input to the WNP can be translated to a graph as follows. Let the locations taken as input to the WNP be the vertices and let the a_{ij} matrix specify the edges. The costs of a vertex are given by the connection costs of a location in this case. Additionally it can be assumed that the capacity for each location is known and that each location can be duplicated only once. Using this translation it can be demonstrated that the CVC does not describe the problem at hand.

In the CVC all edges should be covered by a vertex to come to a valid solution. However, for the WNP the goal is to connect each location. When two locations are opened than it is no longer necessary to cover the edge which might be existent between these locations. Thus the formulation of the CVC is not sufficient to describe the WNP.

4 Solution Approach

For both the MSLSCP and WNP several solution methods will be proposed. The solution methods will be compared on their ability to find good solutions and on their computation times. The solution method which yields the most promising results for the MSLSCP will be used as input to the WNP.

4.1 MSLSCP Methods

For the MSLSCP multiple methods are proposed ranging from an exact solution method using a commercial Mixed Integer Programming solver to two greedy heuristics and a sophisticated solution method which sequentially solves SCPs.

The methods approach the problem from different perspectives. The method based on a SCP in section 4.1.2 approaches the problem from the perspective of the services. The greedy heuristic in section 4.1.3 approaches the problem from the location perspective and the greedy heuristic in section 4.1.4 approaches the problem from the demand perspective.

4.1.1 Exact Algorithm

The presented problem formulation can be solved by means of an exact algorithm in reasonable time for problem instances of a likely limited size. The problem formulation will be implemented and the commercial Mixed Integer Programming solver Gurobi will be used to search for an optimal solution.

The results of the exact algorithm on a small problem are very useful. These can be used to determine the effectiveness of the other methods. This can be done based on the resulting solution as well as on the computation time which was required.

4.1.2 Sequential Set Covering Heuristic

When solving the MSLSCP for only one service the resulting problem can be characterized as a SCP. A naive implementation would be to solve a SCP for each service independently without taking into account the solutions for other services. However, this fails to take into account the preferred grouping of services at a single location. To achieve an efficient distribution while also grouping the services together the following method is proposed.

In the first iteration only one service is considered. For this service a SCP is solved where each of the demand points should be covered from the given locations. The costs of selecting each location is given by $f_j + c_j^u$. This takes into account that some locations might be preferred over others due to the relatively low costs of enabling a location to be equipped with services. In this iteration a selection has been made from the locations which are required to provide coverage for the first service. This information can then be used for the next service.

For the next service the fixed costs of enabling a previously selected location to be equipped with services are set to zero. So a location that has been equipped with the previous service can now be equipped with the new service at a cost of only c_j^u instead of the original $c_j^u + f_j$. This makes it so that locations which have been selected for the previous service are preferred to be equipped with the new service.

Then again all locations which are selected to be equipped with the new service which were not yet equipped with the previous service can be set to a fixed cost of zero. These steps should be repeated till each of the services is distributed in such a way that coverage is provided for all services.

This method is likely to result in short computation times when each SCP is solved heuristically. Good algorithms exist which obtain a good heuristic solution to a SCP. Considering only one service at a time is however a simplification of the general MSLSCP and will void the optimality of the results, especially when a heuristic solution is used for each of the resulting SCPs.

An outline for the Sequential Set Covering (SSC) Heuristic is given in Algorithm 4.1. In this outline the set X^u contains the locations which are connected to an existing network when solving the SCP for service $u \in \mathcal{F}$.

In the SSC only one service is considered at a time. For the first service the costs of equipping a location with this service consists of the connection costs f_j and the equipping costs c_j^u. With these costs a SCP is solved. For all locations which have been connected in an iteration the connection costs f_j are set to zero for future iterations. This makes it more likely that, ultimately, services are grouped together on connected lampposts as much as possible.

This is done for all services, such that all services are provided while satisfying the coverage requirements. Based on the final outcome a total costs can be calculated for the solution.

Algorithm 4.1 Outline for the Sequential SCP Heuristic.

```
1: $X^u = \emptyset$
2: for $u \in \mathcal{F}$ do
3:     Solve SCP for service using costs $c_j^u + f_j \rightarrow X^u$
4:     Set $f_j = 0$ for locations in $X^u$
5: end for
```

4.1.3 Likelihood Heuristic

The MSLSCP can also be approached from the location perspective. A certain location is likely to be enabled to be equipped with services when from this location the largest amount of demand can be satisfied at the least amount of enabling costs. This results in a greedy heuristic. Greedy heuristics are known for their intuitive implementations and are also likely to result in quick solutions.

In this greedy heuristic, which is called the Likelihood Heuristic, a ratio is calculated for each unconnected location in each iteration which represents the likeliness of lampposts to occur in good solutions.

In each iteration it is known for each of the services which demand points still require coverage. This means that for any location not yet equipped with a service it can be calculated how many demand points can be covered when the specific location is equipped with the service. When this number is divided by the costs of enabling a location to be equipped with services then the resulting ratio indicates the likeliness of the location to be equipped with a service.

Now the costs of enabling a location to be equipped with services can either be f_j when the considered service would be the first to be placed at the location or a small number when the location is already equipped with a service. The small costs of equipping a location with a service indicates that the location can be equipped with another service at relatively low costs.

A set $C^u \subseteq \mathcal{G}^u$ is determined in each iteration consisting of the demand points still requiring coverage for service $u \in \mathcal{F}$. The set X^u contains the locations which have been connected for service $u \in \mathcal{F}$. The calculated ratio is as follows.

$$L_j^u = \frac{\sum_{i \in C^u} a_{ij}^u}{f_j} \qquad \forall j \notin X^u, \forall u \in \mathcal{F} \tag{4.1}$$

The resulting values can be sorted in decreasing order. Based on these values it can be found for which location and which service the highest value of L_j^u has been achieved. This location is thus able to cover the most demand at the least amount of costs. For the resulting service the found location is added to X^u and the connection costs for the location should be set to ϵ, to identify this location as a cheap location to equip with additional services while preventing division by zero. This approach should be repeated till all demand is satisfied, i.e. till all sets C^u containing the demand points which still require coverage, are empty.

With the resulting solution it might be the case that some demand points are satisfied multiple times. To see whether this is the case, one can take the solution and calculate for each demand point the number of times which it is covered. If there is a location which is equipped with a service which only covers demand points which are covered too often, than this service is considered to be redundant at this location. When unequipping the location with the service the solution remains feasible and the total costs are reduced. This step should be repeated for each location in the solution and for all services with which it is equipped.

For the resulting solution the total costs can be calculated. An outline for the Likelihood Heuristic can be found in Algorithm 4.2.

Algorithm 4.2 Outline for the Likelihood Heuristic.

```
1: $X^u = \emptyset$
2: Determine $C^u$ $\forall u \in \mathcal{F}$
3: **while** $C^u \neq \emptyset$ $\forall u \in \mathcal{F}$ **do**
4:     Calculate $L_j^u$ $\forall j \notin X^u, u \in \mathcal{F}$
5:     $j, u = \text{argmax}\{L_j^u\}$
6:     $X^u = X^u \cup \{j\}$
7:     Set $f_j = \epsilon$
8:     Update the sets $C^u$
9: **end while**
10: Remove redundant services
```

4.1.4 Connection Heuristic

The MSLSCP can also be approached from the perspective of the demand points. Some demand points might only be covered from a small number of locations. To find the demand points which might only be covered from very few locations the same sets X^u and C^u are used in the same way as in the Likelihood Heuristic. These sets contain the connected locations and the demand points which should still be covered, respectively.

Again, an uncovered demand point can only be covered from a location within range that has not yet been equipped with the service. For each demand point it can be calculated how many locations are within range that might cover the demand point for the specific service. This value is then stored in the D_i^u variable.

$$D_i^u = \sum_{j \notin X^u} a_{ij}^u \qquad \forall i \in C^u, \forall u \in \mathcal{F} \tag{4.2}$$

Then the demand point and service are searched for which the minimum value of D_i^u is found. This demand point is then the demand point which can be covered from the lowest amount of potential locations. For the demand point it can be found which locations might cover it for the resulting service, denoted by the set S. Then it is searched which of these locations covers the most additional uncovered demand for the same service. This location is then connected and the sets C^u are updated. This step is then repeated till all demand points are covered.

Just as in the Likelihood heuristic, which is described in section 4.1.3, the resulting solution can be inspected on the presence of redundant services. These should be deleted to improve the solution quality. An outline for the Connection Heuristic is given in Algorithm 4.3.

Algorithm 4.3 Outline for the Connection Heuristic.

```
1: $X^u = \emptyset$
2: Determine $C^u$
3: while $C^u \neq \emptyset \; \forall u \in \mathcal{F}$ do
4:     Calculate $D_i^u$
5:     $i, u = \text{argmin}\{D_i^u\}$
6:     $S = \{j | a_{ij}^u > 0\}$
7:     $X^u = X^u \cup \{\text{argmax}_{j \in S}\{\sum_{i \in C^u} A_{ij}^u\}\}$
8:     Update the sets $C^u$
9: end while
10: Remove redundant services
```

4.1.5 Improvement Step

The proposed solution methods are mostly heuristics. Results from early iterations are taken into account in the later iterations of each of the heuristics. However, the heuristics are unable to take into account the results from later iterations in early iterations. This might mean that in an early iteration a location is connected and equipped with certain services even though it turns out that it is redundant in later iterations. To measure the effectiveness of the heuristics the solutions of the heuristics are taken as input to the exact algorithm.

From the heuristics it is known that the obtained solution is sufficient to provide coverage, however it need not be an optimal solution. The locations which have been chosen in the solution to be equipped with services can be taken as input to the exact algorithm. Taking the selected locations results in a new MSLSCP, however of a greatly reduced size.

It can then be investigated whether the exact algorithm is able to greatly improve the solution based on the selected locations from the heuristic solutions or not. When a large improvement is possible than this indicates that in the heuristic many locations have been selected which are not necessary to provide coverage. However, when the exact algorithm is unable to improve much on the heuristic solution then this indicates that the heuristic has been able to efficiently identify locations and equip these with services.

When a very large instance of the MSLSCP is solved by a heuristic then even the reduced problem size after this step might be too large to be efficiently solved by the exact algorithm. Because of this the exact algorithm can merely be used to gauge the quality of the presented heuristic solutions instead of being a viable alternative to coming to good solutions.

4.2 Common Steps for Metaheuristics

For the WNP multiple metaheuristics will be implemented of varying complexity. In the literature metaheuristics are recognized as efficient approaches to hard optimization problems. Metaheuristics are usually defined by a general framework in which the same set of steps are executed to solve an optimization problem. The steps which are executed may be adapted to better fit the optimization problem.

A good metaheuristic should be able to find sufficiently good solutions by sampling solutions for an optimization problem in a structured manner. In the sampling of the solutions it is important to obtain the right balance between diversification and intensification. Diversification refers to the ability of a metaheuristic to sample solutions from a large portion of the search space. A metaheuristic should also be able to improve already good solutions, which is called intensification.

Most metaheuristics rely on similar steps which are applied in a slightly different manner for each of the metaheuristics. Common steps consist of initializing solutions, generating neighboring solutions and perturbing a solution to move to a slightly different part of the search space. A general implementation for these steps will be presented and these implementations will then be used in a selection of metaheuristics.

Due to the common implementation of important steps for metaheuristics it will be interesting to see which of the metaheuristics is able to obtain the best balance between diversification and intensification for the WNP. The common steps are discussed in the following sections.

4.2.1 Initial Solution

An initial solution to the WNP is generated using a (randomized) greedy heuristic. In the first step locations are identified which are always connected to an existing network. These are the locations which are not in range of any other location, i.e. for which $\sum_i a_{ij} = 0$. All locations for which this holds are added to the set $\mathcal{F}$, containing all locations which are either connected to an existing network or to another location.

Then, using the remaining locations $j \notin \mathcal{F}$, a value is calculated which represents the likeliness of a location to occur in good solutions. This is calculated as follows.

$$V_j = \min\{1 + \sum_{i \notin \mathcal{F}} A_{ij}, k_j\}/f_j \qquad \forall j \notin \mathcal{F} \tag{4.3}$$

The presented ratio can be seen as the number of locations one can add over the costs of connecting a location to an existing network. The location for which the highest value of V_j is reached is most likely to occur in a good solution and is chosen to be connected to an existing network. This results in a greedy heuristic.

