A Multi Service Capacitated Facility Location Problem with Stochastic Demand

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

L.J. Kim (427684jk)

Supervisors:
Dr. D. Huisman, Dr. F. Phillipson & R. Wezeman

Co-reader:
Dr. O. Kuryatnikova

Erasmus School of Economics

12 March, 2021

Contents

1 Introduction

1.1 Motivation

The development of smart city planning and infrastructure is of high interest in the modern day. For example, there recently has been a great amount of enthusiasm in the introduction of 5G (5th generation) cellular networks, which are designed to perform with very low latency, higher speeds, availability and reliability compared to its predecessors. The number of internet connected devices per user is continuously rising. According to Connectivity and Mobile Trends Survey by Deloitte [11], the average household has 11 connected devices, of which 7 are smart screens to view online content. The survey also reports that "nearly half of consumers, and 62 percent of Gen Z, say they would find a wireless connection as fast as fiber broadband "very" or "extremely" valuable." This indicates that physical fiber connections could gradually be replaced almost completely by wireless connections, as improvements in wireless connections speeds are made. Common wireless services that are used today include the Wi-Fi internet access and of course, the telecommunication network. The majority of telecommunication services are currently wireless. As mobile phones have become a necessity, landline telephone usage has been on a decline. A demographic from 2018 provided by the U.S. Department of Health and Human Services [14] states that 55.2% of all adults in the U.S. reside in a home with only wireless telephones. It is also mentioned that this percentage decreased with age, with only 29.2% of adults aged 75 and over living with only wireless phones. These observations illustrate the growing demand in wireless services within the current generation.

However, to bring all this digital prosperity to the average user, a commercial supplier is met with the problem of designing service networks such that demand is fulfilled in the most cost efficient manner. A simple problem example is the distribution of public Wi-Fi routers in a city that can be utilized by households, instead of the traditional wired connection. These routers can be equipped in suitable areas throughout the city. A good option may be public lampposts. Each household in the city requires a certain amount of signal for daily usage. Installed routers can transmit signals to all households within a certain vicinity. It is undesirable to overload one router with an overwhelming number of households to service, because there is a limit on how much signal a single router can provide. Taking this restriction into account, we wish to meet the demand of all customers. However, demand for wireless services is not guaranteed to be equal every day in a household, which makes it it difficult to measure demand in a simple, deterministic way. This calls for the need of coverage in cases of uncertainties in consumer demand. There are two types of costs involved in this matter. The first is making a lamppost accessible for installation of any device, called opening costs. The second is the cost of installing a router on to a prepared lamppost, called installation costs. For the city and service provider, it is of interest on which lampposts the routers should be installed such that all households are satisfied with their service needs, but at minimal cost.

Nowadays, there are numerous types of wireless services such as Wi-Fi that needs to be delivered to consumers. In a smart city, the aforementioned decision problem need to be solved for several types of services. It is very possible that the optimal allocation of service installations of one type of service may depend on an already existing service. Let's say that other than Wi-Fi, a set of alarm devices need to be installed in the same city. It may be beneficial to install some alarm devices on lampposts that have been equipped with Wi-Fi routers, because there is no need to make the lamppost accessible for installation of devices again. We can denote the described problem as a capacitated multi-service distribution problem.

1.2 Research Question

Looking at the whole picture, we have a question of interest: *What is an efficient method to solve the capacitated multi-service distribution problem in case of stochastic demand?*. This is the main question that we will address in this thesis. Although methods to solve wireless network problems have been studied in the past, there is not a lot of research done in the setting of uncertainty in demand as well as multiple services to be handled. It is a question of increasing importance as technologies in fields such as data transmission continue to head towards wireless alternatives. In order to address it, we use several heuristic methods such as local search and simulated annealing.

Furthermore, we wish to extend this topic by adding realistic complications to the problem. It is possible that the further away a household is located, the more service capacity it takes to serve the household. Another possible scenario is that it costs more to install a router if it is highly occupied. The more signal one router needs to deliver, the more expensive. A sensitivity analysis on the problem using these variations on the cost model could be of interest. We will observe whether the computation times needed and final resulting costs increase significantly.

Finally, we take a look at an additional way of using our results of the first research question. The current capacitated multi-service distribution problem describes a scenario where the demand per household is defined for a single period of time. In real life, it is often the case that demand is variable across time; there may be more demand for service at a certain time period compared to the previous or the next. This scenario can be denoted as the multi period setting. The question *"Does the solution to the capacitated multi-service problem provide good results for a multi period setting?"* may be of practical interest, since it enables our results to be used for more realistic instances. We wish to observe whether the single period solution which takes stochastic demand into account is able to cover for the realized demand uncertainty in the multi period setting. The performance of our results can be validated by means of simulation.

1.3 Thesis Structure

We continue by elaborating on how the problem at hand is modeled in Chapter 2. Afterwards, we present the background research done for this study in Chapter 3. Before the designed methods are presented, linear programming formulations of the problem are given in Chapter 4. The explored methods and implementations of extensions are explained in detail in Chapter 5. The data that is collected to use for this research is described in Chapter 6, and the results are presented in Chapter 7. Afterwards, application of the methodology to a multi period setting is introduced in Chapter 8. Finally, we close this thesis by briefly looking back at our results and giving conclusions and suggestions for future research in Chapter 9.

2 Problem Description

In this chapter, we explain the problem we wish to tackle. We assume the problem setting described in Section 1.1, but with several types of services to distribute. The problem of allocating the installations for all services efficiently can be modeled as a Multi Service Capacitated Facility Location Problem, abbreviated as MSCFLP.

2.1 Multi Service Capaciated Facility Location Problem

The MSCFLP is a generalisation of the Capacitated Facility Location Problem (CFLP). In the CFLP, there are a set of customers with individual demand for a single product that need to be satisfied. The customers are denoted as demand points. There are multiple locations where facilities can be opened such that they can service the demand points. The locations where it is possible to open a facility are called access points. Once a facility has been opened at an access point, the location is called a service point. These facilities have a capacity on the amount of product that can be produced. The MSCFLP allows for multiple types of products. The problem consists of deciding where to open facilities such that operation costs are minimized. Two types of costs are incurred: the setup costs for an access point to be prepared to be a service point, and the installations costs to equip a prepared service point with a device. In this study, we model the problem of distributing wireless service installations across a city as a MSCFLP. The products correspond to wireless services. The demand points correspond to households and other users who have devices that need connection. An access location is a lamppost, whilst a facility is an installation. Instead of a limit on how much product a facility can make, we have a limit on the number of connections to devices one installation can make.

As an example, a small illustration is given in Figure 1. There are three lampposts, labeled left (L), center (C) and right (R). There are 5 houses in the picture. To keep things simple, we focus on just the first type of service. This can for example be Wi-Fi. The green circle represents the signal range of a Wi-Fi router installed on the lampposts. Currently, only lamppost L and R are equipped with routers, and are servicing houses 1 to 5. The possibility to install a router on lamppost C exists, but is likely to be more expensive since it will involve making lamppost C available for setup. But perhaps the value that lamppost C brings as an active service point will be greater the additional costs. The aim of the MSCFLP is to identify which lampposts to use as service points and which houses should be serviced by each service point. The main objective is to minimize the total costs, and if we have several viable options that deliver the same costs, we wish to choose the setup which spreads out the service loads smoothly over the used service points.

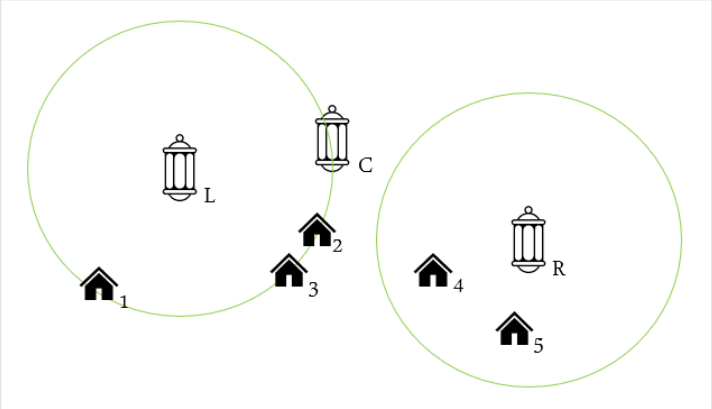


Figure 1: Basic Instance of a MSCFLP

The CFLP is in general NP-hard. The problem without capacity restrictions per facility, the Uncapacitated Facility Location Problem (UFLP), can be shown to be NP-hard via reduction from the Set Cover Problem (SCP), which is one of Karp's 21 NP-Complete problems, as listed by Karp [1]. The UFLP is a relaxation of the CFLP of the capacity restrictions, and is thus not more difficult than the CFLP. As a CFLP is a special case of the MSCFLP where the number of services equal 1, hence the complexity of the MSCFLP is NP-hard.

2.2 Demand Distribution

Often in a facility location problem, a demand point only needs to be covered once by a (single) service point, with no variable quantity in demand. Instead of this scenario, in our case each demand point has a certain level of demand that needs to be fulfilled. As mentioned, demand is modeled as number of devices requiring connection. For example, imagine one demand point has a smartphone, smart television and two laptops that need internet access. The demand for the service Wi-Fi corresponds to 4 for this demand point. The demand is discrete valued, since it is not possible to have a fraction of a device in need of service.

We assume that the demand is not given to be deterministic but has an underlying distribution. The assumption of a normal distribution is used, but other distributions such as the Weibull distribution, which can take a similar shape as the normal distribution with the correct choice of parameters, may also be fitting candidates. The Normal distribution is chosen based on certain features it has that makes it a realistic assumption. Characteristics of a normal distribution include (i) the values being concentrated around a mean value (ii) the distribution being symmetric. These are both plausible assumptions to have about behavior of demand for wireless services. We have a mean number of devices per demand point, and a conventional household is assumed to have a little more or less around this mean. The means may differ by a small amount per household.

An issue with the normal distribution is that it may in some cases simulate negative demand. Lin [9] also uses normally distributed data which in some cases may simulate negative values. Although in reality it is not possible to have a negative demand for a service, it is simply regarded as no demand at that location (which is a possibility in reality) since the deviations from zero are very minimal. Furthermore, if the standard deviation divided by the mean is considerably less than 1, there is a relatively low probability for negative demand values.

The normal distribution requires a mean and standard deviation parameter. In this study, the mean values of the assumed normal distribution is made to be dependent on the number of persons at a demand point. Not every household is assumed to be the same size, and a bigger household is expected to have more connected devices since it naturally has more users. The number of members in a household (corresponding to a demand point) is simulated based on estimated probabilities of it being 1, 2, 3, 4 or more than 5 people. The probabilities are estimated using actual household configuration data. The mean values for a demand point are decided based on both the number of persons and the service type, while standard deviations are only defined per service type.

2.3 Services

This study uses location data in two major cities of the Netherlands: Amsterdam and Rotterdam. Several random subareas are selected off the map as our instances. Public lampposts are assumed to be viable access points. When at least one type of service is installed on an access point, it is opened and becomes a service point. An opened service point can only service demand points within a certain range that differs per service type. Whether a demand point is

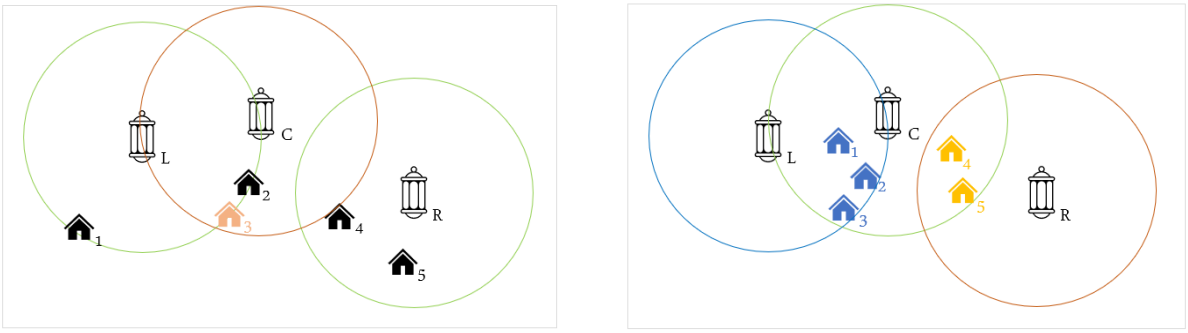
within range is decided by assuming a circular range with a predefined radius per service type. A service installation can only emit a certain amount of signal of a specific service type. This is taken into account by putting a limit on the number of devices that can be served. Finally, multiple service points can serve a demand point for the same service type.

We have three different service types: "Wi-Fi", "Alarm" and "Telecommunication Network". The Wi-Fi service provides internet access to devices connected to it. The Alarm service is a new service designed to keep track of bicycle theft. If a bicycle is mounted with a special smart lock, this service detects whether any of them are being broken. An alerting sound is made if any are detected. The Telecommunication Network service is designed to work just like cell towers: they deliver signals to mobile devices such that calls can be made between users. The Telecommunication Network service may not only used by residents at the households, but also everyone else who are at the demand points. Examples are potential guests and bypassers. The three services all have different signal ranges as well as maximum number of connections per installation.

2.4 Sensitivity Analysis

Until now, we assume that every demand point takes equal amounts of resources to service from a certain demand point as long as it is within range for the relevant service type and there is positive remaining capacity at the service point. In reality, there may be additional obstacles that influence the composition of the optimal final network. An example is the emitted signal strength not being equally strong throughout the whole radius of a service point. Another is that it may cost more energy to operate a service installation in case it is operating at high capacity, such that it may be cost wise favorable to open a different or extra service point to split the load. We demonstrate how the outcome may change with two different model alterations that may be necessary conditions representing real life behavior. The two scenarios considered are (i) the required capacity to serve a demand point varying with distance between the service and demand point, and (ii) service cost varying with occupancy of a service point.

An example of case (i) is shown in Figure 2a. The green circles indicate the signal radius of currently active service points. Assume for this example that a unit of needed capacity is added for every unit of distance the signal has to travel to a demand point.



(a) Case (i)

(b) Case (ii)

Figure 2: Two Different Cases of the MSCFLP

In the original case of Figure 1, lamppost L had enough capacity to service houses 1, 2, and 3. However, houses 1 and 2 are far away from the lamppost L, on the edge of its reachable range. In case (i), serving houses 1 and 2 now costs more capacity for lamppost L than in the original problem, such that it has no capacity left to service house 3. Given the range of

lamppost C marked by the orange circle, it is possible to service house 3 from lamppost C if it is opened for service. Therefore, it is necessary to equip lamppost C with service and let it service house 3. Next, take a look at Figure 2b showing a scenario of case (ii). Assume now that instead of needed capacity increasing with signal travel distance, the service costs increase as more connections are made with one installation. With the original problem, it is possible to service all houses from lamppost C, so only lamppost C would be opened. This is shown by the green circle. All five houses are being serviced by a single installation. Lets say for example, that a lamppost can service 3 demand points with ease and after that threshold, the costs for service are doubled. With the new type of cost calculation, the current allocation is expected to be very expensive. It may be profitable to open service points at lampposts L and R, so they can take over the service of all houses, but split among each other such that both lampposts service 3 or less houses. The blue circle depicts the range of lamppost L after opening, and the houses marked blue are taken over by lamppost L. Idem goes for lamppost R, but with its elements marked in orange. Lamppost C can then be closed as a service point.

The illustrated effects may occur in large real life instances. It is interest to be able to find good solutions with these extra conditions, as well as if the presence of them lead to big alterations to the resulting network.

3 Literature Review

In this chapter, we show the literary research done for this study.

3.1 Capacitated Facility Location Problem

The MSCFLP is a special case of the Capacitated Facility Location Problem (CFLP), which is a popularly studied topic. The classic CFLP is the decision of opening a number of facilities such that all demand points are serviced and costs are minimized. The demand points are normally considered to have singular demand: one demand point simply needs to be covered by an open facility, and all demand points do not have different levels of demand. A common variation to this is that the volume of demand can vary per demand point, but is fixed and known beforehand.

