Master thesis

Routing and scheduling of the parking enforcement in Amsterdam

Author: Jan Groeneveld

ARS | Traffic & Transport Technology
Supervisor: D.G. Speekenbrink

University of Twente
Supervisor: Dr. Ir. J.M.J. Schutten
Supervisor: Dr. Ir. M.R.K. Mes

Graduation date: 18.10.2017

ARS | Traffic & Transport Technology

Management summary

In December 2016, Egis Parking Services B.V. (EPS), who was hired by the municipality of Amsterdam to manage the parking enforcement within Amsterdam, tasked ARS T&TT (ARS) with the development of a planning tool that supports their work. The Smart Parking unit of ARS started to work on this project and additionally, requested a separate research on how such a planning tool can be developed. From February 2017 until September 2017, we conducted this research.

The idea behind the parking enforcement is that more parking visitors pay the parking fee. Whenever a parking visitor in Amsterdam wants to pay the parking fee, the visitor has to register the license plate of his/her car. The license plate is uploaded to a database afterwards. The on-street agents of EPS visit different neighborhoods of Amsterdam, i.e., they drive through neighborhood in parking enforcement vehicles (PEVs) and scan parked cars in different neighborhoods. During this process, the license plates of the parked cars are uploaded to a different database. By comparing both databases, it can be determined whether a visitor, whose car was scanned by a PEV, paid the parking fee. If a visitor did not pay, a penalty charge notice (PCN) is generated. For some exceptional cases, it is required that another agent who follows the PEVs on a scooter (PEF) checks the parked car on-site.

The output of our routing algorithm is a schedule of all neighborhood visits for all PEVs. In order to do this in a smart manner, we have to consider EPS' objective. The municipality of Amsterdam measures EPS' performance regarding the parking enforcement based on two Key Performance Indicators (KPIs): the payment rate, which is the ratio between paying visitors and all visitors, and the control chance, which is the probability that a non-paying visitor receives a PCN. From the control chance target, we can derive the number of PCNs that is needed in order to meet the control chance target. This number is called the PCN target, which we use instead of the control chance. Concerning the evaluation of EPS' performance, it is important that every neighborhood belongs to one of 10 KPI areas. Within one KPI period, which lasts three months, EPS has to meet certain targets of the KPI that are determined by the municipality. The municipality of Amsterdam takes random samples of neighborhoods and examines the payment rate. Whenever the payment rate measured by the municipality is below the pre-set target, the PCN target is considered. The rationale behind this is that EPS cannot directly influence the payment behavior of the visitors and therefore they have to show that their effort of fining the non-payers is at least high enough. If both targets are below the pre-set targets, then the KPI area is in a malus state and EPS will receive a fine. If in all KPI areas at least one of the targets is met, then EPS receives a bonus for those KPI areas where the payment rate exceeds the pre-set target. Apart from the KPI targets, EPS tries to visit every neighborhood once a week. Therefore, we derive the following three priorities in the following order:

1. Meet either the payment rate target or the PCN target of every KPI area.
2. Maximize the control chance in chosen KPI areas in order to eventually increase the payment rate and maximize the performance bonus.
3. Visit every neighborhood once a week.

Finally, it is required that our routing algorithm does not only maximize the number of PCNs but also takes these priorities into account. For that reason, we do not only consider the expected number of PCNs that can be obtained by visiting a neighborhood but also the neighborhood's Target Factor and Visit Day Factor, which take the mentioned priorities into account. Since the routing algorithm requires inputs, we first have to compute:

- Travel times (the travel time from one neighborhood to another)
- Service Times (the time that is needed to scan a neighborhood)
- Number of PCNs (the expected number of PCNs by visiting a neighborhood)
- Margin of error (an increase of the KPI targets to account for uncertainty)

For predicting the number of PCNs, we concluded that it is required to know the following ratios:

- The occupancy ratio, which is the ratio between occupied parking spots and all parking spots in a neighborhood at a given time
- The PCN ratio, which is the ratio between PCNs and all scans in a neighborhood at a given time

This PCN ratio can also be split into the visitor ratio, which is the ratio of visitor scans (scanned cars that belongs to paying or non-paying visitors) and all scans, and the non-paying ratio, which is the ratio of all non-paying visitor scans (PCNs) and all visitor scans. Our prediction model (or forecasting method) is based on an estimation of this occupancy ratio and a neural-network based prediction for the PCN ratio. Multiplying the number of parking spots with the occupancy ratio and the PCN ratio results in our forecast of the number of PCNs. This research also contains an extensive data analysis of the PCN ratio. We observed in our data analysis that:

- The PCN ratio depends on the time of the day

- The PCN ratio depends on the weekday
- The PCN ratio does not depend on the weather
- The payment rate increased from approximately 89% to 90% within one year (1.6.2016-1.6.2017)

The routing algorithm that we present in this research, first creates a solution based on a greedy algorithm and then tries to optimize this solution by constructing new solutions based on an ant colonization optimization (ACO) algorithm. Since it is possible to visit one neighborhood multiple times a day, one ingredient of our greedy algorithm is very important, namely the stability function. Our proposed stability function accounts for the fact that when the PEVs goes to a neighborhood that has been visited already the same day, it is possible that some non-paying visitors from the earlier visit are still in that neighborhood. Even though we show that the ACO algorithm performs well for one vehicle, it had difficulties to find better solutions than the greedy algorithm when 12 vehicles are deployed, which is the standard number of vehicles used by EPS (from Monday to Saturday). Finally, we perform a sensitivity analysis, a simulation study, and compare our prediction model and routing algorithm to the ones currently used at ARS. We prove that our neural network has a more accurate prediction (+4%) regarding the PCN ratio and that our routing algorithm leads to better results (+34%) than the current implementation of ARS algorithm when the same inputs are used (assuming that our stability function is correct). We are confident that our planning tool improves the current situation at EPS by automating the planning process, increasing the number of obtained PCNs, and faster reaching the KPI targets. This will finally lead to less fines and more rewards. Furthermore, we have shown that our greedy algorithm can create a planning for 90 days within 6 hours.

We recommend ARS to make use of our presented greedy algorithm with our presented parameter settings in combination with our neural network forecast. Finally, we have different ideas for future research. The most important ones are to investigate in the parking duration of non-paying (or at least paying) visitors in order to improve the stability function and to develop a more accurate method to estimate the occupancy ratio. Furthermore, we advise to do keep observing and analyzing the travel times because it seems that they are underestimated.

Preface

With this thesis, I am not only finishing my master program but also my entire study period at the University of Twente. On the one hand that makes me sad because I am closing a great period of my life, during which I have learned a lot about the world of industrial engineering, lived in three different countries, and I have met and worked with very special people. On the other hand, I am happy that I succeeded, I am proud to present my master thesis and I am looking forward to the future.

I thank ARS T&TT and Rolf Appel for the opportunity and giving me such an interesting project. Even though I never expected that one day I would optimize parking enforcement, I really liked the technical challenges that came with it. In this regard I also thank Ahmad Al Hanbali, who supervised my bachelor thesis and helped me finding this project.

My special thanks go to my supervisor Dennis Speekenbrink at ARS T&TT, who made always time for feedback and discussions despite of his busy agenda. It was a pleasure working with you and your critical thoughts and feedback were most helpful.

Of course, I also owe a great deal of thanks to my supervisors Marco and Martijn. You both spent a lot of time in proof-reading and finding every grammatical, writing, or essential error and answered my emails in no time. After hearing about other students' experiences, I know that this should not be taken for granted.

Lastly, I thank my friends and family who accepted and supported my decision to live and study abroad and visited me even when I was living in Portugal or Mexico. That also included my girlfriend Ellian, for whom I am most grateful. You always supported and comforted me during the stressful periods of my master program and motivated me to keep going.

Definitions & Notations

Ant colonization optimization (ACO): An optimization technique that is inspired by the pheromone trails that ants leave, in order to attract other ants to the ways that used to work well in the past.

ARS T&TT (ARS): The company where the research is conducted and which provides traffic and transport technology solutions to businesses and government.

Center point: The middle point of a neighborhood based on the scans of 3 months.

Chinese postman problem (CPP): An arc routing problem in which the route of a postman that has to deliver mail to different streets is optimized.

Control chance: The average number of PCNs within a not-paid-for parking hour, which is estimated.

Control chance target: A pre-set target regarding the control chance that is determined by the municipality for one KPI area regarding one KPI period.

Egis Parking Service (EPS): The company that is responsible for most of the operational aspects of on-street fiscal parking in Amsterdam, such as on-street parking meter enforcement.

Key Performance Indicator (KPI): A type of performance measurement that evaluates the success of an organization or of a particular activity.

KPI period: A recurring period of three months within which the KPI targets must be met.

KPI area: The 10 different areas in Amsterdam that are measured.

KPI targets: Include the PCN target, which is derived from the control chance target, and payment rate target.

Neighborhood: A small geographical unit within a KPI area.

Non-paying ratio: The ratio between non-paying visitors and all visitors in a neighborhood at a given time.

Parking enforcement: In this thesis we limit the parking enforcement to the activities related to the on-street parking meters in Amsterdam.

Occupancy ratio: The ratio between the occupied parking spots and all parking spots in a neighborhood at a given time.

Parking enforcement follow-up scooter (PEF): A scooter that is driven by a parking enforcement follow-up agent that go to the parked cars that need further investigation and/or where a PCN must be issued locally.

Parking enforcement vehicle (PEV): A vehicle that is driven by a parking enforcement agent and that scans parked cars.

Parking regime times: Indicates the time interval of a certain region (independent from KPI area and neighborhood) during which visitors have to pay for parking.

Payment rate: The ratio between paying visitors and all visitors in a neighborhood at a given time.

Payment rate performance: The performance of the payment rate in a KPI area, which is measured by dividing the current payment rate by the payment rate target.

Payment rate target: A pre-set target regarding the payment rate that is determined by the municipality for one KPI area regarding one KPI period.

Penalty charge notice (PCN): A parking fine that is issued whenever a non-paying parking visitor is detected.

PCN performance: The performance of the number of PCNs in a KPI area, which is measured by dividing the current number of PCNs by the PCN target.

PCN ratio: The ratio between non-paying visitor scans and all scans in a neighborhood at a given time.

PCN target: A target that is derived from the control chance target. Indicates the number of PCNs that has to be achieved in a KPI within a KPI period.

PEV driver: Parking enforcement agent who drives the PEV.

PEF driver: Parking enforcement agent who drives the PEF.

Service time: The time needed to scan a certain neighborhood.

Target Reached Parameter: A parameter that determines how important the KPI areas get after they have reached one of the KPI targets.

Team orienteering problem (TOP): A problem in which many vehicle have to maximize the rewards within a certain a time by choosing a set of vertices and the sequence of visiting them.

Travel Distance Restriction: A parameter that restricts the travel distance between two scheduled neighborhoods.

Travel time: The time needed to travel from the center point of a neighborhood to the center point of another neighborhood.

Travel Time Reduction: A parameter that reduces the accounted travel time when the PEV leaves a break location.

Travel Time Restriction: A parameter that restricts the travel time between two scheduled neighborhoods.

Traveling salesman problem (TSP): A problem in which a salesman has to travel to a certain set of cities and the travelled distance has to be minimized by choosing the best sequence of visits.

Visitor ratio: The ratio between visitor scans and all scans in a neighborhood at a given time.

Visitor scan: A scan of car that belongs to a paying or non-paying visitor.

Vehicle routing problem (VRP): The same problem as the TSP but usually with multiple vehicles.

Parking Rights Database (PRDB): A database that contains all important payment parking information.

Performance bonus: A bonus that increases when the payment rate target is exceed. The bonus is only given when the control chance and the payment rate target are met.

Stability parameter: A parameter that determines the fraction non-paying visitors that stay at a visited neighborhood for a certain amount of time.

Swap Threshold: A parameter that determines whether a swap is performed.

Visit Day Parameter: A parameter that determines the growth of the visit day function.

Table of Contents

1 Introduction

This project is part of the Master program Industrial Engineering and Management at the University of Twente. It has a limited time span of 6 months. We conduct this research at the Smart Parking business unit of ARS Traffic & Transport Technology. Within this research, we develop a planning tool to support the parking enforcement activities that concern the on-street parking in Amsterdam. In this thesis, we denote these activities as parking enforcement.

This first chapter provides an introduction to this research. Section 1.1 introduces the stakeholders who are involved in this project and describes core activities associated with parking enforcement. In Section 1.2, we identify the problem this research addresses. Finally, we define the scope of this research in Section 1.3 and explain our approach to tackle this problem including our research questions in Section 1.4.

1.1 Context

In order to better understand the background of this research, this section describes the stakeholders involved in the parking enforcement and how the parking enforcement in Amsterdam is executed and how its performance is currently measured.

1.1.1 Stakeholders

In this research project about the parking enforcement in Amsterdam, there are three important stakeholders: the municipality of Amsterdam, ARS Traffic & Transport Technology (ARS), and Egis Parking Services B.V. (EPS).

ARS is a company in The Hague that provides traffic and transport technology solutions to business and governments. Since 1997 it is active in its home market, the Netherlands, but also internationally (ARS T&TT, 2017). Concerning on-street parking, ARS is a partner in a joint venture with Egis Project, called Egis Parking Services B.V. (EPS). EPS operates from the shared service center in Amsterdam. In January 2016, the municipality of Amsterdam hired EPS to manage all operational aspects of on-street fiscal parking, such as for permit management, ticket machine maintenance, and parking enforcement. In December 2016, EPS tasked ARS with developing a planning tool regarding the parking enforcement in Amsterdam.

1.1.2 Parking enforcement in Amsterdam

Amsterdam has over 140,000 on-street parking spaces, dispersed amongst 10 fiscal parking areas. Each of these areas is divided into neighborhoods. In total there are 538 neighborhoods with different sizes (see Figure 1) of which 320 have fiscal (paid) parking.

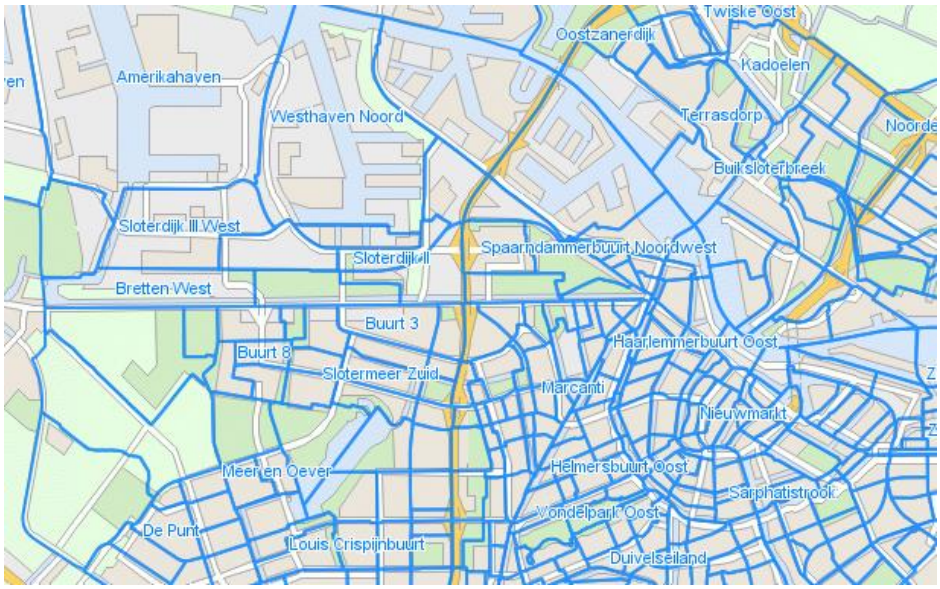


Figure 1 – Fiscal parking neighborhoods in Amsterdam

Every visitor who travels to Amsterdam by car and parks in a fiscal parking space has to pay a parking fee. The fee depends on the time the vehicle remains in the parking space and the parking space itself. A visitor can pay the parking fee through several electronic systems, such as parking meters, mobile payment, call-payment, and online visitor registration. The payment information of these systems, including the vehicle's license plate, is uploaded to a "Parking Rights Database" (PRDB).

Figure 2 – Parking enforcement vehicle

Parking enforcement vehicles (see Figure 2) drive through the neighborhoods, scan parked cars in the fiscal parking space, and take pictures of these. We further denote these vehicles as PEVs and their driver as PEV drivers. While scanning, the license plates of the parked cars are uploaded to a central system, which stores the recognized license plates. By comparing both databases, it can be determined which visitors did not pay a parking fee so that a Penalty Charge Notice (PCN) can be issued. The amount of the PCN is the sum of a fixed amount (the penalty) and one hour of the parking fee that should have been paid for that parking spot. In general, parked cars as scanned by the PEV can be classified as:

- Parking permit holders
- Exceptions
- Visitors
  - Paying visitors
  - Non-paying visitors
    - Domestic
    - International
  - Unclear situation

Parked cars with a parking permit belong to inhabitants that pay on a long-term basis. Some vehicles are exempted from parking payment because they are considered as exceptions (e.g., loading/ unloading vehicles and emergency vehicles). For this research, the most important groups are the domestic and international visitors who did not pay for their parking and visitors whose situation is unclear at first sight. Domestic non-paying visitors receive a PCN that is issued automatically. This automatic process is not possible for international non-paying visitors.