This procedure can also be randomized. Based on the values V_j a probability can be assigned to each of the locations which can be used to choose which location to add to the solution. This probability is calculated as in (4.4). This is a common way of assigning probabilities, see Murata et al. [46].

$$P_j = \frac{(V_j - \min_j V_j)^2}{\sum_j (V_j - \min_j V_j)^2} \qquad \forall j \notin \mathcal{F} \tag{4.4}$$

When a location is selected to be connected it is checked which other locations are within range of the newly connected location. When there are less lampposts within range than the specified capacity, then all lampposts within range are connected to the newly connected lamppost. When there are more lampposts within range than the specified capacity than randomly lampposts are selected to be connected to the newly connected lampposts with equal probability till the capacity is reached.

All newly connected locations are added to the set $\mathcal{F}$ and the previous steps are repeated till all lampposts are either connected to an existing network or to another lamppost.

Algorithm 4.4 (Randomized) Greedy Heuristic.

```
1:  Initialize $\mathcal{F}$ with all locations which are always opened
2:  Set $y_j = 1\ \forall j \in \mathcal{F}$
3:  while Not all locations connected do
4:      Calculate $V_j\ \forall j \notin \mathcal{F}$
5:      if Greedy then
6:          Identify location to open $j = \text{argmax}\, V_j$
7:      else if Randomized Greedy then
8:          Calculate $P_j\ \forall j \notin \mathcal{F}$
9:          Randomly choose location $j$ to open using $P_j$
10:     end if
11:     Set $y_j = 1$
12:     Identify all locations within range $I = \{i | A_{ij} = 1, i \notin \mathcal{F}\}$
13:     if $|I| > k_j$ then
14:         Randomly remove $|I| - k_j$ locations from $I$
15:     end if
16:     Set $x_{ij} = 1\ \forall i \in I$
17:     $\mathcal{F} = \mathcal{F} \cap \{I \cap j\}$
18: end while
```

4.2.2 Defining the Neighborhood

The neighborhood of a feasible solution to the WNP is defined as follows. For any feasible solution it is given that there are small clusters of locations of which one is connected to an existing network. In the (randomized) greedy heuristic which is used to create an initial solution, there are always as much as possible locations connected to an opened location. Intuitively this might seem the best thing to do. However, this need not lead to the best possible solution.

To try and improve on a solution, the following method is proposed. Of all locations which are connected to an opened location, choose one at random to investigate whether disconnecting from the opened location is profitable. Disconnecting can only be profitable when the disconnected location is within range of one or more opened locations other than the location it was connected to, before disconnecting.

When a location has been identified for which disconnecting might be profitable then this location will be disconnected and consequently opened. All opened locations, other than the location the disconnected location was connected to, will be connected to the newly opened location if possible. Now these opened locations which have been connected to the newly opened location could have locations connected to them. These will then be disconnected and using the greedy heuristic will be opened again.

The total creation of a neighboring solution is as in Algorithm 4.5.

Algorithm 4.5 Neighborhood Heuristic.

```
1: Identify all locations connected to an opened location $C = \{i | \exists x_{ij} = 1\}$
2: Randomly choose $i \in C$, was connected to $J = \{j | x_{ij} = 1\}$
3: Open the location $i$, $x_{ij} = 0 \ \forall j \in \mathcal{L}$, $y_i = 1$
4: Identify all opened locations in range of $i$, $O = \{k | A_{ik} = 1, y_k = 1, k \neq J\}$
5: Disconnect all $k \in O$, $y_k = 0$ and $x_{jk} = 0 \ \forall j \in \mathcal{L}$
6: Let $N = \{i | y_i = 0, \nexists x_{ij} = 0\}$
7: Identify all locations within range of $i$, $I = \{j | A_{ij} = 1, j \in N\}$
8: if $|I| > k_j$ then
9:     Randomly remove $|I| - k_j$ locations from $I$
10: end if
11: Set $x_{ji} = 1 \ \forall j \in I$
12: $N = N \backslash I$
13: Greedily connect all remaining locations in $N$
```

4.2.3 Perturbation of a Solution

In the perturbation of a solution the goal is to introduce a large amount of diversification in the solution. In this perturbation step large changes are made to a given solution by purposefully breaking and consequently repairing the solution in a randomized greedy way.

For any feasible solution the locations can be divided in a set of opened locations and a set of locations connected to an opened location. In the set of opened locations there might be a subset of locations which are opened in all feasible solution and locations which are opened in the specific solution. The opened locations which are not fixed in any feasible solution is the part of a solution which will be perturbed.

Of these non-fixed, opened locations 20% is closed. Additionally, all locations connected to these closed locations are disconnected. This results in a large part of the locations being disconnected and an infeasible solution. All locations which are closed and not connected to an opened location are added to the set N, thus indicating the locations requiring a repair.

The purposefully broken down solution is then rebuilt using a randomized greedy heuristic for all locations still requiring a connection. The complete algorithm is given in Algorithm 4.6.

Algorithm 4.6 Perturbation Heuristic.

```
1: Initialize $\mathcal{F}$ with all locations which are always opened
2: Initialize $C = \{j|y_j = 1, j \notin \mathcal{F}\}$
3: Randomly pick 20% of locations in $C$ to obtain $T$
4: Set $y_j = 0\ \forall j \in T$, $x_{ij} = 0\ \forall i \in \mathcal{L}, \forall j \in T$
5: Let $N = \{j|y_j = 0, \nexists x_{ji} = 0\}$
6: while $N \neq \emptyset$ do
7:     Calculate $V_j\ \forall j \in N$
8:     Calculate $P_j\ \forall j \in N$
9:     Identify location to open $j \in N$
10:    Set $y_j = 1$
11:    Identify all locations within range $I = \{i|A_{ij} = 1, i \notin \mathcal{F}\}$
12:    if $|I| > k_j$ then
13:        Randomly remove $|I| - k_j$ locations from $I$
14:    end if
15:    Set $x_{ij} = 1\ \forall i \in I$
16:    $N = N \backslash \{I \cap j\}$
17: end while
```

4.3 Selected Metaheuristics for the WNP

The common steps which have been discussed in the previous section will be fitted into the framework of several selected metaheuristics. Which metaheuristics are selected will be discussed in this section. Each of the selected metaheuristics uses the common steps in a slightly different way to find a good balance between diversification and intensification. For each of the proposed methods an explanation will be given on how both the intensification and diversification is reached. Also, each of the metaheuristics is classified according to being single-solution based or population based as in Boussaïd et al. [11].

4.3.1 Iterated Local Search

Local Search (LS) is an intensification-oriented, single-solution based metaheuristic. In a LS method it is repeatedly attempted to improve the current best solution by applying slight changes and accepting these changes when they lead to improvements. This is done till either a local optimum is found or a stopping criterion is reached. Whereas the LS method is strong in its intensification of a solution, it lacks in the diversification of the found solutions.

Iterated Local Search (ILS), proposed by Lourenço et al. [43], is an extension on the well-known LS method. The ILS method tries to improve on the LS method by introducing diversification through perturbation of a solution stuck in a local optimum. When for a solution a local optimum is found or a stopping criterion is reached, the search is restarted using a perturbed version of the current best solution. A good perturbation of a solution is achieved when the perturbed solution has the right balance between being a randomly chosen solution and the original solution. Usually, the restart is performed a fixed number of times, always starting from the current best solution.

A framework for the ILS method is given in Algorithm 4.7.

Algorithm 4.7 Iterated Local Search [43].

```
1:  Initialize $f(s^*) = \infty$
2:  Create an initial solution $s$ using a Greedy Heuristic
3:  Apply the local search method to $s$ to obtain $s'$
4:  if $f(s') < f(s^*)$ then
5:      $s^* = s'$
6:  end if
7:  for A fixed number of iterations do
8:      Perturb solution $s^*$ to obtain $p$
9:      Apply the local search method to $p$ to obtain $p'$
10:     if $f(p') < f(s^*)$ then
11:         $s^* = p'$
12:     end if
13: end for
14: Return the overall best solution
```

4.3.2 GRASP

Greedy Randomized Adaptive Search Procedure (GRASP) is another single-solution based metaheuristic which has been introduced by Feo and Resende [25, 26]. A GRASP procedure repeatedly initializes a solution s using a randomized greedy heuristic which is consequently improved using a simple local search method to obtain s'. Each time the local search method terminates the resulting solution is compared against the current best solution s^*. If it is found that the newly created solution is an improvement compared to the current best solution then the current best solution is replaced with the new solution.

This is done for a specified number of iterations, after which the algorithm terminates and the overall best found solution is returned. A general framework is given in Algorithm 4.8.

Algorithm 4.8 Greedy Randomized Adaptive Search Procedure [25, 26].

```
1: Initialize $f(s^*) = \infty$
2: for A fixed number of iterations do
3:     Create an initial solution $s$ using a Randomized Greedy Heuristic
4:     Apply the local search method to $s$ to obtain $s'$
5:     if $f(s') < f(s^*)$ then
6:         $s^* = s'$
7:     end if
8: end for
9: Return the best solution
```

4.3.3 Simulated Annealing

Another often used metaheuristic is Simulated Annealing (SA). This metaheuristic has been independently proposed by Čern𝑦̌ [15] and Kirkpatrick [39]. SA is a nature-inspired, single-solution metaheuristic, based on annealing metals. Annealing is a technique used in the field of metallurgy in which a metal is heated in order to reshape it and consequently cooling it in a controlled manner, such that the metal keeps its improved shape. The idea is that, when working a heated metal, sometimes the quality of the shape should decrease to ultimately end up with an improved shape.

SA uses this analogy by introducing controlling parameter T, called the temperature. The temperature indicates the state of the system. When the temperature is high a large amount of diversification can take place. As the temperature is decreased during the search process, the diversification is decreased and the intensification is increased.

After the initialization of a solution and a high temperature a neighboring solution is searched. When the objective value of the neighboring solution is better than the objective value of the original solution, then the original solution is replaced by the neighboring solution. When the neighboring solution is worse than the original solution, then the neighboring solution is accepted as the new solution with a probability based on the temperature and the objective values of both solutions. A general form of a probability function is $P(T, f(s'), f(s)) = \exp(-(f(s')-f(s))/T)$ where T is the temperature, s' is the neighboring solution and s is the original solution. When the temperature is high the probability of accepting a worse solution is large and as the temperature decreases the probability of accepting a worse solution decreases.

Accepting worse solutions can be seen as a diversification step to escape from local optima. The SA method should ultimately converge to a good solution, which is why the intensification of a current solution is increased throughout time. For a temperature $T = 0$ a worse solution is never accepted and the SA method is equal to a LS method. Generally the SA method is terminated when either the temperature is below a certain threshold value or when the objective value no longer improves.

Simulated annealing has been around for a long time and has been a topic of interest to many researchers. Many variants have been proposed. See Suman and Kumar [53] for a recent survey.

A general framework for SA can be found in Algorithm 4.9. In this algorithm the value η represents the threshold temperature. When the temperature drops below this temperature it is expected that a good solution has been found by this time.

Algorithm 4.9 Simulated Annealing [15, 39].

```
1: Create an initial solution $s$ using a Greedy Heuristic
2: Initialize temperature T
3: while $T > \eta$ do
4:     Search neighboring solution $s'$ of $s$
5:     if $f(s') < f(s)$ then
6:         $s = s'$
7:     else
8:         $s = s'$ with probability $P(T, f(s'), f(s))$
9:     end if
10:    Decrease T
11: end while
12: Return the best solution
```

4.3.4 Genetic Algorithm

An example of a nature-inspired population-based metaheuristic is a Genetic Algorithm (GA), first designed by Holland [37]. Since its original introduction there have been many adaptations and extensions of GAs. However, each GA has some specific characteristics, which are present in most of the implementations.

In principle, the GA is based on the theory of evolution found in nature. In each iteration of the algorithm, or the so-called generation, there is a set of solutions, otherwise known as a population of individuals. Each of these individuals is more or less fit for their collective environment. The fitness of any individual is generally expressed using the objective value corresponding to the solution. Following the theory of natural selection it follows that individuals which are more fit to their environment have a higher probability of passing its desirable genes over to the next generation.