A well studied case of the CFLP is the Single Source Capacitated Facility Location Problem (SSCFLP). In this scenario, a demand point is serviced by a single facility. Several solution approaches have been proposed for the SSCFLP. For example, Holmberg et al. [2] propose a Lagrangean heuristic that consists of two components. After a mathematical model is formulated, a Lagrangean relaxation in combination with subgradient optimization is used to obtain lower bounds of the objective. Then, a heuristic is used to find primal feasible solutions which also are the upper bounds. The found lower and upper bounds are subsequently used in a branch and bound framework in order to find the optimal objective. Baldacci et al. [3] give a reformulation of the original problem as a set covering formulation. The set covering formulation looks at possible clusters of demand points, such that demand points within a cluster are covered by the same service point. The resulting form cannot be solved directly due to a large number of possible cluster combinations. Several heuristics are employed to obtain upper bounds and solutions to the dual of the relaxed set covering problem. With the upper bound and the dual solutions, the set covering formulation is reduced by defining a smaller set for the solution space. The found objective values have been found to be sub optimal in many of the test instances, but intermediate upper and lower bounds give an idea on how far the current solution is from optimality. A similar idea of grouping demand points is used by Díaz and Fernández [19]. A branch and price algorithm is introduced for the SSCFLP which consists of two stages: deciding what service points to open, and given the opened service points which demand points to assign to each service point. They define a concept of "allocation pattern", which links a service point with a number of demand points it shall service. Given a subset of open service point and allocation patterns, column generation is used to find what allocations patterns to use is best. The pricing problem finds the best allocation pattern for each open service point. Their biggest test instance consists of 30 possible access points and 90 demand points, for which good results are found within short computation times. As a final example we look at Ahuja et al. [4]. Instead of attempting to solve the problem, relaxations of it or subproblems using simplex methods, a large neighborhood search is conducted on a given initial solution. The neighborhoods that are explored include the opening and closing of new facilities, reassignment of demand points from one facility to another. The authors comment that this algorithm outperforms Lagrangean heuristics used by other studies such as Hindi and Pienkosz [18] in terms of average deviations from the expected optimal value. The computation times appear fast for instances with 100 access locations and 1000 demand points.

If a single demand point can be serviced by different service points simultaneously, we have a Multi Source Capacitated Facility Location Problem (also abbreviated as MSCFLP), also commonly called the Multi Source Capacitated Plant Location Problem. In such an instance, there are volumes of demand per demand point that can be split among several service points. Chyu and Chang [15] present two local search heuristics for this problem: The Simulated Annealing method (SA-method) and Variable Neighborhood Local Search (VNLS). The SA-method gener-

ates neighbors of the initial solution using both random selection and average costs assessment of currently opened/closed facilities. Simulated annealing is incorporated to explore a bigger search space. VNLS defines three different neighborhood structures and assesses whether any of them generate better solutions. If none are found, new neighborhoods are generated. Neighborhoods returning bad solutions are kept track of by incorporating tabu search. The VLNS outperforms the SA-method and the exact approach using a commercial solver in most instances.

3.2 Multi Service Capacitated Facility Location Problem

Considering multiple service types for the CFLP gives us the Multi Service Capacitated Facility Location Problem. Although facility location problems with single service have been intensively studied, the topic of MSCFLP, or Multi Service Facility Location Problems in general, is relatively new. One of the earliest mentions of a multi-service instance is looked into by Suzuki and Hodgson [8]. A single source model is described where a service point can offer three types of human services. The number of facilities that are open is decided beforehand. Demand is described as a discrete set of demand points which need to be covered. Suzuki and Hodgson use a p-median model, which minimizes the distance between demand points and their closest open facility. The problem is solved with the p-median model for all three services separately, which appears to give sub optimal results. Facilities for different services that could operate from the same location are built separately close to each other. It is attempted to improve the solution by introducing different types of fixed opening costs to the objective function.

In the case of wireless services with only fixed location opening costs and installation costs, the distance between service and demand points are irrelevant as long as the demand point is within range of the service point(s) it is serviced by. A series of intensive research about the MSCFLP for wireless services has been conducted by or in partnership with TNO. As of recently, Veerman [10] proposes three heuristics to tackle the problem at hand: Extended Pricing Heuristic (EPH), Extended Linear Relaxation Heuristic (ELRH), Extended Sequential Covering Heuristic (ESCH). All three heuristics have a structure of (i) Finding an intermediate solution, (ii) Using the intermediate solutions to reduce the solution space for an exact mixed integer programming approach. The EPH starts by using a value-cost ratio of opening a service point to construct a feasible solution in a greedy fashion. Using the obtained open locations from this intermediate solution, the MSCFLP is solved within a reduced solution space. The ELRH assumes that that the LP relaxation of the MSCFLP is much easier than with the integrality constraint. It solves the LP relaxation first, and again uses the intermediate open locations of this results to solve the MSCFLP with integrality constraints in a smaller solution space. The results show that all solution methods terminate either due to the optimality condition or the optimality gap termination criteria. The setup incorporates a modularity parameter κ which controls how many access points can be installed at a single service point of a certain type of service. It is reported in the results that with $\kappa = 1$, the exact method returns a smaller optimality gap, but in case $\kappa > 1$, the heuristics return smaller gaps. Although these methods return a solution within more favorable computation times, there is no research yet on how they could be modified in case of stochastic demand.

In Vos and Phillipson [17], a similar problem is tackled. Here, a network needs to be designed which is able to service all demand points in the area, and the service points are to be connected to an existing network. The problem is split into two sub problems. As a starting problem the Multi Service Location Set Covering Problem (MSLSCP) is introduced. This is the problem of distributing a set of services on predetermined potential service points such that all demand points for each service are satisfied, while keeping costs as low as possible. The methods employed to solve the MSLSCP include the exact formulation using a solver and several heuristics. Three different heuristics are proposed: Sequential Set Covering Heuristic (SSCH), Likelihood

Heuristic (LH), Connection Heuristic (CH). The SSCH performs a set covering heuristic for each of the services, whilst adjusting the fixed costs in each step when a location is opened. The LH uses a likelihood measurement equal to a ratio of number of potential connections to demand points over the opening costs of a location. These ratios are sorted in decreasing order such that the locations with highest ratio are selected to be opened. The CH prioritizes guaranteeing service to demand points which do not have many access locations in its area. It activates a service point which is able to reach such lone demand points while providing the most potential to service other demand points. After such a service point is found, the next service point to open is found by searching for an access location which can service the most demand points for the same service type. These steps are repeated until all demand points are covered. The solutions resulting from the heuristics are then put through an improvement step which attempts to find a better solution with an exact algorithm, comparable to Veerman [10]. In Vos [7], an additional problem is solved on top of the MSLSCP. Vos [7] uses the concept of 'hubs' - a distinction is made between service points which have a wired connection with an existing network and those which have a wireless connection with the points that have a wired connection. The service points which are directly connected to the networks are called 'hubs', and are assumed to have higher equipment costs than opening a service point active on a wireless connection from one of these hubs. What service points are to act as hubs are determined by formulating the Wireless Network Problem (WNP). This is the second component of the main problem. The WNP takes the set of active lampposts selected by the MSLSCP and assigns some of them to be a hub. The problem has to keep certain restrictions such as bandwidth restrictions and maximum number of possible hubs into account. Techniques selected to find a solution include Greedy Randomized Adaptive Large Neighborhood Search, Iterated Local Search, Simmulated Annealing and Genetic Algorithms. These heuristic methods seem to perform with fast computation times for most instances, but do not incorporate having certain quantities of demand per demand point.

Hoekstra [5] works with a model which does incorporate quantity in demand. It is measured in number of connections, which is the most similar to the situation that is going to be looked into. The methods mentioned are the Sequential Solving Heuristic (SSH) and Ordered Sequential Solving Heuristic with Updating (OSSHU), where the latter extends the prior method. The SSH splits the multi-service problem into single service subproblems, and combines the solutions per service into a solution for the MSCFLP. The single service subproblems are solved exactly. The OSSHU works in a similar manner, except that the change in opening costs are adjusted to zero for opened access points after each service is optimised. Looking at the results, the heuristics appear to have little advantages in computation time for small instances, but has a noticeable gap in objective value with the exact method. With bigger instances, the difference in objective values decreases but the computation time is longer for the heuristic than the exact approach. There is again no stochasticity involved in the demand.

3.3 Facility Location Problems with Stochastic Demand

Most studies conducted about the facility location problem use the assumption of known, deterministic demand. As mentioned in Chapter 1, instances with stochastic demand are worthy of research for applications in practice. Lin [9] presents a SSCFLP with stochastic demand. Two distributions for demand are considered. The first distribution is the Poisson distribution, where each demand point has a known mean. The Poisson distribution is often used to model arrival processes. Lin [9] mentions the application examples the request for medical/emergency service and incoming calls for a telecommunication service. The second distribution is the normal distribution with known mean and standard deviations per demand point. Lin [9] then proposes a heuristic called H-SSSCFLP. H-SSSCFLP sequentially solves the problem multiple times with a given number of facilities that are open on each run. A lower bound on the number of facilities to open is estimated, and with it a feasible solution is constructed, of which the objective value

is considered an initial upper bound. The algorithm then continuously determines whether it should open more facilities or close open facilities depending on the changes in upper bound as the relaxed model is solved with the current number of open facilities. The H-SSSCFLP shows good performance, with all test instances except one returning an optimality gap of less than 5%, and the exception having a gap of 6.51%. The longest taking instance is solved in approximately 28 hours. However, the algorithm is developed for a single source case, and the test instances are quite small, with the largest number of demand points being 89.

An exact formulation is presented by Verhoek [6] on a MSCFLP with stochastic demand. The setting of the research is a scenario that has stochastic demand based on data collected for a major 3G network system in Mexico. The demand in this study is measured in connection speed, and can be provided by multiple sources. The thesis covers three demand situations: a single service is provided, multiple services are provided, and multiple services are provided and active sensors are added to the service points. It attempts to satisfy the demand at a given reliability level by incorporating it into a constraint in the mathematical program. After the problem is defined, we attempt to reduce the problem in size with procedures including the removal of redundant constraints and variables, checking for infeasible cases in the primal and dual form, and shrinking the feasible region by adding cuts. An upper bound on the solution is predetermined with a heuristic of choice, after which a branch and bound algorithm is employed in order to find the optimal solution. Although an advantage of the exact approach is solution quality, the computation times in most cases appear to be quite long. One of the instances could not be solved on time in the single service setting, and in the multi service settings some other instances could not be solved for a reliability level of 0.95.

The problem studied in this thesis resembles that of Verhoek [6], whose exact approach suffers of high computation times. This topic therefore still requires further research in order to find methods that deliver good solutions within feasible time. Successful methods in the covered literature tackling similar problems may be good alternatives if adjusted to be suitable for the problem at hand. Adopting elements of the branch and price method by Díaz and Fernández [19] may be viable, but the mathematical models will need to be changed to allow demand points to be served by multiple sources. The multi source aspect makes the use of this method difficult since it makes the formulation of a suitable pricing problem very difficult. Another approach we could consider is the neighborhood search of Chyu and Chang [15]. Local search methods are generally very fast and can easily deliver intermediate solutions, but have less guarantee on optimality. If problem instances get very large, it may be favorable to employ such a local search heuristic over other methods.

4 Mathematical Model

In this chapter, the mathematical model describing the problem at hand is presented.

4.1 Parameters and Variables

Tables 1, 2 and 3 show the notation of sets, parameters and variables that are used to describe the base mathematical model respectively.

Set	**Description and Indices**
S	The set of services: Wi-Fi (1), Alarm (2), Telecommunication Network (3), $u \in S$
L	The set of potential service locations, $j \in L$
D^u	The set of demand points for service u, $i \in D^u$
$\mathbb{H}^u$	Combinations of (i, j) for which demand point i can be reached by service point j with service type u.

Table 1: List of Sets

Parameter	**Description**
a_{ij}	Binary value which equals value 1 if demand point i can be reached from service point j in terms of distance, independent of how occupied j is.
f_j	Fixed cost of opening service location j
c_j^u	Cost of installing service type u at service point j.
d_i^u	Demand for service type u at demand point i
N^u	Service capacity for one installation of service type u, in number of connections available.
α	Reliability level to meet demand.
M	Arbitrarily large constant.
μ_i^u	Mean number of wireless devices in a household i requiring signal service type u.
σ^u	Standard deviation of number of wireless devices per household requiring signal service type u.

Table 2: List of Parameters

Variable	Description
y_j	$\begin{cases} 1 & \text{if service point } j \text{ is opened} \\ 0 & \text{else} \end{cases}$
x_j^u	$\begin{cases} 1 & \text{if service point } j \text{ is equipped with service type } u \\ 0 & \text{else} \end{cases}$
z_{ij}^u	Numerical value representing the number of connections to devices for service type u provided by service point j to demand point i

Table 3: List of Variables

4.2 MSCFLP with Stochastic Demand

$$\min \quad \sum_{u \in S} \sum_{j \in L} c_j^u x_j^u + \sum_{j \in L} f_j y_j \tag{1}$$

$$\begin{aligned}
s.t. \qquad x_j^u &\leq y_j && \forall j \in L, \forall u \in S && (2)\\
\textstyle\sum_{i \in D^u} z_{ij}^u &\leq N^u x_j^u && \forall j \in L, \forall u \in S && (3)\\
P(\textstyle\sum_{j \in L} z_{ij}^u \geq d_i^u) &\geq \alpha && \forall i \in D^u, \forall u \in S && (4)\\
x_j^u &\in \{0,1\} && \forall j \in L, \forall u \in S && (5)\\
y_j &\in \{0,1\} && \forall j \in L && (6)\\
z_{ij}^u &\in \mathbb{N} && \forall \{i,j\} \in \mathbb{H}^u && (7)
\end{aligned}$$

Above equations (1) to (7) describe the basic MSCFLP. The problem minimizes the total costs for opening the required allocation points to become active service points and the costs incurred to install the wireless services on to these points, summarized by equation (1). Constraints (2) only allow service installations to be equipped in places which are open to be active. At the same time, it also ensures that only one installation of service type u can be made at location j, since the value of y_j is bounded from above by 1. Constraints (3) make sure that an installation cannot deliver signal to more devices than it is compatible with. The demand requirement is enforced by constraints (4), with $\alpha = 0.95$ in this study. Finally, constraints (5), (6) and (7) define the space of the decision variables. All variables in our model are considered to be integer valued, because the demand and service units are measured in number of devices. Notice that the variables z_{ij}^u can only exist in case demand point i can be reached by signal type u from service point j. In order to decrease the number of unnecessary variables, we define a separate set $\mathbb{H}^u$. The set $\mathbb{H}^u$ contains all combinations $\{i,j\}$ where the demand point i is within signal range of service point j for service type u.

Currently, constraints (4) contain a probability term which cannot be directly implemented in to a solver. However, it is possible to rewrite constraints (4) into a more easily implementable form. We take a look at a method for modelling with uncertain variables in case of the normal distribution. Postek [12] explains that "a safe way to implement our 95% constraint satisfaction requirement is to make sure that the 95% quantile of this random variable is less than or equal to b_i", where the random variable in our case is the demand level at each demand point. This is a common trick with unknown variables which follow a distribution with known parameters. In mathematical notation, the following relation is brought to light:

$$a^{iT}x + q_{0.95}\sqrt{(A^{iT}x)^T \sum A^{iT}x} \leq b_i \rightarrow \mathbb{P}((a^i + A^i z)^T x > b_i) \leq 0.05 \tag{8}$$

The notation $q_{0.95}$ refers to the 95^{th} percentile of the standard normal distribution. This is for the case where the constraint to be satisfied is formulated as $(a^i + A^i z)^T x \leq b_i$, with z being the variable containing the uncertainty, x the decision variable, and a^i, b^i, A^i known parameters. In this example, $z \sim N(0, \Sigma)$; if the mean were not to be zero which is likely in this case, the mean μ can be easily taken out and added as $A^i\mu$ to the constant a^i. The property of linear transformation of normally distributed random variables is used here. The random variable z undergoes a linear transformation in the form of $cz + b$ where $c = A^i x$ and "constant" $b = a^i x$ is added. Relation 8 suggests that we can ensure a similar constraint in our IP to ensure the demand is covered for 95% certainty.