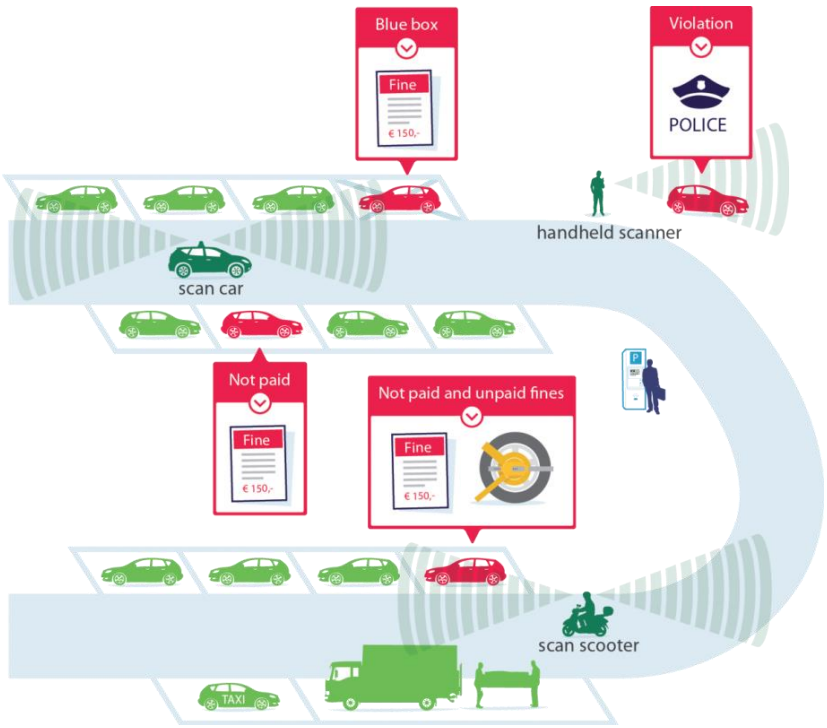


Figure 3 – Scanning process

That is why an off-street agent will contact a follow-up parking enforcement agent who drives with a scooter to the location of the international car to issue a PCN on-site (see Figure 3). As for the PEV, in this research, we denote the scooters as PEFs and their drivers as PEF drivers. For some vehicles, it is unclear if they paid for their parking. Possible reasons are that the license plate is unreadable in the provided images or it is unclear whether there is a loading/unloading process. Since it has to be determined whether the vehicle belongs to a visitor that did not pay, these unclear parking situations require an on-site visit of the PEF driver as well. At this moment, there is a 1-to-1 relation between the PEVs and PEFs, i.e., one PEF follows one PEV. If a visitor receives a second PCN and did not pay the first one, the municipality may request that a wheel clamp is placed on the vehicle.

1.1.3 Measurement method of the parking enforcement in Amsterdam

The purpose of parking enforcement is to ensure that citizens and visitors pay for their parking. The municipality of Amsterdam measures this by means of a Key Performance Indicator (KPI), namely the "payment rate". The payment rate is the willingness of visitors (non-permit holders) to pay for their parking. In other words, it is the ratio of paying visitors in relations to the total number of visitors. In this research, we denote it as the payment rate and not payment ratio because this is the term that is currently used at ARS. The municipality measures the payment rate in all 10 fiscal parking areas for a period of 3 months by taking random samples. In this research, we denote these areas as KPI areas and the 3 month period as the KPI period. The performance of EPS with regards to the parking enforcement is evaluated by the payment rate that the municipality measures for every KPI in a KPI period. Unfortunately, there is no direct relationship between the parking enforcement and the payment rate because it is unknown to what extent the parking enforcement affects the payment rate. In theory, it could happen that EPS does a great job and every visitor who does not pay a parking fee gets a PCN but the payment rate does not increase. Even though, this is very unlikely because the general assumption is that people pay for parking if the chance of getting a fine is too high. This assumption, that enforcement influences the payment rate, is confirmed in the literature, as Adiv and Wang (1987) show that parking non-compliance level increases as the level of enforcement decreases. Peliot (2004) indicates that this phenomenon can be described as a relational economic choice (portfolio choice), i.e., the driver asses the risk of getting a fine and the amount of the fine versus the regular parking costs. This theory is confirmed by Adiv and Wang (1987) and Elliot and Wright (1982) in an empirical study. Since it is not desirable for the municipality to increase the amount of the PCNs or the parking fee, they want to increase the risk of getting a fine (PCN). For that reason, the municipality does not only consider the payment rate but also the "control chance", which is the probability that non-paying visitors receive a PCN. Only if the payment rate is not met, the municipality will consider the control chance as a means to establish that "enough" effort has been put into enforcement. Only when for all KPI areas either the payment rate or control chance target has been achieved, the municipality will give EPS a performance bonus for any KPI area where the payment rate exceeds the pre-set target (not for the control chance). Note that every KPI area has different targets for the payment rate and control chance. These targets increase after every KPI period (up to certain maximum values).

1.2 Problem identification

In this section, we define the problem which enables us to formulate an approach to tackle the problem.

In the context of this thesis, EPS has two planning tasks, namely the daily routing of the PEVs and the staff scheduling. Currently, EPS handles both planning tasks manually. Since capacity is limited and enforcement targets are rising, there is a need for an automated planning tool that delivers the following three outputs:

- PEV routing: The planning of the daily PEV routes indicates at which time drivers need to be in a certain neighborhood. Obviously, this will require inputs to determine realistic and smart routes.
- Staff scheduling: The staff has to be assigned to the vehicles. This can be done separately from the PEV routing.
- Estimation of the KPI results: An indication to what extent the KPI targets will be met at the end of the KPI period. This supports the decision process with regards to the needed capacity for the short term.

The goal of this planning tool is to increase the efficiency of the available capacity and reducing the effort of manual scheduling. Increasing efficiency is always linked to an objective. As we derive from Section 1.1.3, it is the objective to meet either the targets of the payment rate or the control chance in all KPI areas. Furthermore, EPS wants to visits every neighborhood once a week. In fact, EPS has an order of priority with regards to their targets:

1. Meet either the payment rate target or the control chance target of every KPI area.
2. Maximize the control chance in chosen KPI areas in order to eventually increase the payment rate and maximize the performance bonus.
3. Visit every neighborhood once a week.

As a higher number of PCNs means that EPS is performing better with regards to the control chance (the exact formula will be explained in Section 2.1.8), the planning tool has to maximize the numbers of PCNs in such a way that the required KPI targets are met in all KPI areas and eventually a performance bonus is achieved.

ARS has started the development of such a planning tool in December 2016. This thesis, which started in February 2017, can be seen as a part of this project that aims to develop a more accurate and smarter planning tool that is supported by scientific literature.

1.3 Research scope

In this section, we discuss which aspects we do and do not consider in this project. The objective of this graduation project is to create a planning tool for the parking enforcement in Amsterdam. In Section 1.2, we stated that the planning tool should deliver the PEV routing, the staff scheduling, and an estimation of the KPI results. The staff scheduling, however, is an independent smaller and less crucial problem and therefore we only focus on the development of a routing algorithm that determines the daily routing of the PEVs and also estimates the KPI results. The development of this planning tool involves three main steps:

First, before creating any outputs, we need inputs for the routing algorithm. To this end, we have to analyze which inputs are needed. For example, we need to know how long it takes to travel between the neighborhoods and how long it takes to scan one. Furthermore, we need to know how many PCNs we expect to generate when scanning a neighborhood. For the development of these inputs, we can make use of the literature, the knowledge of EPS, and data of all parked cars that were scanned since January of 2016.

Second, we develop the routing algorithm. This routing determines for every PEV the sequence of neighborhood visits (including the time). The routing algorithm implies also embedding this algorithm in a programming platform. EPS requires that the total computational time must not exceed a daily limit of six hours. Apart from the routing, the routing algorithm needs to estimate whether the KPI targets will be met at the end of the KPI period.

Third, the planning tool needs to be validated afterwards, such that the functionality and contribution of the tool can be proven.

Consequently, the planning tool consists of useable input data and a routing algorithm that is embedded in an application. In this regard, there are some related aspects that are beyond the scope of this research. First of all, we only assign drivers to neighborhoods and not to streets. Even though we do have data of all scans since January of 2016 including GPS coordinates, EPS asks for a system that is based on neighborhoods. The reason behind it is that a street based planning is too strict and cannot be executed accurately it practice. A neighborhood routing gives EPS more flexibility. Additionally, creating street-based routes would increase the solution space of the problem and therefore the computation time. Furthermore, we do not consider decisions made on an online operational level, i.e., we do not take dynamic aspects into account. For instance, if an accident occurs, the original route is not adjusted. Such online changes require detailed real-time instructions (e.g., a navigation tool in the car) but this is currently not possible. Neither, do we make decisions on a strategic level such as reducing the number of PEVs. Nevertheless, the capacity might change in the future and therefore we use number of deployed PEVs as a parameter. Furthermore, we do not consider reducing the number of PEFs and assigning them to multiple PEVs because we focus on the routing of the PEVs. Moreover, we do not consider the fact that visible presence of the PEV possibly prevents parking violations (comparable with the presence of police cars preventing possible crimes).

1.4 Problem approach

This section describes the plan of approach of this research, which also includes the research questions.

Before introducing all research questions, we present our research goal, as concluded in Section 1.2:

"Develop a planning tool for the parking enforcement that maximizes the number of PCNs in such a way that the required KPI targets are met in all KPI areas and eventually a payment rate bonus is achieved"

From this research goal, we derive research questions, which are discussed in the following chapters:

<u>Chapter 2 – Current situation</u>

This chapter describes the current situation of the planning and routing. By conducting interviews and reviewing the available data, we answer the following two questions:

- How does the current routing of the PEVs look like?
- How does the current planning process look like?

Chapter 3 – Literature

This chapter introduces the required literature of this thesis. First of all, we want to know if this problem or similar ones have been introduced to the literature and how these problems have been tackled and solved. Furthermore, we investigate how inputs for the routing algorithm can be developed by using available data. Furthermore, we are interested in theories about the parking and payment behavior of visitors. Consequently, we answer the following questions:

- What is known in the literature about problems regarding the routing of parking enforcement or similar routing problems?
  - Which solution methods does the literature suggest?
- What is known in the literature about the parking and payment behavior?
- What is known in the literature about developing input data?
  - What is known about travel times or speed models?
  - What is known about prediction models?

For the purpose of this literature research, we use Scopus and Google Scholar.

Chapter 4 – Computation of inputs

This chapter analyzes the gathered data and investigates patterns and statistical characteristics in order to compute inputs that can be used for the routing algorithm. We answer the following questions:

- How can we use the historical data to develop inputs for the routing algorithm?

In order to answer these questions, we first clean the available data and make some transformations if necessary. Afterwards, we analyze the data to find patterns and develop a data model. By means of this data model, we create inputs for the planning tool.

Chapter 5 - Routing algorithm

Within this chapter, we design a routing algorithm that can be embedded in the planning tool. Furthermore, we discuss the choices with regards to the algorithm and the strategies behind it.

- What kind of algorithm is most suitable for this problem?
- How can we measure the performance of the algorithm?

Chapter 6 - Results

In this chapter, we analyze the results of the planning tool to make sure that this tool is actually working and improving the current situation.

- To what extent is the proposed planning tool improving the current situation?

Chapter 7 – Conclusion and Recommendations

This chapter summarizes the project, discusses important points, indicates possibilities for future studies, and finally lists our recommendations.

2 Current situation

This chapter describes the current situation of the routing (Section 2.1) and planning process (Section 2.2) at EPS. This chapter is based on interviews with the manager and drivers from EPS and data that they provided.

2.1 Routing

In this section, we describe the routing of the PEVs in terms of different routing characteristics proposed by Van der Heijden and Van der Wegen (2011). These general characteristics are applicable to every routing problem.

2.1.1 Fleet characteristics

The fleet consists of a certain number of homogenous PEVs and PEFs, which is set a priori. Currently, one PEF is assigned to one PEV and 12 of both are used in the daily operations. A vehicle can be unavailable due to maintenance or other reasons. Every morning the vehicles are refuelled.

2.1.2 Depot characteristics

The fleet of PEVs and PEFs starts and finishes the daily routes at the same depot. In between, the drivers have breaks, which are further explained in Section 2.1.5. For these breaks, the fleet may return to the depot or to two extra break locations, which are exclusively used for breaks. The fleet must take every break at one of the three possible break locations. The break locations are scheduled based on the shortest extra travel time with regards to the scheduled route. Figure 4 shows all neighborhoods, the depot (marked in red), and the two break points (marked in green).

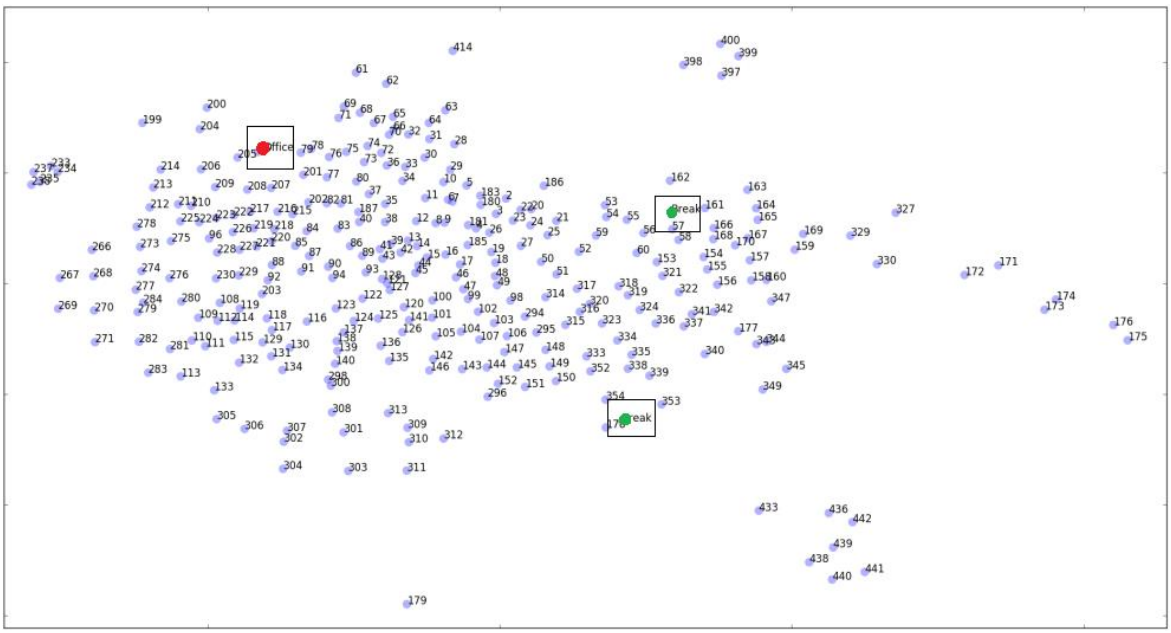


Figure 4 – Overview of neighborhoods, the depot, and the two break locations

2.1.3 Customer characteristics

The "customers" in this problem are the 10 KPI areas with certain KPI targets that have to be met. Achieving the KPI targets, which is further explained in Section 2.1.8, involves the PEV visiting the neighborhoods within these KPI areas in order to scan the parked cars. Every KPI area is divided into several neighborhoods with different sizes. As mentioned in the Section 1.3, we consider the neighborhoods globally and not every specific street in it. According to EPS, the number of visitors and their payment behavior is affected by various factors, such as the time of the day, weather, holidays, markets, special days of sale, and short-term events. Chapter 3 discusses what is known in the literature

about factors influencing the number of visitors and the payment behavior. The data analysis of these factors is part of Chapter 4. This analysis helps us to make predictions about how many PCNs we can expect when scanning a neighborhood. The time that a PEV driver needs to scan a neighborhood, is denoted as the service time. The service time depends on the average speed of the PEV and the length of the route within the neighborhood. The PEV drivers indicate that these service times are time-dependent due to traffic congestion. Chapter 4 deals with estimation of this service time and the prediction of the expected PCNs while scanning. Since it is possible to go more than once a day to one neighborhood but it is not possible to issue multiple PCNs on one day for the same vehicle, another problem arises, namely: How many PCNs can we expect the second time? Or more correctly: How many non-paying visitors remain on the parking spot until the next visit? This problem is discussed further in Section 4.3.4.

Regarding the locations of the neighborhoods, the neighborhoods have the shape of polygons and ARS has the GPS coordinates of polygons' edge points. The sizes, shapes, and number of corner points are different for every neighborhood. ARS created an extra GPS coordinate for every neighborhood that is the center point of all scans during 3 months and therefore we denote this point as center point.