Passing over to a next generation occurs through a recombination of two individuals, called parents, to produce children. The idea here is that combining favorable traits of two parents results in children which are even better suited to the environment. Another way in which improvements in the solutions can occur is through mutation. Mutation changes an individual slightly which might possibly improve its fitness for the environment.

A GA generally is run for a fixed number of generations, after which the best found solution is returned.

Due to the highly adaptable nature of GAs it is possible to implement them for a large range of problems. However, this also means that it can be hard to find the right settings for a problem. Choices have to be made on how to recombine two solutions, how a mutation is defined, the probability of mutation, the probability of crossover, and so on.

For the WNP the GA has been implemented as follows. First the exogenous parameters are initialized. These are the population size S, the maximum number of generations G, the crossover probability p_c and the mutation probability p_m.

The population is then initialized by calling a randomized greedy heuristic S times. For all the individuals in the population a fitness is calculated by evaluating the objective function of the WNP.

Based on the fitness of each of the individuals a selection is made on which individuals to advance to the next generation. This is done by assigning to each of the individuals a probability of being selected as a parent. The probability of selecting solution $p \in S$ is given in equation (4.5). The second parent is chosen by assigning equal probabilities to the remaining parents. This follows the methodology presented in Murata et al. [46] and can be classified as tournament selection.

$$P_p = \frac{(f(p) - \min_{s \in S} f(s))^2}{\sum_s (f(p) - \min_{s \in S} f(s))^2} \qquad \forall p \in S \tag{4.5}$$

When two different parents are selected it is determined whether they should recombine in the next generation or not. To do this a uniform random variable x is drawn which is uniformly distributed between [0,1]. When $x < p_c$ the solutions are recombined, otherwise the solutions advance to the next generation as they are.

Recombining two solutions is not necessarily straightforward. This follows from the constraints which may not be violated. However, solutions should retain their favorable characteristics. For the WNP it is chosen to implement the recombination as follows. Let parent 1 be represented by x_{ij}^1 and y_j^1 and parent 2 is x_{ij}^2 and y_j^2. For each location in these solutions it follows that they are either connected to an opened location, or they are opened themselves. These are the choices which are deemed characteristic in the WNP.

A breakpoint is randomly chosen for the two solutions. Then locations are crossed by taking all choices made for the locations from the first parent till the breakpoint and all choices made for the locations from the second parent from the breakpoint till the end. For the second new solution this is done by taking the choices made for the locations till the breakpoint from the second parent and additionally all choices made for the locations from the breakpoint till the end from the first parent.

This recombination may result in infeasible solutions. For example, when in the first solution the choice is transferred that a location before the breakpoint is connected to an opened location after the breakpoint but this location is not opened in the parent with which is recombined than the solution is no longer valid. Thus after recombination each solution should be checked for feasiblity and, if necessary, undergo repair.

Repair is done by searching for all locations which are connected to a location which is not opened in the new solution. These are the invalid locations. For each of these locations it is checked whether there is another opened location within range which has not yet reached its capacity. If so, the location is connected to this opened location. Otherwise, the location is disconnected from all opened locations and opened itself. This is repeated till there are no more invalid locations.

When the recombination phase has ended a new population is obtained. However, before moving to the next generation, each of the individuals in the new population might undergo mutation. To each of the solutions in the population a random variable is assigned drawn from a uniform distribution between [0,1]. When for any solution the random variable is smaller than p_m the individual undergoes mutation. This is done by moving the individual to a neighboring solution as has been described in the implementation.

These steps are then repeated for the prespecified number of generations G. At the end the best found solution is returned.

Algorithm 4.10 Genetic Algorithm [11].

```
1:  Initialize exogenous parameters $S$, $G$, $p_c$ and $p_m$
2:  Initialize population by calling the Randomized Greedy Heuristic $S$ times
3:  for $G$ generations do
4:      Select parents
5:      if $x \in U(0,1) < p_c$ then
6:          Recombine parents
7:      else
8:          Pass parents to next generation as they are
9:      end if
10:     Select individuals to mutate from new population
11: end for
12: Return the best solution
```

5 Solving Set Covering Problems

The Sequential Set Covering method introduced in section 4.1.2 is based on sequentially solving a SCP. This method is motivated by the similarity between the SCP and the MSLSCP. In section 3.2 a literature review has been given of the Set Covering Problem. In this literature review it is stated that SCPs are known to be NP-hard problems [45]. This means that large scale SCPs are not easily solved to optimality.

However, the SCP has been studied extensively and many good methods exist for solving a SCP. In this section the implementation of a heuristic solution method for the SCP will be presented. Afterwards, the performance of the implemented method will be compared to other implementations by solving the problems in Beasley's OR-Library [6].

5.1 Solution Method

From the literature review in section 3.2 it can be deduced that many heuristic solution methods for the SCP exist. The ones which show the best performance are based on a combination of Lagrangian relaxation, subgradient optimization and solving a pricing problem to define the core problem. The solution method which is implemented here is for a large part based on Beasley [5], which is improved by first defining a core problem by following the steps of Ceria et al. [14].

5.1.1 Lagrangian Heuristic

Lagrangian relaxation is a proven method for quickly calculating lower bounds. The right to exist for Lagrangian relaxation stems from the fact that it is often more efficient in finding a lower bound than solving a linear relaxation. Up until the introduction of Lagrangian relaxation in the 1970s most methods used a linear relaxation to calculate a lower bound in, for example, a branch and bound method. After the introduction of Lagrangian relaxation many methods were easily improved by the implementation of Lagrangian relaxation.

Lagrangian relaxation is based on the view that most hard problems would be easy to solve if it were not for a small set of side constraints. In Lagrangian relaxation a problem is simplified, i.e. relaxed, by dualizing the set of constraints which make a problem hard to solve. What this effectively means is that one is allowed to violate the dualized constraints. However, this comes with a penalty in the solution value. The penalization of violating the dualized constraints is caused by the so-called Lagrangian multipliers.

The key to the Lagrangian relaxation method is the fact that the Lagrangian multipliers, i.e. the penalization, may be varied. By varying the Lagrangian multipliers one can influence the solution to the relaxed problem. For solving the relaxed problem it is known that solving it always results in a valid lower bound to the original problem. This means it might pay off to investigate whether stronger lower bounds can be found by varying the Lagrangian multipliers. A method to find good Lagrangian multipliers in a structured way is subgradient optimization.

A detailed introduction to the use of Lagrangian relaxation combined with subgradient optimization for hard optimization problems is given by Fisher [27].

Lagrangian relaxation has been applied to many hard optimization problems, among which is the SCP. An early heuristic based on Lagrangian relaxation is given by Beasley [5]. In the paper by Beasley, the constraint (3.8) is relaxed which requires that all rows have to be covered. This gives the following Lagrangian Lower Bound Program (LLBP) where λ_i are the Lagrangian multipliers associated with the relaxed constraint.

$$\text{Min } \sum_{j\in\mathcal{N}} c_j x_j + \sum_{i\in\mathcal{M}} \lambda_i \left(1 - \sum_{j\in\mathcal{N}} a_{ij} x_j\right) = \tag{5.1}$$

$$\text{Min } \sum_{j\in\mathcal{N}} \left(c_j - \sum_{i\in\mathcal{M}} \lambda_i a_{ij}\right) x_j + \sum_{i\in\mathcal{M}} \lambda_i \tag{5.2}$$

$$\text{s.t. } x_j \in \{0,1\} \qquad \forall j \in \mathcal{N} \tag{5.3}$$

In (5.1) the dualization of the coverage constraint can be seen, which is penalized by λ_i when the constraint is violated. However, for ease of notation it is common to use the objective function as in (5.2). This notation is useful in solving the LLBP.

From (5.2) it can be seen that there is a fixed term, namely the sum of the Lagrangian multipliers, which can not be influenced. Now let $C_j = c_j - \sum_{i\in M} \lambda_i a_{ij}$ represent the coefficient of x_j in (5.2). Using this term it can be seen that an optimal solution to the LLBP can easily be found by setting $x_j = 1$ if $C_j < 0$ and $x_j = 0$, otherwise. For any set of multipliers $\lambda \geq 0$ it is known that the optimal solution to the LLBP is a valid lower bound to the original SCP instance. Let $L(\lambda)$ be the optimal solution value of the LLBP given λ. The best lower bound is found by solving $\max L(\lambda)$.

Subgradient optimization is used to find the vector λ which maximizes the Lagrangian function. Subgradient optimization is an iterative procedure which incrementally updates the vector of multipliers. Given an initial vector of multipliers λ^0 a sequence of multipliers is generated by equation (5.4). In this equation, λ^k is the current vector of multipliers, T is the step size and G_i is the subgradient.

$$\lambda_i^{k+1} = \max\{0, \lambda_i^k + TG_i\} \tag{5.4}$$

In each iteration of the subgradient optimization the subgradient G_i is calculated as in (5.5). Here x_j^k is the optimal solution to the LLBP for the vector of multipliers λ^k in the kth iteration.

$$G_i = 1 - \sum_{j \in \mathcal{N}} a_{ij} x_j^k \qquad \forall i \in \mathcal{M} \tag{5.5}$$

A suitable step size T is calculated as in (5.6). Here Z_{UB} is the objective value of the best found solution to the original SCP and Z_{LB} is the objective value found by solving the LLBP for the current set of multipliers. Parameter f is a controlling parameter which influences the step size. Throughout the execution of the algorithm the parameter is decreased which results in smaller step sizes.

$$T = f \frac{(1.05 Z_{UB} - Z_{LB})}{\sum\limits_{i \in \mathcal{M}} (G_i)^2} \tag{5.6}$$

Note that, in calculating a suitable step size T a value for Z_{UB} is required. Ideally the value for Z_{UB} is frequently updated as it is preferred to have a smaller step size when the difference between Z_{UB} and Z_{LB} becomes smaller. In the Lagrangian heuristic as proposed by Beasley this is done by making use of the information contained in the solution to the LLBP. Namely, in each iteration of the algorithm the solution to the LLBP is transformed into a solution to the original SCP instance.

A solution to the LLBP is not necessarily a feasible solution to the original SCP. This follows from the fact that in the LLBP the coverage constraint is relaxed, meaning it is allowed to violate this constraint. To transform the solution to the LLBP the following steps are executed. First, let $S = \{j | x_j = 1, \forall j \in \mathcal{N}\}$, where x_j is a solution to the LLBP. Based on this it can be checked whether there are any rows which are not covered by the columns in S. For each row $i \in \mathcal{M}$ not covered by S, a column is added to S covering row $i \in \mathcal{M}$ at the least amount of costs. This gives $S = S \cup \text{argmin}(c_j | a_{ij} = 1)$. This results in a feasible solution, however some of the columns in S might turn out to be redundant.

A column in S is said to be redundant if the set $S = S \backslash \{j\}$ is still a feasible solution to the original SCP. For each of the columns in S it is checked whether it is redundant in descending order of c_j and removed if it is found to be redundant. Finally, it is checked whether the found solution has a lower solution value than the best found solution so far. The best found solution to the original SCP is updated by $Z_{UB} = \min(Z_{UB}, \sum_{j \in S} c_j)$.

As stated the best value for Z_{UB} is used for calculating the step size. The calculated values are then used to update the multipliers. With the new multipliers the same procedure is executed again. When the subgradient optimization method is unable to improve the best found solution to the original SCP instance for 30 consecutive iterations, the value for f is halved.

The entire algorithm for Beasley's Lagrangian Heuristic can be found in Algorithm 5.1. In the algorithm the Lagrangian multipliers are initialized as $\lambda_i = \min\{c_j | a_{ij} = 1, \forall j \in \mathcal{N}\} \;\; \forall i \in \mathcal{M}$. The algorithm terminates when either an optimal solution is found (i.e. $Z_{UB} = Z_{min}$, where Z_{min} is the best found lower bound) or when $f \leq \epsilon$, which is an arbitrarily chosen stopping criteria. At the end the best found solution is returned.