Given that $d_i^u \sim N(\mu_i^u, \sigma^u)$, we can derive a similar relation. This simplifies the probability containing constraint into a linear constraint which is comparable to the case of deterministic

demand. In the linear transformation of random variable d_i^u, a "constant" of $\sum_{j \in L} z_{ij}^u$ is deducted. Following relation (8), we can enforce constraints (4) using the following reformulation instead:

$$\mu_i^u + q_{0.95} \sqrt{(\sigma^u)^2} \leq \sum_{j \in L} z_{ij}^u \tag{9}$$

By doing this, we have translated the stochastic constraint into a deterministic one, which is easy to implement into a commercial solver. The size of this problem grows quickly with the increase in elements in S and D^u, with every additional service type u adding $|S| \times |D^u|$ variables and $3|L| + |D^u|$ constraints. In practice, the sets D^u and L are also expected to be large, and in combination with the fact that CFLPs are in general NP-hard, we cannot expect a direct IP input into a solver to return (good) results in reasonable time. In the next chapter, we introduce a heuristic method that may deliver good results much faster.

5 Solution Approach

As mentioned previously, solving the whole mathematical model using an exact approach is expected to result in undesirably long computation times for large instances. However, the results of Hoekstra [5], who uses the same mathematical model, suggest that the model solves in very short time in case of small instances. The exact evaluation took less than 1 second for less than 100 access locations (and around double the amount of demand points) with a 0.0% gap, and for around 100 access locations a solution with a <5% gap was found within 1 second. Using this observation and the idea of neighborhood search, we propose an Adaptive Large Neighborhood Search with Exact Subproblems (ALNS-ES). In this chapter, we briefly introduce the concept of Adaptive Large Neighborhood Search, followed by details on the ALNS-ES algorithm.

5.1 Adaptive Large Neighborhood Search

The general Adaptive Large Neighborhood Seach (ALNS) framework is an extension of Large Neighborhood Search (LNS). LNS describes a neighborhood search method starting with an initial feasible solution, and iteratively attempts to improve the current solution by destroying and repairing parts of it. This method explores a much bigger search space compared to generic local search methods which modify a small part of the solution. It keeps track of a current best solution by evaluating the objective values of found neighborhoods. The ALNS involves the use of multiple destroy and repair methods. Which destroy and repair methods are used at each turn are determined by probabilities of choosing a certain method pair, which are adjusted throughout the algorithm after each iteration. The probabilities of being selected depend on chances of success in improving the solution in the past iterations. The algorithm continuously explores neighborhoods that are found via successive uses of destroy and repair methods, until a stopping criterion is met. There are two commonly used stopping criteria: the number of iterations and the number of consecutive iterations performed without finding an improvement.

The ALNS-ES algorithm uses the ALNS framework with a simulated annealing component, tabu search and two types of objective values. A candidate solution is a set of open access points with service allocations connecting service points with demand points that satisfy the demand requirements of the problem instance. An overview of the whole algorithm is given by Algorithm 1.

Algorithm 1: ALNS-ES

```
Initialize costs f_j, c_j, demand points D^u, service points L;
Sort services types S in ascending order of service range;
for u ∈ S do
    Initialize temperatures T_c, T_b, cooling ι and weights w_k.
    noChange ← 0;
    tabu_1 = ∅, tabu_2 = ∅
    initialSolution ← constructiveHeuristic();
    bestSolution ← initialSolution;
    currentSolution ← bestSolution;
    while Stopping criteria not met do
        Choose destroy method destroy_k() with probability p_k;
        neighborSolution ← destroy_k(currentSolution);
        neighborSolution ← repair(neighborSolution);
        if Best cost solution found then
            currentSolution ← neighborSolution;
            bestSolution ← currentSolution;
            noChange ← 0;
        else
            if Better balance solution found then
                currentSolution ← neighborSolution;
                noChange ← 0;
            else
                p_accept ← max{exp(−Δ_c/T_c), exp(−Δ_b/T_b)}
                if Solution accepted then
                    currentSolution ← neighborSolution;
                    noChange ← 0;
                else
                    noChange ← noChange + 1;
                end
                T_c ← ιT_c;
                T_b ← ιT_b;
            end
        end
        update w_k;
    end
    return bestSolution;
    update f_j;
end
```

We now take a look at the steps in Algorithm 1. In order to start the search process, we need to define a starting point. The starting point is a feasible solution which is not necessarily (and most likely not) optimal. An initial feasible solution is found using the constructive heuristic described in Algorithm 2, denoted as *constructiveHeuristic()* in Algorithm 1. In practice, the design of the constructive heuristic is not of importance as long as it is able to deliver a feasible allocation. The term "accessibility" refers to how many service points for service type u a demand point has as possible connections, based on the distance between them. The algorithm attempts to connect the demand points by allocating as much demand as possible to the next service point in the list of nearby service points B. If a service point is selected for the first time, it is opened and added to O, the set of open service points. Demand quantities of a single demand point may be split among service points, if the first service point in B cannot

accommodate all remaining demand of a demand point. If a whole list B is traversed through and the remaining demand that is not allocated to a service point is higher than zero, the demand point is labeled as unsatisfied and the heuristic fails to return a feasible initial solution.

Algorithm 2: Constructive heuristic

```
O ← {∅}
Sort all demand points d ∈ D^u in order of ascending accessibility.
for d ∈ D^u do
    Retrieve set of nearby service points B ∈ L
    remainingDemand ← Remaining demand of d
    demandMet ← 0
    for s ∈ B do
        remainingCapacity ← Remaining capacity of s
        if remainingCapacity > 0 then
            O ← O ∪ s
            minVal ← min(remainingDemand, remainingCapacity)
            if minVal > 0 then
                remainingCapacity ← remainingCapacity - minVal
                remainingDemand ← remainingDemand - minVal
                demandMet ← demandMet + minVal
            end
        end
    end
    if demandMet < totalDemand then
        No service point can be selected, failed to create solution
    else
        return L
    end
end
```

5.2 Destroy Methods

Next, the created initial solution undergoes a iterative destruction and reparation process. ALNS-ES uses two destroy methods. The destroy methods are denoted as $destroy_k()$ in Algorithm 1, with $k = 1, 2$.

The first type of destroy method chooses a main demand point i_M randomly and removes all of its current connections to service points. For each disconnected service point, its links to all other demand points are also destroyed. The motivation behind this destroy method is that it is perhaps possible to find more efficient allocations within a certain sub-network, and doing it for many sub-networks can get the solution closer to the optimal allocation for the whole region. The results of Hoekstra [5] for running the SSH for small instances suggests that if we make our sub-regions contain around 100 access locations, they can be solved in very little time.

The first group of connections to destroy are found by looking at the chain of $i_M \Leftrightarrow$ service point $\Leftrightarrow$ demand point. Denote this sequence as the "base chain". The demand point i_M is the start of this base chain, and the other demand point is the end of this base chain. From a demand point at the end of the base chain, we can see what service points are connected to it and consequently what other demand points are connected to these service points. The chain can then be extended from the demand point, adding all service points it is connected it and all demand points connected to these service points. This can be done multiple times, as every time

we extend a chain, we have a new demand point at the end of the chain. Denote by "depth" the number of service points and demand points explored past an ending demand point of a base chain. An illustration is given in Figure 3.

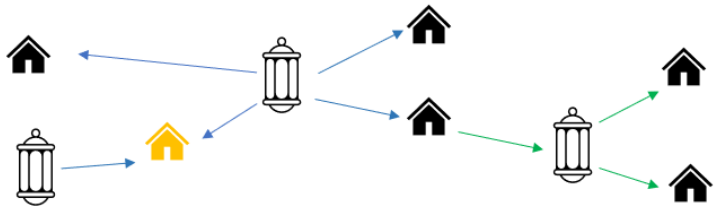


Figure 3: A Base Chain with Depth = 1.

Given that the yellow house represents i_M, and the blue arrows denote the base chain. One of the demand points at the end of the base chain are extended with 1 depth, indicated by the green arrows. The depth of destruction depends on the environment of selected i_M. We start with depth = 0, and look further with higher depth until the total number of access locations within reach of the un-allocated service points exceeds a certain threshold. This threshold is a random number between β_1 and β_2, with around 100 expecting to result in short computation times. Different choices of β_1, β_2 may be eligible and are also explored in this study.

The pseudocode of $destroy_1()$ is given by Algorithm 3.

Algorithm 3: Destroy method 1

$n_D \leftarrow 0$
$n_S \leftarrow 0$
$min_S \leftarrow 80$
$limit_D \leftarrow 300$
$limit_S \leftarrow$ *random selection* $\in [\beta_1, \beta_2]$
$queue = \emptyset$
$i_M \leftarrow$ random selection $\in D^u \setminus tabu_1$
Add i_M to $tabu_1$
Add i_M to *queue*. **if** $|tabu_1| > 100$ **then**
  Remove first element of $tabu_1$
end
while $n_S < min_S$ **do**
  while $n_D < limit_D$ *and* $n_S < limit_S$ *and* $n_D < |queue|$ **do**
    $n_D \leftarrow n_D + 1$
    for *Every service point that is connected* s **do**
      Remove service connection between s and $(n_D + 1)^{th}$ element in *queue*.
      $n_S \leftarrow n_S + 1$
      for *Every service connection* d *of* s **do**
        if $d \notin queue$ **then**
          Add d to *queue*.
        end
      end
    end
  end
  Select new $i_M \notin tabu_1$
  Add i_M to *queue*.
end

The counter variables n_D and n_S denote the number of demand and service points that have been disconnected respectively. The parameters $limit_D$ and $limit_S$ decide the maximum number of demand and service points to disconnect, such that the IP instance for the subproblem does not grow too large. For the number of demand points, this is set to 300. For the service points, it is randomly selected between the upper bound $UB_{destroy_1}$ and lower bound $LB_{destroy_1}$ parameters, which are defined by the user. The list *queue* contains the demand points of which the service connections are to be explored. This includes not only the demand points that are in the chain of i_M, but a new i_M that is not in tabu list $tabu_1$ is selected and not put in the queue if after one destruction process based on the initial i_M, less than min_S service ponts have been disconnected. This insures that if we happen to select an i_M that is part of a very small chain, the algorithm does not waste an iteration solving a IP that is too small. For example, i_M is connected to 2 service points and these 2 service points are not connected to any other demand point, the IP will have an input dimension of 1 demand point and 2 service points. This is a likely scenario especially in the first iteration directly after the constructive phase.

The second destroy method chooses a random service point j_M and destroys all links between several service points and their corresponding demand points that are close to j_M. The 300 service points closest to j_M are identified (or the maximal amount of service points in case the problem instance contains less than 300 service points in total) in the instance initialization step and a number between β_1 and β_2 of them are disconnected. Again, favorable values β_1, β_2 are expected to lie around 100 as suggested by results of Hoekstra [5]. The parameter values β_1, β_2 used are the same as in the first destroy method such that solution repair times remain similar. The motivation behind this selection of service points is that after the repair method, the demands served by j_M can be taken over by the service points near j_M such that j_M can be closed.

The pseudocode of $destroy_2()$ is given in Algorithm 4.

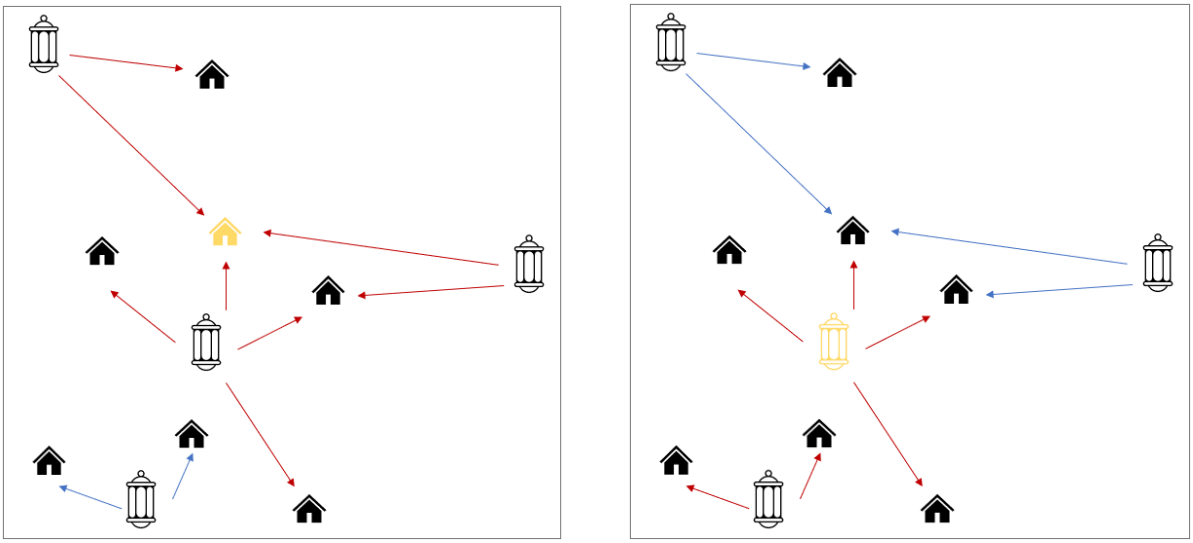
Algorithm 4: Destroy method 2

$n \leftarrow 0$
$limit \leftarrow$ *random selection* $\in [\beta_1, \beta_2]$
$j_M \leftarrow$ *random selection* $\in L \setminus tabu_2$
Add j_M to $tabu_2$
if $|tabu_2| > 100$ **then**
  Remove first element of $tabu_2$
end
for *Every demand point d that is connected to* j_M **do**
  Remove connection between d and j_M
  $n \leftarrow n + 1$
end
while $n < limit$ **do**
  $s \leftarrow$ next closest service point to j_M
end
for *Every demand point d that is connected to s* **do**
  Remove connection between d and s
  $n \leftarrow n + 1$
end

In both destroy methods, the starting points i_M and j_M are picked randomly. It can occur that the same service or demand point is selected multiple times almost consecutively by chance, causing the algorithm to repeat unnecessary iterations which return the same result. In order to avoid such behaviour, a tabu list is employed for both the selection of i_M in the first destroy

method and j_M in the second. Once a point has been selected, it is put in a tabu list which the algorithm does not choose as i_M or j_M for the next 100 iterations. Tabu lists are denoted as $tabu_1$ and $tabu_2$ in Algorithms 3 and 4.

The main difference between this destroy methods $destroy_1()$ and $destroy_2()$ is that in the first destroy method, some service points within the considered sub-region around i_M may not be disconnected, whilst in the second destroy method all service points within a certain distance of j_M are disconnected. This is illustrated in Figure 4. Let us say that a current solution has the connections denoted by arrows in both sub-figures. Again, the lamps represent service points and houses represent demand points. The yellow house represents the i_M selected with the first destroy method in Figure 4a. In this figure, the arrows marked in red denote the connections to be destroyed using the first destroy method. This consists of the connections directly to i_M, and the connections made by service points directly connected to i_M to other demand points. The two connections made by the service point at the very bottom are not removed. In Figure 4b, a selected j_M is marked yellow. Let us say that for this small example, instead of between β_1 and β_2 service points, maximally 2 service points are disconnected including j_M. The connections to be destroyed using the second method are also marked in red in Figure 4b. The connections made by j_M are destroyed, and we look for the closest service point to j_M, corresponding to the service point at the bottom. The two connections made by the service point at the bottom are also destroyed. Because we reached the limit of 2 service points, the connections made by the two service points located higher than j_M in the figure are not destroyed.