2.1.4 Travel characteristics

As for the service time, the travel times between two neighborhoods depend on the distance between the neighborhoods and the average speeds of the PEV.

The speed is influenced by time-dependent traffic congestion. Note that we do not consider the allowed speeds of the streets since we do not consider streets in this research. Moreover, due to one-way streets or one-way traffic congestions, the speeds might be directed, i.e., it matters if the PEV goes from neighborhood i to j or from j to i. Chapters 4 discusses whether it is necessary and possible to include this.

The distance between two neighborhoods depends on the last scanned street of the previous scheduled neighborhood and the first scanned street of the following neighborhood. ARS created a distance matrix using the center point of every neighborhood that we explained in Section 2.1.3. This distance matrix considers the actual distance traveling through all streets from one center point to another. This leads to a problem because the center point lies within the neighborhood. Therefore, this distance includes also the distance between the point when the PEV enters or leaves the scheduled neighborhood and its center point. Consequently, if we measure the distance between the two centers points of the two neighborhoods, we will calculate twice the redundant distance from the entry/exit point of the neighborhood and its center point. This distance, however, is difficult to determine as it depends on the last scanned street of the previous neighborhood and the first scanned street of the following neighborhood. Chapter 4 further discusses this problem and the development of useful travel time input.

Furthermore, we must not forget that the PEV keeps scanning parked cars while traveling to other neighborhoods. For instance, the PEV travels through other neighborhoods to get to the next destined neighborhood. However, in that case, the PEV drivers mostly use through-streets that are less dense with regards to the number of PCNs. In Chapter 4, we also answer the question whether it is useful to include this phenomenon.

2.1.5 Time restrictions

In order to create proper routes, we have to take several time restrictions due to shifts, breaks and parking regimes into account. Regarding the shifts, there are three regular shifts from every day of the week:

Regular shifts (from Monday until Sunday):

- Day shift: 8.00-16.30
- Evening shift: 15.30-23.40
- Night shift: 23.30-4.00

However, there is also on additional shift on Sundays due to different parking regimes:

Additional shift (on Sundays):

- Sunday afternoon shift: 11.30-20.00

Even though on Sundays there are additional Sunday afternoon shifts, the total number of deployed PEVs is usually less on Sundays than during the week.

At the start of every shift, the drivers get a briefing at the depot. The drivers of the day shift also have to refuel the vehicles. When the day shift ends, the PEV driver of the evening shift drives with a scooter to the current location of the PEV, then they switch, and the driver of the day shift returns with the scooter. The shift change occurs between 16.00 and 16.15 and requires 15 minutes at the same location. Therefore,

the PEV driver must not have a visit that starts before 16.00 and finishes after 16.15. The shift change from evening to night shift is done at the depot, hence we do not have to take it into account. Except for the night shift, all shifts including one 20-minute break and one 35-minute break. The night shift drives without a break. EPS decided to vary break times a bit in order to spread the breaks around and to avoid moments in which no PEV is driving around. The reason behind this is that it lead to visitors not paying during break times of the PEV drivers. Currently, this is done by means of the vehicle number, where even numbers have one break regime, and uneven numbers have another. The break times are as follows:

Even numbers:

- Dayshift breaks: 11.30 – 12:05 and 13.45 – 14.05
- Evening shift breaks: 18.15 – 18.50 and 21.15 – 21.35
- Sunday afternoon shift breaks: 14.15 – 14:35 and 17.00 – 17.35

Uneven numbers:

- Dayshift breaks: 11.15 – 11.50 and 14.00 – 14.20
- Evening shift breaks: 17.45 – 18.20 and 20.45 – 21:05
- Sunday afternoon shift breaks: 14.00 – 14.20 and 16.45 – 17.20

Even though EPS uses fixed break times, we can use a tolerance of 15 minutes to increase the flexibility of the planning. The PEV driver should choose the break location which leads to the shortest total travel time. The PEF drivers have to do the follow-up work. Correspondingly they go to the same break location and their breaks start a bit later than the ones of the PEV drivers.

Furthermore, there are different parking regime times, which indicate at which time a visitor has to pay a parking fee. The parking regimes times of Amsterdam can be seen in Table 1.

KPI area	**Subarea within KPI area**	**Parking regime**	
		Days	**Hours**
Centrum	A	Monday - Sunday	09.00-24.00
	B	Monday - Sunday	09.00-04.00
Nieuw-West		Monday - Saturday	09.00-19.00
Noord	A	Monday - Saturday	09.00-19.00
	B	Monday - Sunday	12.00-19.00
	C	Monday - Sunday	09.00-24.00
Oost 1/ Oost 2	A	Monday - Saturday	09.00-19.00
	B	Monday - Saturday	09.00-21.00
	C	Monday - Saturday	09.00-24.00
West 1	A	Monday - Saturday	09.00-24.00
	B	Monday - Sunday	09.00-24.00
West 2	A	Monday - Saturday	09.00-24.00
	B	Monday - Saturday	09.00-19.00
Zuid 1	A	Monday - Saturday	09.00-19.00
	B	Monday - Saturday	09.00-21.00
	C	Monday - Saturday	09.00-24.00
Zuid 2	A	Monday - Saturday	09.00-21.00
	B	Monday - Saturday	09.00-24.00
	C	Monday - Friday	09.00-19.00
Zuid Oost		Monday - Sunday	09.00-21.00

Table 1 – Parking regime times

Sometimes the parking regime times differ within one KPI area. Therefore, some of them are split into two or three subareas. ARS already assigned the neighborhoods to different parking regimes.

2.1.6 Route restrictions

We do not have restrictions with regards to the length of the route. The route is only restricted by the time as mentioned before.

2.1.7 Costs factors

Cost factors are factors that need to be minimized. In a usual VRP, this would be the travel times or travel distances. As we further explain in Section 2.1.8, the maximization of the number of PCNs is the crucial output in this routing problem. Even though EPS has no interest in minimizing the travel distances, it is important that the travel distances are reasonable such that the PEF driver can still follow the PEV on the scooter. A smart maximization of the number of PCNs will automatically minimize travel times to some extent to improve the efficiency of the route. Nevertheless, it could be interesting to keep track of the travel distances for two reasons. First, the routes will not have the same amount of kilometers. So assuming all PEV are interchangeable, at a later time it may make sense to arrange vehicles amongst schedules such that they do not all reach their next maintenance requirement at the same time. Second, if we assume that some vehicles have less range than others (e.g., electric vehicles) it may be useful to assign specific vehicles to routes with less travel distance. Therefore, the travel distance would be nice to have but should not impact the core of the routing algorithm.

2.1.8 Optimization criteria

As already mentioned in Section 1.2 the most important KPI for the municipality is the payment rate, the fraction of visitors that pay for parking. The problem of this KPI is that we do not have the information of all visitors. The number of paying visitors is known as they are saved in the PRDB. Logically, the non-paying visitors are not registered in the PRDB. Only those non-paying visitors are known that are scanned and issued with a PCN but these are not all of them. Basically, the payment rate (p) should be measured with the following formula:

$$p = \frac{Number\ of\ paying\ visitors}{Number\ of\ paying\ visitors + Number\ of\ non\ paying\ visitors},$$

but since we do not know the number of non-paying visitors, it is measured with the following formula:

$$p = \frac{Number\ of\ scanned\ paying\ visitors}{Number\ of\ scanned\ visitors}.$$

Therefore, we can only estimate the payment rate for a certain sample of scans. For example, a PEV starts to scan a small neighborhood with 100 parking spots at time t = 0. After 10 minutes at t = 10, the PEV has scanned all 90 cars. Out of these 90 cars, 40 cars had a parking permit and 50 cars were visitors. 10 of these visitors, did not pay the parking fee and will receive a PCN. Assuming that nobody left a parking spot or arrived to the parking spot within these 10 minutes, we can make the following conclusions from this example:

- The occupancy ratio is 90%, which is the ratio between occupied parking spaces and the total number of parking spaces
- The visitor ratio is 50/90%, which is the ratio between visitor scans (all scanned cars that belong to a paying or non-paying visitors) and all scans
- The non-paying ratio is (40/50%), which is the ratio between non-paying visitors and all visitors (equal to 1-p)

In this context, we thought of a pyramid (see Figure 5) that explains the components of the number of PCNs. If we consider a specific time interval and take the known number of parking spots N, multiply it by the occupancy ratio S/N (number of scans divided by number of parking spots), by the visitor ratio V/S (number of visitors divided by the number of scans), and then by the non-paying ratio 1-p, then we get the number of PCNs. Therefore, the number of PCNs can be expressed as PCN = N*(S/N)*(V/S)*(1-p) or PCN = V*(1-p). As mentioned in Section 2.1.3, the number of visitors (or the visitor ratio) and payment rate are affected by different factors, which we investigate in Chapter 3 and Chapter 4.

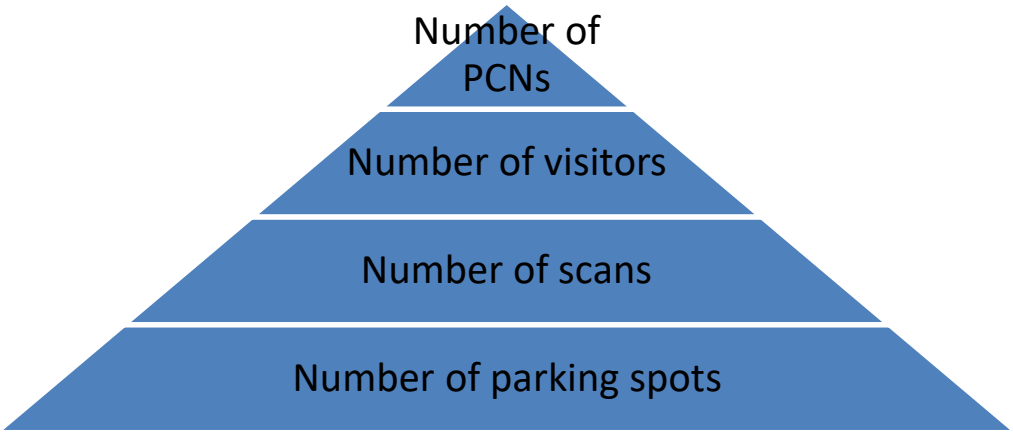


Figure 5 – Pyramid showing the factors on the number of PCNs

Regarding the number of PCNs, approximately 10% of the PCNs that are immediately generated are removed afterwards. A common reason for this is that a visitor pays the parking fee but registers the wrong license plate and therefore receives a PCN. After a complaint, these PCNs will be deleted and therefore not accounted with regards to the KPI targets. Within this research, we only consider the number correctly issued PCNs.

As stated in Section 1.2, EPS tries to increase the control chance in order to eventually increase the payment rate and show the municipality that the enforcement effort is at least high enough. The control chance is supposed to indicate the fraction of non-paying visitors that are "caught" and issued a PCN. However, the municipality, and consequently also EPS, computes the control chance (c) by dividing the number of correctly issued PCNs in a KPI area by the estimated number of not-paid-for visitor parking hours. Consequently, this means that the control chance is not a probability. More accurately, we should call it the average number of PCNs per not-paid-for visitor parking hour. This could be improved by dividing the number of not-paid-for visitor hours by the average parking duration of non-paying visitors. However, since this is the way the performance is measured by the municipality and EPS, we continue explaining how this number is estimated. For this purpose, the municipality uses the payment rate (p). First, they estimate the total number of visitor parking hours by dividing the paid-for visitor parking hours, which can be retrieved from the PRDB, by the payment rate:

$$\text{Visitor parking hours} = \frac{paid\ for\ visitor\ parking\ hours}{p}.$$

In order to estimate the number of not-paid-for parking hours, they subtract the paid-for visitor parking hours from the total number of visitor parking hours:

$$not\ paid\ for\ visitor\ parking\ hours =$$

$$\frac{paid\ for\ visitor\ parking\ hours}{p} - paid\ for\ visitor\ parking\ hours =$$

$$\left(\frac{1}{p} - 1\right) * paid\ for\ visitor\ parking\ hours\ .$$

Finally, the formula of the control chance (c) is:

$$c = \frac{Number\ of\ PCNs}{\left(\frac{1}{p} - 1\right) * paid\ for\ visitor\ parking\ hours}.$$

The measured payment rate, the control chance target (c_{target}), and the paid-for visitor parking hours can be inserted in the formula. By doing so, the number of PCNs needed in every KPI area for the 3 month of the KPI period can be estimated. This number is called the PCN target:

$$\text{PCN target} = c_{\text{target}} * \left(\frac{1}{p} - 1\right) * \text{paid for visitor parking hours.}$$

Since the PCN target is always derived from the control chance target, we always refer to the PCN target and the payment rate whenever we speak about KPI targets further in this research.

There is one major problem with regards to the KPI targets, namely the fluctuation of the payment rate and the paid-for visitor parking hours. Both can be estimated but a false estimation can lead to not achieving the PCN target. Every KPI period, the municipality measures the payment rate by taking random samples to check whether the target is reached. The size of the samples and when and where they are taken is unknown. Consequently, EPS has the problem that they do not know which payment rate the municipality finally uses to measure their performance. This makes it difficult to set a fixed PCN target. The good thing is that EPS can estimate the payment rate based on a large number of recent scans, namely all scans of their daily planned routes. As the daily planned routes are not planned randomly, one can say that they use non-random samples but with large sample sizes to represent the entire population of the KPI areas. The paid-for visitor parking hours are based on historic data saved in the PRDB. We will tackle the uncertainty problem of the payment rate and the paid-for visitor hours in Chapter 4.

Considering the KPI targets, a KPI area can have one of the three following statuses:

- Malus – None of the targets is reached. In this case there is a fine for the difference between the control chance target and the actual performance since the effort of EPS is not big enough.
- Neutral – The PCN target is reached and the payment rate measured by the municipality does not exceed the
- Bonus – The payment rate measured by the municipality exceeds the payment rate target.

Note that EPS only receives a bonus for a KPI area if none of the KPI areas has a malus status. In this regard, it is important to remember that due to the uncertainty of the KPI targets it is possible that they turn out to be lower than expected at the end of the KPI period (further discussed in Chapter 4). Finally, as already mentioned in Section 1.2, the objective of the routing follows a certain order of priority:

1. Meet either the payment rate target or the PCN target (respectively the control chance) of every KPI area (bring all KPI areas at least to a neutral status).
2. Maximize the PCN target in chosen KPI areas in order to eventually increase the payment rate and maximize the performance bonus.
3. Visit every neighborhood once a week.

The order of these priorities must not be interpreted as a sequence of actions, i.e., first we only act on the first priority, then on the second, and finally on the third. All priorities should rather be taken into account at all times. Even though the third point of the priority list ("visit every neighborhood once a week") is only a soft constraint, it is important to visit all neighborhoods in order to collect data. Otherwise, the following scenario might happen:

KPI area A consists of 5 neighborhoods A1, A2, A3, A4, and A5. The daily measured payment rate of KPI area A is below the required payment target. Consequently, the PEVs have to scan this area to reach the PCN target. If the PEV drivers know that it is likely that they can issue a lot of PCNs in the neighborhoods A1 and A2, they will probably drive there, in order to reach the PCN target faster. If they keep doing this, they create blind spots because they do not scan neighborhoods A3, A4, and A5 anymore. Hence, they do not collect data of the payment rate in these neighborhoods. These blind spots are dangerous since they lead to a misconception of average payment rate of the entire KPI area. If the municipality measures the payment rate only in one of these "blind spot neighborhoods" (A3, A4, A5), it might happen that the payment rate that is measured by the municipality is actually lower than expected. Considering the formula of the PCN target, this target increases with a lower payment rate. In the end, this could lead to EPS not meeting neither of the KPI targets.

2.2 Planning

In this section, we briefly explain how EPS currently manages the planning of the PEV routes.

EPS determines the staff scheduling for one year. Normally, they deploy around 12 PEVs from Monday to Saturday and less on Sundays. For one shift with 12 PEVs, they usually need 28 people: 12 driving the PEVs, 12 driving the PEFs, and 4 operating as off-street agents. For the night shift, EPS usually schedules only one PEV. If they know that there will be a shortage of drivers in the next two week, they can hire extra drivers from an external company.

EPS needs to determine the tasks of the staff and the routes of the PEVs for the following day on a daily basis. The PEV routes are based on recent scan results. Every day, EPS receives the results regarding the scans, PCNs, and payment rate of the previous day. Furthermore, they receive how many paid-for parking

hours there have been in the previous week (Tuesday till Monday) every Tuesday. As described in Section 2.1.8, they use all this recent data to estimate both KPI targets of the current KPI period. EPS adjusts the PCN targets every day such that they know many PCNs they should have issued until the day of the planning. For instance, if they estimated that the PCN target of KPI area A is 900 at the end of the KPI period, which is day 90, then they would have required 400 if today was day 40. This means that they assume in this planning that every day they can issue the same number of PCNs. Every day, they plan the routes for the next day as follows:

1. The results of the past two days are analyzed.
2. They check whether all drivers are available for the next day. Reasons for not being available are holidays, illness, or appointments.
3. The availability of the PEVs and PEFs is checked. Sometimes they are unavailable due to maintenance or damage.
4. They determine the routes by assigning the PEV drivers to certain neighborhoods that they should visit between the breaks. The routes are based on the analysis of the last days, the results of the KPI targets, and their priority order as discussed in Section 2.1.8.