Algorithm 5.1 Beasley's Lagrangian Heuristic.

```
1: Initialize $Z_{min} = -\infty$, $Z_{UB} = \infty$, $\lambda_i = \min\{c_j | a_{ij} = 1, \forall j \in \mathcal{N}\} \;\; \forall i \in \mathcal{M}$,
   $f = \eta$
2: while $Z_{UB} \neq Z_{min}$ and $f > \epsilon$ do
3:     Solve LLBP for the current set of multipliers, gives $Z_{LB}$
4:     if $Z_{LB} > Z_{min}$ then
5:         $Z_{min} = Z_{LB}$
6:     end if
7:     Construct a feasible solution $S$ to the original SCP
8:     Remove redundant columns from $S$
9:     if $\sum_{j \in S} c_j < Z_{UB}$ then
10:        $Z_{UB} = \sum_{j \in S} c_j$
11:    end if
12:    Calculate the subgradients $G_i$
13:    Calculate step size $T$
14:    if $Z_{min}$ has not increased in the last 30 iterations then
15:        $f = f/2$
16:    end if
17:    Update the Lagrangian multipliers $\lambda_i$
18: end while
```

5.1.2 Defining the Core Problem

The Lagrangian Heuristic by Beasley is found to be a very efficient method, which is able to reach near optimal results with little computational effort. However, the problem size of many SCP instances has increased since the introduction of the Lagrangian Heuristic. Many large-scale SCP instances are still solved with a Lagrangian Heuristic, which is applied to a so-called core problem, e.g. Caprara et al. [12], Ceria et al. [14].

The goal of defining a core problem is to reduce the problem size, while maintaining similar solution quality. For a SCP instance, a core problem consists of a subset of the columns which are found in the original SCP instance and are able to produce feasible solutions. Solving the core problem by any method, such as a Lagrangian Heuristic, yields a solution to the core problem. In turn, a solution to the core problem is a valid solution to the original SCP instance. This is due to the fact that the core problem consists of columns found in the original SCP instance.

To obtain good solutions to the original SCP instance the core problem should be determined in an effective way. Any good core problem should consist of columns which can be found in good solutions, supplemented with additional columns to ensure feasible solutions. Any column which is not found in the core problem is assumed to not be found in good solutions and may never be selected.

Defining a core problem greatly reduces the problem size of a SCP instance. However, note that defining a core problem is only efficient when the time required to determine the core problem and solving the core problem offsets the time required to solve the original SCP instance. Also, solutions to the core problem should be of similar quality as the solutions to the original SCP instance.

To define the core problem a pricing procedure is used. The pricing procedure is in the spirit of the well-known column generation technique, as introduced by Bixby et al. [10]. Both Ceria et al. [14] and Caprara et al. [12] used a pricing procedure specifically tailored for large-scale SCP instances to define the core problem. The difference between the implementation of Caprara et al. and Ceria et al. is that Caprara et al. update their core problem dynamically, whereas Ceria et al. define their core problem once and try to find the best solution to this core problem. The added benefit to updating the core problem dynamically is that better solutions are found, however this comes at an increase in computational effort. As the aim of this research is to obtain quickly generated solutions it is decided to use the implementation of Ceria et al. to define the core problem.

Ceria et al. [14] use Lagrangian relaxation to quickly calculate a good lower bound for the original SCP and a corresponding set of optimal multipliers λ^*. Based on the optimal multipliers they calculate the reduced costs for each of the columns, as in (5.7). All columns which have a reduced costs which is below a threshold value $R_j \leq \gamma$ are added to the core problem. To ensure that the columns in the core problem are able to result in a feasible solution to the original SCP they add columns to the core problem till each row is covered by at least θ columns. This is done by identifying how often rows are covered. If any row is covered less than θ times, it is identified which columns cover those specific rows. Those columns are then added in ascending reduced costs order till all rows are sufficiently covered.

$$R_j = c_j - \sum_{i \in \mathcal{M}} \lambda_i a_{ij} \qquad \forall j \in \mathcal{N} \tag{5.7}$$

So far the method for determining a good set of multipliers λ is similar to the method used by Beasley. However, for large problems it is inefficient to create a feasible solution to the original SCP in each iteration. Also, in defining the core problem it is not yet an objective to find a solution to the original SCP. The goal is to find a good set of multipliers with which a core problem can be defined. To determine a good set of multipliers an upper bound on the original

SCP is required. To quickly calculate an upper bound to the original SCP Ceria et al. use Lagrangian relaxation for the dual of the original SCP.

The dual to the linear relaxation of the SCP is formulated as follows.

$$\text{Max} \sum_{i \in \mathcal{M}} y_i \tag{5.8}$$

$$\text{s.t.} \sum_{i \in \mathcal{M}} a_{ji} y_i \leq c_j \qquad \forall j \in \mathcal{N} \tag{5.9}$$

$$y_i \geq 0 \qquad \forall i \in \mathcal{M} \tag{5.10}$$

Where a Lagrangian relaxation is given by relaxing the constraint set (5.9), resulting in the following Lagrangian upper bound program (LUBP). The multipliers corresponding to the relaxation are given by μ_j.

$$\text{Max} \sum_{i \in \mathcal{M}} (1 - \sum_{j \in \mathcal{N}} a_{ij}^T \mu_j) y_i + \sum_{j \in \mathcal{N}} c_j \mu_j \tag{5.11}$$

$$\text{s.t. } y_i \geq 0 \qquad \forall i \in \mathcal{M} \tag{5.12}$$

For any set of multipliers μ the solution value to the LUBP is a valid upper bound to the original SCP. Given a set of dual multipliers μ a solution is easily calculated. Let $\bar{c}_i = \max\{c_j | a_{ij} = 1, \ \forall j \in \mathcal{N}\}$. Note that the following is implied $0 \leq y_i \leq \bar{c}_i$ due to constraint set (5.9), even though it is not explicitly stated. A solution to the LUBP is then given by setting $y_i = \bar{c}_i$ if $(1 - \sum_{j \in \mathcal{N}} a_{ij}^T \mu_j) > 0$ and $y_i = 0$ otherwise.

Note that, when maximizing, the choice should actually be made to set $y_i = \infty$ as this choice is in no way prohibited and this would result in the true maximum of the LUBP. Doing so, however, would result in erratic behavior in the convergence to an optimal set of multipliers λ. When the found upper bound constantly switches between ∞ and a suitable upper bound there is no useful meaning to the calculated step size. To prevent this erratic behavior the choice is made to set $y_i = \bar{c}_i$.

Let $DL(\mu)$ be the solution value to the LUBP for the dual multipliers μ. The best upper bound is found by solving $\min DL(\mu)$. A good set of dual multipliers μ can be found using subgradient optimization in a similar fashion to Beasley's Lagrangian Heuristic.

In each iteration of the algorithm the dual multipliers are updated as in (5.13).

$$\mu_j^{k+1} = \max\{0, \mu_j^k + US_j\} \qquad \forall j \in \mathcal{N} \tag{5.13}$$

The subgradients for the dual Lagrangian problem are calculated as in (5.14). In this equation y_i^k is the optimal solution to the LUBP for the dual multipliers

μ^k.

$$S_j = \sum_{i \in \mathcal{M}} a_{ij} y_i^k - c_j \qquad \forall j \in \mathcal{N} \tag{5.14}$$

A step size for the dual subgradients is calculated as follows. Here Z_{UB} is the solution value of the LUBP for the current set of dual multipliers μ. Z_{min} is the best lower bound found in the previous iterations. Parameter $g = \eta$ initially, however this value is halved if the tightest upper bound has not improved in the last 30 iterations.

$$U = g \frac{(1.05 Z_{ub} - Z_{min})}{\sum_{j \in \mathcal{N}} (S_j)^2} \tag{5.15}$$

Ceria et al. created a Primal-Dual Lagrangian algorithm which solves the Lagrangian relaxations of the original SCP and the dual of the original SCP simultaneously. In the algorithm Z_{min} represents the best lower bound and Z_{max} represents the best upper bound. The algorithm stops when either an optimal solution is found (i.e. $Z_{UB} = Z_{min}$) or when either $f \leq \epsilon$ or $g \leq \epsilon$. The algorithm is implemented as in Algorithm 5.2, where λ, f and T correspond to the Lagrangian relaxation of the original SCP.

When the algorithm terminates the final multipliers λ are used to construct the core problem. This core problem is then extended with additional columns to ensure the feasibility of a possible solution. This core problem can then be solved by any method which can be used to solve a SCP instance.

5.2 Performance of the Solution Method

The performance of the implemented method can be compared to the other methods by solving the SCP problems found in Beasley's OR-library. These SCPs are solved by most of the in the literature available methods for SCPs and for many problems an optimal solution is known.

In the comparison three solution methods will be compared to one another. First there is the method by Beasley [5], which is shown in Algorithm 5.1. This method is indicated by Be. For this method the parameter ϵ is set to 0.005, just as in the paper by Beasley [5]. Secondly, there is the method where first a core problem is identified as is described in Algorithm 5.2. For defining the core problem the parameters $\gamma = 0.1$, $\theta = 10$ and $\epsilon = 0.005$ are used, which are equivalent to the parameters used by Ceria et al. [14]. This core problem is then solved using the Lagrangian heuristic as in Algorithm 5.1. This method is indicated by BeCe. Thirdly, the objective value and solution times of the commercial Gurobi solver are added to compare the solutions by the heuristics to the optimal solutions.

Algorithm 5.2 Ceria's Primal-Dual Lagrangian Algorithm.

```
1: Initialize $Z_{min} = -\infty$, $Z_{max} = \infty$, $\lambda_i = \min\{c_j | a_{ij} = 1, \forall j \in \mathcal{N}\}$ $\forall i \in \mathcal{M}$,
   $\mu_j = 1/\sum_{i \in \mathcal{M}} a_{ij}$ $\forall j \in \mathcal{N}$, $f = g = \eta$
2: while $Z_{UB} \neq Z_{min}$ and $f > \epsilon$ and $g > \epsilon$ do
3:     Solve LLBP and LUBP for the current multipliers, gives $Z_{LB}$ and $Z_{UB}$
4:     if $Z_{LB} > Z_{min}$ then
5:         $Z_{min} = Z_{LB}$
6:     end if
7:     if $Z_{UB} < Z_{max}$ then
8:         $Z_{max} = Z_{UB}$
9:     end if
10:    Calculate the subgradients $G_i$ and $S_j$
11:    Calculate step size $T$ and $U$
12:    if $Z_{min}$ has not increased in the last 30 iterations then
13:        $f = f/2$
14:    end if
15:    if $Z_{max}$ has not increased in the last 30 iterations then
16:        $g = g/2$
17:    end if
18:    Update the Lagrangian multipliers $\lambda_i$ and $\mu_j$
19: end while
20: Determine core problem based on final $C_j < \gamma$
21: Extend the core problem till each row is covered $\theta$ times
```

	Test Problem Details			
Set	Rows	Columns	Density (%)	No. of Problems
4	200	1,000	2	10
5	200	2,000	2	10
6	200	1,000	5	5
A	300	3,000	2	5
B	300	3,000	5	5
C	400	4,000	2	5
D	400	4,000	5	5
E	500	5,000	10	5
F	500	5,000	20	5
G	1,000	10,000	2	5
H	1,000	10,000	5	5

Table 5.1: Details for each of the test sets.

Some details about the SCPs found in Beasley's OR-Library can be found in Table 5.1. In this table the density is defined as the percentage of ones in the a_{ij} matrix. For each of the problem sets the costs of a column lie within the interval [1,100].

As can be seen from this table, the problem sets differ in size (i.e. number of rows and number of columns) and density. Generally, the computational effort required to solve a problem increases with the size and the density of the problem.

For most of the problems in the problem set an optimal solution is known in the literature. However, for the hardest problems considered, namely the problems in problem sets G and H, no optimal solutions are known and only a best known objective value is reported in the literature. As a best known solution is available it is still possible to compare the performance of the heuristics to this value. For the Gurobi solver it will be interesting to see whether it is able to solve the problems in the problem set within 10 minutes, which is the time limit set for the solver.

In Table 5.2 the problems of sets 4, 5 and 6 are solved. These are relatively small problems and require little computational effort. The Gurobi solver is able to solve these problems to optimality in a small amount of computation time. The heuristics Be and BeCe are comparable in their performance. Both have similar computation times and are able to obtain optimal or near-optimal results. From these problem instances it can be seen that, for problems of a limited size, no improvement is made in terms of computation times by first defining a core problem. However, first defining a core problem also does not increase the computation times of the heuristic.