(a) Destroy Method 1

(b) Destroy Method 2

Figure 4: Differences in Connections Destroyed Between the Two Destroy Methods.

Which of these two destroy methods are to be employed depends on a probability p_k for method k. The probability is described by Equation (10).

$$p_k = \frac{w_k}{\sum_{l=1}^{2} w_l} \tag{10}$$

The weight of selecting a certain destroy method w_k is calculated as:

$$w_k = \delta w_k + (1 - \delta)\Psi \tag{11}$$

The δ is a decay parameter with a value between 0 and 1. The Ψ is an outcome of a score

function which is determined as:

$$\Psi = max \begin{cases} 20, \text{ if the new solution is the best obtained so far in terms of costs} \\ 15, \text{ if the new solution is better than the current solution in terms of balance} \\ 10, \text{ if the new solution is accepted} \\ 5, \text{ if the new solution is not accepted} \end{cases} \tag{12}$$

Weights are initialized to be 10 in the default case for both methods such that a solution accepted in the simulated annealing process does not increase the initial weight of the destroy method used.

5.3 Repair Method

Both destroy methods are followed up by the same type of repair method, denoted by *repair*() in Algorithm 1. The two destroy methods result in incomplete solutions with demand points which are either completely or partially not satisfied and service points that are completely disconnected from their previous customers. Recall that we have made the stochastic demand element deterministic using Equation (9), therefore our instance has known individual demand values per demand point. For partially unsatisfied demand points, it can be easily calculated how much demand is unsatisfied by looking at how much service was provided to them before destruction, and subtracting it from their total demand. Then we solve a mixed integer program as described in Chapter 4, but for the smaller sub-regions:

$$\min \quad \sum_{j \in L'} c_j^u x_j^u + \sum_{j \in L'} f_j y_j \tag{13}$$

$$\begin{aligned}
s.t. \qquad x_j^u &\leq y_j && \forall j \in L' && (14)\\
\textstyle\sum_{i \in (D^u)'} v_{ij} &\leq N^u x_j^u && \forall j \in L' && (15)\\
\sum_{j \in L'} v_{ij} &\geq min\{\mu_i^u + q_{0.95}\sqrt{(\sigma^u)^2}, \sum_{j \in L'} \zeta_{ij}\} && \forall i \in (D^u)' && (16)\\
x_j^u &\in \{0,1\} && \forall j \in L' && (17)\\
y_j &\in \{0,1\} && \forall j \in L' && (18)\\
v_{ij} &\in \mathbb{N} && \forall \{i,j\} \in \mathbb{H}' && (19)\\
& && && (20)
\end{aligned}$$

The notation L' represents all service points that have been affected and $(D^u)'$ is the collection of demand points that have been affected by the destroy methods. The ζ_{ij} values correspond the amount of service provided by service point j to demand point i before the destroy method. Variable v_{ij} is the amount of service provided by service point j to demand point i by the repair method. Constraints (14) again ensure that service points are active only if the locations are opened first. Constraints (15) check that the capacities of service points are not exceeded by the new allocation. Constraints (16) are the adjusted demand satisfaction constraints. After the destroy method has been executed, all or a part of demand point i's demand is not satisfied. The amount that is not satisfied is given by the sum of service that was provided by service points that have been disconnected by the destroy method, equal to $\sum_{j \in L'} z_{ij}$. In case where a demand point has been completely disconnected, this corresponds to $\mu_i^u + q_{0.95}\sqrt{(\sigma^u)^2}$. As a safety measure in case the previous allocation provides more service to a demand point than necessary (since excess service provided does not affect costs, it's permitted as long as it doesn't

exceed the capacity constraint), we take the minimum of these two values. Constraints (17) to (19) define the decision variables. Set $\mathbb{H}'$ is defined the same way as $\mathbb{H}$ in Chapter 4, but limited to the pairs i, j that are in $(D^u)'$ and L'. A time limit of 5 seconds is enforced for the MIP subproblem, such that the theoretical maximum running time for the solely solving the subproblems is 25 minutes. In a case where the demand points and service points selected for the subproblem make it difficult for the exact IP to be solved to optimality, there are two options. The first is that the IP has found a feasible IP solution, which may not be optimal, but is the best solution so far. The subproblem then returns this solution as the best solution found. The other is that the IP has not been able to find a feasible IP solution. In this case, the IP does not return a solution and the algorithm reverts to the current solution before destruction. In practice, most subproblems are solved within 500 milliseconds, and almost all subproblems are solved within 1 second.

5.4 Improvement Step

After we have repaired a solution, we need to assess whether the new solution is better than the current best candidate. We use two criteria to determine whether a newly constructed solution is "better" than the previous one. The first and foremost one is of course, the total costs. Note that in our model, costs are only incurred for the opening of an access point, and the single installation costs of a service at an opened access point. In other applications of LNS, there often exists other costs that are incurred for the supply of service from a service point to a demand point. For example, in Chyu and Chang [15], there exists transportation costs from a service point to a demand point. In these cases, the total costs often differ between two solutions which use the same number of facilities. This is not applicable to our problem and we use a secondary measure of solution quality. We define a new measure "balance" of a solution, which indicates how well distributed the service load is across the service points. In order to assess this measure, let's say we have a set of L^* service points which have undergone change in an iteration of the algorithm. The service points in L^* are ordered in descending order of amount of service provided. Then, we take a look at the top and bottom 20% in the ordered list. Denote the collection of service points in the top 10% as L_{top}, and L_{bot} for the bottom 20%. We then define the measure "Balance" for a set of service points as in Equation (21). Since we would like have an even spread the service intensity if possible, a low value of this criterion is preferred.

$$Balance = \sum_{i \in D^u} \Big(\sum_{j \in L_{top}} z_{ij} - \sum_{j \in L_{bot}} z_{ij} \Big) \tag{21}$$

Given these two criteria, the current best solution is updated if the neighbor solution being explored returns a lower value in either one. Furthermore, a simulated annealing framework is used in order to allow for sufficient exploration of neighborhoods that may not deliver improvements in the earlier iterations but possibly can in later ones. Simulated annealing allows for neighborhoods which are not accepted according to the previous two acceptance criteria to still be accepted with a certain probability. This probability p_{accept} depends on the difference in criterion values between the neighbor solution and the current best solution Δ, and a temperature T, as in Equation 22. Δ_c corresponds to the difference in cost criterion and Δ_b to the difference in balance criterion. We consider both probabilities corresponding to differences in both total costs and balance, and take the bigger probability of the two as the acceptance probability. The temperature T is adjusted every time the simulated annealing process is called. Using a cooling scheme ι, after each call T is updated to $\iota \times T$. The temperature is determined separately for the two criteria, distinguished as T_c for the cost criterion and T_b for the balance criterion. Cooling scheme ι is a parameter between 0 and 1 which controls how fast the probability of acceptance

decreases, and set to 0.95 in this research.

$$p_{accept} = max\{exp(\frac{-\Delta_c}{T_c}), exp(\frac{-\Delta_b}{T_b})\} \tag{22}$$

The ALNS-ES performs neighborhood searches per service type, starting with the service type with lowest range. This is because it is most likely that the locations that need to be opened are the least flexible for services with low range. After the algorithm has terminated for a type of service, the opening costs f_j of opened locations in the solution are set to 0 for the next service type. The explained steps are performed for a maximum of 300 iterations for each service, and maximally 100 consecutive iterations without improvement.

5.5 Sensitivity Analysis

We now take a look at how the solution approach needs to be altered in case we look at the situations described in Section 2.4. In the first extension, a demand point that is located far away from a service point costs the service point more capacity to service than a demand point that is close by. In this experiment, we define a certain extra capacity penalty π^u of which the value is dependent on the distance between demand point i and service point j, denoted as $dist(i, j)$. The penalty is enabled or disabled depending on the proportion of the distance over the maximum range a service point j can reach for service type u. Since j cannot service demand points outside of this range at all, this returns a fraction less or equal to 1. In this setup, each unit of service of type u provided by a service point is multiplied by a penalty value π^u if the demand point to service is sufficiently far away. A demand point is said to be "sufficiently far away" if the distance between the demand point and a service point is more than half the service point's range. We can incorporate this feature by adjusting Constraint (15) with the penalty to make constraints (23). The $round(\cdot)$ function denotes the rounding function, such that the input is rounded up or down to the closest integer value. This means that the rounding function term will either return value 0 or 1; 0 if the demand point is within half the service point's range, and 1 if it is not. In expression (23), the penalty value acts as a multiplier to the service amount value. For example, a π^u value of 2 adjusts the capacity requirements such that it takes double the capacity in order to service demand points that are further away than half the possible radius. In the case that there is no penalty issued, this multiplier will equal to 0, and multiplying it directly to the variable v_{ij} will disable the corresponding constraint. Therefore, the multiplier is defined to be the maximal value between 1 (corresponding to no multiplier) and the penalty multiplier value.

$$\sum_{i \in (D^u)'} [v_{ij} \cdot max(1, round(\tfrac{dist(i,j)}{range^u})\pi^u)] \leq N^u x_j^u \qquad \forall j \in L' \tag{23}$$

For the original problem, a simple constructive heuristic is used to create an initial feasible solution which looks for any available service point in the vicinity of every demand point and allocates all of a demand point's demand to a found service point. This can lead to densely occupied service points which cannot accommodate the extra space taken by the distance penalty in the subproblem. Certain constructive heuristics may even have difficulty creating an initial feasible solution, since the occupancy of service points in the initial solution is dependent on the order of service points that the constructive heuristic considers while creating a solution. For this reason, when working with the distance penalty, the initial maximum capacities of service points are increased to be at least π^u times the original value. A separate solution is calculated with the amplified capacities as a control counterpart. By comparing the solutions with and without the distance penalty feature, we observe whether the number of open locations significantly changes.

In the second extension, extra monetary costs are incurred if a service point is particularly busy. The overloading of service points is a practically unwanted trait and with this penalty,

we attempt to avoid the consumption of high capacities at each service point. The objective function is altered to be Equation (24). An extra cost parameter c_o^u is defined, which scales with the occupancy level of a service point defined as $\frac{\sum_{i \in D^u} v_{ij}}{N^u}$. This fraction equals the proportion of available capacity used for a service point. The value of c_o^u is set to be close to the opening costs f_j of an access point such that if the service point is overly occupied, it costs nearly as much as opening an additional access point, and the algorithm may choose to lessen the load on the current service point and open another access point.

$$\min \quad \sum_{j \in L'} (c_j^u x_j^u + \frac{c_o^u \sum_{i \in D^u} v_{ij}}{N^u}) + \sum_{j \in L'} f_j y_j \tag{24}$$

With the addition of $|L'| \times |D^u|$ variables in the objective function, the solver used takes a short amount of time to find a very good solution, but is not able to solve the subproblem to optimality within 10 seconds. Therefore, the gap tolerance for the solver in this case is set to 5%.

Since both extensions add extra restrictions and penalties to our original problem, we expect the extensions to results in higher costs and more open locations.

6 Data Description

In this chapter, we introduce the datasets and preset parameter values used in our experiments.

6.1 Location Data

The location data of public lampposts is obtained from open sources such as Dataplatform. Specifically, we look at two major cities in the Netherlands: Amsterdam and Rotterdam. The locations are provided in the form of longitude and latitude coordinates, such that the distance between two locations can be converted to meters using the Haversine formula, which calculates the great-circle distance between two locations. A list of coordinates of addresses in the Netherlands was provided by TNO.

6.2 Service Installations

For the services Wi-Fi and telecommunications network, a limit exists in reality for the number of wireless devices one router installation can support at the same time. According to American internet service provider Fusion Connect, theoretically most wireless routers can connect up to 250 devices. However, the more devices are connected to a single router, the poorer the connection quality is going to be. They state that as a general rule of thumb, the number of connections should be limited to for instance, 45. The website of American telecommunications company Centurylink states that a 2.4GHz 802.11n signal theoretically covers a range of 820ft, but a real world distance of 410ft (approximately 125m) is reported. Based on this, we could designate a Wi-Fi router to have a signal radius of 125m. However, due to the fact that several houses in the instances do not have an access point within a 125m range, this is extended to 150m. The bicycle theft alarm service is set to have a range of 300 meters from its source. The telecommunication network installations are meant to be smaller in size compared to conventional cell towers for the sake of urban aesthetics. The structure will be no where near the size of a regular tower or a cell on wheels (C.O.W), which could potentially reach callers up to 15km away, according to Motorola [13]. We take a 10^{th} of the potential range of a C.O.W, which is 1500m.

Table 4 shows the number of available access points $|L|$ and number of demand points $|D^{all}|$ per instance. These are found by selecting small regions of the cities of which we have the location data, and taking all recorded public lampposts as access points and addresses as demand points that are in the subarea spanned by the region coordinates.

Instance	$\|L\|$	$\|D^{all}\|$
Amsterdam 1	110	306
Amsterdam 2	1728	3479
Amsterdam 3	3322	4529
Amsterdam 4	6200	8545
Amsterdam 5	18487	11147
Rotterdam 1	1227	1715
Rotterdam 2	2005	2956
Rotterdam 3	10660	14542

Table 4: Instance Dimensions

6.3 Demand Points

For each demand point, the number of inhabitants is simulated using the proportions in Table 5, which contains the proportions of various household sizes in the Netherlands. The statistics have been obtained from CBS: Statistics Netherlands [16]. On average, there are 2.2 persons per household. The household group of 2 persons is assumed to have the average number of connected devices reported by Deloitte [11], which is 11 connected devices. An additional person in a household is not expected to double the number of devices compared to one person. There are often devices that are shared in one household, such as smart TVs and gaming consoles. The bicycle theft alarm is designed to prevents bicycle thefts from homes. According to the Ministry of Water and Water Management of the Netherlands, the Netherlands accommodates 17 million inhabitants and 23 million bicycles. This averages to 1.35 bicycles per person. We expect the number of Wi-Fi connected devices to be more varying for a given household size than the number of bikes, hence assume a smaller standard deviation for the number of bikes. For the Telecommunication Service, the average number of devices in demand are set to the number of inhabitants boosted by a potential number of guests or bypassers at the demand point. This is currently arbitrarily chosen to be 10 extra persons and may be subject to change with more educated guesses, or may be simulated separately.

Number of Persons	Number of Households	Proportion
1	3079778	0.38
2	2610601	0.33
3	938515	0.12
4	407592	0.05
5+	961314	0.12

Table 5: Proportions of Household Sizes in Netherlands 2020

The set of demand points for each instance are found by using the list of addresses in the Netherlands, and identifying all available addresses within a selected sub area. Random rectangular areas of different sizes have been selected in Amsterdam and Rotterdam, of which the longitude latitude coordinates are recorded in Table 20 in the Appendix for each instance.

6.4 Costs and Other Parameters

In order to define the objective function, the cost parameters such as setup costs and operation costs are also needed. These are predefined in our experiment and summarized in Table 6. The term "*size*" indicates the number of inhabitants in demand point i.