Even though the planning is done one day in advance, it can still happen that employees are suddenly unavailable the following day. In this case, employees who were initially scheduled to operate as an off-street agent need to drive a PEV, to ensure that the capacity of the PEVs is efficiently used.

2.3 Conclusion

We conclude from the current situation that there are several inputs needed for routing algorithm.

- KPI targets: Every 3 months the fixed KPI targets for the payment rate and control chance (respectively PCN target) are changed. The PCN target is recomputed every day to ensure that it contains the most recent data.
- Available capacity: In this project, the capacity depends on the availability of PEVs, PEFs, and shifts. On the one hand, this includes daily information concerning the available staff and equipment. On the other hand, this includes a staff roster, which is set a priori.
- Restrictions: We have several restrictions, such as the parking regime of the neighborhoods, the shifts and break times of the drivers. Also, the drivers must return in the end to the depot and for their two breaks, they have to go to one of the three break locations.
- The expected number of PCNs: We want to know how many PCNs can be expected when a specific neighborhood is scanned at a certain time.
- Service time: The service time is the time needed to scan a neighborhood at a certain time.
- Travel times: In order to calculate the time needed for every route, we need to calculate the travel time from one neighborhood to another at a certain time.

Regarding all these inputs, the three inputs service times, travel times, and future PCNs have yet to be established. Since we have historical data available of all scans made since 1.1.2016, we can use this data to develop inputs that can be used for the routing algorithm. In should be taken into account that the travel and service time both depend on the travel speed and consequently on traffic congestion. The expected number of PCNs is more complicated, as it involves more factors and probably not all of them are measured in the current data set, such as weather circumstances. Regarding the expected number of PCN, the question how long visitors that already received a PCN remain at the same parking spots, needs to be answered. Furthermore, we have to solve the problem concerning the uncertainty of the PCN targets. The development of all these inputs is the scope of Chapter 4. The objective and the design of the routing algorithm are discussed in Chapter 5.

3 Literature

This chapter presents the findings of our literature research. Section 3.1 discusses similar routing problems in the literature and Section 3.2 possible solution approaches. Furthermore, Section 3.3 contains information about speed models, prediction models, and factors that influence parking and payment behavior.

3.1 Problem review

In this section, we review the literature regarding our problem such that we can define the problem and find solution approaches for it. In our literature research, we only found one article that deals with the routing of parking enforcement (searching method is shown in Appendix A). Summerfield, Dror, and Cohen (2015) state that the problem of designing an online parking enforcement algorithm that maximizes the revenue collection has not yet been introduced to their knowledge. Summerfield et al. (2015) model the task of designing parking permit inspection routes as a revenue collecting Chinese Postman Problem. The original Chinese Postman Problem (CPP) aims to find the shortest route for a postman with the requirement that the postman covers every street (denoted as edge or arc) at least once (Gendreau & Laporte, 1994) and returns to the start location. In the problem of Summerfield et al (2015), every edge has certain weights. Since not every edge has to be traversed, it is the goal to maximize the total amount. We also know that in Portugal a research group is working on a similar problem with weighted arcs in the parking enforcement. Considering this problem as an arc routing problem (such as the CPP) with weighted arcs is a logical approach in most countries, as they plan routes for walking agents going through streets (similar to the postman). In this research, however, we consider the neighborhoods within a city and therefore our problem focusses on nodes (also denoted as vertices). Nevertheless, for future studies, this (weighted) arc routing problem might be interesting. For instance, once the routing of all neighborhoods is done, algorithms to solve the CPP could create the route within a neighborhood. To this end, we can use different variations like:

- Open CPP: postman does not return to the original destination (Thimbleby, 2003).
- Windy or directed CPP: edges are directed, meaning that it matters in which direction you are traversing the street (Eiselt, Gendrau & Laporte, 1994).
- Mixed CPP: edges can be both directed and undirected (Wang, Yan, Hollister & Zhu, 2008).
- Multiple CPP: Multiple postmen have to traverse each street once such that every street has only one postman assigned to it, except for the starting street (Zhang, 2011).

If we only considered one car in one neighborhood, the problem would be an undirected CPP and therefore solvable in polynomial time (Gendreau & Laporte, 1994). This is interesting because in the future, one might want to plan also the routes within the scheduled neighborhood. Since the problem is then solvable in polynomial time, this could probably be implemented as an online application

A well-known problem that does consider the optimal route planning of nodes is the Travel Salesman Problem (TSP). The TSP deals with a salesperson that has to travel to every given city exactly once and return to the starting city. As for the CPP, it is the objective to minimize the total travel distance (Graham, Joshi & Pizlo, 2000). Unlike the CPP, the TSP has to visit once every node (in this case: neighborhoods) instead of every edge (e.g., streets). As for the CPP, there is also a multiple version of the TSP, the mTSP, where every node has to be visited once by one salesman (Bektas, 2005). The mTSP seems to be a better fit as we have to schedule multiple vehicles. Again, there are a lot of different variations defined for the TSP and mTSP. The most widely studied generalization is the vehicle routing problem (VRP) in which the car has limited capacity and has to deliver goods to the customer at the nodes (Braekers, Ramaekers & Van Nieuwenhuyse, 2016). Braekers et al. (2016, p.304) state that the VRP is one of the most widely studied topics in Operations Research and also present a list with different characteristics of the VRP that were most often reviewed in the last years:

- Capacitated vehicles
- Heterogeneous vehicles
- Time windows
- Backhauls
- Multiple depots
- Recourse allowed
- Multi-period time horizon
- Precedence and coupling constraints

- Subset covering constraints
- Split deliveries allowed
- Stochastic demands
- Unknown demands
- Time-dependent travel times
- Stochastic travel times
- Unknown travel times
- Dynamic requests

Unfortunately, these variations are still focused on minimizing the travel time and number of vehicles, whereas we strive to maximize the number of PCNs in every neighborhood. However, these variations are still relevant because we also have to deal, for instance, with time-dependent travel times and multiple depots (break locations).

Another interesting variation of the VRP is the milk collection problem. Claassen and Hendriks (2007) modeled the milk collection problem as a periodical VRP (PVRP). PVRP considers a planning period of T days instead of a single day. Therefore, clients are not necessarily visited every day. The demand of customers can be different for every customer at every day and the frequency of visits can be different for every customer. They do not minimize the travel time but minimize the weighted sum of deviations on demand level. Our problem is also a periodic problem (if we choose to make a planning for the whole KPI period) and the neighborhoods can be visited more than once during the KPI period. However, in our case, one neighborhood can even be visited more than once a day. Even though the milk collection problem comes closer to the problem of this research, this problem deals with capacity, which we can neglect. Moreover, minimizing the deviation of demand and maximizing the number of PCNs are not quite the same. Nevertheless, some parts of this article might be useful.

Another generalization of the TSP, namely the Traveling Salesman Problem with Profit (TSPwP), seems to be a better fit. Feillet, Dejax, and Gendreau (2005, p.189) discuss three different variations:

1. Both objectives, profit and travel costs, are combined in the objective function; the aim is to find a circuit tour that minimizes travel costs minus collected profit
2. The travel cost objective is stated as a constraint; the aim is to find a circuit tour that maximizes collected profit such that travel costs do not exceed a preset value.
3. The profit objective is stated as a constraint: the aim is to find a circuit that minimizes travel costs and whose collected profit is not smaller than a preset value.

As we want to maximize the amount of PCNs and we are restricted by the length of a shift, we choose the second option which they call the orienteering problem (OP). Furthermore, they state that the OP can be found in the literature under different names such as selective TSP (STSP) or the maximum collection problem (MCP). However, Feillet et al. (2005) explains that the OP and MCP differs from these as the OP is generally defined as a path rather than a circuit. However, by "adding a dummy arc from the destination to the origin of the paths makes the two problems equivalent" (Feillet et al., 2005, p.189). Even though the route has to finish at one particular depot, there are three locations (including the depot) where a break can be held. Therefore, it might be better to consider the problem including different break locations as separate paths instead of one circuit tour. For example, a circuit tour that starts and finishes at the depot (point A) and that includes two breaks, one at point B and one at point C, could be described as three paths (A->B, B->C, and C->A). Note that in our problem the break location should be chosen while constructing the route. As for TSP, there is a multiple-variant of the OP or MCP called the team orienteering problem (TOP) (Chao et al., 1996) or the multiple tour maximum collection problem (MTMCP) (Butt & Ryan, 1997). In the literature, we find the following applications:

- Scheduling maintenance technicians problem (Tang, Miller-Hooks & Tomastik, 2007)
- Tourist route planning problem (Gavalas et al., 2015; Vansteenwegen et al., 2009a; Vansteenwegen et al., 2009b; Vansteenwegen et al., 2009c)
- Bank robber problem (Awerbuch, Azar, Blum & Vempala, 1998)
- Home fuel delivery problem (Tang & Hooks, 2005)
- Athlete recruiting problem (Tang & Hooks, 2005)

Verbeeck et al. (2014b) present some generalizations of the TOP that include time windows and time-dependent (and/or stochastic) rewards, travel times, and services times. Considering the mathematical structure, the OP has a set of vertices, which are connected by edges. In order to travel these edges a certain travel time is needed. Unlike the TSP, time is limited and therefore it is the goal to visit a selected set of vertices in such a way that the total collected score is maximized (Verbeek, Aghezzaf & Vansteenwegen, 2014). Usually, the vertices in the OP are visited at most once (Vansteenwegen et al., 2011). In our problem, we do not have this constraint since the PEVs may go to the same neighborhoods

multiple times a day. The usual OP "is a combination of the knapsack problem (KP) and the traveling salesperson problem" (Verbeeck, Sörensen & Aghezzaf, 2014a). The knapsack problem maximizes an objective function by choosing items subject to a packing constraint (Hochbaum, 1995). That mean it is a combination of the selection of nodes and the determination of the sequence of these selected nodes. Since we do not have the constraint that a node is only chosen at most once a day, we have an additional scheduling problem because it is also required to determine the number of times a selected node is visited a day. Furthermore, in case that a node is visited multiple times a day, the reward that can be obtained during a visit depends on the time-difference to the earlier visit, hence the rewards are inter-related. The reason behind this is that one visitor can receive at most one PCN a day, therefore the expected reward at a node decreases after it has been visited. In addition, we consider time-dependent service times, travel times, and rewards and a periodical planning. Due to different parking regime times of the neighborhoods, we also have to take different time windows into account. As mentioned before in this section, our problem also requires visiting three possible break locations and therefore it is also a kind of multi-depot problem. Finally, we call this generalization the time-dependent and periodical TOP with multiple visits and multiple constraints, which we denote as the TD-PTOPMVMC. To the best of our knowledge, such a generalization is not discussed in the literature. Especially, the TOP with multiple visits are an interesting contribution to the literature, as it could have different applications, such as various inspection, collection, or salesmen problems. The problem could be further extended to a Mixed TOP (Vansteenwegen, Souffria & Van Oudheusden, 2011) that includes also the number of PCNs that are generated while traveling from one neighborhood to another.

Regarding the running time complexity, the TSP and VRP are known to be NP-hard. As stated before, in order to solve the TOP, not only the determination of the route is required but also the selection of the subset of nodes that will be visited. Because of this added element of complexity, it follows that the TOP is also NP-hard (Butt & Ryan, 1999). With another added element of complexity due to the multiple visits, the same holds logically for the TOPMV. The additional impact of breaks on the running time is discussed by Kok, Hans, Schutten, and Zijm (2010). They state that if the number of existing entries without breaks was O(np), the total number of entries with at most one break scheduled would be O(n2p). Analogously, considering 4 breaks would result in the running time complexity of O(n5p). This leads to the question whether this problem is solvable by an exact algorithm. It is known that exact algorithms can only solve NP-hard problems with relatively small instances and that heuristic are the more reliable approach in practical instances (Cordeau, Gendreau, Hertz, Laporte & Sormany, 2004). An exact algorithm to solve the TOP, using column generation, has been published by Butt and Ryan (1999). They were able to solve problems with up to 100 vertices and stated that the "solution procedure works well on realistic size problems, particularly when the number of nodes visited in any tour is relatively small" (Butt & Ryan, 1999, p.440). In their case, the average number of nodes per tour for 100 nodes was 3. In our case, we have 320 neighborhoods with probably more than 20 nodes per tour, a time-period of 90 days and multiple breaks at different break locations. Intuitively, it seems unlikely that an exact algorithm can solve this problem in reasonable time. Besides, an exact solution cannot be executed in practice as it is unlikely that they always arrive at the scheduled neighborhood in time. Consequently, we choose to apply heuristic algorithms, which are discussed in the Section 3.2.

3.2 Routing heuristics

As stated in 3.1 we are looking for a heuristic approach to solve our problem. This section first introduces the basic principles of routing heuristics in Section 3.2.1 and then presents literature dealing with solving TOP or MTMCP as our generalization is not yet introduced to the literature.

3.2.1 Introduction to routing heuristics

Cardeau et al. (2005) state that VRP heuristics usually combine some of the following four components:

1. Constructive heuristics
2. Improvement heuristics
3. Population mechanisms
4. Learning mechanisms

Except for the capacity constraint, the TOP is strongly related to the VRP and therefore we can apply the same heuristics and mechanisms. To this end, we briefly introduce the concepts of these four components before presenting heuristics that have been proven to be effective for TOP problems.

Constructive heuristics, as the name implies, construct an initial solution. Often, improvement heuristics are then applied to find better solutions than the initial solution. The combination does not necessarily mean that one improvement heuristic is applied on one constructed solution. It is also possible to use one or more improvement procedures on one or more solutions. Improvement heuristics can be divided into

(local optimum) heuristics and metaheuristics. The local optimum heuristics function in a descent mode until a local optimum (maximum or minimum) is reached. Metaheuristics, on the other hand, work in such a way that they try to avoid being trapped in a local optimum. Therefore, these heuristics sometimes accept worse or even infeasible solutions. Three well-known metaheuristics are tabu search, simulated annealing (SA), and variable neighborhood search (VNS), which will also be discussed in Section 3.2.2. SA is inspired from annealing in metallurgy and therefore works with cooling parameters. While exploring more and more solutions, these cooling parameters decrease the probability of accepting worse solutions. The tabu search always chooses the best neighbor that is not on the tabu list and adds this neighbor to the tabu list. The tabu list usually has a limited length and deletes the oldest ones from the list. Since the tabu list prohibits going back to old solutions, the tabu search avoids getting stuck in local optima, and is therefore a metaheuristic. In the VNS, introduced by Mlandenovic and Hansen (1997), the neighborhood structure is able to change while exploring solutions. These heuristics are often combined with each other or other heuristics.

Another kind of heuristics is population-based, which belong to the class of evolutionary algorithms (EA). The widest known population-based algorithm is the genetic algorithm (GA). The classical GA is a metaheuristic that operates on a population of solutions called chromosomes or individuals. In each iteration (generation), the following operations are applied k times (Cardeau et al., 2005; Kumar et al., 2005):

1. Select two parent chromosomes
2. Use crossover operators to generate two offspring from these parents
3. Apply a random mutation to each offspring with a small probability
4. Remove the 2k worst elements of the population and replace them with the 2k offspring

The idea of combining solutions to generate new ones is also used in the adaptive memory procedure (AMP), which was introduced by Rochat and Taillard (1995). The only difference is that they can generate new solutions from more than two parents (Golden et al., 1997).

Learning mechanisms are heuristics that are inspired by different learning paradigms in the world. For instance, neural network models, which are inspired by the way how the brain works. Another example are ant colonization optimization (ACO), which belong to the ant colonization optimization algorithms. The ACO algorithms are inspired by the way ants collect food. Ants use trails of pheromone to mark their travel paths. As time passes, the best paths will have the strongest trail since more and more ants are using these paths.

3.2.2 Team orienteering problem heuristics

A lot of different heuristics have been introduced to solve different variations of OP, MTMCP, TOP, MTMCP, and the selective traveling salesman problem (STSP). In this section, we describe the most important ones. Gendreau, Laporte & Semet (1998) describe some difficulties when applying heuristic approaches to the OP. They state that "profits and distances are independent and a good solution with respect to one criterion is often unsatisfactory with respect to the other" (Gendreau et al., 1998, p.540). This makes it hard to accurately select nodes. Furthermore, Vansteenwegen (2009a) indicates that the most difficult OP instances to solve are those where the selected number of nodes is a little more than half of the total number of nodes. According to Vansteenwegen et al. (2011), the best-performing TOP algorithms are discussed in Tang and Miller-Hooks (2005), Archetti et al. (2007), Ke et al. (2008), Vansteenwegen et al. (2009c), and Souffriau et al. (2010). The computational results of these algorithms are shown in Table 2.