In Table 5.3 the problem sets A, B, C and D are considered. For these problem sets the Gurobi solver is still able to solve them to optimality in only a couple of seconds. However, the Gurobi solver is no longer noticeably more

		Be		BeCe		Gurobi	
Problem	Opt	Obj	Time (s)	Obj	Time (s)	Obj	Time (s)
4.1	429	429	0.281	429	0.401	429	0.026
4.2	512	512	0.290	512	0.230	512	0.016
4.3	516	516	0.089	516	0.186	516	0.015
4.4	494	495	0.924	494	0.284	494	0.017
4.5	512	512	0.128	512	0.104	512	0.015
4.6	560	561	0.903	561	0.712	560	0.026
4.7	430	430	0.230	430	0.442	430	0.015
4.8	492	493	0.810	492	0.885	492	0.039
4.9	641	641	1.349	641	1.008	641	0.044
4.10	514	514	0.151	514	0.263	514	0.016
5.1	253	254	1.511	254	1.115	253	0.107
5.2	302	305	1.841	302	1.326	302	0.105
5.3	226	226	0.166	226	0.173	226	0.024
5.4	242	242	1.225	242	0.902	242	0.037
5.5	211	211	0.184	211	0.184	211	0.025
5.6	213	213	0.203	213	0.353	213	0.025
5.7	293	293	1.083	294	1.045	293	0.043
5.8	288	288	1.352	288	0.865	288	0.040
5.9	279	279	0.315	279	0.402	279	0.024
5.10	265	265	0.106	265	0.167	265	0.023
6.1	138	140	1.359	141	0.803	138	0.322
6.2	146	149	1.137	148	0.903	146	0.594
6.3	145	145	1.070	145	0.718	145	0.118
6.4	131	131	0.785	131	0.744	131	0.054
6.5	161	166	1.143	163	0.892	161	0.657

Table 5.2: Comparison of the SCP solution methods for the problem sets 4, 5 and 6.

		Be		BeCe		Gurobi	
Problem	Opt	Obj	Time (s)	Obj	Time (s)	Obj	Time (s)
A.1	253	255	2.980	256	1.493	253	0.852
A.2	252	256	2.536	255	1.655	252	0.592
A.3	232	235	2.670	234	1.741	232	0.417
A.4	234	234	2.449	234	1.729	234	0.203
A.5	236	237	1.968	237	1.371	236	0.090
B.1	69	70	2.436	70	1.133	69	1.446
B.2	76	76	3.766	76	1.576	76	1.978
B.3	80	81	2.959	81	1.229	80	0.862
B.4	79	81	3.651	81	1.318	79	2.256
B.5	72	72	3.353	72	1.372	72	1.219
C.1	227	230	4.415	229	2.932	227	0.660
C.2	219	222	3.966	222	2.041	219	1.264
C.3	243	248	5.492	247	2.490	243	2.165
C.4	219	223	4.363	223	2.193	219	0.958
C.5	215	217	4.005	217	1.940	215	0.970
D.1	60	60	6.641	60	1.799	60	1.776
D.2	66	68	5.525	67	1.687	66	3.684
D.3	72	73	6.311	74	2.240	72	4.200
D.4	62	64	5.944	64	1.751	62	2.709
D.5	61	61	5.259	61	2.288	61	0.902

Table 5.3: Comparison of the SCP solution methods for the problem sets A, B, C and D.

efficient in coming to a solution than the BeCe method. In terms of solution quality both the Be and BeCe method show similar performance by obtaining optimal or near-optimal solutions. Based on the time in which the solution is obtained, it is now evident that first defining a core problem already improves the efficiency of the BeCe method.

Finally, in Table 5.4 the hardest problem sets are considered. The problems in problem sets E and F are still solved to optimality by the Gurobi solver within the time limit of 10 minutes. Yet for the problems in problem sets G and H, which are the hardest problems, the Gurobi solver is unable to find the optimal solution within the time limit. For these problems best known solutions (BKS) are reported which originate from existing literature on SCP problems. Most of the times the Gurobi solver is not able to find the best known solutions within the time limit.

For all the problems in the problem sets E, F, G and H the computational efficiency of the heuristics is finally apparent. Again the heuristics are able to obtain optimal or near-optimal results while requiring only little computational

		Be		BeCe		Gurobi	
Problem	BKS	Obj	Time (s)	Obj	Time (s)	Obj	Time (s)
E.1	29	29	20.450	29	2.599	29	44.150
E.2	30	32	21.746	32	2.911	30	267.470
E.3	27	29	19.794	29	2.578	27	22.996
E.4	28	30	24.612	29	2.849	28	35.761
E.5	28	28	20.618	28	3.288	28	25.786
F.1	14	15	37.496	15	4.032	14	37.658
F.2	15	16	33.035	15	3.651	15	28.512
F.3	14	15	40.721	15	3.579	14	17.500
F.4	14	15	37.539	15	3.619	14	76.569
F.5	13	15	32.630	15	4.274	13	403.435
G.1	176	183	40.960	183	6.127	176	†600.000
G.2	154	162	41.011	164	6.364	155	†600.000
G.3	166	174	44.269	175	6.023	167	†600.000
G.4	168	178	39.816	176	6.003	168	†600.000
G.5	168	177	53.299	177	6.326	169	†600.000
H.1	63	68	110.769	69	8.404	65	†600.000
H.2	63	66	100.165	66	7.722	64	†600.000
H.3	59	65	101.599	65	7.978	61	†600.000
H.4	58	63	88.943	63	7.963	59	†600.000
H.5	55	59	82.091	59	7.674	55	†600.000

Table 5.4: Comparison of the SCP solution methods for the problem sets E, F, G and H. For solution times marked with †, the Gurobi solver reached the time limit before finding an optimal solution.

effort. For these problem sets the BeCe method really distinguishes itself in terms of computational efficiency. For the BeCe method the computations times are only a fraction of the computations times of the Be method and the Gurobi solver.

6 Application and Test Instances

In this section the design of several fictitious test instances is discussed. These are used to gain experience with the application of the models which are proposed to distribute the services over the lampposts in a city. In the test instances there are several key ingredients required to obtain a situation in which the proposed models are applicable. These ingredients are chosen in such a way that they resemble the real world conditions as close as possible.

For each test instance the list of services which are to be distributed will be the same. The test instances differ on which city is used as an area of interest. The cities are selected in such a way that the test instances are ranging from small rural towns to a moderately large city. The selected cities influence the size of the test instances. The test instance of a small rural town will have much less lampposts than the test instance of a moderately large city. The range of test instances can be used to investigate the scalability of the solution methods.

6.1 List of Services

For the test instances it is important to first define the services which will be considered. For each of the services an equipping cost should be known, which is assumed to be equal over all locations. Additionally, the shape of the coverage area is required.

In this research it is assumed that all services have a circular coverage area which is defined by the range of a service. For many services this will be an accurate representation of reality. However, it should be noted that in theory any other shape could be incorporated in the a_{ij}^{u} matrix.

For most of the services it is the case that the quality of the service diminishes when one is further away from the source. For example, an alarm can be assumed to be insufficiently audible from a certain distance. Hence, a range is specified for which it is assumed that the quality of the service is acceptable within the range. This does not mean that the service is no longer available just outside the range, however coverage is deemed inadequate from this point onward.

A final list of services which are considered in this research can be found in Table 6.1. For each of the services in this table information can be found on the range of the service and on the equipping costs. The specifications for each of the services are chosen to be plausible in a real world situation, however the list of services which will be considered is by no means exhaustive and serves an exemplary purpose.

		Specifications	
Service	Abbreviation	Range (m)	Costs (€)
Wi-Fi	Wi	50	300
Motion Detection	MD	100	350
Alarm	AL	300	150
Air quality	AQ	650	400
Weather Station	WS	1,500	950

Table 6.1: Technical specifications and costs of services.

6.2 The Locations of Lampposts

For many cities in the Netherlands the locations of lampposts are made publicly available. The data on the locations of lampposts is accessible through the Open Data Initiative Dataplatform[1]. For the test instances a diverse range of cities is chosen as an area of interest. The selection consists of Schiermonnikoog, Rozendaal, Noordwijk, Lisse, Amsterdam Center and Delft.

As an example, the city center of Amsterdam is used. The city center has 8,604 lampposts which are displayed on a map of Amsterdam in Figure 6.1.

6.3 Regional Representation

Some additional attention should be paid to the representation of the area which needs to be covered. The results from Murray et al. [48] indicate that representing the state space as regularly spaced points on a street map works best in terms of computational efficiency. A visualization of a regularly spaced regional representation of the city center of Amsterdam, The Netherlands for the Wi-Fi service can be found in Figure 6.2. The known location of lampposts can then be used to calculate the distance from each lamppost location to each of the demand points. Additionally, with the coverage of a service it can be checked whether a demand point might be covered by a service from a lamppost location.

A choice should be made for the density, i.e. the resolution, of the regularly spaced points on the street map. A high resolution results in an accurate regional representation. However, it also leads to an increase in the problem sizes.

From Table 6.1 it can be seen that the difference in range between services can be very large. The Wi-Fi service for example only has a range of 50 meters, whereas the weather station service has a range of 1,500 meters, which is 30 times as large. For services with a small range it is imperative to use a very detailed state space, i.e. the demand points should be close to one another. This ensures an accurate representation of a city. For services with a large range, the demand points can be further apart. This will still ensure an accurate representation of a city.

[1]`https://www.dataplatform.nl/` (Accessed on April 1, 2016)

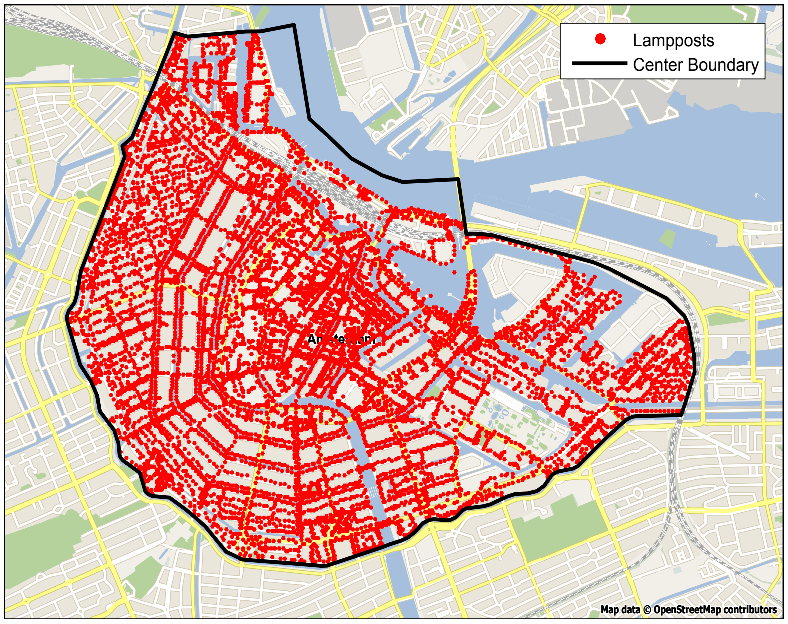

Figure 6.1: The locations of lampposts in the city center of Amsterdam, The Netherlands.

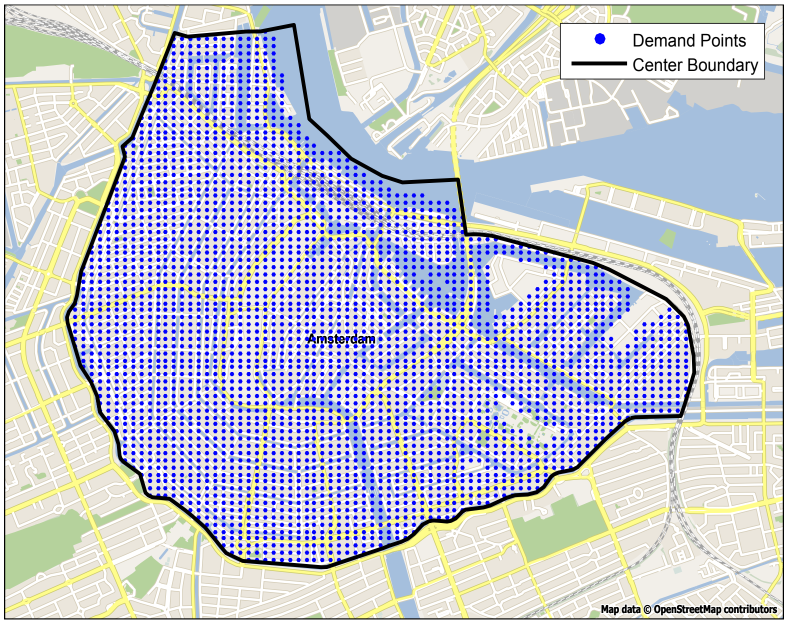

Figure 6.2: Regional representation of the city center of Amsterdam, The Netherlands for the Wi-Fi service.