Parameter	Preset Value
α	0.95
f_j	1000
c_j^u	$\begin{cases} 350 & u = 1, \text{ Wi-Fi} \\ 150 & u = 2, \text{ Alarm} \\ 500 & u = 3, \text{ Telecommunication Network} \end{cases}$
N^u	$\begin{cases} 45 & u = 1, \text{ Wi-Fi} \\ 50 & u = 2, \text{ Alarm} \\ 62 & u = 3, \text{ Telecommunication Network} \end{cases}$
μ_i^u	$\begin{cases} 8 & \text{size} = 1, u = 1, \text{ Wi-Fi} \\ 11 & \text{size} = 2, u = 1, \text{ Wi-Fi} \\ 12 & \text{size} = 3, u = 1, \text{ Wi-Fi} \\ 14 & \text{size} = 4, u = 1, \text{ Wi-Fi} \\ 15 & \text{size} = 5+, u = 1, \text{ Wi-Fi} \\ 1.35 \times size & u = 2, \text{ Alarm} \\ size + 10 & u = 3, \text{ Telecommunication Network} \end{cases}$
σ^u	$\begin{cases} 2 & u = 1, \text{ Wi-Fi} \\ 1 & u = 2, \text{ Alarm} \\ 3 & u = 3, \text{ Telecommunication Network} \end{cases}$
c_o^u	$\begin{cases} 950 & u = 1, \text{ Wi-Fi} \\ 1000 & u = 2, \text{ Alarm} \\ 1200 & u = 3, \text{ Telecommunication Network} \end{cases}$
π^u	$2 \qquad u = 1, 2, 3$

Table 6: List of Preset Parameter Values with Constant Installation Costs

Setup costs are assumed to be equal across all access locations and in all three scenarios. Installation costs have been decided on based on Vos [7]. The access point opening cost have been set to an amount higher than the maximum installation costs, such that it is unfavorable to open more access points than actually needed. Capacities N^u based on literature, and expected to be equal across all access locations. The demand mean parameters μ_i^u are designed such that for different service types, the shape of expected demands vary depending on the size of the household, with some general assumptions as described in Section 6.3. The standard deviation values σ^u in Table 6 are default values used to evaluate the ALNS-ES performance for Table 10. The occupancy penalties c_o^u are set to arbitrary values that are close to f_j. The distance penalties π_u are chosen to be maximally 2; higher values often lead to infeasibility of certain instances.

7 Results

In this chapter, we present the results for the ALNS-ES algorithm applied to a single period instance, as well as the tuning of its parameters and both penalty extensions. The solution approaches have been coded in Java, execution environment SE-1.8 (64-bit JVM), and makes use of the CPLEX ILOG package (version 2010) developed by IBM. We first identify the optimal parameters, and follow by displaying the results found using these optimal parameters. Lastly, we show the results of the problem instances with the distance and occupancy penalty extensions.

7.1 Tuning of Parameters

The ALNS-ES heuristic makes use of several parameters that are to be predefined by the user. In the base implementation of the model, the default values of the parameters are decided based on educated guesses or examples from previous literature. The optimal values for these parameters may lie somewhere else, and may even differ per problem instance. In this study, we explore a number of possibilities for these parameters as means of fine tuning our algorithm towards the best possible performance.

The first parameter subject to inspection is the score function ψ defined by Equation (12). The score values control how dominant a certain destroy method can become over the course of the algorithm depending on its performance. The default values in combination with decay $\delta = 0.9$ provide an initial increase of 1 weight unit on the first improvement in costs, and this increment is decreasing as the number of improvements made increases. Such small steps have as advantage that the model is relatively stable, meaning that a single successful iteration does not heavily bias the algorithm to prefer the corresponding destroy method. Especially because the scores are incorporated heavier in the early iterations than in later iterations, high scores for improvements are likely to increase weights of whatever destroy method is used more often in the early iterations. This is especially an issue for local search with a sub-optimization component like the ALNS-ES where it is relatively easy to find an improvement in the early stages, and gets more difficult as the algorithm progresses. On the other hand, scores that are too low provide not enough movement in the weights of the destroy methods, such that the adaptive aspect of the neighborhood search may add little to no value compared to for example, a random selection of destroy methods.

Additionally, we also consider different upper and lower bounds β_1, β_2 on the number of disconnected service points in both destroy methods. Initially, we start with values (β_1, β_2) = (80, 150), such that the drawn value is expected to lie close to 100. This boundary has been determined in order to ensure short computation times, but the size of this boundary has close ties to the resulting solution quality; fundamentally, the ALNS-ES algorithm solves the MSCFLP for a single source for a sub area of the problem instance repeatedly, and delivers the union of the sub area allocations as the final result. As these sub areas span a larger proportion of the whole instance, the sub problems deliver partial solutions that are closer to the actual optimal solution. Perhaps in certain situations, it is profitable to opt to sacrifice computation time for better solution quality. When an actual service point network is designed, it is likely to be used on long term. Hence spending up to several hours in computation time may be worth if it means that the solution which is implemented for several years is more close to its optimum. In this study, we attempt to keep computation times below 1 hour, which is still very low considering the fact that an instance with more than 2000 access points and a comparable number of demand points most likely fails to optimize using an exact IP evaluation after 12 hours.

The specific values designated in Section 5.2 are the default values. Three other cases are considered which are adjustments from these default values. The performance of the algorithms

for different combinations are compared in cost objective, and the best combination is selected to generate results for all instances. The running time is taken into account as a secondary aspect; a combination may not be selected if they return unacceptable computation times (setting the limit at > 1 hour), but in general we prioritize combinations returning the lowest costs. In case there is no pronounced optimal combination, we look at the few best combinations in the pool of the instances used for the experiment and select a combination present in the best groups for both instances.

The different variants are summarized in Table 7; Since we have four variants of both parameter types, it gives us 16 possible combinations, hence we perform the tuning analysis on two representative instances. The selected instances are Amsterdam 3 and Rotterdam 3, which are medium and large scale instances. Smaller instances may contain access locations that have an insufficient number of other nearby access locations for bigger cases of (β_1, β_2). The default values for (β_1, β_2) used in Section 5.2 is (80, 150), which is a small interval and has a low upper bound. The new (β_1, β_2) values are designed such that given that we go through the following options: large interval with high upper bound (Variant 2), medium interval with high upper bound (Variant 3) and small interval with high upper bound (Variant 4). The value of 300 as the high upper bound is chosen, due to the observation that sub problems with more than 300 service points often cannot be solved within several minutes. The values of ψ are chosen such that compared to the default case, we have 3 relative cases: high reward for improvements (Variant 2), high penalty for not finding an improvement (Variant 3) and high rewards with high penalties simultaneously (Variant 4).

Variant	ψ	β_1, β_2
1	20,15,10,5	(80,150)
2	30,20,10,5	(80, 300)
3	20,15,10,1	(150, 300)
4	30,10,15,1	(220, 300)

Table 7: Explored Ψ and Destruction Sizes

The cost results and computation time taken are displayed in Table 8. The rows denote the values of (β_1, β_2). In each cell, the first number denotes the total costs found and the number in brackets shows computation times taken in seconds.

	ψ_1	ψ_2	ψ_3	ψ_4
(80,150)	2932700 (737)	2914450 (770)	2908950 (807)	2962050 (716)
(80,300)	2877700 (1460)	2869050 (1855)	2878700 (1654)	**2861400 (1634)**
(150,300)	2862850 (2053)	**2856400 (2050)**	2864700 (1990)	2865550 (2104)
(220,300)	2868200 (2531)	2872200 (2810)	**2821700 (2818)**	**2835350 (2665)**

Table 8: Results for Different Combinations of Ψ and (β_1, β_2): Amsterdam 3

	ψ_1	ψ_2	ψ_3	ψ_4
(80,150)	9923900 (2032)	9877350 (1749)	9853800 (1757)	9970350 (1665)
(80,300)	9598850 (2196)	9566350 (2465)	9654500 (2273)	9626800 (2388)
(150,300)	9644500 (2566)	9620250 (2693)	9672500 (2497)	**9517800 (2566)**
(220,300)	**9472400 (2623)**	**9420250 (2868)**	9567800 (2790)	**9501500 (3029)**

Table 9: Results for Different Combinations of Ψ and (β_1, β_2): Rotterdam 3

Although increasing the upper bound β_2 returns lower costs and less locations opened in all cases, it comes at a cost; all runs with $\beta_2 = 300$ resulted in computation times that were at least twice as much as the default case. In fact, for $(\beta_1, \beta_2) = [220, 300]$, computation times were often 3000 seconds or higher, which is more than 5 times the computation time of this default case. This an expected result, as mentioned in Section 7.1. During the run, it was observed that in this case the number of demand points considered per sub problem reached nearly 1000 in some cases, whilst in the default case this number rarely surpassed 300.

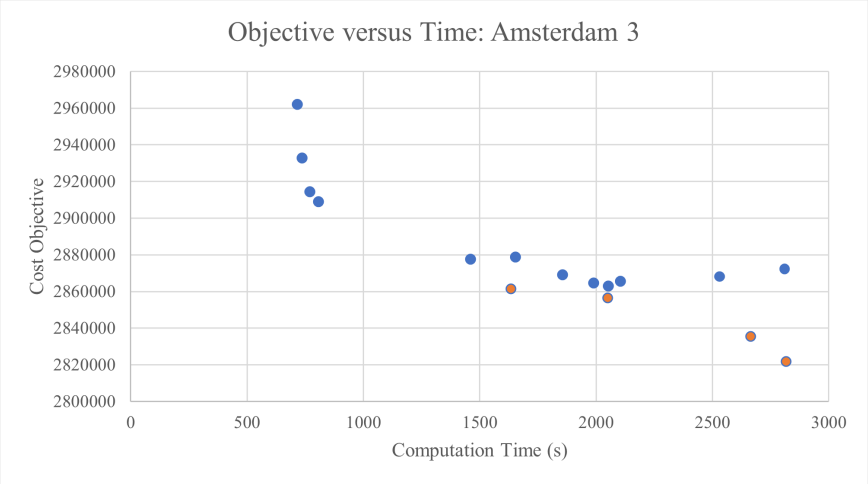


Figure 5: Trade off between cost objective and computation time (Amsterdam 3)

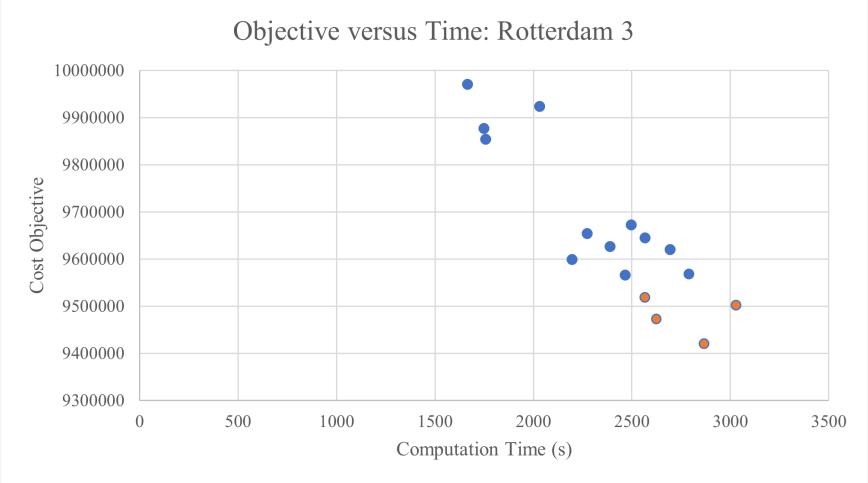


Figure 6: Trade off between cost objective and computation time (Rotterdam 3)

Figures 5 and 6 illustrate the results reported in Table 8 and 9. The vertical axis and horizontal axis contain the costs and computation times respectively. In general, there is a linear pattern displaying that the algorithm returns lower costs with longer computation times. The cases with the longer computation times are primarily with $(\beta_1, \beta_2) = (220, 300)$, since these parameters have the highest chance of creating larger sub problems which take naturally longer to solve. The four combinations for each instance that are the top performers in terms of costs are marked in orange. These combinations are marked in bold in both tables. There is one combination that is included in the top four combinations in both instances: (ψ_4, [(β_1, β_2) = (220,300)]). This combination is used in the ALNS-ES to generate results for the rest of the instances.

7.2 ALNS-ES Results

Using the best selection of parameters tested in the previous section, we now take a look at the performance of the ALNS-ES heuristic compared to an exact approach on the same problem. The exact approach is implemented as the MIP described in Section 4.2. We take a look at the differences in optimal cost objective, balance objective and the computation time taken between the two methods.

Table 10 displays the ALNS-ES results for the base problem for the instances in Table 4. For each service type, the algorithm was run for 300 iterations. The column 'Costs' denotes the cost objective and "Balance" denotes the balance objective, summed over all service types. The runtime for the algorithm is given under the "Time" column, which gives the run time in seconds. The last column describes the percentage of available access locations that are opened to be active service locations, with in brackets the exact number that are opened. An exception is made for instance Amsterdam 1; this small instance has less than 220 service points, and thus the parameters (β_1, β_2) have been set to 110.

Instance	Costs	Balance	Time (s)	% Open (num)
Amsterdam 1	182250.0	0	428	89.1 (98)
Amsterdam 2	2177800.0	70.0	639	70.3 (1215)
Amsterdam 3	2835350.0	58.0	2665	47.5 (1579)
Amsterdam 4	5385550.0	256.0	2850	48.5 (3009)
Amsterdam 5	7783350.0	1610.0	3024	25.3 (4679)
Rotterdam 1	1039500.0	21.0	414	46.1 (566)
Rotterdam 2	1812250.0	53.0	2505	49.4 (991)
Rotterdam 3	9501500.0	684.0	3029	51.2 (5459)

Table 10: Computational Results Base Setting

Table 11 shows the results of attempting to solve the problem described by Equations (1) to (7) in Chapter 4 with an exact approach. The ILOG CPLEX package has been used in order to solve the linear program with a 1% gap tolerance. The column "IP solution" denotes the cost objective returned by CPLEX within the given time limit, which has been set to 12 hours. The superscript LB indicates that CPLEX was not able to find a feasible integer solution in the given time using its branch and bound framework. Instead, it returns an objective on the lower bound that is expected over all unexplored nodes in the tree. Column "IP Time" shows the time it took in order to solve the program in seconds. A dagger indicates that the program stopped after it hit the time limit. If the corresponding IP solution has no superscript, this indicates that the IP solution value is the best found integer solution so far. The LB superscript denotes that the solver could not find a feasible integer solution within the given time, and a (non integer) lower bound value is returned instead. The column "IP Gap" denotes the difference between the current upper and lower bound in the solution tree upon termination. As long as this gap is not equal to 0, a found integer solution most likely does not correspond to the optimal solution. The column "Difference" shows the percentage difference in cost objective between the found IP solution and the ALNS-ES heuristic solution, with a negative difference denoting that the heuristic solution had a lower cost than the best solution found by the IP. Lastly, the "Time Difference" column denotes the difference in computation time between the exact approach and ALNS-ES heuristic in seconds. A positive difference indicates that the exact approach was solved in a shorter time, whilst a negative difference means that the ALNS-ES heuristic was faster. The instances Amsterdam 4, Amsterdam 5 and Rotterdam 3 could not be evaluated exactly with an IP solver due to memory limitations.

Instance	IP Solution	IP Time	IP Gap	Difference	Time Difference
Amsterdam 1	182400.0	13	0.54%	-0.08%	+417
Amsterdam 2	2096800.0	24163†	0.47 %	3.86%	-24664
Amsterdam 3	2719604.16^{LB}	43205†	-	4.25%	-
Rotterdam 1	1035000.0	3059	0.10%	0.43%	-470
Rotterdam 2	1779850.0	14645	0.07%	1.82%	-14207

Table 11: Comparison with IP results for small instances.

Looking at Table 11, we see that except for the smallest instance Amsterdam 1, the ALNS-ES finds a good solution within a much faster computation time than the exact method. Since instance Amsterdam 1 contains around 100 service points, with the knowledge that for such instance sizes the IP can be solved in reasonable time, the use of the heuristic is not necessary. However, it is also noticeable that the best found IP solution for instance Amsterdam 1 is higher than the heuristic solution. This is because due to the 1% tolerance gap setting, the IP solver terminated after finding a solution satisfying this gap. For this specific instance, it was observed that the (non integer) lower bound found so far upon termination was 181752.65, but the solver failed to optimize past the IP gap of 0.35% within 12 hours. Therefore, even for this instance there is still enough motivation to favor the use of the heuristic over an exact evaluation; when it comes to integer solutions, the heuristic has found a better solution than the exact method.