Reference	Computer specifications	Technique	Algorithm	# best	Avg gap (%)	Avg CPU (seconds)
Tang and Miller-Hooks (2005)	DEC Alpha XP1000, 1 GB RAM, 1.5 GB swap	Tabu search	TMH	34	1.32	336.6
Archetti et al. (2007)	Intel Pentium 4, 1 GB RAM, 2.8 GHz	Tabu search	TSF	94	0.20	531.5
			TSU	69	0.49	318.0
		Variable neighbourhood search	SVN	**128**	**0.05**	906.1
			FVN	97	0.18	**63.6**
Ke et al. (2008)	PC, 3.0 GHz	Ant colony optimisation	ASe	**130**	**0.08**	252.3
			ARC	81	0.40	204.8
			ADC	80	0.35	213.8
			ASi	84	0.32	215.0
Vansteenwegen et al. (2009c)	Intel Pentium 4, 1 GB RAM, 2.8 GHz	Variable neighbourhood search	SVNS	44	0.97	**3.8**
Souffriau et al. (in press)	Intel Xeon, 4 GB RAM, 2.5 GHz	Greedy randomised adaptive search procedure with path relinking	FPR	78	0.39	**5.0**
			SPR	**131**	**0.04**	212.4

Table 2 – Summary of the best-performing TOP algorithms (Vansteenwegen et al., 2011, p.5)

Concerning these articles, Vansteenwegen (2009b) argues that the local search moves used in these TOP solutions are not effective when applied to TOPTW because they include local search moves that become useless when time windows are considered. In our case the time windows are quite the same except for the evenings and Sundays (see Table 1 in Section 2.1.5), therefore this is not a problem. On the other hand, all the travel time, service time, and expected number of PCNs are time-dependent and that means that by applying local search techniques, such as swapping, we need to calculate the whole route again and have to take into account that no time restrictions are violated. Moreover, as explained in Section 2.1.3, visiting a neighborhood has influences on other visits of the neighborhood if they are on the same day. Therefore, local search moves will be difficult to implement but we do consider these algorithms as well (except for the article of Archetti et al. (2007) because is not accessible for us) to get a broader impression of possible solution heuristics.

Vansteenwegen et al. (2011) state that there are four articles published in the literature that deal with the TOP with time windows. We do not include the exact algorithm from Boussier et al. (2007) in our research because we focus on heuristic approaches, as stated in Section 3.2.1. However, we add another interesting article about a simulated annealing heuristic from Lin and Yu (2015). Therefore, we add the following four articles to the four articles from Table 2:

- Montemanni & Gambardella (2009): An ACO algorithm for the hierarchical TOP with time windows
- Vansteenwegen et al. (2009b): Iterated local search heuristic for the TOP with time windows
- Tricoire et al. (2010): VNS algorithm for the multi-period OP with multiple time windows
- Lin & Yu (2015): Simulated annealing heuristic for the multi-constraint TOP with multiple time windows

In the following, we discuss the algorithms of these eight articles.

3.2.2.1 Tabu search embedded in an adaptive memory procedure

Tang and Miller-Hooks (2005) apply a tabu search heuristic embedded in an AMP, which we introduced in Section 3.2.1., to the TOP. However, instead of reviewing this article, we review the article from Tang, Miller-Hooks & Tomastik (2007) because they use the same approach and extend the TOP by considering time-dependency and a periodical planning, which fits better to our problem.

Tang et al. (2007) tackle the problem of scheduling technicians for planned maintenance. They consider a planning period of 3 weeks and time-dependent rewards to better describe the reality. Greater rewards are assigned to locations that have not been maintained for a longer time. The travel times between locations and service times at every location are different but not time-dependent. Their approach includes three AMP steps:

1. Partial solution generation and storage:

   Partial solutions are defined as one single tour of the m tours. First, a set of partial solutions is generated and stored. The first non-depot vertex is randomly chosen. Random vertices are added in between a pair of vertices, which depends on a ratio with regards to the added tour duration and the added reward.

2. Solutions construction:

   Afterwards, solutions are constructed by combining partial solutions. The selection preference is biased to those single tours with preferred objective values. All constructed solutions are improved by tabu search afterwards. Both random and greedy procedures are applied in the neighborhood solution exploration.

3. Partial solution update:

   The solutions maintained in the adaptive memory are updated with these improvements. Low-reward tours in the adaptive memory are replaced by the improved tours.

3.2.2.2 Ant colonization optimization

Montemanni and Gambardella (2009) apply an ACO heuristic to the team orienteering problem with time windows. They define their problem as a hierarchical TOP, which requires the same input as the TOP does but it requires a set of non-overlapping elementary paths, which have an ordered sequence of nodes starting from node 1 and ending at node n.

The construction phase is performed by sending out all ants sequentially. Iteratively, every ant goes probabilistically from node i to node j based on the *pheromone trail* and the *desirability*. The pheromone

trails contain the trails of previous ants that travelled there and indicate how good this path has been in the past. The desirability is a formula regarding the associated profit, the distance, and the time window of node j. The possible nodes for j are selected out of a set of feasible nodes, which still need to be visited and are within the time window. Note that only the best ant, which collected the most rewards, is allowed to leave a trail that is updated to all arcs. While the ant builds the solution, the pheromone trail is updated as well. Each ant removes pheromone trails of the visited arcs to make sure that there is a variety of generated solutions. Afterwards, the constructed solutions are being optimized by a local search algorithm. They apply a CROSS exchange procedure that exchanges two sub-chains of customers of the giant tour.
Ke et al. (2007) also apply ACO but to the regular TOP without any time-dependencies or time windows. They state that sending the ants sequentially results in the best results. Furthermore, they performed a benchmark of their algorithm with the one of the Archetti and Tang et al. (2005) with the result that the quality of their solution could compete with the others but with a much faster computational time. The results can be seen in Appendix B. Another interesting aspect of their approach is that in their heuristic function they include the angle at neighborhood i between the way to the depot n and the next neighborhood j. By doing so, the algorithm can send the driver in the desired direction. First leaving the depot and then forcing the driver more towards the depot.

Verbeeck et al. (2014a) apply ACO to a TOP with time-dependent travel times. They speed up the time-dependent insertion procedure by using a local evaluation metric. Verbeeck et al. (2014b) tackle the TOP with time-windows and time-dependent and stochastic rewards and time-dependent travel times by using a greedy randomized adaptive search procedure and a stochastic version of the ACO.

3.2.2.3 Simulated annealing

Lin and Yu (2015) apply an SA heuristic, which we briefly introduced in Section 3.2.1., for the multi-constraint TOP with multiple time windows. Their heuristic starts by creating a random initial solution. Afterwards, the initial solution is optimized by means of SA including a swap, insertion, or inversion procedure in every iteration. Additionally, they add a restart strategy as an extra diversification to avoid local optima. They state that sometimes accepting worse solutions is not enough to escape the local optima. The current temperature, which determines the probability of accepting worse solutions, decreases after every iteration. The algorithm restarts if the current best solution has not improved for a pre-determined number of consecutive temperature decreases. Once the algorithm restarts, the current temperature is reset to the initial temperature and a new initial solution is generated randomly to initiate a new SA run. They show that SA with a restart strategy is a promising heuristic method to solve multi-constraint TOP with multiple time windows and that the restart strategy enhances the performance of the SA.

3.2.2.4 Variable neighborhood search

Tricoire, Romauch, Doerner and Hartl (2010) deal with a multi-periodic TOP with multiple time windows and use a VNS. Before applying the VNS, they first construct solutions. To this end, they use the best insertion heuristic. The insertion heuristic is based on two criteria. One is the lowest increase in distance and the other one the lowest increase in time. The feasibility of the insertions is checked by means of an exact feasibility algorithm, which operates in polynomial time. Afterwards, VNS is applied to improve the initial solution. A stopping condition can be a limit on computational time, the number of iterations, or the number of iterations without improvement. They apply the number of iterations as a stopping criterion. For every iteration of the VNS algorithm, an improvement method that depends on the number of iterations, that have been performed already (iteration 1-8: cross-exchange, iteration 9-12: optional exchange, iteration 13-17: optional exchange), uses random nodes to create a new solution. If the new solution is better, it replaces the initial solution. In a benchmark, they show that their VNS algorithm is a viable option for all kind of orienteering problems, with or without time windows.

3.2.2.5 Path relinking heuristic with a greedy randomized adaptive search procedure

Souffria et al. (2010) use a path relinking metaheuristic in combination with a greedy randomized adaptive search procedure because path relinking heuristics have been proved to work well on knapsack problems. Their approach works as follows:

While the number of iterations without improvement is not exceeded:

1. Construct: The construction heuristic is based on a greedy randomized adaptive search procedure. This procedure depends on a "greediness" parameter that lies between 0 and 1. This parameter indicates the level between randomness (0) and greediness (1). The parameter is determined randomly before the construction.
2. Local search: The local search algorithm uses 2-Opt, swap, replace and insert procedures until a local optimum is reached.
3. Link to elites: This procedure combines the solution, that was constructed and improved in the prior two phases, with one of the solutions out of the elite pool. The two solutions are first combined, then adapted, and finally improved to create a new feasible solution. This procedure is done for all possible combinations, therefore for all members of the pool of elites.
4. Update elite pool: The best solution found in the prior step is considered for the insertion into the pool of elite solutions. If the pool is full, it replaces the worst elite solution if it leads to an improvement. Every solution is assigned to an age and it increases with every time the "Link to elites" is performed. At a certain age, the solution is deleted from the pool.

3.2.2.6 Iterated local search heuristic

Vansteenwegen et al. (2009b) apply an iterated local search heuristic algorithm to the TOPTW with the purpose of developing an electric tour guide. The electric tour guide required a short computation time and therefore they chose an algorithm that is very simple, fast, and effective. They achieved this goal with an average performance gap of 1.8% to the best-known solutions and the average computation time is more than a 100 times faster than the best-known solutions. Gavalas et al. (2014, p.19) state that it is "the fastest known algorithm proposed for the TOPTW". Their approach includes an insert step in combination with a shaking step to escape from local optima that perform performs very well on a large and diverse set of the instance.
The insertion step adds one by one new visits to a tour. Before a new visit can be added, the time windows need to be checked for feasibility. A feasible node with the cheapest insertion time will be inserted. For each node, a ratio is calculated that incorporates the profit and the delay of adding this node. Afterwards, a shake step is used to escape from local optima. In this shake step, random node(s) are removed in every tour to make space for nodes that might improve the solution.

3.3 Input models

This section addresses the development of the three inputs: scanning time, travel time, and the number of PCNs. As scanning time and travel time both depend on speed, Section 3.3.1 is about speed models. The number of PCNs is more complicated because some factors are yet to be determined and finally we have to analyze the impact of all factors. Therefore, Section 3.3.2 discusses possible factors from the literature, and Section 3.3.3 presents data analysis tools and prediction models.

3.3.1 Speed models

As described in Chapter 2, a speed model, which also accounts for traffic congestion, is required for the estimation of the service time and travel time.

Kok, Hans and Schutten (2012) used a speed model to calculate travel times of a VRP. In order to avoid traffic congestion, they use time-dependent travel times as an input for their VRP. They state that peak hours depend on location and time of the day, therefore "traffic congestion avoidance is all about not being at the wrong place at the wrong time" (Kok et al., 2012, p.1). Therefore, their speed model accounts for traffic congestion depending on time and location. They introduce four different strategies regarding the calculations of the shortest paths and finally the travel times of the VRP as can be seen in the following Table 3.

Strategy	Shortest paths	Travel times input for VRP	Accounting for congestion	Avoiding congestion
1	Time-indep.	Time-indep.	No	No
2	Time-indep.	Time-indep.	Yes	No
3	Time-indep.	Time-dep.	Yes	Yes
4	Time-dep.	Time-dep.	Yes	Yes

Table 3 – Travel time strategies by Kok et al. (2012)

Their results have proven that accounting for traffic congestion leads to more reliable travel times in terms of punctuality at the customer locations. They also indicate that avoiding traffic congestion leads to improved routes in terms of traveled time.

In our case, we do not have to calculate the shortest path because we can make use of an existing distance matrix. Nevertheless, we can use this model by making the travel times of the TOP time-dependent to avoid the traffic congestion in Amsterdam. However, we do not consider such a big network as in the research of Kok et al. (2012) and therefore it is questionable if it is necessary for us to let the traffic congestion depend on different locations within Amsterdam. An alternative model was introduced by Verbeeck et al. (2014a) who also use a time-dependent speed model but their model accounts for different arc categories:

- Always busy
- Morning peak
- Two peaks a day
- Evening peak
- Seldom traveled

Assigning every arc to such a category might be useful for our problem as well.

3.3.2 Payment and parking behavior

This section discusses what the literature provides about factors that have a possible impact on the number of PCNs. In Section 2.1.8, we concluded that the number of PCNs can be derived from the number of visitors and the payment rate and that the number of visitors can also be derived from the occupancy ratio of the parking spots and the visitor ratio of all scans. Since it is also possible to visit a neighborhood more than once during a day, it is interesting to know how long visitors, who do not pay, stay at the same parking spot. To this end, we investigate what is written in literate about parking and payment behavior.

As already mentioned in Section 1.1.3, Peliot (2004) states that the choice whether to pay for parking depends on the amount of the fine, the risk of getting caught, and the regular parking costs. We already consider the control chance and therefore the risk of getting caught. As discussed in Section 1.2, the amount of the fine consists of a fixed amount plus one-hour parking fee. Both the fee and the fine cannot be changed. However, it might be interesting to investigate if differences of the parking fee in different neighborhoods influence the payment behavior.

Summerfield et al. (2015), who consider the parking enforcement problem as a CPP, use two probabilities in their model. The first is a Bernoulli distribution that indicates whether a car is parked in a parking space. For the Bernoulli distribution, they experiment with occupancy ratios of the parking space from 90% until 100%. Lower occupancy ratios decrease the revenue of a route. However, with these lower occupancy ratios the parking enforcement agents could actually finish their routes earlier as they did not have to investigate 100% of the parking spots. This does not apply to our case due to the fact that the PEV passes the parking spots regardless of whether the parking spot is occupied or not. The second distribution Summerfield et al. (2015) use is a triangle distribution that represents the distribution of the time between the parking of a car and the return of its owner. They assume that everybody is a visitor that pays but not everybody returns in time to their car. The authors use parking permit times between 30 and 120 minutes and a memory of the parking enforcement agent. There are two memory options:

1. Wait on-site for parked cars that are about to expire
2. Go back to the parked cars that were about to expire

This memory option is not relevant for us, as the PEV driver does not know when the parked cars are about to expire because they just scan the license plates and have no immediate access to the PRDB, which contains the payment information.

Aikoh, Abe, Kohsaka, Iwata, and Shoji (2012) investigate the factors influencing visitors going to suburban open space areas near a northern Japanese city. Even though visitors going to suburban spaces is not quite the same as going to Amsterdam or another big city, their results are quite interesting for us. They used a multiple regression analysis to analyze social factors and meteorological factors, such as:

- Day of the week and holidays
- School vacation
- Rain, Snow
- Temperature
- Weather conditions at the departure site
- Weather forecasts

They state that the day of the week and holidays have the biggest influence on the number of visitors. Moreover, the results seem to depend on the season of the year. For instance, higher temperatures increase the number of visitors in the green season but they have an opposite effect in the snow season as they expect snow in the snow season. Their results also show that not only the weather but also the weather forecast affects the number of visitors in the summer. Furthermore, school vacations, rain at departure site and depth of snow influenced the number of visitors in the winter.

Probably some of these factors also play a role in the number of visitors of Amsterdam. However, not all people who visit Amsterdam are actual visitors in a touristic sense but also business people. Probably the number of business people does not depend that much on seasons and the weather as it is the case for tourists. It could be that their number is affected by economic growth or special business events.

3.3.3 Data analysis tools and prediction models

This section deals with how to make a prediction model from available data. First of all, it is important to understand the data set. In the field of prediction models, we have a matrix representing a data set. This matrix consists of several columns representing either a feature or an output. The rows are filled with observations that provide information about the features, the output, and more importantly their relation to each other. Features can be either quantitative or qualitative (Jain, Murty & Flynn, 1999):

- Quantitative features
- Continuous values (e.g., weight)
- Discrete values (e.g., the number of computers)
- Interval values (e.g., the duration of an event)
- Qualitative features
- Nominal or unordered (e.g., color)
- Ordinal (e.g., qualitative evaluations of temperature (“cool” or “hot”))

An output can also be quantitative or qualitative. In our case, we have quantitative outputs because we want to know the number of visitors and the probability that they pay for a given space and time period. To this end, we discuss different prediction models for quantitative outputs later in this section but first we address the field of the *clustering analysis*.

3.3.3.1 Clustering analysis

A clustering analysis helps to better understand a data set by investigating its observations and establishing certain patterns. A pattern can be a represented as a multidimensional vector or a point in a multidimensional space that measures an output (Jain, Murty & Flynn, 1999). For instance, looking at the number of traffic jams in a big city and the hour of the day, one could probably find patterns that show an increase in the morning and afternoon. Xu and Wunsch (2005) state that it is essential to classify the observations of the data into a set of categories or clusters. Clustering analysis is an unsupervised (unlabeled) classification, i.e. the names of the clusters are unknown. The objective of clustering analysis is to group a given collection of unlabeled patterns into meaningful clusters.