The number of demand points which might be covered by a location is dependent on the range of a service and the density of the demand points. To be precise, it is the ratio of the range of a service and the distance between demand points which determines the potential number of demand points within reach. This can be seen from the following. Consider a circle with a range of 20 meters containing equally spaced demand points which are 1 meter apart. Then the same number of points are within this circle as in a circle with a range of 10 meters in which each point is 0.5 meters apart. However, when the range of a circle is 20 meters while the range between points is 0.5 meters the number of points within the circle has increased.

To illustrate how the number of demand points within range is dependent on the ratio of the range of a service and the distance between demand points a small test case has been created. Consider a square area where the sides have a length which is two times the range of a circle. In this square area a grid is created where the points have an equal distance from one another. When drawing a circle with as center the middle of the square area, one can count the number of points within the circle. When increasing the range, and thus the area which is considered, but keeping the distance between points the same, the ratio increases. With this also the number of points within range increases.

Figure 6.3 illustrates the increase in the number of demand points within range as the ratio increases. It can be seen that the ratio entirely explains the number of demand points which can be covered from a location. Given any ratio x, the number of demand points within range, y, is given by equation (6.1). This curve is also shown in Figure 6.3. The fitted equation is, not coincidentally, similar to the equation describing the area of a circle.

$$y = 3.169x^2 - 1.11x - 0.1088 \tag{6.1}$$

This result indicates that while, for example, a distance between demand points of 20 meters is suitable for the motion detection service, which has a range of 50 meters, this is not suitable for the air quality service. The air quality service has a range of 1,500 meters which would result in up to 17,743 demand points being covered from one location when the demand points are only 20 meters apart. Hence, the distance between demand points should be suitably adjusted for each of the services.

For this research it is chosen to have a distance between demand points which is the range of a service divided by 2. This results in a potential number of 10 demand points within range of any location and thus an accurate representation of the state space. Note that this is not a perfect representation. This is due to the fact that the demand points should be covered, however, the area between demand points does not have to be covered. When the distance between demand points increases, also the area which is potentially not covered increases. This is because only coverage guarantees for the demand points can be given with the current model. This concession has to be made, however, as the problem might turn computationally intractable if the number of demand points is too large.

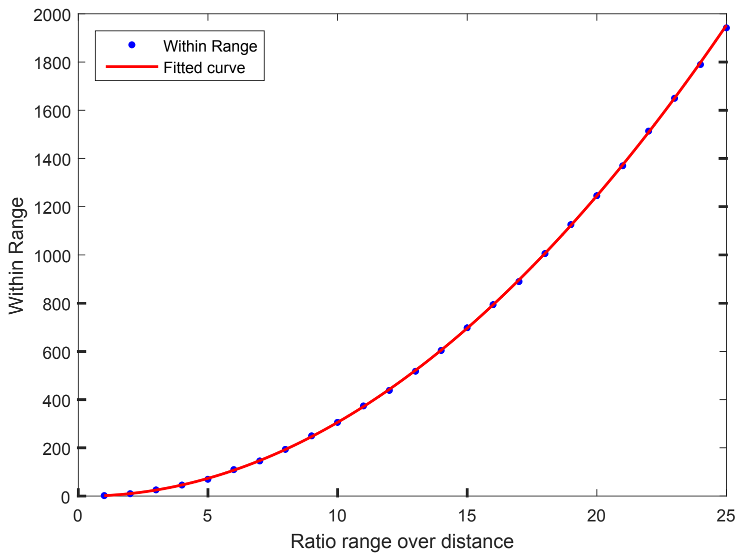

Figure 6.3: Relationship between the ratio range over distance and the number of demand points within range.

6.4 Places

Combining the information in sections 6.1 to 6.3 enables the creation of test instances. First a region should be specified within which the MSLSCP and WNP should be applied. This is specified by a boundary of (a part of) a city or a town. Using this boundary a grid of demand points is created for each of the services. For each of the services, the distance between the demand points is specified by half the service range.

Then a distance is calculated from each location of a lamppost to all demand points. This is done to create the coverage matrix for a service. When for a service a demand point might be covered from a location then this information is stored in the coverage matrix. It might be that some demand points are never within the range of any location of a lamppost for a service. These demand points are obsolete and thus deleted from the formulation. All resulting demand points are required to be covered at least once in any valid solution to the MSLSCP. This is repeated for all services, such that each service has its own demand points and coverage matrix.

The costs of connecting a lamppost to an existing network are assumed to be uniformly distributed between €750 and €5,000. A uniform distribution is chosen as no additional information on the costs of connecting a lamppost to an existing network is known at the time.

In Table 6.2 it can be found which places are considered. For each of the places a test case has been created. In the table information can be found on the number of lampposts and the number of created demand points per service.

		Demand Points				
Service Area	Lampposts	Wi	MD	AL	AQ	WS
Schiermonnikoog	233	1,704	783	242	106	52
Rozendaal	523	1,736	653	116	40	16
Noordwijk	1,162	4,156	1,902	488	187	72
Lisse	4,273	10,172	3,692	724	194	43
Amsterdam Center	8,604	11,744	3,243	391	96	19
Delft	13,885	22,489	6,806	983	248	53

Table 6.2: Test instance details.

From this table it can be seen that the selection of cities results in varying sizes of the test instances.

7 Computational Results

The proposed solution methods are applied to the test instances to solve the problem where a set of services is to be distributed over the lampposts in a city such that coverage can be provided. As stated in the introduction this will be done in a two-step approach. First the MSLSCP is solved for the test instances to make a selection from the lampposts which are required to provide coverage. Then it is investigated whether a cost reduction is possible by applying the solution methods for the WNP to the solution of the MSLSCP obtained using the most promising solution method.

For each of the solution methods a comparison will be made on the computation time as well as on the quality of the solution. It is preferred to come to good solutions in a relatively small amount of time.

7.1 MSLSCP Results

First the results for the MSLSCP to the test instances are presented. The results for each of the solution methods on the test instances can be found in Table 7.1. In the first phase the results for the solution methods, as described in section 4.1.1 to 4.1.4, are presented. The second phase is the improvement step which has been described in section 4.1.5 This is used to gauge the quality of the solutions obtained by the proposed solution methods. Another measure to determine the solution quality is by comparing the solution to the lower bound for each of the test instances, which is calculated by solving the linear relaxation of the MSLSCP. For each solution a percentage deviation from the lower bound is presented.

The MSLSCP Exact method is the exact algorithm as described in section 4.1.1. The time limit for the Gurobi solver has been set to an overnight computation time of seven hours. Optimal solutions have been found for the four smallest test cases, namely for Schiermonnikoog, Rozendaal, Noordwijk and Lisse. The results are indicative of the rapidly increasing computational effort which is required to solve larger problems.

The SSC Heuristic method is the Sequential Set Covering method as described in section 4.1.2. In each iteration the resulting SCP is solved using the heuristic method described in section 5. For the smallest test case Schiermonnikoog, this method is able to find the optimal solution. For the test cases Rozendaal, Noordwijk and Lisse the found solution is very close to the optimal solution. For the larger test cases Amsterdam Center and Delft the SSC Heuristic method is able to find solutions within reasonable computation times. For these test cases no optimal solutions are known. However, the found solutions are very close to the best found solutions by the exact algorithm.

The results for the greedy Likelihood and Connection heuristics are also shown. These methods are described in section 4.1.3 and section 4.1.4, respectively. Both these methods are able to quickly find feasible solutions to the test cases. The Likelihood heuristic tends to require a longer time till a solution is found, however the solution is always of a better quality then those found by the Connection heuristic. This can be attributed to the fact that the Likelihood heuristic takes into account the costs of opening a location, whereas the Connection heuristic does not. Even though the solutions found by the Likelihood heuristic are always of a lesser quality than the solutions found by the SSC method, they are somewhat close. The same can not be said for the Connection heuristic.

In the second phase the improvement step is applied to the solutions found in the first phase. Using the connected locations it is investigated whether a redistribution of the services leads to an improvement in the solution quality. The improvement is not applied to the solutions found by the MSLSCP Exact method as the optimal solutions can not be improved.

For the smaller test cases Schiermonnikoog, Rozendaal, Noordwijk and Lisse the solutions are quickly improved. For the larger test cases Amsterdam Center and Delft the improvement step is less efficient. For the Delft test case, all solutions from the first phase can not be optimally improved within the time limit. However, some feasible solutions are found which are improvements of the initial solutions. The computation times which are required indicate that the improvement step might not be useful to apply to large test cases.

For the SSC method, the improvements are not very large. This indicates that a redistribution of the services over the opened locations is not very influential on the final result. This shows that the services are already distributed quite effectively over the opened locations by the SSC method in the first phase. For the Connection method there is much room for improvement. However, even the improved solutions for the Connection method do not match the solution quality of the solutions found by the SSC and Likelihood methods in the first phase.

In Figure 7.1 the results for the Air Quality service are visualized for the test case Amsterdam Center which has been solved by the Likelihood method. Indeed all demand points are covered which is as expected of a feasible solution. Some coverage gaps can be found in the solution. However, as explained in section 6.3, these were to be expected.

7.2 Wireless Network Problem

From the results for the MSLSCP it follows that the SSC method yields the most useful results. The solutions to each of the test instances obtained through applying the SSC method are taken as input to the WNP. The most important objective of the WNP is to see whether a cost reduction can be found in enabling lampposts to be equipped with services. In the MSLSCP the assumption has been that all lampposts have to be connected to an existing network in order

			Phase 1			Phase 2		
TC	LB (€)	Method	Obj (€)	Dev (%)	Time (s)	Obj (€)	Dev (%)	Time (s)
Schier monnikoog	528,291	MSLSCP Exact	528,889	0.11	0.12	-	-	-
		SSC Heuristic	528,889	0.11	2.33	528,889	0.11	0.07
		Likelihood	542,154	2.62	0.08	535,359	1.34	0.06
		Connection	592,924	12.23	0.19	554,192	4.89	0.06
Rozen daal	592,871	MSLSCP Exact	593,402	0.09	0.17	-	-	-
		SSC Heuristic	596,727	0.65	1.57	594,734	0.31	0.10
		Likelihood	636,001	7.27	0.16	610,854	3.03	0.14
		Connection	748,627	26.27	0.13	654,739	10.44	0.10
Noord wijk	1,692,027	MSLSCP Exact	1,692,594	0.03	0.55	-	-	-
		SSC Heuristic	1,706,116	0.83	3.66	1,696,606	0.27	0.11
		Likelihood	1,783,309	5.39	0.90	1,721,095	1.72	0.14
		Connection	2,017,122	19.21	0.89	1,823,728	7.78	0.17
Lisse	2,957,457	MSLSCP Exact	2,963,737	0.21	2,736.75	-	-	-
		SSC Heuristic	2,991,293	1.14	21.71	2,977,416	0.67	3.96
		Likelihood	3,167,800	7.11	6.43	3,056,816	3.36	234.32
		Connection	4,032,367	36.35	5.26	3,533,453	19.48	167.22
Amsterdam Center	3,236,138	MSLSCP Exact	3,255,959	0.61	$^{\dagger}25,200.00$	-	-	-
		SSC Heuristic	3,303,342	2.08	26.18	3,280,118	1.36	981.73
		Likelihood	3,609,602	11.54	14.27	3,449,529	6.59	$^{\dagger}25,200.00$
		Connection	5,130,907	58.55	10.24	4,362,125	34.79	4,083.67
Delft	6,918,375	MSLSCP Exact	6,947,421	0.42	$^{\dagger}25,200.00$	-	-	-
		SSC Heuristic	7,019,789	1.47	96.92	6,984,609	0.96	$^{\dagger}25,200.00$
		Likelihood	7,576,794	9.52	43.59	7,259,661	4.93	$^{\dagger}25,200.00$
		Connection	10,425,661	50.70	30.79	8,939,547	29.21	$^{\dagger}25,200.00$

Table 7.1: Results for each of the test cases. For solution times marked with †, the Gurobi solver reached the time limit before finding an optimal solution.