Regarding the rest of the instances, the ALNS-ES heuristic returns solutions for which the differences in cost objective are negligible for medium size instances (around 1000 service points). For larger instances, this difference is generally higher but remain below 5%. With even bigger instances such as Amsterdam 4, 5 and Rotterdam 3 for which the exact solutions could not be evaluated at all, the difference is expected to increase as well. A 5% difference from the optimal solution may be a significant gap because of the scale of the objective values, but it remains difficult to debate that this implies we should prefer exact IP method; after all, the heuristic is much faster and recall that for bigger instances, no solution could be found using the exact approach. For instance Amsterdam 3 already, no feasible integer solution could be found within 12 hours.

Furthermore, since the heuristic is iterative and improves with more runs, it is possible that a better solution can be found within more runs than the current preset value of 300. Table 12 show the ALNS-ES results for instances Amsterdam 2, 3 and Rotterdam 2 with doubled number of maximal iterations and iterations without improvement. These two instances are the ones with the highest percentage differences between the heuristic and IP output. The first and second column show the found costs and time taken and the third column shows the proportion of access locations that are open. The column "Difference vs 300" shows the percentage costs decrease compared to a run for the same instance with 300 iterations, and column "Difference vs IP" shows the percentage difference with the best integer solution or lower bound found using the exact IP.

Instance	Costs	Time (s)	% Open (num)	Difference vs 300	Difference vs IP
Amsterdam 2	2173450.0	3608	64.8% (1121)	-0.19%	3.65%
Amsterdam 3	2798700.0	4567	46.47% (1544)	- 1.29%	2.9%
Rotterdam 2	1810100.0	4698	49.3% (989)	-0.12%	1.69%

Table 12: Computational Results with Doubled Iterations.

Looking at the last column, we see that the gap between the best integer solution found by an IP and the heuristic solution can be reduced by simply more runs. For instance Amsterdam 3,

the difference between the non integer lower bound returned by the solver is at 2.9%; hence the actual gap between the heuristic and optimal solution is expected to be even lower. Instances Amsterdam 2 and Rotterdam 2 also found a slight improvement. On the downside, naturally the computation times have increased, and for all instances, the runs took more than 1 hour. Compared to this increase in computation time, we the decrease in objective value compared to the run with 300 iterations is less than 1% for two of the three tested instances. We therefore retain 300 maximal iterations for our algorithm. We however use these results as motivation to employ ALNS-ES as a good alternative to an exact analysis, since it is able to find good solutions for instances that the exact approach is unable to evaluate.

7.3 Sensitivity Analysis

In Section 2.4, we introduced the concept of distance and occupancy penalties and explained how we will analyze its effects in Section 5.5. We now present the mathematical results and compare the solution in terms of the cost objective and number of open locations compared to a case with no extension.

7.3.1 Distance Dependent Occupancy

Table 13 shows the results for the control group and penalty group with amplified capacities. With control group, we denote the original instances with no distance penalty incorporated, that lets us observe the effect of the distance penalty by comparing the control group instances to the penalized instances. In this case, the control group is not equal to the original instances as in Table 10 since it also has amplified capacities. We compare the differences in costs and number of open locations, and observe whether significant difference and patterns are observable. Noticeable differences and/or patterns imply that the incorporation of a distance penalty in the optimization process is advantageous, if a similar condition were to be present in a real life setting, as well as whether the ALNS-ES is able to still find good solutions with this additional measure.

Instance	NP Cost	Cost	NP Open	Open	Difference Cost	Difference Open
Amsterdam 1	91200.0	91200.0	49	49	0% (0)	0
Amsterdam 2	1533400.0	1068900.0	600	584	-30.29% (-464500)	16
Amsterdam 3	1577850.0	1560950.0	948	922	-1.07% (-16900)	26
Amsterdam 4	2842000.0	2864250.0	1652	1652	+0.78% (+22250)	0
Amsterdam 5	3992650.0	4416550.0	2428	2648	10.62% (-423900)	-220
Rotterdam 1	518250.0	518750.0	280	280	+0.00%(+500)	0
Rotterdam 2	932800.0	906300.0	522	494	-2.84% (-26500)	28
Rotterdam 3	4971150.0	5055800.0	2948	2903	-1.67% (-84650)	45

Table 13: Comparison of results with distance penalty

The column “NP Cost” displays the cost objectives for the instances that serve as a control group (NP being shorthand for No Penalty); they have enlarged capacities, but no penalty. The column “Cost” shows the cost objectives for the instances with the same enlarged capacities but with the distance penalty enabled. The columns “NP Open” and “Open” denote the number of open locations in the two different settings respectively. The “Difference Cost” and “Difference Open” columns summarize the differences in the cost objectives and number of open locations between the two settings. The differences in costs are expressed in percentages followed by the unit differences in brackets.

Surprisingly, we observe that the cost objectives and number of open locations are not consistently higher in the cases with distance penalty compared to the control case. Except for

the instance Amsterdam 5, the instances even return equal or less number of opened locations with the distance penalty enabled. A possible cause of this output is that the control group instances are not well optimized with the amplified capacities. For the penalized instances, since there is a restriction on capacities due to the distance penalty, the constructive heuristic as well as the sub-optimization have restricted choices on which locations can be opened, while the control instances do not. This may lead to the instances with distance penalty converging to an optimal solution faster than the control instances that have a bigger solution search space. This theory can likely be tested by letting the control group's optimizations run for extended periods of time and checking whether after sufficient iterations, the control group instances return lower costs and number of open locations. Another possibility is that many of the service points in the initial solutions were not fully occupied to start with. Since this penalty is applied to the capacities rather than costs, if a service point has sufficient remaining space to accommodate the extra capacities consumed by the penalties, there would be no need to open a new access point which harbors additional costs. If this were to be the case for many active service points, the presence of the distance penalty may not be significant. However, although there is no consistent pattern, for some instances such as Amsterdam 2 and Amsterdam 5, there are cost differences of over 10% compared to the control solutions. This indicates that if in reality, such a penalty were to be present, it is advised to take it into account in the optimization process.

7.3.2 Occupancy Dependent Costs

Table 14 shows the results for the original instances with regular capacities with the occupancy penalty incorporated into the objective function. Note that because the penalty in this case is incorporated into the objective function, the capacities do not need to be adjusted and we can use the original instances as a control group. The columns "Cost", "Balance" and "Open" denote the new costs, balance and number of open locations in the new solutions. The columns "Difference Cost", "Difference Balance" and "Difference Open" show the percentage differences between the costs, balance and number of open locations of the solutions found with the occupancy penalty versus the original solutions in Table 10. We observe whether the incorporation of this penalty has the desired effect of the intuition behind it; solutions with improved balance.

Instance	Cost	Balance	Open	Difference Cost	Difference Balance	Difference Open
Amsterdam 1	414823.54	71.0	98	127.6% (+232573.54)	(+71)	0
Amsterdam 2	4868908.10	640.0	1228	150.6% (+2691108.1)	814.2% (+570)	+13
Amsterdam 3	6352330.60	1237.0	1604	124.0% (+3516980.6)	2032.76% (+1179)	+25
Amsterdam 4	12049789.49	2666.0	3059	123.7% (+6664239.49)	941.4% (+2410)	+50
Amsterdam 5	16385278.10	6134.0	4665	110.5% (+8601928.1)	280.9% (+4524)	-14
Rotterdam 1	2366407.92	386.0	563	127.6% (+1326907.92)	1738.1% (+365)	-3
Rotterdam 2	4133314.91	789.0	1030	128.1% (+2321064.91)	1388.6% (+736)	+39
Rotterdam 3	20890497.31	6626.0	5581	119.8% (+11388997.31)	868.7% (+5942)	+122

Table 14: Comparison of results with occupancy penalty

In the case of an occupancy penalty, we see consistent expected increases in costs compared to the base case results in Table 10. It is noticeable that the cost increases are between 120% and 150% in most instances; this is not a surprising number given that the occupancy penalties are designed such that a fully occupied service point levers an extra cost of $c_o^u = 950, 1000, 1200$, depending on the service type. These percentage cost increases may be an indication that on average, a service point is occupied for approximately 20% to 50% of its maximal capacity.

Looking at the balance scores and number of open locations, the results are very unexpected; the balance scores are immensely higher for the runs with the penalty enabled. This is most likely due to the fact that although the overloading of service points is punished by increasing the objective, the algorithm takes reducing costs as the main goal. Instead of averaging out the penalties paid by distributing the services such that each service point has a similar occupancy level, the algorithm has likely paid high penalties for many service points and "compensated" for these costs by paying low penalties at other service points. This leads to non extreme occupancy levels on average, but the mode occupancy levels are likely to be scattered at both extremes (barely in service and fully in service). If this were the correct reasoning behind this, a solution could be also penalizing the under-usage of open access points, such that service levels per service points are pushed to equate around an average value. This however requires the prior knowledge of the optimal average value, or the existence of a target average value. We conclude out of these results that more than penalizing the objective is necessary if we wish to prioritize finding balanced allocations. Solely penalizing the usage of capacities even appears counterproductive for balancing out capacity usage, without extra measures that control for the described compensation effect.

8 Application to a Multi Period Setting

The current model assumes a single period setting where we have a stochastic demand value which has been converted into a deterministic variant that is given for that period. However, in the real world, demand fluctuates over periods of time. By means of simulation, we observe whether the assumption of stochastic demand in a single period setting is able to provide good approximation for the overall demand uncertainty in the multi period setting. In this chapter we describe how the performance of the single period ALNS-ES solution in a multi period simulation is assessed.

8.1 The Traffic Model

A common technique used to model real life arrivals of demand for a service is the Poisson traffic model. In this model, the Poisson Process plays a key role. The Poisson process is a counting process where arrivals of events happen with a certain rate λ but the exact moments of arrivals are at random. The probability of a number of arrivals occurring within a specific time period can be evaluated using the Poisson distribution with rate λ. The system can have a single or a number of servers, which provide service to arrivals. Furthermore, the arrivals can carry individual demands of which the sizes can be modeled by some other compounding distribution.

For our wireless service distribution problem, the concept of traffic models can also be used to simulate behavior of demand points. We now consider a multi period setting of a certain number of time periods. However, demand points are not expected to need service in every single time period. Define two states a demand point can be in in regards to a service type u: "ON" and "OFF". In state OFF, the demand point is not in need of service. In state ON, the demand point has a certain level of demand for service type u. The action of a demand point transitioning from ON to OFF and vice versa is seen as an arrival the transition process. For each service type, an transition time ϕ_1 is drawn from the exponential distribution with rate λ, such that average transition time is $\frac{1}{\lambda}$. Once a demand point is in state ON for a certain service type, the number of consecutive periods that the status remains ON is drawn from an exponential distribution with rate η. After this time, the demand point returns to state OFF. Afterwards, the transition time to the next ON period is drawn and the process continues until it has reached the total number of periods. A demand point before the first period is in state OFF, and the state of a demand point for a certain service type is independent of the states for other service types. The described behavior deviates a bit from the classic Poisson traffic model. In the classic Poisson traffic model, interarrival times of customer arrivals are drawn independent of whether a server is occupied or not. The concept of a server being occupied is comparable to a demand point being in state ON, whilst arrivals are comparable to demand points moving to state ON. In our simulation, transition times to transition to state ON are only drawn after the demand point returns to state OFF, there for the arrival process is not completely independent of the service process, and the number of arrivals cannot be represented by a Poisson distribution. Instead, this two state traffic model can be seen as a Markov chain with transition rates λ and η. The fraction of time a demand point is in state ON, given by Equation (25), which can be interpreted as the probability that a demand point is in state ON in the Markov Chain.

$$\textit{Probability demand point is in state ON} = \frac{\lambda}{\lambda + \eta} \tag{25}$$

As aforementioned, we would like to know whether the ALNS-ES algorithm returns solutions that is able to cover fluctuating demand in a multi-period model. To achieve this, we first look at how the ALNS-ES solution can be applied to the new situation. Since the heuristic finds a solution with specific links between service points and demand points, we only allow demand

points to receive service from service points that are allocated to them in the ALNS-ES solution. Recall that in the traffic model, the quantities of demand for a service type in a time period has its own distribution. Denote μ'^u as the expected value of this demand quantity for a certain service type u. If $\mu^u = \mu'^u$, μ^u being the mean values in ALNS-ES for the same service type, the solutions of ALNS-ES will be too conservative to apply to the traffic model; The presence of states ON and OFF simply reduce the total amount of demand, and the solution will always cover at least as much demand as in the single period case without switching of states. We therefore calculate adjusted μ^u values for the ALNS-ES method in order to use the results for a traffic model with parameters λ and η, by equating the total amount of traffic that is expected to occur with the demand parameters used in the ALNS-ES algorithm. Recall that the term in Equation (25) is equivalent to the probability that a demand point is in state ON at a period in time. We set the mean values μ^u in the ALNS-ES to be as in Equation (26).

$$\mu^u = \frac{\lambda}{\lambda + \eta} \cdot \mu'^u \tag{26}$$

The standard deviation of the adjusted demand, which are also dependent on the values of λ and η are found using simulation. Values for transition times $\phi_1 \sim Exp(\lambda)$ and $\phi_2 \sim Exp(\eta)$ are drawn for 500,000 time periods, and the sample variance of the right hand side expression of Equation (26) is evaluated and used to calculate σ^u values to be used in the ALNS-ES model.

In this study, several combinations of values for λ and η are explored. These parameter values have direct influence on the mean and variance parameters used in the simulation model, and therefore the performance results are expected to vary per combination. Explored values are set between 0 and 1 for both parameters. Several combinations are explored as below in Table 15. The "OFF" and "ON" columns give a quick summary of the expected behavior of the traffic simulation. The tag "short" indicates that the exponential times drawn for sessions ON or OFF are expected to be less than 2 times periods based on the mean value of the distribution, which correspond to $\frac{1}{\lambda}$ and $\frac{1}{\eta}$. The tag "regular" indicates that these times are expected to be between 2 to 4 times periods long for ON sessions, and 2 to 3 periods for OFF sessions. Lastly, the "long" tag indicates that these times are expected to be more than 4 times periods for ON sessions, and 3 time periods for OFF sessions. The combination of long OFF sessions and short ON sessions are not explored since it is expected to simulate very little demand which is less realistic. However, it is expected that in future smart cities, we are able to work with bigger capacities at each service point thanks to improvements in technology. If the amount of total simulated demand were to remain the same, with increases in service point capacities, demand peaks tend to be bigger and less frequent, since more demand can be fulfilled at a time. The analyses for varying λ, η combinations is done using instance Amsterdam 3. A total of 200 time periods are simulated for all service types.

Scenario	λ	η	OFF	ON
1	0.7	0.15	short	long
2	0.8	0.6	short	short
3	0.8	0.3	short	regular
4	0.4	0.2	regular	short
5	0.5	0.5	regular	regular
6	0.5	0.7	regular	short
7	0.3	0.5	long	regular

Table 15: Explored Combinations of (λ, η)

8.2 Demand Simulation

The simulation of demand in each time period is done in two ways: (i) constant throughout a single ON session (ii) drawn from a known distribution for each period throughout an ON session. The demand for a service is set to 0 if for that service type, a demand point is in state OFF. For the first case, the traffic model only controls whether a demand point is in state ON or OFF for every time period. The level of demand for a demand point whenever it is in state ON is set to be μ'^{u}, which may vary across demand points depending on the household size they were simulated to be. For the second case, the demand for a single period is randomly drawn from the Poisson distribution with mean $\mu_i'^{u}$. The Poisson distribution with parameter λ can be estimated pretty well by a $N(\lambda, \sqrt{\lambda})$ distribution. Additionally, the Poisson distribution draws discrete, positive values, which are qualities the normal distribution does not have as discussed in Section 2.2. Hence in essence, we are attempting to draw demand values from a Normal distribution just like in Section 2.2, but on positive integers. A question on this decision may be, why the Poisson distribution is not considered for the initial model in Chapters 2 to 5. This is because the assumption of normal distribution allows the easy translation of a constraints (4) to a deterministic variant while keeping the reliability level α into account. A possible issue is that the aforementioned Normal approximation for a Poisson distribution is accurate for $\mu > 20$, with the approximation improving as μ increases. The preset demand mean values for each type of service in our main model do not exceed 20, as shown in detail later in Chapter 6. A simple workaround for this is changing the scale of demand. An example alternative for demand quantity is Mbps as in Verhoek [6]. The Mbps unit works with larger numbers in general compared to number of devices, which makes the Normal approximation for a Poisson distribution more accurate.