Xu and Wunsch (2005) divide clustering algorithms into four tasks.

1. Feature selection:

   Sometimes the data needs some kind of transformation. Even simple transformations “can yield significantly improved clustering results” (Jain, Murty & Flynn, 1999, p.7). Furthermore, it is not necessary to use all available features. The feature selection is the process of selecting the most effective subset of original features in order to reduce the number of dimensions. By doing so, the results of the clustering and the pattern representation can be improved. A well-known method of feature selection is the primary component analysis (PCA), also called the Karhunen-Loeve transformation. The objective of the PCA is to select only the most important features that say the most about the variance of the data set. By not considering the other features in the data analysis, the dimensions of the data set are reduced.

2. Clustering algorithm:

   The next step is applying a clustering algorithm in order to cluster the data. There are several different algorithms due to the fact that there are many different types of data. These algorithms format clusters by using proximity measures and constructing a criterion function. We consider especially the algorithms that deal with clustering of time series and geographical clustering. Anderson (2009) use kernel density estimation and K-means clustering for clustering road

accident hotspots. Liao (2005) discusses relocation clustering, agglomerative hierarchical clustering, K-means, fuzzy c-means, and self-organizing maps for clustering time series.

3. Validation of clusters:

   After clustering, we can validate whether these clusters are appropriate. The assessments should be objective without having preferences for any algorithm or outcome.

4. Results interpretation:

   The ultimate goal of clustering is to provide users with meaningful insights from the original data.

Xu and Wunsch (2005) present these steps in a figure (Figure 6). All steps are inter-related due to feedback pathways.

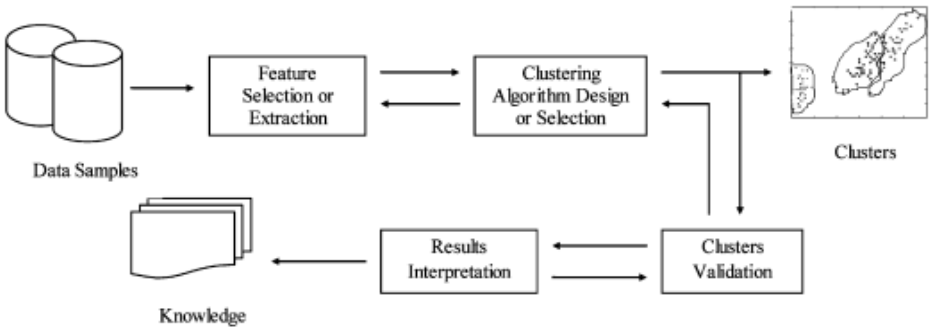


Figure 6 – Clustering analysis procedure (Xu & Wunsch, 2005)

We can use these steps as a framework for our data analysis.

3.3.3.2 Prediction models

A common way to develop prediction models for quantitative outputs are regression models (Larsen & Marx, 2012). Regressions models analyze the effect of the independent variables, also called predictors or features, on the outcome variable. A regression analysis can include one (simple regression) or more predictors (multiple regression). The relationship between predictors and the outcome variable can be linear, curvilinear, or nonlinear. For the prediction of probabilities, logistic regressions can be applied. For numerical outputs usually a (multiple) linear regression or nonlinear regression is used.

There are also machine learning techniques that are using regression models such as artificial neural networks. These networks are inspired by the architecture of biological neural networks (Mair et al., 2000). Every network consists of neurons which are interconnected by strings. A neuron receives an input which is associated with a weight. If the sum of these weighted inputs exceed a certain threshold, the neuron fires and creates a positive or negative output for other neurons in the network. This process stops when one or more outputs are generated. An example of this process is shown in Figure 7. This example shows n inputs. If the threshold is exceeded, the output becomes 1, otherwise, it is 0. If an output is incorrect, a process called backpropagation starts. In this process, the output is corrected by adjusting the weights. In this way, the networks learn from a data set.

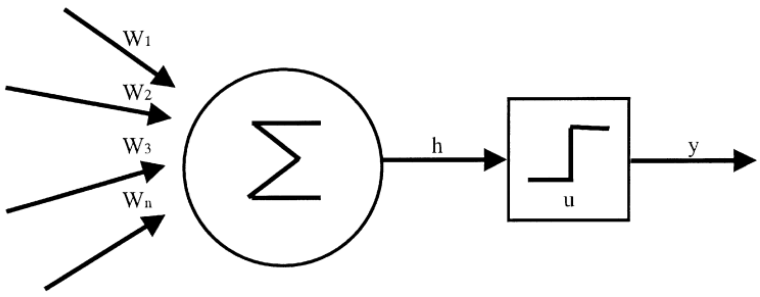


Figure 7 – An example of a neural network (Mair et al., 2000)

Sarkar, Ghalia, Wu, and Bose (2009) applied a neural network to predict fiber diameters by using different inputs. In this case they use a *multilayer network* as proposed by White (1992). A multilayer network has hidden layers between the original input and the final output variable(s). These hidden layers are functions that use the previous inputs to create an intermediate output node, which can be used as an input for another hidden layer or for the final output variable. Within one hidden layer there can be many layer nodes. It is also hard to tell how many layers and nodes a neural network should have because in the end the neural network determines what happens in the layer nodes within the layer. Sarkar et al. (2009)

determine the number of nodes and layers by conducting experiments. The results were 12 nodes in the first hidden layer and 7 nodes in the second one. The final neural network can be seen in Figure 8.

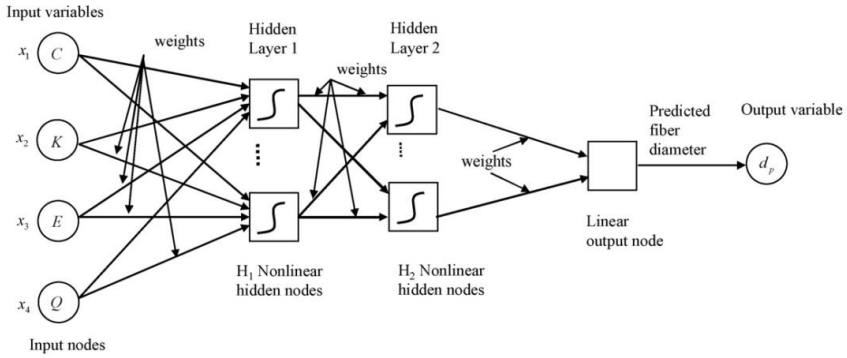


Figure 8 – An applied neural network with hidden layers (Sarkar et al., 2009)

Le Cun et al. (2012) have some recommendation for applying a neural network. For instance, shuffling the data set helps the network to learn faster from unexpected samples (LeCun et al. 2012) and normalization of the presented input data can also increase the learning process. It is well-known that, the data set should be split into a training set and a test set, in order to avoid overfitting, i.e., the neural network learns too much from the data set in a sense that it also learns from outliers and noise instead of creating a general applicable prediction model. Furthermore, White (1992) and LeCun et al. (2012) state the use of too many parameters and too many layers can also lead to overfitting.

Another prediction model is introduced by Van Urk, Mes and Hans (2013). They use a prediction model for an application, which is quite similar to our planning tool, namely the development of a decision support application for the Dutch Aviation Police and Air Support unit for routing their helicopters in anticipation of unknown future incidents. Their research is similar as it involves a forecasting method and a routing method that maximizes the likelihood of being close to a future crime. Even though parking violation and crime are not quite the same, the principle can be applied here as well. The second part of their research also deals with a kind of TOP but they combine it with a Location Covering Problem (LCP), as the helicopters have to cover certain areas to intervene quickly in case of emergencies. For us, the LCP is not relevant, as we do not deal with that kind of emergencies. More interesting, however, is the first part regarding the forecasting. In order to predict future crime intensity, they use a forecast based on the moment of the day, days of the week, and months of the year, which have an impact on the crime rates. They convert every past incident in order to use this information for future predictions. To this effect, they use two conversion factors, namely the FactorMonth (month, hour) and the FactorWeekday (weekday, hour). Additionally, they apply generalization techniques because they assume that an incident at one specific location and time is similar to the neighboring areas and some time periods around the incident. Analogously, they apply this for the time dimension. This model with some modifications can be used for this research as well. The incidents can be replaced by the number of PCNs and it would be required to check whether the generalization also applies in our case. A similar approach is discussed in the master thesis of van Hal (2015). In this research, the forecasting method is based on the fact that the relative distribution of incidents regarding the Netherlands does not depend on time. For that reason, the forecasting method is split into a time problem, which is solved by linear regression with different time-related factors, and a space problem, which is solved by means of the kernel density method.

3.4 Conclusion

This chapter discussed a number of articles that tackle a similar routing problem. We defined our problem as a team orienteering problem with a periodic planning horizon, time windows, and time-dependent (and/or stochastic) travel times, service times, and rewards. This problem is in the literature also known as the scheduling maintenance technicians' problem, tourist route planning problem, bank robber problem, home fuel delivery problem, and athlete recruiting problem.

Moreover, we concluded that we use a heuristic approach due to the problem's complexity (NP-hard) and the incapability of finding an exact solution. We discussed different algorithms that can be used to solve such a problem:

- Tabu search embedded in an adaptive memory procedure
- Ant colonization optimization
- Simulated annealing
- Variable neighborhood search

- Path relinking heuristic with a greedy randomized adaptive search procedure
- Iterated local search heuristic

Furthermore, we addressed the topic of input models. We learned from other articles that including time-dependency in speed models improves the routing problem in such a way that the reliability and the travel time are increased. Looking for factors that might influence the number of PCNs, we found that the number of visitors seems to depend on time and meteorological circumstances. The payment rate is described as a portfolio model in which the visitors assess the costs of the fee versus the costs of getting a fine, and the probability of getting a fine.

In order to actually analyze the data with regards to these factors, we presented a clustering framework. Finally, we discussed different options to predict the expected number of PCNs, such as regression models, neural networks, and two other forecasting models that were applied to similar problems.

4 Computation of inputs

The objective of this chapter is to compute the necessary inputs that are needed for our routing algorithm. For all inputs that are time-dependent, we apply interpolation to make predictions across different time periods. We have access to a database that contains information (such as time, GPS location, and whether it was a PCN) of all parked cars that were scanned by EPS between the beginning of 2016 and today. As part of a prior analysis, ARS has computed travel times, service times, and expected number of PCNs for their own routing algorithm. We use these inputs to compare them to our approach. Also, ARS encountered some challenges due to gaps in the data.

Section 4.1 discusses the quality of the source data and examines ways to cope with it which are further elaborated in the subsequent sections. Section 4.2 tackles the question of the size of the error margin with regards to the KPI targets, which has been discussed in Section 2.1.8. In Section 4.3, we present our prediction of the expected number of PCNs. Section 4.4 explains how the travel times are computed. Lastly, Section 4.5 discusses our approach to estimate the service times.

4.1 Challenges of the historical data

In this section, we discuss the challenges that ARS already had encountered of using historical data to predict future results. One of these challenges is that PEV drivers did not always scan all parking spots of the assigned neighborhoods in the past. We denote the scanning of a neighborhood also a visit. Since it is the intention to make predictions about the future number of PCNs of visits, in which all parking spots are scanned, it is difficult to make predictions based on visits, in which only a fraction of the neighborhood is scanned. Let us say that on a random day, we have the following information of a visit of one neighborhood:

- 200 parking spots
- 100 scans
- 4 PCNs
- Service time of 10 minutes

While these figures precisely indicate that 4 PCN's were issued, we cannot conclude that the total number of potential PCNs in that neighborhood was since we do not know if all parking spaces of the neighborhood have been scanned. This applies equally for the service time and the apparent occupancy ratio. If the neighborhood was actually fully scanned, then the occupancy ratio would be 50%. This could be a realistic value for some neighborhoods, but for busy areas, such as "Centrum", this is very unlikely. In conclusion, we cannot derive from the data to what extend the neighborhood has been scanned. This problem could be handled in a better way if the occupancy ratio of the neighborhood was known. Let us assume that the occupancy ratio was 80% in that neighborhood at that time. With this information, we can estimate that 160 parking spaces should have been scanned, meaning that only 62.5% (100 scans/160 expected scans) of the neighborhood were scanned. This information enables us to extrapolate the number of PCNs and the service time by multiplying them with 1.60 (160/100). This is the approach that ARS used with the assumption that the occupancy ratio of all neighborhoods is 80% at any time. The problem with this approach is that this assumption is not based on data but on the expert opinion of EPS. Moreover, an extrapolation is not a perfect solution in this case, as the fractions of the neighborhood differ from each other. For instance, if the PEV driver only scans one shopping street of a neighborhood, which is attractive in terms of the number of PCNS, it will result in an overestimation of the number of PCNs of that neighborhood due to the extrapolation.

4.2 Margin of error

In this section, we calculate a margin of error for the two KPI targets: the payment rate target and the PCN target. As already stated in Section 2.1.8, EPS has the problem that both KPI targets depend on uncertain variables. Therefore, we estimate in this section a margin of error for the payment rate in Section 4.2.1 and the PCN target in Section 4.2.2.

4.2.1 The payment rate

The problem of the payment rate is that EPS performance regarding this target is based on the payment rate measurements of the municipality. The measurements of the municipality are similar to the scanning process of EPS. And while it is known that the municipality's sample size is much smaller than one of EPS, which reduces the accuracy, the time and place of these measurements are unknown. Yet, these

measurements are used to conclude on the payment rate for the whole KPI period of that KPI area. Since EPS scans a lot of parked cars during the KPI period, with a generally good geographical distribution, they have a good estimation of the actual payment rate of every KPI area. However, it might happen that the relatively small sample of the municipality is taken in a neighborhood that is considered to be an outlier at that time. An example from the past is a summer day when a lot of people went swimming in a neighborhood, where the parking meter was broken. Therefore, there were more visitors than normal, and also they did not pay in large part which caused the payment rate measured by the municipality to be very much lower than expected. Or conversely, EPS overestimated the payment target that the municipality would derive and met neither of the KPI targets in that KPI period. Even though these extreme cases can hardly be consistently prevented, we want to account for this uncertainty. We achieve this by increasing the payment rate target by a margin of error. The method with which we determine the margin of error is described below.

As said, EPS scans a large number of parked cars in all KPI areas with a good distribution over time and location. Therefore, we assume that the measurement of EPS represents the true value of the payment rate for all KPI areas. This payment rate is denoted as p. The payment rate of the municipality's sample is denoted as X/n, where n is the sample size and X is the number of paying visitors within this sample. In the following, we want to determine the margin of error, which is the half width of the confidence interval d. For this purpose, we use a reliability (denoted as α) of 95%. The size of d should be big enough that we can conclude that the payment rate of the municipality's sample is higher than the true value of payment rate minus the margin of error d, therefore: $p - d \leq \frac{X}{n}$.

Since the distribution of the payment rate is the probability that visitors of a certain sample pay, we can apply a binomial distribution. Larsen and Marx (2012) describe the binomial distribution with the following formula:

$$P\left(-d \leq \frac{X}{n} - p \leq d\right) = 1 - \alpha\,.$$

In order to determine d, we can use their formula for estimating the sample size n and solve it for d:

$$n = \frac{Z_{\alpha/2}^2}{4d^2} \rightarrow d = \frac{Z_{\alpha/2}}{2\sqrt{n}},$$

where $Z_{\frac{\alpha}{2}}$ is the value of the standard normal distribution function for which $P(Z \geq z_{\frac{\alpha}{2}}) = \frac{\alpha}{2}$. Even though the sample size is unknown beforehand, we do know the sample sizes of three quartiles in 2016. We use the minimal value instead of the average value of the three samples as our sample size n because we rather have a margin of error that is too big than too small. With this formula, we are able to say that if d is, for instance, 2% and the payment rate measured by EPS is 90%, it will mean that with a reliability of 95% the payment rate of the sample measured by the municipality is at least 88%. The results of the margin of error for all KPI areas are presented in Table 4.

KPI area	Minimal sample (n)	Margin of error of the payment rate target (d)
Centrum	3,581	1.64%
Nieuw-West	1,763	2.33%
Noord	1,054	3.02%
Oost-1	1,288	2.73%
Oost-2	2,148	2.11%
West-1	1,982	2.20%
West-2	2,227	2.08%
Zuid-1	3,337	1.70%
Zuid-2	2,116	2.13%
Zuidoost	418	4.79%

Table 4 – Margin of error calculation

In Section 4.2.2, we determine also the margin of error of the PCN target.