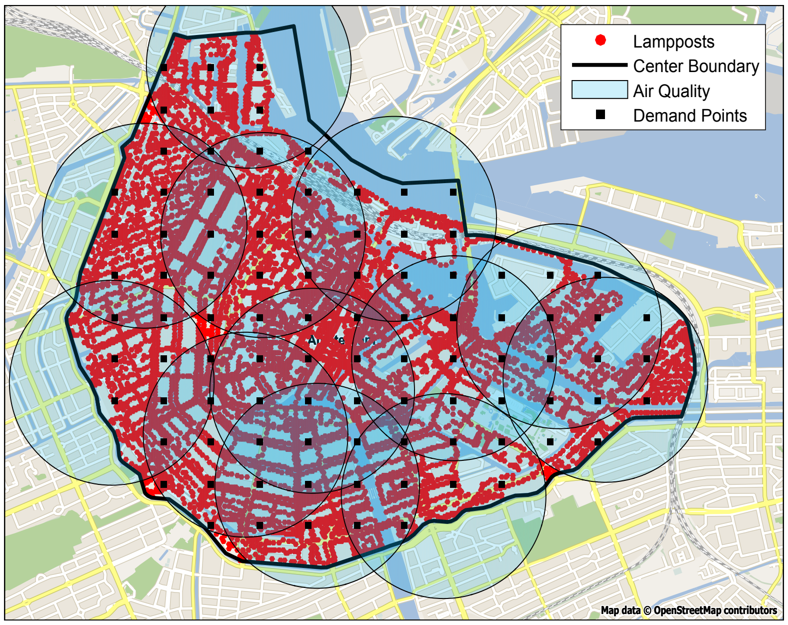

Figure 7.1: Result for the Air Quality service for the Likelihood heuristic. Test case Amsterdam Center.

to be equipped with services. This is a costly endeavor and any possible cost reductions should be exploited.

In the application of the WNP to the solutions of the MSLSCP it is assumed that a lamppost connected to an existing network can act as a hub to at most four other lampposts which are within a range of 100 meters of the connected lamppost.

In Table 7.2 the starting position of the WNP is shown. This starting position is defined by the results of the MSLSCP obtained through the SSC method on the test instances. In these solutions a certain amount of lampposts is connected to an existing network and there is a corresponding connection costs associated with this. In the table also the results of the exact algorithm for the WNP can be found. The results indicate an increasing computational effort required to solve the WNP. However, the results also give a useful benchmark for the implemented heuristic methods.

The results for each of the heuristic methods can be found in Table 7.3. As all methods are randomized to a certain extend it is chosen to apply the methods three times to each of the test instances. In the table the best objective value from the three solutions is presented, as well as the average objective value and the average time required to come to a solution.

First, the ILS and GRASP methods are shown. The implementation of both these methods has been discussed in section 4.3.1 and section 4.3.2, respectively. For the ILS it has been chosen to restart the search 10 times from a perturbed

	MSLSCP		Exact WNP		
Service Area	Connected	Costs (€)	Connected	Costs (€)	Time (s)
Schiermonnikoog	163	441,049	55	120,478	0.15
Rozendaal	196	506,227	48	73,958	0.29
Noordwijk	544	1,434,866	142	233,330	2.00
Lisse	1,075	2,511,543	261	393,003	89.09
Amsterdam Center	1,401	2,729,674	307	357,287	1536.70
Delft	2,792	5,851,939	648	855,452	$^{\dagger}25,200.00$

Table 7.2: Comparison of connection results after the MSLSCP and after WNP. For solution times marked with †, the Gurobi solver reached the time limit before finding an optimal solution.

version of the best solution. At the end the overall best solution is returned for which the results are shown in the table. The GRASP method has also been restarted 10 times, however for this method the search starts from a solution obtained through a randomized greedy procedure.

Secondly, the SA and GA methods are shown. The implementation of both these methods has been discussed in section 4.3.3 and section 4.3.4, respectively. These methods are more advanced than the ILS and GRASP method in the sense that they allow some control over the behavior of the methods through the tuning of several parameters. For the SA a choice can be made on the initial temperature, the cooling scheme which is used and when the method should terminate. The GA requires a population size, a mutation rate, a crossover rate and a maximum number of generations.

For the SA method the same initial temperature has been used for all test instances which is chosen to be a temperature of 10,000. This initial temperature yields suitable acceptance probabilities of worse solutions. Dependent on the test instances differing cooling schemes have been used. For the Schiermonnikoog, Rozendaal and Noordwijk test cases the temperature is decreased by 50 after each iteration. For Lisse the temperature is decreased by 25 in each iteration and for Amsterdam Center and Delft the temperature is decreased by 10 in each iteration. When the temperature is decreased with a smaller number then this ultimately results in a larger number of total iterations, which for the larger test instances is useful as the search space is larger for these test instances. For all test instances the search is terminated once the temperature falls below 500.

For the GA it is chosen to have the same parameters for all test instances. The population size is initialized to be 20. In accordance with existing literature the mutation rate has been selected to be 5% and the crossover rate is set to 80%, see Boussaïd et al. [11]. The GA terminates after 50 generations. These parameters are chosen in such a way that there is a sufficient number of iterations to allow the algorithm to investigate a large part of the search space. A high crossover rate is chosen to give good solutions the opportunity to obtain

	ILS			GRASP		
Service Area	Best	Avg	Time (s)	Best	Avg	Time (s)
Schiermonnikoog	134,851	135,450	0.21	132,874	134,140	0.76
Rozendaal	76,791	81,091	0.37	85,829	87,312	1.17
Noordwijk	255,015	256,550	2.15	267,084	279,010	10.94
Lisse	501,235	511,790	15.93	545,337	555,790	64.53
Amsterdam Center	516,017	538,388	32.25	550,556	561,990	136.65
Delft	1,186,504	1,196,515	147.39	1,276,910	1,287,202	318.24

	SA			GA		
Service Area	Best	Avg	Time (s)	Best	Avg	Time (s)
Schiermonnikoog	143,149	146,025	0.14	145,901	147,714	3.38
Rozendaal	93,514	96,734	0.23	95,074	99,251	5.17
Noordwijk	308,813	320,027	1.72	342,396	346,415	19.43
Lisse	550,193	551,091	12.37	612,893	620,898	75.62
Amsterdam Center	565,008	583,935	53.23	617,867	629,034	119.78
Delft	1,225,352	1,241,628	201.21	1,395,782	1,413,018	333.49

Table 7.3: Results on WNP for ILS, GRASP, SA and GA.

good characteristics of other solutions. Mutation of a solution is not necessarily preferred as this might result in a decreasing quality of a solution which is why this rate is set relatively low.

From the results it can be seen that the ILS method returns the most promising results. Generally, this method is able to find the best solutions while requiring the least amount of computation time. This indicates that this method is able to find a good balance between intensification and diversification. The other methods are unable to find a similar balance between intensification and diversification, however this can be explained.

For the GRASP method it is found that it is able to obtain solutions which are not far off from the solutions found by the ILS method. This can be attributed to the fact that these methods use the same local search method. However, the solutions times for GRASP are noticeably worse than the solution times which are required for the ILS method. This can be attributed to the initialization of these solutions.

The heuristic used to obtain initial solutions could be implemented either greedy, randomized greedy or totally random. When comparing the quality of the initial solutions the results as in Figure 7.2 could be seen for all the test instances. The greedy implementation is always able to find better initial solutions than the randomized greedy implementation. Hence, any method starting from a randomized greedy solution requires more local search iterations before a solution of a similar quality of even the greedy heuristic is found. The GRASP method always restarts from a randomized greedy solution, whereas

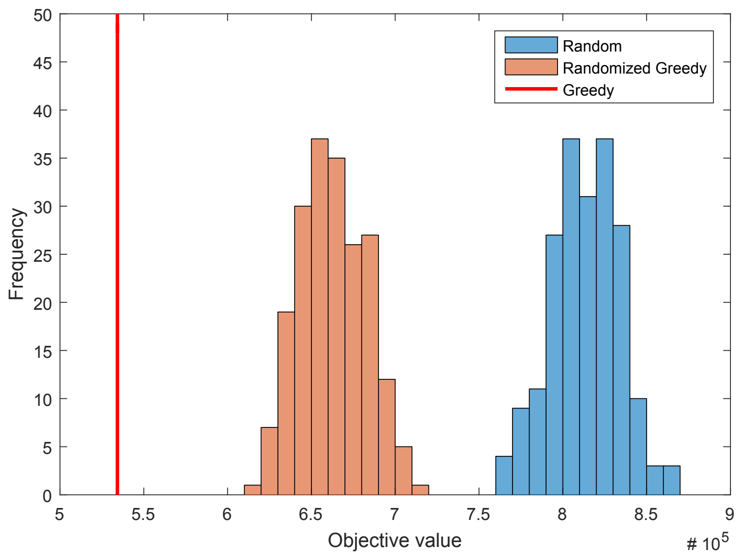

Figure 7.2: Comparison of initial solutions for the test instance Lisse.

the ILS method starts from a greedy solution and restarts from a perturbed version of the current best solution which is likely to still be of a better quality than the randomized greedy solutions. This leads to a smaller number of overall iterations for the ILS method and thus better computation times.

The SA method also starts from a greedy solution. However, this method sometimes moves to worse solutions for the purpose of diversification. As the algorithm progresses the amount of diversification is lowered and the amount of intensification is increased. Apparently the method is unable to obtain a suitable starting point for the intensification of a solution after the diversification of the solutions.

The GA method is overall the method which results in the least favorable solutions and computation times. The GA method is also the method which is the most distinct from the other methods. The reason why the GA method is unable to find good solutions can be attributed to the fact that it is rather difficult to suitably recombine two solutions. Generally after each recombination the solutions have to be repaired in order to adhere to the restrictions imposed by the WNP. This repairing of a solution prevents the GA method from converting to better and better solutions.

This shows that ILS is the preferred method for solving the WNP. For this method the solutions are of a sufficient quality while requiring the least amount of computation time. A visual representation of the WNP result obtained with the ILS method for the test instance Amsterdam Center is shown in Figure 7.3.

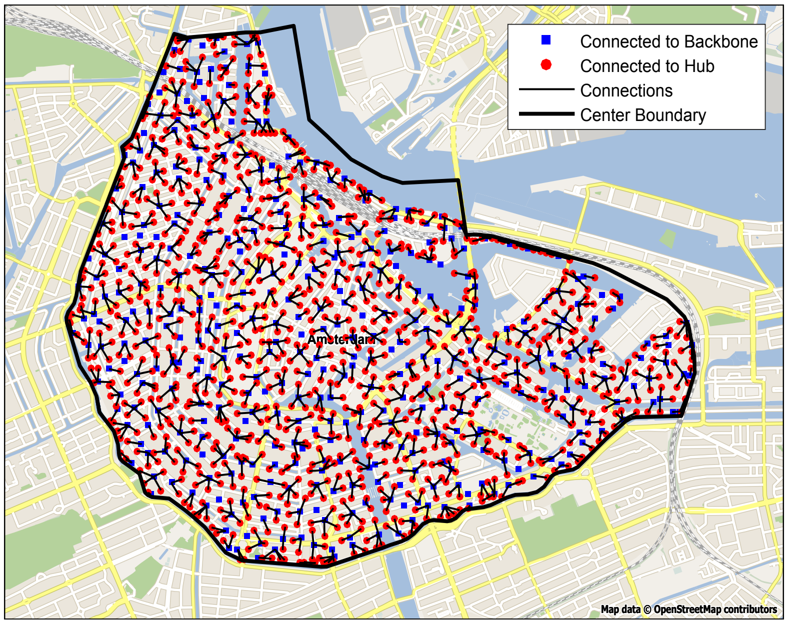

Figure 7.3: Results for the WNP for the test instance Amsterdam Center using ILS.

7.3 Two-Step Approach Results

When combining the results for both the MSLSCP and the WNP it has been found that the best combination of solution methods is the Sequential Set Covering method and the Iterated Local Search method. Combining these methods leads to a final result in terms of total costs required to supply the cities in the test instances with the services and a total number of connected lampposts. These results are shown in Table 7.4 and are based on the best solution obtained with the ILS method.

Service Area	Total Costs (€)	Connected
Schiermonnikoog	222,701	65
Rozendaal	167,291	56
Noordwijk	526,265	166
Lisse	980,985	323
Amsterdam Center	1,138,676	420
Delft	2,354,354	824

Table 7.4: Final results.