8.3 Quality Assessment

We implement our ALNS-ES solution to our Poisson traffic model by taking the connections made by the algorithm between demand points and service points. Demand point i can only be serviced by the service points $j \in L$ that have $z_{ij}^u > 0$. For each instance of the Poisson traffic model, we evaluate how much of demand is able to be satisfied by our solution. The reliability of a solution on an instance of the Poisson traffic model depends on the proportion of simulated demand that is not able to be satisfied. This proportion can be assessed from two different perspectives; the first one being the amount of demand simulated, and the other being the number of periods in which unsatisfied demand occurs. Denote demand of demand point i at time period t for service type u as d_{it}^u. When looking at demand that is not able to be met, we have two options of assessment. The first is from the perspective of a demand point, by looking at how many demand units cannot be allocated to service points. The second is from the perspective of service points, where we look at how much capacity is exceeded if we allocate all the simulated demand units to the opened service points. In this study, we evaluate unmet demand from the perspective of the service points. There is a practical reason behind this choice: in reality, if such a study is done in order to design a service point network, the properties regarding the service points such as capacities are within the network designer's control, while the behavior or demand points in the area that is of interest is often not. Therefore, it is more advantageous for the designer to acquire information on necessary adjustments that can be applied to service points rather than performance statistics per household. For example, if it is observed that within the simulation the capacity of a certain service point is exceeded regularly, the designers may look into options to increase this service point's capacity.

To start with, we assume that demand not met at a time period t is immediately lost. Denote each demand unit lost by d_{jt}^{u*}, i.e. the amount of capacity short at service point j at time period t for service type u. The reliability level R_1 of a solution is calculated as in Equation (27), where

P denotes the number of time periods simulated.

$$R_1 = 1 - \frac{\sum_{t=1}^{P} \sum_{u \in S} \sum_{j \in L} d_{jt}^{u*}}{\sum_{t=1}^{P} \sum_{u \in S} \sum_{i \in D^u} d_{it}^{u}} \tag{27}$$

The fraction value is equal to the short in capacity over all periods, service types and service points divided by the total amount of demand simulated across all periods and demand points.

A second reliability measure is based on the number of time periods we encounter a shortage in capacity. This is calculated as R_2 as in Equation (28).

$$R_2 = 1 - \frac{\sum_{t=1}^{P} \sum_{u \in S} \sum_{j \in L} I_{\{d_{jt}^{u*} > 0\}}}{\sum_{u \in S} P \times |L_{open}^{u}|} \tag{28}$$

The term $I_{\{d_{it}^{u} > 0\}}$ denotes an indicator function having value 1 if there is any capacity shortage d_{jt}^{u*} for service type u at demand point i during period t, else 0. The term $|L_{open}^{u}|$ in the denominator denotes the number of access points that are open for service type u. The calculation of R_2 does not take the amount of capacity shortage into account whenever it occurs, only the frequency of a shortage happening.

For both R_1 and R_2, a value close to 1 indicates a high reliability level. Although the value of R_1 also gives a notable impression of the solution's performance, it is especially of interest whether R_2 approaches the $\alpha = 0.95$ reliability threshold as predefined in the ALNS-ES algorithm. This is because substantially, the constraints (4) describe that the demand satisfaction requirement is to be held in $\alpha \times 100\%$ of all scenarios, independent of the degree to which the constraint is violated in the remaining $(1 - \alpha) \times 100\%$.

For both calculations R_1 and R_2, we need to determine d_{jt}^{u}: the demand that cannot be serviced by service point j due to the capacity restriction although it is allocated to j. In order to determine the quantity d_{jt}^{u} per service point j, we need to define how a demand point is serviced. Given simulated demand quantities and the links between service points and demand points alone, we cannot identify (yet) how much demand of each demand point is being serviced by which service points. The complete ALNS-ES solution provides the following connection characteristics: (i) what service points a demand point can receive service from, and (ii) how much service is expected to be allocated by a connected service point to the demand point. This is a directly applicable solution to the case of single period deterministic demand which is used in the ALNS-ES algorithm, but not in the new multi period case. As mentioned in the beginning of this section, the simulation does not and likely cannot strictly enforce the quantity assignments z_{ij}^{u} due to fluctuations in the simulated demand values which differ from the stationary values used in the heuristic. However, since we have now identified which access points are active service points for every service type, the problem of finding an allocation of demand across these open service points is not difficult. The question is, in what way is the allocation to be done, such that we can evaluate the d_{jt}^{u} values? An intuitive idea is to aim to reduce the capacity shortages that we can experience as much as possible. Optimizing an allocation with this objective can be conveniently done by a linear program formulation and the help of a commercial solver. We consider two options alongside this main allocation goal; one where we allocate freely, and another where we restrict the allocations to be single source.

For the first option where we have no restriction on the allocation pattern, Equations (29) to (33) describe a linear program that solves the service allocation problem for a single period.

$$\min \quad \sum_{j \in L} \upsilon_j^u \tag{29}$$

$$\begin{aligned}
s.t. \qquad \textstyle\sum_{i \in D^u} k_{ij}^u - N^u \leq \upsilon_j^u \qquad & \forall j \in L & (30)\\
\textstyle\sum_{j \in \{i,j\} \in \mathbb{C}^u} k_{ij}^u \geq d_i^u \qquad & \forall i \in D^u & (31)\\
\upsilon_j^u \in \mathbb{N} \qquad & \forall j \in L & (32)\\
k_{ij}^u \in \mathbb{N} \qquad & \forall \{i,j\} \in \mathbb{C}^u & (33)
\end{aligned}$$

Given the viable connections between demand points and service points and simulated demand of each service point in a certain period, it finds a service allocation pattern which minimizes the quantity of total shortage in capacity in this period. Variable υ_j^u denotes the capacity shortage at service point j for service type u. The objective is to minimize the value of υ_i^u for all service points. The variable k_{ij}^u denotes the amount of service service point j provides to demand point i for service type u. It is only defined for pairs of $\{i, j\}$ in set $\mathbb{C}^u$. The set $\mathbb{C}^u$ contains the connections $\{i, j\}$ that are established by the ALNS-ES solution, i.e. all pairs $\{i, j\}$ for which $z_{ij}^u > 0$. In this particular case, there is a strict demand satisfaction requirement and no strict capacity limit. This means that the program described by Equations (29) to (33) return service levels at each used service point j that leaves no unsatisfied demand, but may have exceedance in capacities represented by υ_j. This is a considerably smaller problem compared to the general MSCFLP and solves in negligible time. The above program is solved per time period t and per service type u. The subscript for t under every variable has been left out for convenience. Constraints (30) defines the the shortage in capacity, which is at least the positive difference between the sum of service a service point j is providing and its total capacity. Constraints (31) ensure that demand of every household is allocated somewhere. Finally, Constraints (32) and (33) define the domains of the variables.

The second option takes the realistic assumption that all demand of one demand point for a certain service type is fulfilled by one service point. Although the original problem optimizes under a multi-source assumption, it is still common today that a household receives service from a single source, especially with contract based services. In this scenario, a demand point receives all its demand for a single service type from a single service point that is available for connection. A demand point is able to connect to different service points per service type. Which service points are available for choice are again dependent on the ALNS-ES solution; a demand point can only choose from service points that are connected by the solution. An issue with this allocation rule is that the original ALNS-ES algorithm does not take this restriction into account. Since the heuristic is able to split demand quantities of demand points in order to limit the number of open access locations, it is probable that the current heuristic solution will result in very high shortages at certain service points. This will be reflected by R_1, whilst the R_2 measure may be misleadingly low. The amount with which the capacity at a service point is exceeded certainly increases, but the frequency with which the exceedances occur do not necessarily increase. It would theoretically be optimal to re-design the heuristic such that the subproblems described in Section 4 takes the single source constraint into account.

The allocation problem with single source constraints is given by Equations (34) to (41) below. It again minimizes the total capacity shortage, but with a few additional constraints. Constraints (36) and (37) together ensure that a demand point is serviced by a single service point. The variable g_{ij}^u is a binary variable which takes on value 1 if demand point i is serviced fully for the designated period by service point j, else 0. By restricting the k_{ij}^u to be smaller

than g_{ij}^u multiplied by a large constant M and limiting g_{ij} to be 1 for only a single j value, we enforce the single source allocation rule.

$$\min \quad \sum_{j \in L} \upsilon_j^u \tag{34}$$

$$\begin{aligned}
s.t. \quad & \textstyle\sum_{i \in D^u} k_{ij}^u - N^u \leq \upsilon_j^u && \forall j \in L && (35)\\
& k_{ij}^u \leq M g_{ij}^u && \forall \{i,j\} \in \mathbb{C}^u && (36)\\
& \textstyle\sum_{j:\{i,j\} \in \mathbb{C}^u} g_{ij}^u = 1 && \forall i \in D^u && (37)\\
& \textstyle\sum_{j:\{i,j\} \in \mathbb{C}^u} k_{ij}^u \geq d_i^u && \forall i \in D^u && (38)\\
& \upsilon_j^u \in \mathbb{N} && \forall j \in L && (39)\\
& k_{ij}^u \in \mathbb{N} && \forall \{i,j\} \in \mathbb{C}^u && (40)\\
& g_{ij}^u \in \{0,1\} && \forall \{i,j\} \in \mathbb{C}^u, \forall u \in S && (41)
\end{aligned}$$

8.4 Results: Poisson Traffic Model

We take a look at the performance of the ALNS-ES solutions on the multi period traffic model described in Chapter 8. Two instances are used of medium and large size: Amsterdam 3 and Amsterdam 5. The Tables 16 and 17 show the values R_1 and R_2 as described in Section 8.3, where there is constant demand levels across ON sessions. Table 16 corresponds to the case of free allocation, where the allocations every period are determined by Equations (29) to (33). Table 17 shows the results for single source allocation, defined by Equations (34) to (41). We compare the calculated values R_1 and R_2 to see if in this setting they match up to the reliability criterion $\alpha = 0.95$ which has been used to obtain solutions from the stationary equivalent. In each table, the columns correspond to combinations of (λ, η) listed in Table 15.

Instance	R_1							R_2						
	1	2	3	4	5	6	7	1	2	3	4	5	6	7
Amsterdam 3	1.0	0.998	1.0	1.0	0.999	0.992	0.994	1.0	0.980	1.0	1.0	0.985	0.954	0.969
Amsterdam 5	1.0	0.996	1.0	1.0	0.997	0.986	0.988	1.0	0.984	1.0	1.0	0.986	0.964	0.974

Table 16: Traffic Model: Free Allocation with Constant Demand

Instance	R_1							R_2						
	1	2	3	4	5	6	7	1	2	3	4	5	6	7
Amsterdam 3	0.999	0.996	0.999	0.999	0.997	0.987	0.990	0.999	0.961	0.998	0.997	0.963	0.912	0.954
Amsterdam 5	0.999	0.994	0.999	0.999	0.994	0.980	0.857	0.999	0.975	0.998	0.997	0.976	0.946	0.917

Table 17: Traffic Model: Single Source Allocation with Constant Demand

For instance Amsterdam 3, almost all periods return both R_1 and R_2 above $\alpha = 0.95$. For (λ, η) cases 1, 3 and 4 with free allocation, both R_1 and R_2 are equal to 1.0, which is very high and even suggests that our solution may be too conservative. We still see that for some periods, there is excess demand that cannot be covered by the ALNS-ES solution. Specifically, $R_2 = 0.912$ for (λ, η) case 6, which corresponds to $(\lambda, \eta) = (0.5, 0.7)$. For instance Amsterdam 5, there are two occurrences where $R_2 < 0.95$: these are (λ, η) cases 6 and 7, which correspond to $(\lambda, \eta) = (0.5, 0.7)$ and $(0.3, 0.5)$ respectively. As for R_1, there is single case where it does not meet the $\alpha = 0.95$ requirement. The single source allocation result with (λ, η) case 7 for instance Amsterdam 5 returns $R_1 = 0.857$. This is not too surprising, looking at the value of the fraction $\frac{\lambda}{\lambda+\eta}$ for $(\lambda, \eta) = (0.3, 0.5)$. To explain, we recognize that the R_2 values for $(\lambda, \eta) = (0.5, 0.7)$

and $(\lambda, \eta) = (0.3, 0.5)$ are lower compared to the other instances. These (λ, η) values return a low value for the fraction value $\frac{\lambda}{\lambda+\eta}$, which is used in Equation (26). For higher fraction values $\frac{\lambda}{\lambda+\eta}$, the ALNS-ES algorithm insures its solution for higher demand values since it implies that demand points are expected to be in status ON often. On the other hand, the demand points in the corresponding traffic models do have a higher ON rate, such that cumulative demand per period is higher than in the case of lower $\frac{\lambda}{\lambda+\eta}$ values. Since the single period IP of the ALNS-ES is based on the assumption that all demand points are in state ON, the ALNS-ES solution is also more robust for busy periods. Therefore the solutions created with (λ, η) values returning low $\frac{\lambda}{\lambda+\eta}$ values suffer more from the low demand levels it is insured against in the first place, compared to how the solutions created with (λ, η) values returning high $\frac{\lambda}{\lambda+\eta}$ values suffer from the high demand intensity it must cover in the multi period setting. This is visualized in Figures 7 and 8 for instance Amsterdam 3, where we see that generally lower $\frac{\lambda}{\lambda+\eta}$ values tend to see lower R_1, R_2 values. We can thus suspect that in the case of $(\lambda, \eta) = (0.5, 0.7)$ and $(\lambda, \eta) = (0.3, 0.5)$, the ALNS-ES algorithm has produced outputs based on low expected demand values, and in the multi period model this solution is unable to provide enough service for periods where many demand points are in state on simultaneously.

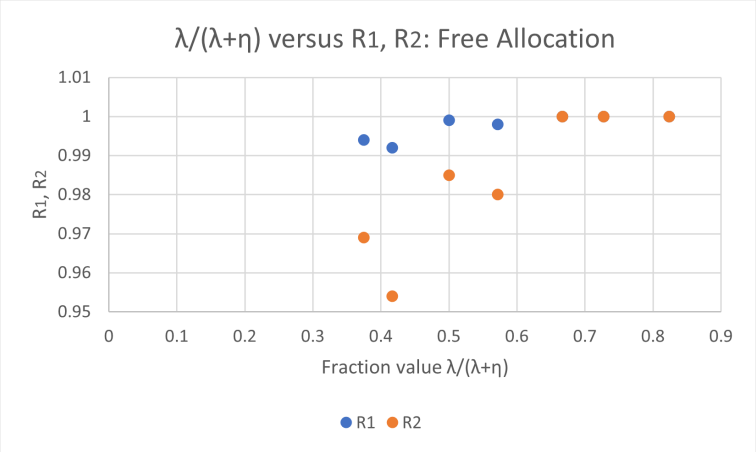


Figure 7: $\frac{\lambda}{\lambda+\eta}$ versus R_1, R_2, Free Allocation (Amsterdam 3)

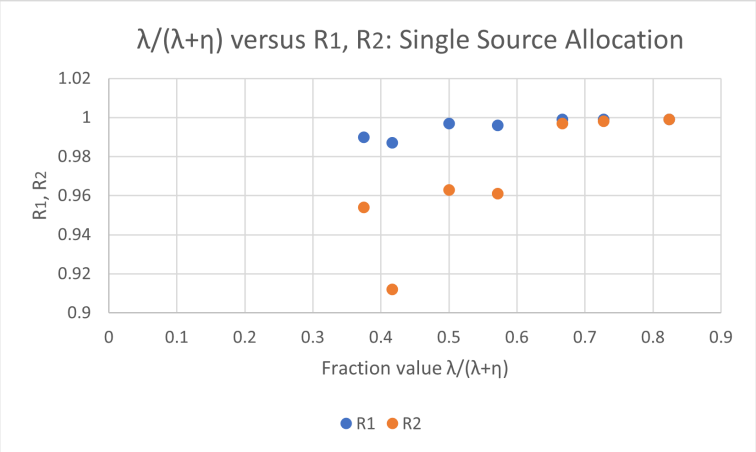


Figure 8: $\frac{\lambda}{\lambda+\eta}$ versus R_1, R_2, Single Source Allocation (Amsterdam 3)

Additionally, it appears that all R_1, R_2 values are slightly lower for the case of single source allocation compared to the free allocation. This is naturally expected since the the single source case is a more restricted version of the latter. The R_1, R_2 values for the single source allocation still appear relatively high compared to the strict requirement that a demand point can be only serviced by one of the service point the ALNS-ES solution has linked it to per period; only a single case returns a value below 0.95.