4.2.2 The PCN target

Determining the upper bound for the PCN target is more complicated, as the PCN target depends on the control chance target (c), the payment rate (p), and also the number of paid-for visitor hours (h). We consider two different PCN targets. The first one is the basic PCN target that is the PCN target that EPS uses today, which is based on the EPS' estimation of payment rate p and the paid-for visitor hours h:

$$\text{Basic PCN target} = c * h * \left(\frac{1}{\text{p}} - 1\right).$$

The second PCN target is the worst case and the upper bound of the PCN target that copes with the uncertainty of the municipality's payment rate and the paid-for visitor hours. To this end, we have to consider the case that the municipality's payment rate has the lowest value and the paid-for visitor hours has the highest value that we would expect. The number of paid-for visitor hours that EPS uses is the same that the municipality uses. However, the exact number of paid-for visitor hours are unknown until the end of the KPI period. But since weekly updates of the paid-for visitor hours are available, EPS uses these to predict the total paid-for visitor hours of the entire period. Unfortunately, we do not have access to this data. Therefore, we assume that the number of paid-for visitor hours is at most 10% higher than expected. We choose such a high number, which leads to a 10% increase of the basic PCN target, in order to rather overestimate than underestimate this variability. It is important that this percentage will be replaced on the long-term by a variability that is derived from data. As determined in Section 4.2.1, the minimal value of the municipality's payment rate is the payment rate measured by EPS p minus the margin of error d. By inserting the worst case values for h and p in the basic PCN target, we compute the upper bound of the PCN target as follows:

$$\text{PCN target upper bound} = c * 1.1h * \left(\frac{1}{\text{p}-\text{d}} - 1\right).$$

Now, we would like to express the upper bound of the PCN target in terms of the basic PCN target. Therefore, we introduce two variables that increase the basic PCN target. The first one is u_p and increases the basic PCN target due to the uncertainty of the payment rate. The second one is u_p and increases the basic PCN target due to the uncertainty of the paid-for visitor hours (u_h). Finally, our objective is to compute the PCN target upper bound as follows:

$$\text{PCN target upper bound} = \text{Basic PCN target} * u_h * u_p.$$

Since the PCN target increases linearly with h, u_h is equal to 1.1. In order to compute u_p, we form the following equation by inserting the previous two formulas in the last one:

$$h * c * \left(\frac{1}{p} - 1\right) * 1.1 * u_p = c * 1.1h * \left(\frac{1}{p+d} - 1\right).$$

Solving this equation leads to:

$$u_p = \frac{(1-p-d)*p}{(p+d)*(1-p)}.$$

As we have now determined the two values of the variables u_h and u_p, we can multiply these with the basic PCN target in order to compute the upper bound of the PCN target.

4.3 PCN prediction

This section describes our approach on how to predict the expected number of PCNs, which is the number of PCNs that will be generated if one should scan a full neighborhood at specific times in the future. In this research, we use the terms forecast and prediction model interchangeably. As stated in Section 2.1.8, we want to predict the expected number of PCNs for a future visit to a neighborhood by multiplying the

number of parking spots by our predictions of the occupancy ratio, the visitor ratio, and the non-paying ratio. The product of the visitor ratio and the non-paying ratio is equal to the PCN ratio. This approach solves the problem of not knowing to what extent the neighborhoods have been visited. Additionally, the different ratios can be replaced easily. This can be useful if in the future new ratios will be computed due to more or better data. Moreover, we include the parking regime, which was already retrieved by ARS, in our forecast by setting the PCN ratio of the parking hours without a fiscal parking regime to 0.

In Section 4.3.1, we first analyze the data of historic scans with regards to the visitor ratio and non-paying ratio. Section 4.3.2 presents our prediction model of the PCN ratio, including the visitor ratio and non-paying ratio, and Section 4.3.3 our estimation of the occupancy ratio. Finally, Section 4.3.4 tackles the question how long the visitors that receive a PCN stay in the same neighborhood.

4.3.1 Data analysis

For the purpose of the data analysis, we collect scan data of one year between 1.6.2016 and 1.6.2017. At the time of this analysis, this period encompasses the most recent source data available, and a one year period is considered sufficient for taking into account seasonal trends. Moreover, we prefer the recent data because it is more reliable due to the fact that there have been some organizational changes.

We choose to aggregate the scan data for every hour of the year and every neighborhood, as we want to look for certain patterns regarding the neighborhoods, hours, weekdays, weeks, month, and the weather. For every hour in a neighborhood, we retrieve the number of scans, visitors, and PCNs in order to derive the visitor ratio, the non-paying ratio, and the PCN ratio. We start by analyzing the ratios with regards to different time dimensions. Figure 9 shows the behavior of the ratios with regards to the week of the year.

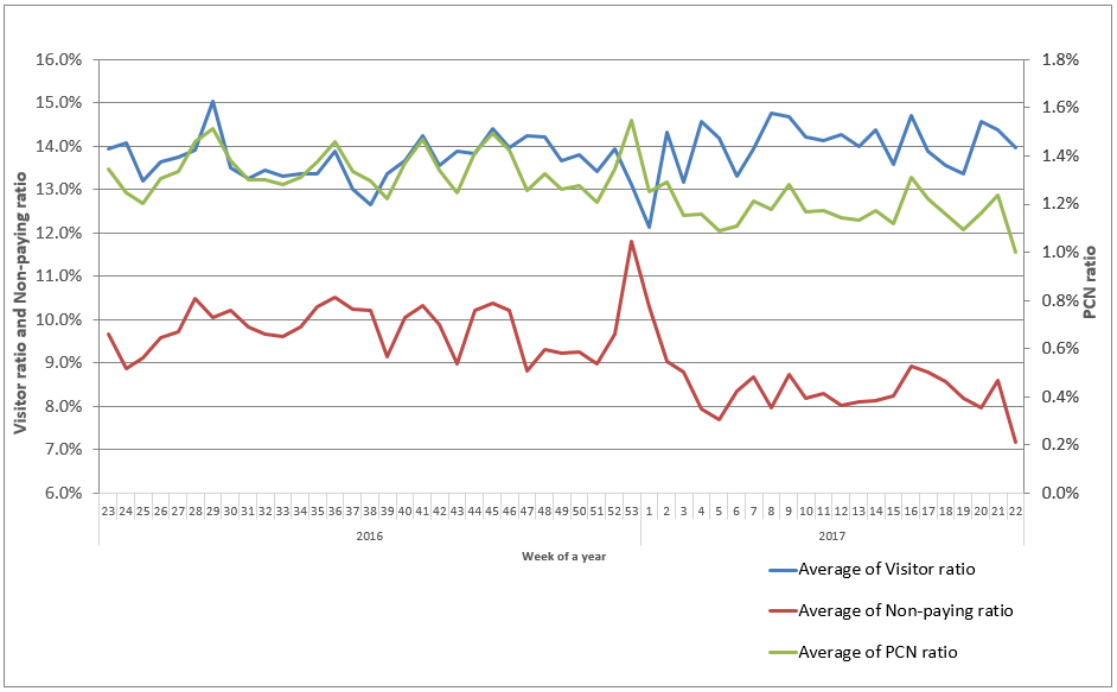


Figure 9 – Week diagram of visitor, non-paying, and PCN ratio

From Figure 9, we observe one big increase of the non-paying ratio at the end of the year 2016, probably due to Christmas and New Year. We cannot distinguish clear patterns in terms of seasons or months from this figure. However, we can clearly see that the non-paying ratio and the PCN ratio are decreasing over time. That means that more and more visitors are paying for their parking. This is an interesting observation. As EPS regularly adapts its current strategies to increase enforcement output, this effect could be due to changes in their enforcement approach. In Figure 10, we look at whether we can see if there is a relationship between the efforts of EPS and the payment rate. We consider the KPI area "Centrum" as it has the most scan data available. Figure 10 shows the average payment rate per week and the effort of EPS in terms of the absolute number of scans.

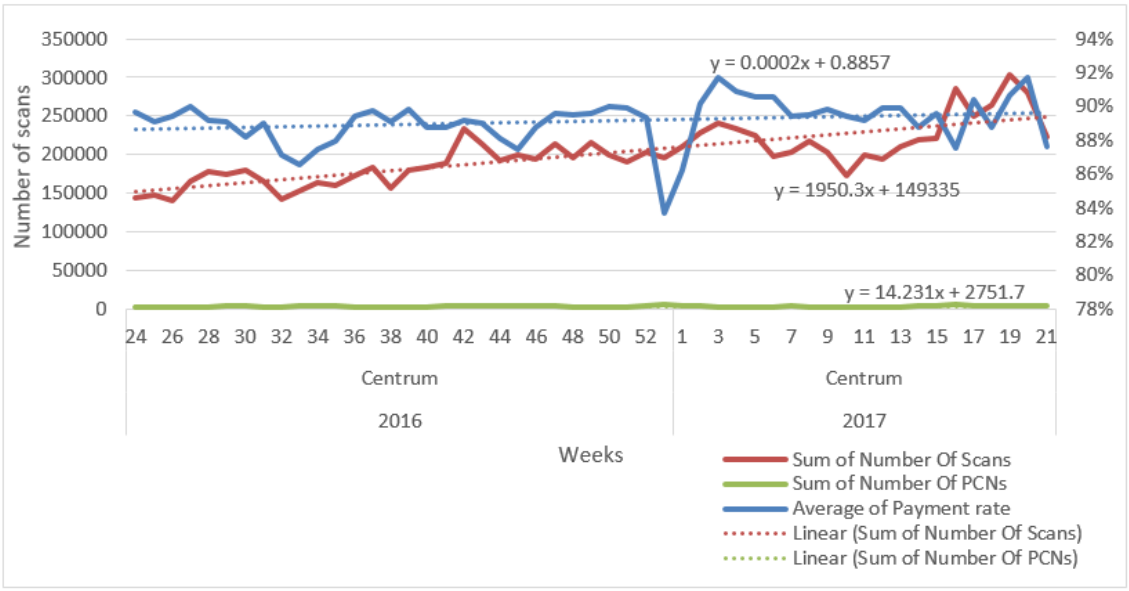


Figure 10 – Number of scans, number of PCNs, and average payment rate per week in "Centrum"

In line what we have seen in Figure 9, the payment rate is increasing with time. Probably the payment rate does not increase linearly as it converges towards 100%. However, Figure 10 shows the linear increase of the payment rate in order to give us an impression to what extent the payment rate increased during that year every week. It appears that the payment rate is increased by 0.02% (absolute number) every week. Moreover, Figure 10 shows that the number of scans and number of tickets increases every week. In Figure 10, we see that some peaks of the payment rate and the number of scans seem to be askew and others seem to be aligned. Consequently, although we determine that the enforcement efforts and payment rate both increase, we cannot conclude that there is a causal relation between the two metrics.

As a next step, we want to have a look at a smaller time dimension. Figure 11 shows how the different ratios behave for the different hours of a day.

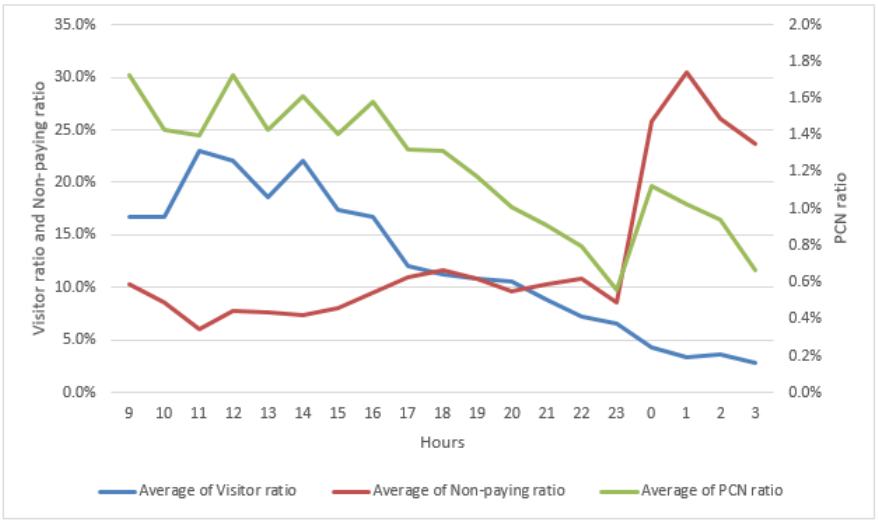


Figure 11 – Hour diagram of visitor, non-paying, and PCN ratio

We clearly see that the visitor ratio and the non-paying ratio depend quite heavily on the hour of the day. In the night there are less visitors, however, relatively more visitors do not pay for parking. Note that the scans at night can only be measured in a few neighborhoods of the KPI area "Centrum". It seems that in the noon there are the most visitors with a slightly increasing non-paying ratio. Even though the number is smaller, we see that there is an impact of the hour on the PCN ratio as is varies between approximately 0.6% and 1.8%.

Figure 12 shows the influence of the weekday. It appears that the ratios are quite the same within the week from Monday to Friday. The visitor ratio does not change a lot, however, on Saturday the visitor ratio seems to be lower. In the weekend, more visitors do not pay, which leads to a peak of the PCN ratio on Sunday (day 1) as the visitor ratio is also relatively high.

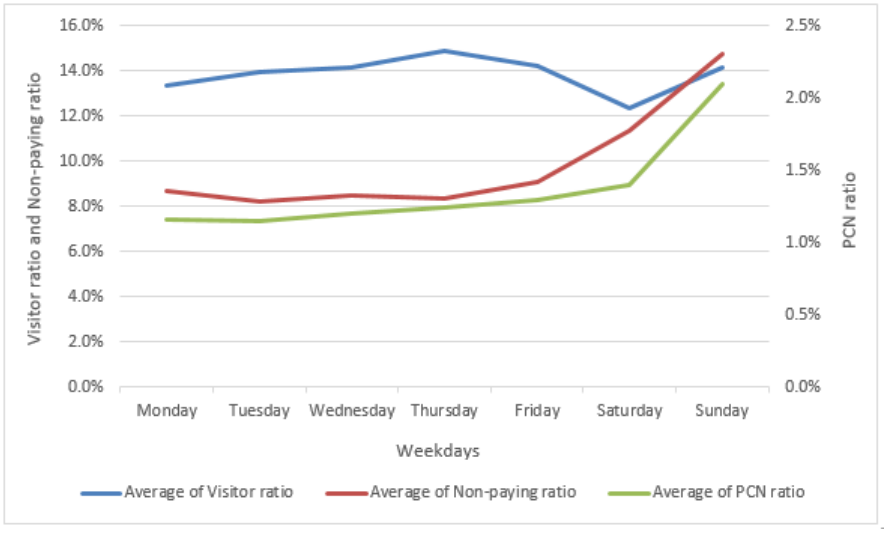


Figure 12 – Weekday diagram of visitor, non-paying, and PCN ratio

Now, we want to investigate if we see the same patterns when combining the hour of the day and the weekday. This is presented in Figure 13.

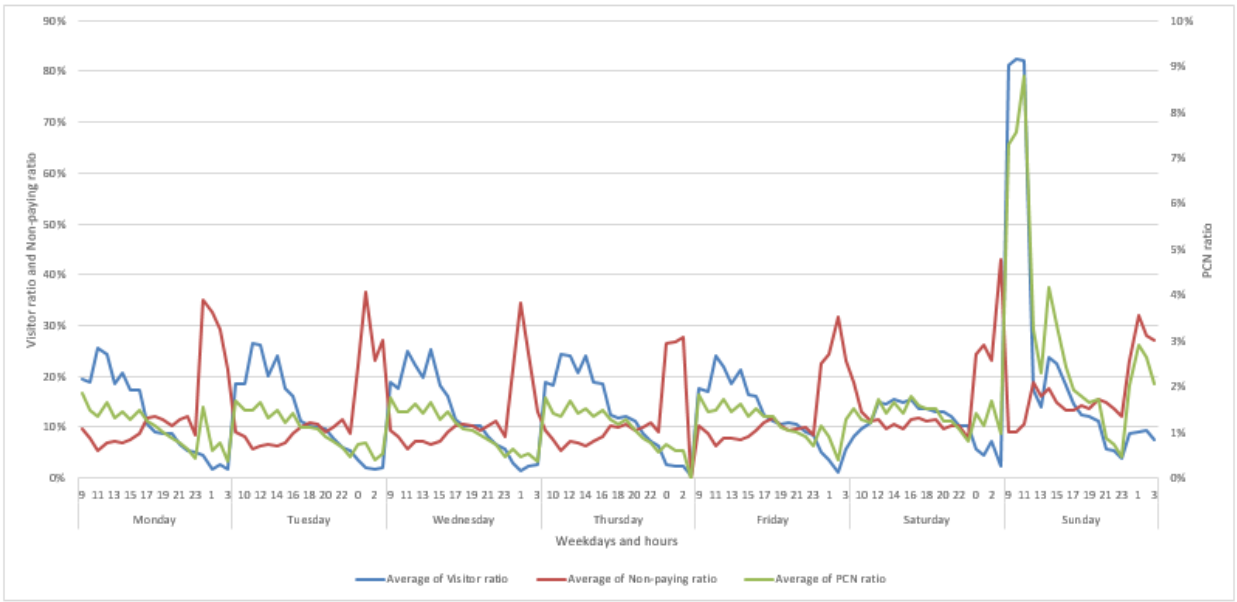


Figure 13 – Weekday and hour diagram of visitor, non-paying, and PCN ratio

We clearly see that the patterns during the week regarding the hour of the day stay the same during weekdays, but are different on Saturday and Sunday. Especially on Sunday in the morning around 10.00, there is an extreme peak of the visitor ratio and therefore also of the PCN ratio. According to the parking regime times, there are only three neighborhoods that have paid parking before 12.00 on Sundays and therefore the few numbers of scans in this neighborhood have a strong impact on this diagram. Next to the fact that there are only a few scans, we assume that most visitors do not know that there is a parking regime at that time or they know that this area is usually not scanned at that time. Moreover, the Saturdays miss the usual visitor peak around noon. Finally, we conclude that the hour pattern that we observed in Figure 11 stays more or less the same during the week and are different on Saturday and Sundays. Therefore, we derive the following time clusters, which we use for some parts of this analysis and in Section 4.3.2.