8 Concluding Remarks

In this research a visionary problem has been considered in which services are to be distributed over the lampposts in a city such that these services can be provided adequately against minimal costs. The difficulties of this problem lie in the shared set-up costs of enabling a lamppost to be equipped with services.

To solve this problem a two-step approach is proposed. In the first step it is assumed that all lampposts which are to be equipped with services need to be connected to an existing network.

A mathematical formulation of the problem in the first step has been presented and this formulation has been termed the Multi-Service Location Set Covering Problem. Based on this formulation and the similarities to some well-known problems from the literature, several solutions methods have been devised. The solution methods were implemented and tested on a range of test instances which consisted of cities in which five services were to be distributed over the lampposts. The implemented methods showed varying degrees of success in their ability to solve the MSLSCP. From the results it is concluded that the Sequential Set Covering method worked best.

For the SSC method to work adequately it is important to be able to solve Set Covering Problems. The SCP is known to NP-hard which means that this is no trivial task. From the literature it is found that the most promising methods use a combination of Lagrangian relaxation, subgradient optimization and the search for a core problem. In this research a method for defining a core problem found in the literature has been implemented which subsequently is solved using a proven method to solve SCPs which has also been found in the literature. The combination of these methods showed efficient solving capabilities of random SCP instances.

In the second step of the two-step approach the assumption is dropped that all lampposts which are equipped with services need to be connected to an existing network. It is now investigated whether several lampposts can act as a hub to other lampposts, such that not all lamppost need to be connected. In the first step of the two-step approach a selection has been made from the lampposts which are required to provide coverage. This selection is taken as input to the problem in the second step.

In this problem several restrictions had to be taken into account, such as a maximum range between the lampposts and a maximum number of lampposts which could be connected to a hub. This problem has been formulated mathematically and was termed the Wireless Network Problem.

For the WNP several metaheuristics have been implemented. A good metaheuristic should be able to efficiently find good solutions to hard optimization problems. A metaheuristic is defined by a framework in which several steps are executed in a structured manner such that solutions are efficiently sampled from the search space. The sampled solutions should have a good balance between diversification and intensification. Some of the frameworks for metaheuristics which were applied executed similar steps and these were implemented in the same way across the different metaheuristics. For the WNP it was found that the Iterated Local Search metaheuristic was able to find the best solutions in a reasonable amount of computation time.

8.1 Future Research

Inherent to the formulation of the MSLSCP is the fact that it can be assured that all demand points are covered. However, no guarantees on the coverage for the regions between the demand points can be given. This was to be expected due to literature already indicating such behavior. However, no solution for this behavior has been found. This means an interesting direction for future research would be to investigate how coverage of an entire region can be guaranteed.

To the WNP several metaheuristics have been applied. These metaheuristics had some similar implementations of steps generally occurring in the frameworks of metaheuristics. These were implemented in the same way and it has been interesting to see how each of the metaheuristics were able to use these steps to find good solutions. However, metaheuristics are solution methods which are designed in such a way that they make as few assumptions about good solutions as possible. To solve the WNP it might be that some methods which exploit the characteristics of the problem formulation might result in better solutions. This can not be said with certainty without performing any research regarding this.

Bibliography

[1] R. K. Ahuja, J. B. Orlin, S. Pallottino, M. P. Scaparra, and M. G. Scutellà. A multi-exchange heuristic for the single-source capacitated facility location problem. *Management Science*, 50(6):749–760, 2004.

[2] S. Bao, N. Xiao, Z. Lai, H. Zhang, and C. Kim. Optimizing watchtower locations for forest fire monitoring using location models. *Fire Safety Journal*, 71:100–109, 2015.

[3] J. Barceló and J. Casanovas. A heuristic lagrangean algorithm for the capacitated plant location problem. *European Journal of Operational Research*, 15(2):212–226, 1984.

[4] J. Bautista and J. Pereira. Modeling the problem of locating collection areas for urban waste management. an application to the metropolitan area of barcelona. *Omega*, 34(6):617–629, 2006.

[5] J. E. Beasley. A lagrangian heuristic for set-covering problems. *Naval Research Logistics (NRL)*, 37(1):151–164, 1990.

[6] J. E. Beasley. Or-library: distributing test problems by electronic mail. *Journal of the operational research society*, pages 1069–1072, 1990.

[7] J. E. Beasley. Lagrangean heuristics for location problems. *European Journal of Operational Research*, 65(3):383–399, 1993.

[8] J. E. Bell, S. E. Griffis, W. A. Cunningham, and J. A. Eberlan. Location optimization of strategic alert sites for homeland defense. *Omega*, 39(2): 151–158, 2011.

[9] H. Bernhard, B. Korte, and J. Vygen. Combinatorial optimization: Theory and algorithms, 2008.

[10] R. E. Bixby, J. W. Gregory, I. J. Lustig, R. E. Marsten, and D. F. Shanno. Very large-scale linear programming: a case study in combining interior point and simplex methods. *Operations Research*, 40(5):885–897, 1992.

[11] I. Boussaïd, J. Lepagnot, and P. Siarry. A survey on optimization metaheuristics. *Information Sciences*, 237:82–117, 2013.

[12] A. Caprara, M. Fischetti, and P. Toth. A heuristic method for the set covering problem. *Operations research*, 47(5):730–743, 1999.

[13] A. Caprara, P. Toth, and M. Fischetti. Algorithms for the set covering problem. *Annals of Operations Research*, 98(1-4):353–371, 2000.

[14] S. Ceria, P. Nobili, and A. Sassano. A lagrangian-based heuristic for large-scale set covering problems. *Mathematical Programming*, 81(2):215–228, 1998.

[15] V. Černỳ. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of optimization theory and applications*, 45(1):41–51, 1985.

[16] C.-H. Chen and C.-J. Ting. Combining lagrangian heuristic and ant colony system to solve the single source capacitated facility location problem. *Transportation research part E: logistics and transportation review*, 44(6): 1099–1122, 2008.

[17] R. Church and C. R. Velle. The maximal covering location problem. *Papers in regional science*, 32(1):101–118, 1974.

[18] J. Chuzhoy and J. Naor. Covering problems with hard capacities. *SIAM Journal on Computing*, 36(2):498–515, 2006.

[19] V. Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of operations research*, 4(3):233–235, 1979.

[20] I. A. Contreras and J. A. Díaz. Scatter search for the single source capacitated facility location problem. *Annals of Operations Research*, 157(1): 73–89, 2008.

[21] M. S. Daskin. What you should know about location modeling. *Naval Research Logistics (NRL)*, 55(4):283–294, 2008.

[22] J. Diaz and E. Fernández. A branch-and-price algorithm for the single source capacitated plant location problem. *Journal of the Operational Research Society*, 53(7):728–740, 2002.

[23] D. Erlenkotter. A dual-based procedure for uncapacitated facility location. *Operations Research*, 26(6):992–1009, 1978.

[24] R. Z. Farahani, N. Asgari, N. Heidari, M. Hosseininia, and M. Goh. Covering problems in facility location: A review. *Computers & Industrial Engineering*, 62(1):368–407, 2012.

[25] T. A. Feo and M. G. Resende. A probabilistic heuristic for a computationally difficult set covering problem. *Operations research letters*, 8(2):67–71, 1989.

[26] T. A. Feo and M. G. Resende. Greedy randomized adaptive search procedures. *Journal of global optimization*, 6(2):109–133, 1995.

[27] M. L. Fisher. The lagrangian relaxation method for solving integer programming problems. *Management science*, 50(12_supplement):1861–1871, 2004.

[28] M. L. Fisher, R. Jaikumar, and L. N. Van Wassenhove. A multiplier adjustment method for the generalized assignment problem. *Management Science*, 32(9):1095–1103, 1986.

[29] R. Gandhi, S. Khuller, S. Parthasarathy, and A. Srinivasan. Dependent rounding in bipartite graphs. In *Foundations of Computer Science, 2002. Proceedings. The 43rd Annual IEEE Symposium on*, pages 323–332. IEEE, 2002.

[30] R. Gandhi, E. Halperin, S. Khuller, G. Kortsarz, and A. Srinivasan. An improved approximation algorithm for vertex cover with hard capacities. *Journal of Computer and System Sciences*, 72(1):16–33, 2006.

[31] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979. ISBN 0716710447.

[32] G. Guastaroba and M. G. Speranza. A heuristic for bilp problems: the single source capacitated facility location problem. *European Journal of Operational Research*, 238(2):438–450, 2014.

[33] S. Guha, R. Hassin, S. Khuller, and E. Or. Capacitated vertex covering. *Journal of Algorithms*, 48(1):257–270, 2003.

[34] H. W. Hamacher and Z. Drezner. *Facility location: applications and theory.* Springer Science & Business Media, 2002.

[35] K. Hindi and K. Pieńkosz. Efficient solution of large scale, single-source, capacitated plant location problems. *Journal of the operational Research Society*, 50(3):268–274, 1999.

[36] S. C. Ho. An iterated tabu search heuristic for the single source capacitated facility location problem. *Applied Soft Computing*, 27:169–178, 2015.

[37] J. H. Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.* U Michigan Press, 1975.

[38] K. Holmberg, M. Rönnqvist, and D. Yuan. An exact algorithm for the capacitated facility location problems with single sourcing. *European Journal of Operational Research*, 113(3):544–559, 1999.

[39] S. Kirkpatrick. Optimization by simulated annealing: Quantitative studies. *Journal of statistical physics*, 34(5-6):975–986, 1984.

[40] J. G. Klincewicz and H. Luss. A lagrangian relaxation heuristic for capacitated facility location with single-source constraints. *Journal of the Operational Research Society*, 37(5):495–500, 1986.

[41] G. Lee and A. T. Murray. Maximal covering with network survivability requirements in wireless mesh networks. *Computers, Environment and Urban Systems*, 34(1):49–57, 2010.

[42] S. Li. A 1.488 approximation algorithm for the uncapacitated facility location problem. *Information and Computation*, 222:45–58, 2013.

[43] H. R. Lourenço, O. C. Martin, and T. Stützle. Iterated local search. In *Handbook of metaheuristics*, pages 320–353. Springer, 2003.

[44] M. Mahdian, Y. Ye, and J. Zhang. Approximation algorithms for metric facility location problems. *SIAM Journal on Computing*, 36(2):411–432, 2006.

[45] R. G. Michael and S. J. David. Computers and intractability: a guide to the theory of np-completeness. *WH Free. Co., San Fr*, 1979.

[46] T. Murata, H. Ishibuchi, and H. Tanaka. Genetic algorithms for flowshop scheduling problems. *Computers & Industrial Engineering*, 30(4):1061–1071, 1996.

[47] A. T. Murray and X. Feng. Public street lighting service standard assessment and achievement. *Socio-Economic Planning Sciences*, 53:14–22, 2016.

[48] A. T. Murray, M. E. OKelly, and R. L. Church. Regional service coverage modeling. *Computers & Operations Research*, 35(2):339–355, 2008.

[49] R. M. Nauss. An improved algorithm for the capacitated facility location problem. *Journal of the Operational Research Society*, pages 1195–1201, 1978.

[50] M. B. Rosenwein. Discrete location theory, edited by PB Mirchandani and RL Francis, John Wiley & Sons, New York, 1990, 555 pp., 1994.

[51] D. B. Shmoys, É. Tardos, and K. Aardal. Approximation algorithms for facility location problems. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 265–274. ACM, 1997.

[52] R. Sridharan. The capacitated plant location problem. *European Journal of Operational Research*, 87(2):203–213, 1995.

[53] B. Suman and P. Kumar. A survey of simulated annealing as a tool for single and multiobjective optimization. *Journal of the operational research society*, 57(10):1143–1160, 2006.

[54] C. Toregas, R. Swain, C. ReVelle, and L. Bergman. The location of emergency service facilities. *Operations Research*, 19(6):1363–1373, 1971.

[55] S. Umetani and M. Yagiura. Relaxation heuristics for the set covering problem. *Journal of the Operations Research Society of Japan*, 50(4):350–375, 2007.

[56] Z. Yang, F. Chu, and H. Chen. A cut-and-solve based algorithm for the single-source capacitated facility location problem. *European Journal of Operational Research*, 221(3):521–532, 2012.