Next, we look at the results with variable demand per period. Tables 18 and 19 display the results for the same instances but with variable demand levels per time period in the traffic model.

Instance	R_1							R_2						
	1	2	3	4	5	6	7	1	2	3	4	5	6	7
Amsterdam 3	0.999	0.994	0.999	0.999	0.995	0.986	0.988	0.997	0.969	0.992	0.992	0.972	0.937	0.950
Amsterdam 5	0.999	0.993	0.999	0.998	0.994	0.983	0.985	0.998	0.984	0.996	0.995	0.985	0.968	0.971

Table 18: Traffic Model: Free Allocation with Variable Demand

Instance	R_1							R_2						
	1	2	3	4	5	6	7	1	2	3	4	5	6	7
Amsterdam 3	0.999	0.992	0.998	0.998	0.991	0.981	0.981	0.995	0.953	0.986	0.986	0.949	0.981	0.908
Amsterdam 5	0.999	0.990	0.998	0.998	0.989	0.975	0.964	0.996	0.975	0.993	0.992	0.974	0.951	0.950

Table 19: Traffic Model: Single Source Allocation with Variable Demand

Here we can see a similar pattern as in the case of constant demand; Combinations that return low $\frac{\lambda}{\lambda+\eta}$ generally have lower R_1 and R_2 values. These are generally even lower than the values found in the traffic model with constant demand for every ON session. For instance Amsterdam 3, there are three cases where the R_2 measure falls below the 0.95 mark, all three belonging to groups with (λ, η) combinations with low $\frac{\lambda}{\lambda+\eta}$ values. For instance Amsterdam 5, there are no cases of either R_1 or R_2 dropping below 0.95. Overall, the results for the simulation with variable demands indicate that the ALNS-ES solution perform great in the multi period setting with both free and single allocation. The simulations with constant demand show more mixed results, where some instances show that the solutions are overly conservative, and other instances return lower R_1, R_2 values than the desired 0.95. Nonetheless, the R_1, R_2 values are generally high such that we can conclude that the ALNS-ES solutions are able to cover the simulated demand effectively.

Figure 9 in the Appendix shows an example of service levels (service type 1) at a random service point from instance Amsterdam 3 which encounters inability to meet all demand in several periods. The figure displays the first 100 periods in the simulation, where the vertical axis contains the periods. The horizontal axis contains the service levels, with the red bars indicating the amount of service that is required at the service point in each period. No bar indicates that there is no service required from this service point at the corresponding period. The vertical black line marks the capacity of this service point; if a red bar crosses this line, we have an exceedance in available capacity.

9 Conclusion and Further Research

In this thesis, we explored the method of using an ALNS-ES heuristic on the Multi Source Capacitated Facility Location Problem with stochastic demands that have been translated into deterministic variants. The problem involved identifying a best set of available access points to open in order to satisfy the demands of households in the most cost efficient manner. A model was designed which found a solution satisfying demands with a 95% level of certainty given that demands are normally distributed random values around a known mean. A couple of parameter options were tested before being made definitive to be used in the final computations. Comparing the heuristic results to the best solutions found with an exact analysis for small instances showed that the heuristic is able to deliver good solutions in much shorter time. For bigger instances, more runs may be desirable than the current preset value for improved solution quality. This of course increases the computation times but are expected to be still significantly shorter than the computation times needed for exact evaluation.

We also looked into the algorithm's behavior in case of two adjustments in the problem setting. The first one was a penalty on servicing demand points located past a certain distance from a service point, and the second being a penalty on the occupancy of each active service point. For the distance penalty extension, we expected an increase in number of open service points in the new solutions compared to the the same instances without the penalty. The corresponding results did not show consistent increases in open locations in the solutions, which is suspected to be either caused by the algorithm failing to converge with the new restriction or the distance penalty simply not having a big impact on many of the instances. For the penalty on occupancy of active service points, the proposed model has shown the opposite behavior of what was intended. Although penalizing the usage of capacity in each service point was expected to suppress the high occupancy of all active service points, the model instead opted to balance out the penalties paid by consuming high capacities on some service points and very low capacities on others, such that the average penalties paid still remains low. We conclude that for balanced solutions, penalizing capacity usage in one direction is counterproductive.

Afterwards, we tested the ALNS-ES solutions in a multi period traffic model and assessed their performance. It was of interest to see whether the single period solution with stochastic demand was able to cover the demand in several periods where the demand was simulated with a certain distribution. In overview, the ALNS-ES performs well in terms of the proportion of total demand quantity satisfied; with the exception of 1 case, all R_1 values are strictly above the 0.95 mark, and in most cases very close to 1.0. When it comes to the proportion of time all demand is satisfied across all time periods, R_2, the ALNS-ES solutions are occasionally not robust enough for model instances with low expected demand per time period caused by low transition rates λ compared to η, but show good performance in most scenarios. The rates R_1 and R_2 were mostly slightly higher for simulations with constant demand levels for each period with positive demand than simulations with variable demand levels with a mean equal to the constant demand level used. They remained however, above the criterion of 0.95 for cases with regular to high λ values compared to η.

We make a distinction between how accurate the ALNS-ES is in finding a solution for the multi period setting with a reliability of $\alpha = 0.95$, and whether the ALNS-ES is able to deliver a well performing solution to apply for the multi period setting given the desired reliability level. In terms of accuracy, for (λ, η) with a high $\frac{\lambda}{\lambda+\eta}$ values, the solutions seem a little too conservative; most R_1, R_2 values are closer to 1.0 than 0.95. For combinations with (λ, η) returning lower $\frac{\lambda}{\lambda+\eta}$ values, more R_1, R_2 are closer to 0.95. However, when we assess whether the ALNS-ES solutions provide a good overall coverage of demand in the multi period model, the results show that the ALNS-ES solution return high demand satisfaction levels which is favorable. We therefore

conclude that the solution of the ALNS-ES can be applied to a multi-period setting with the similar reliability, especially if the demand points in the model display frequent demand behavior.

To conclude, we propose a topic of research for future works. The most apparent anomaly of this study was the results following the implementation of the service point occupancy penalty. Although it is suspected that this was the cause of the algorithm allocating extreme service amounts, it is not yet confirmed. Since the overloading of service installations is practically undesired, finding a good solution to this setting is still of interest.

References

[1] R.M. Karp. Reducibility among Combinatorial Problems. *Proceedings of a Symposium: Complexity of Computer Computations* p. 85–103, 1972.

[2] 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, p. 544-559, 1999.

[3] R. Baldacci, E. Hadjiconstantinou, V. Maniezzo, A. Mingozzi. A new method for solving capacitated location problems based on a set partitioning approach. *Computers & Operations Research*, 29, p. 365-386.

[4] R.K. Ahuja, J.B. Orlin, S. Pallottino, M.P. Scarparra and M.G. Scutellá. A Multi-Exchange Heuristic for the Single-Source Capacitated Facility Location Problem. *Management Science*, 50(6), p. 749-760, 2004.

[5] G. Hoekstra. Location Assignment of Capacitated Services in Smart Cities. *Master's Thesis*, 2018.

[6] M. Verhoek. Optimising the Placement of Access Points for Smart City Services with Stochastic Demand. *Master's Thesis*, 2017.

[7] T.J.C. Vos. Using Lampposts to Provide Urban Areas with Multiple Services. *Master's Thesis*, 2016.

[8] T. Suzuki and J.M. Hodgson. Multi-Service Facility Location Models. *Annals of Operations Research*, 123, p. 223-240, 2003.

[9] C.K.Y. Lin. Stochastic single-source capacitated facility location model with service level requirements. *Int. J. Production Economics* 117, p. 439-451.

[10] B. Veerman. Optimizing the Distribution of Modular Capacitated Services in Smart Cities. *Master's Thesis*, 2019.

[11] K. Westcott, J. Loucks, D. Littman, P. Wilson, S. Srivastava, and D. Ciampa. Build it and they will embrace it - Consumers are preparing for 5G connectivity in the home and on the go. *Connectivity and Mobile Trends Survey*, 2019.

[12] K. Postek. Optimization under uncertainty: Lecture notes v.5, 2019.

[13] Motorola Solutions. Deployable LTE Solution For Fast Connectivity Where and When You Need It. *Data Sheet : LXN 6000 LTE Infrastructure*, 2019.

[14] S.J. Blumberg and J.V. Luke. Wireless Substitution: Early Release of Estimates From the National Health Interview Survey, January-June 2018. *National Health Interview Survey Early Release Program*, 2018.

[15] C. Chyu and W. Chang. Multi-Exchange Neighborhood Search Heuristics for the Multi-Source Capacitated Facility Location Problem. *Industrial Engineering & Management Systems*, 8(1), p. 29-36, 2009.

[16] Centraal Bureau Statistieken. Particuliere huishoudens naar samenstelling en grootte, 1 januari, 2020.

[17] T.J.C. Vos and F. Phillipson. Dense Multi-Service Planning in Smart Cities. *International Conference on Information Society and Smart Cities*, 2017.

[18] H. Hindi and K. Pienkosz. Efficient solution of large scale, single-source, capacitated plant location problems. *The Journal of the Operations Research Society*, 50(3), p. 268-274, 1999.

[19] J.A. Díaz and E. Fernández. A Branch-and-Price Algorithm for the Single Source Capacitated Plant Location Problem. *Journal of the Operational Research Society*, 53, p. 728-740 ,2002.

[20] M. Edous and O. Eidous. A Simple Approximation for Normal Distribution Function. *Mathematics and Statistics*, 6(4), p. 47-49, 2018.

Appendices

A Coordinates of Instances

Table 20 shows the latitudes and longitudes of locations that define the sub areas used as instances. Each coordinate corresponds to one vertex a rectangular span.

Instance	LatLong 1	LatLong 2	LatLong 3	LatLong 4
Amsterdam 1	52.377254, 4.797754	52.375588, 4.797278	52.375352, 4.800335	52.377261, 4.800609
Amsterdam 2	52.384509, 4.845755	52.375491, 4.846356	52.3854439, 4.867370	52.376897, 4.867080
Amsterdam 3	52.415372, 4.919551	52.410844, 4.928478	52.401785, 4.919959	52.404482, 4.909058
Amsterdam 4	52.364323, 4.826967	52.345398, 4.827482	52.345791, 4.845506	52.364270, 4.844305
Amsterdam 5	52.385764, 4.882002	52.372299, 4.873934	52.356680, 4.898138	52.370360, 4.935560
Rotterdam 1	51.878791, 4.465318	51.869729, 4.461434	51.868895, 4.476862	51.877413, 4.478836
Rotterdam 2	51.921718, 4.458600	51.910740, 4.460338	51.917850, 4.481487	51.924811, 4.477517
Rotterdam 3	51.934625, 4.455401	51.909375, 4.472192	51.928512, 4.516523	51.941609, 4.492233

Table 20: Latitudes and Longitudes of instances.

B Figure: Service Level

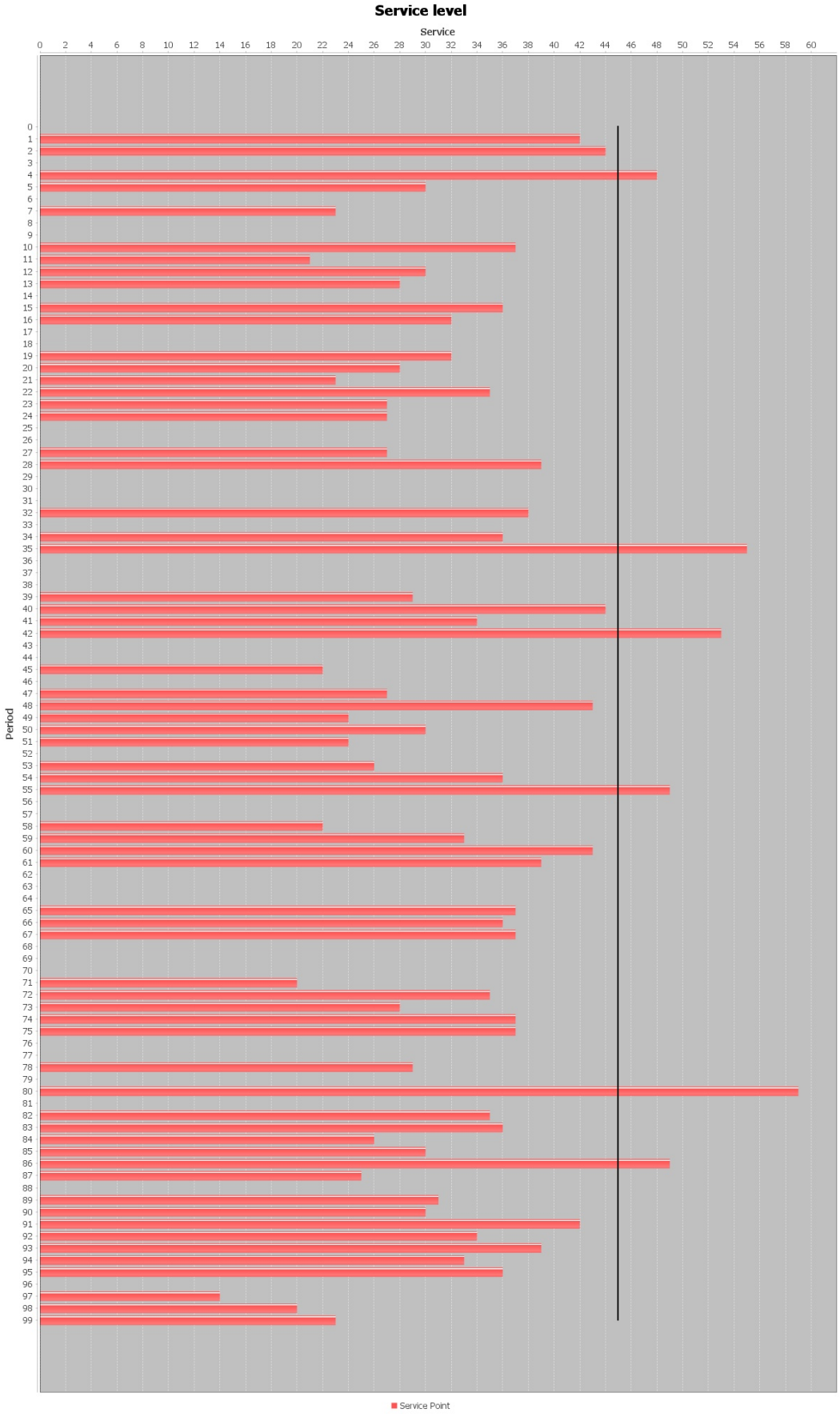


Figure 9: Service Levels for a Service Point in Amsterdam 3