Day periods:

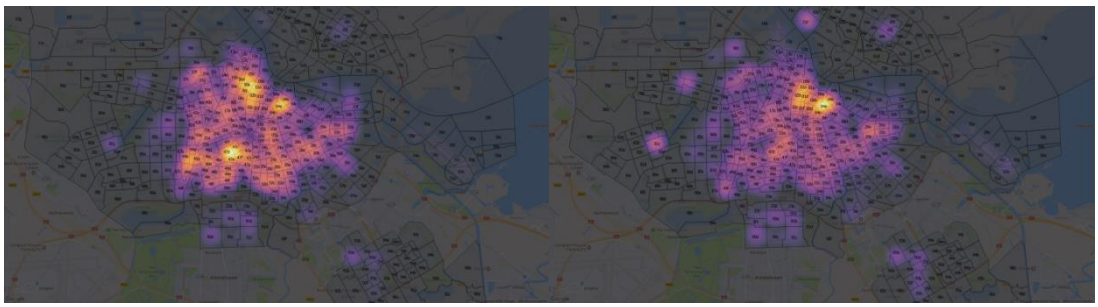
- Morning (9.00-11.00)
- Noon (11.00-15.00)
- Afternoon (15.00-17.00)
- Evening (17.00-21.00)
- Late evening (21.00-0.00)
- Night (0.00-4.00)

These periods are especially derived from the behavior of the PCN ratio in Figure 11. Even though the noon and afternoon could be considered as one cluster, we divide into two because although the PCN ratio remains more or less the same, the visitor ratio is decreasing and the non-paying ratio is increasing.

Week periods:

- During the week (Monday-Friday)
- Saturday
- Sunday

After having analyzed the time dependency, let us consider the spatial impact. First let us have a look at the absolute numbers of scans and PCNs to get an impression where EPS scans the most and where the most PCNs are obtained. Figure 14 shows a heat map that highlights the regions with the most scans and most PCNs in Amsterdam.



Number of scans Number of PCNs

Figure 14 – Heat maps of the absolute numbers of scans and PCNs between 1.6.2016 and 1.6.2017

We see that the most scans have been especially in the center of the map. This is due to fact that not all neighborhoods of Amsterdam have fiscal parking (see Figure 15).

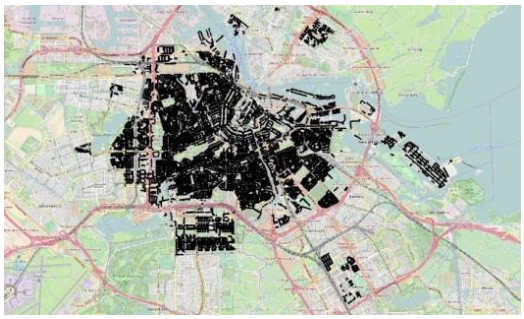


Figure 15 – Fiscal parking spots in Amsterdam

We distinguish three hotspots that are highlighted in yellow. Interestingly, it appears that there are two of these scanning hot spots are in line with the PCN hot spots that can be seen on the right of Figure 14, but one which is not. It seems that the PCN ratio of the third scanning hot spot is lower. Therefore, we continue our analysis by looking at the ratios of the neighborhoods instead. *Figure 16* shows the hot spots with regards to the visitor ratio, non-paying ratio, and PCN ratio.

Visitor ratio

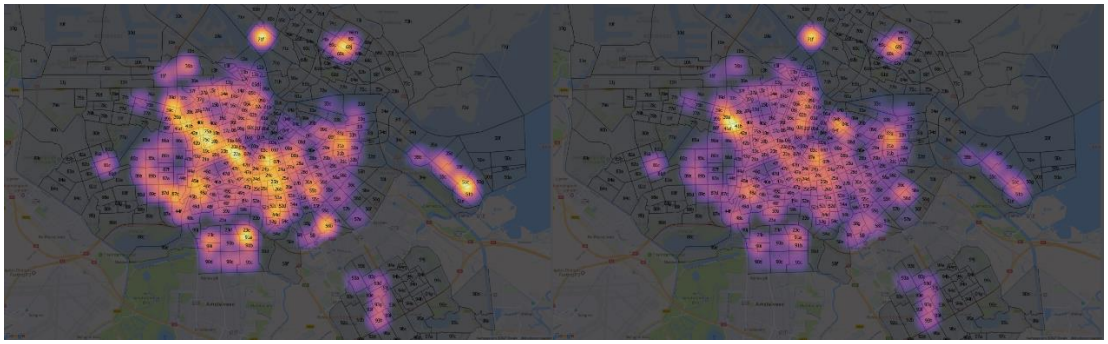
Non-paying ratio

PCN ratio

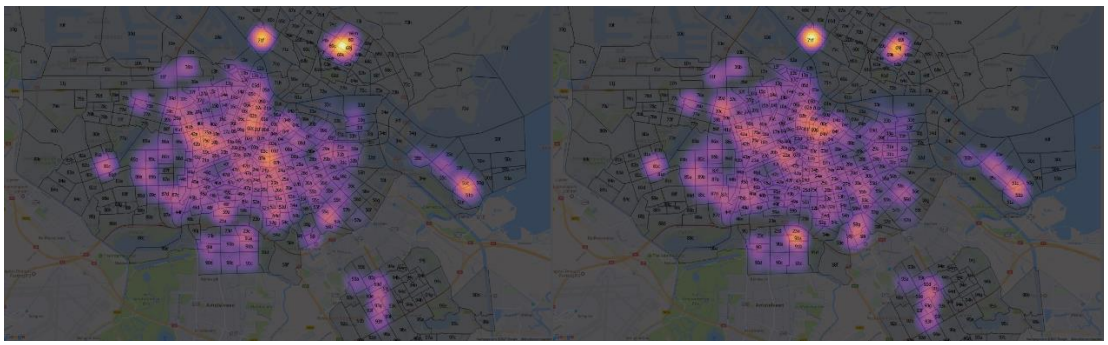
Figure 16 – Heat maps of the average visitor, non-paying, and PCN ratio between 1.6.2016 and 1.6.2017

The PCN ratio hotspots that we see in the north of all maps in *Figure 16* belong to the KPI area "Noord" and are neighborhoods with only a few parking spaces and a few scans that have a strong impact on this heat map. We see that there are less visitors paying in the center of the map. In combination with a slightly higher visitor ratio in the west; this leads to higher PCN ratio in that area.

As Figure 16 does not include the time aspect, we created a new heat map that shows the behavior of the PCN ratio regarding the different day periods. This heatmap is shown in Figure 17.



Morning (9.00-11.00) Noon (11.00-15.00)



Afternoon (15.00-17.00) Evening (17.00-21.00)

Late evening (21.00-0.00) Night (0.00-4.00)

Figure 17 – Heat maps of average PCN ratio for all neighborhoods in different time periods between 1.6.2016 and 1.6.2017

From Figure 17, we make the following observations:

- There are two hotspots in the north of the map that remain constantly until their parking regime is over.
- In the morning, the PCN ratio is higher in the center of the map and then decreases over time.
- The noon, afternoon, and evening are for all neighborhoods more or less the same.
- The later it gets, the less neighborhoods are highlighted because a lot of neighborhoods do not have a parking regime after 21.00.

Neglecting the two outliers in the north, which belong to the KPI area "Noord", and the fact that the neighborhoods have differing parking regimes, we conclude from Figure 17 that space and the hour of the day are independent with regards to the hour patterns, which we observed before.

Let us now check whether this conclusion is correct by looking at the hourly patterns of the different KPI areas. Figure 18 presents the impact of the hour on the ratios of all KPI areas except for "Noord", which is shown separately in Figure 19 due to its higher ratios. The night hours are excluded because only a few neighborhoods in the KPI area "Centrum" have a parking regime at that time.

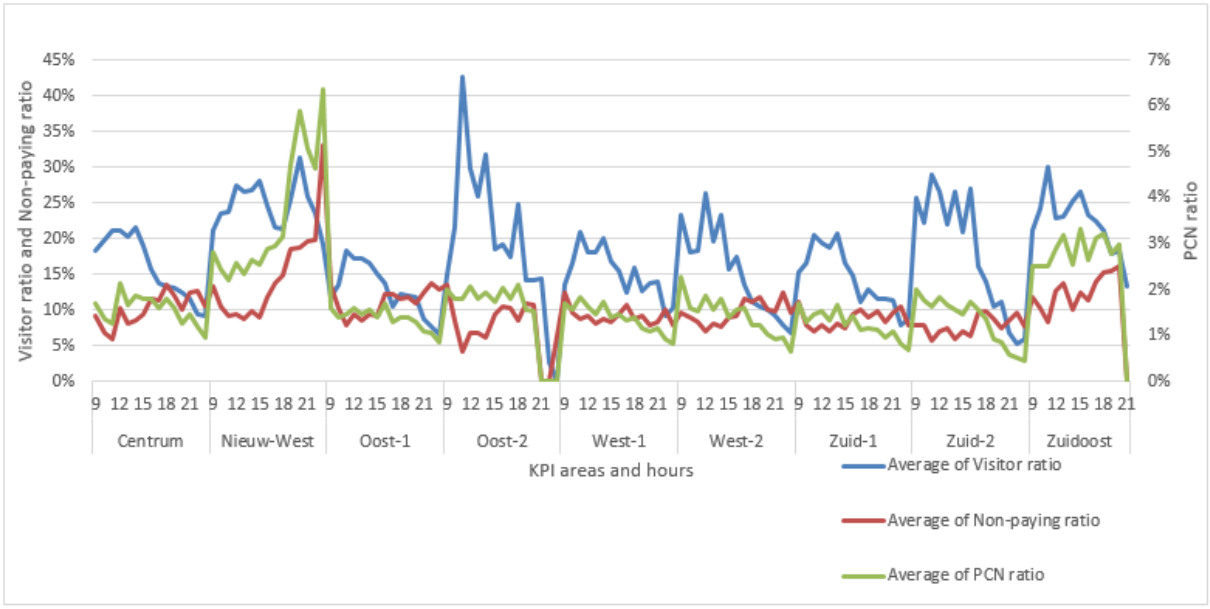


Figure 18 – KPI area (excluding "Noord") and hour diagram of average visitor, non-paying, and PCN ratio (between 9.00 and 24.00) between 1.6.2016 and 1.6.2017

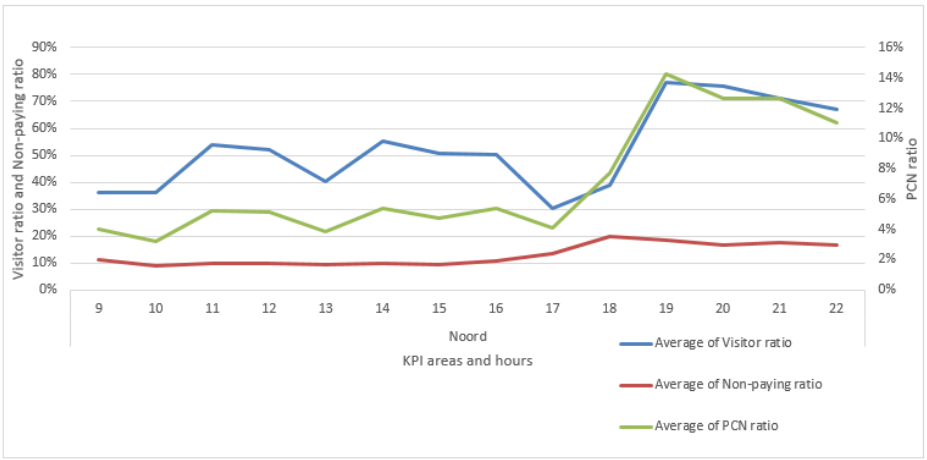


Figure 19 – Hour diagram (between 9.00 and 24.00) of average visitor, non-paying, and PCN ratio for KPI area "Noord" between 1.6.2016 and 1.6.2017

In Figure 18 and Figure 19, we see that KPI area "Noord" has generally a higher visitor ratio and also the PCN ratio at any time. This explains why it was consistently highlighted in Figure 17. Moreover, we see that in most KPI areas the non-paying ratio decreases after 9.00 a bit and then increases until the night. This increase, however, is different for all areas. Whereas the increase of the non-paying ratio is merely visible for KPI area "West-1", the non-paying rate of "Nieuw-West" and "Noord" goes from 0.1 up to 0.2 and therefore increases by roughly 100%. The visitor ratio behaves more or less the same for most KPI areas. The visitor ratio has one or two peaks after each other around noon and then decreases towards the night. Again "Nieuw-West" and "Noord" show a different behavior as they have another peak in the evening. It seems that our conclusion is correct, except that not only the KPI area "Noord" but also "Nieuw-West" behaves differently.

Let us now have a look at the weekdays. We exclude Sundays because on Sundays only some neighborhoods of "Centrum" and "Zuidoost" have a parking regime. The ratios are shown in Figure 20. It appears that "Nieuw-West", "Noord", "Oost-2", "Zuid-2", and "Zuidoost" have higher visitor ratios. This could be explained by less parking decks in these KPI areas, which leads to more visitors on the street. Another reason could be that these KPI areas lay further away from the center of Amsterdam and the KPI areas in the center, such as "Centrum", have generally more parking permits.

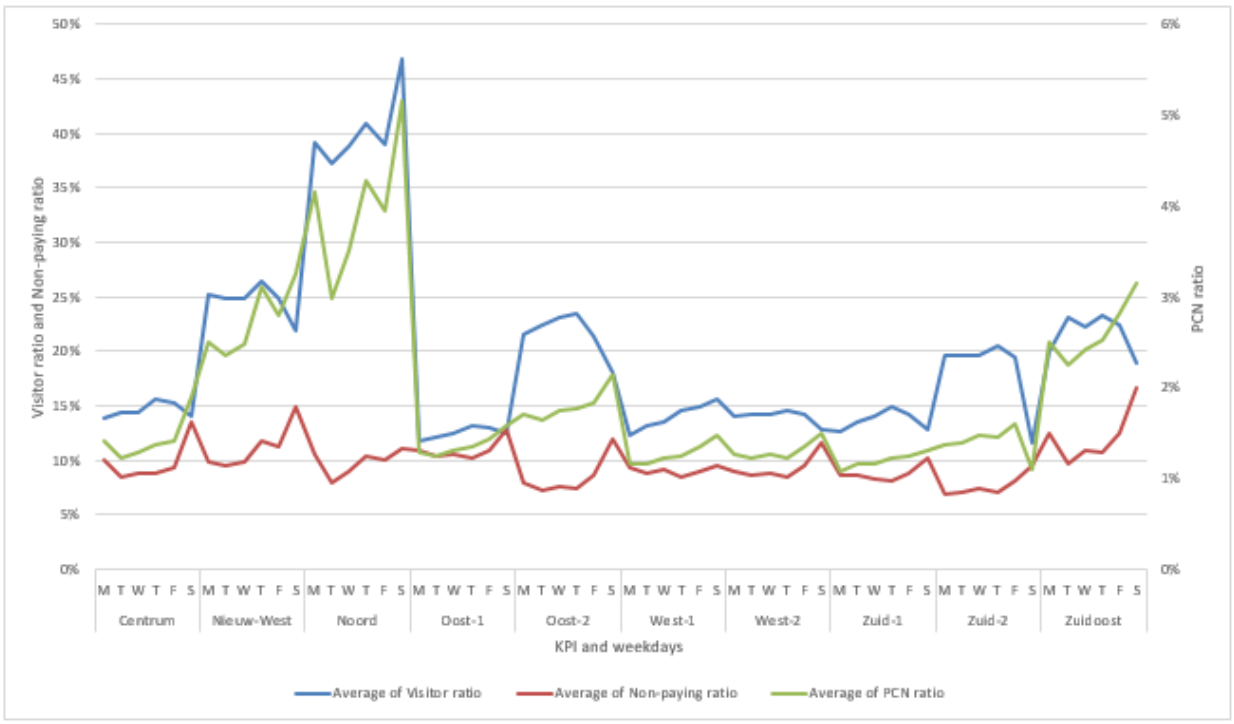


Figure 20 - KPI area and weekday diagram of visitor, non-paying, and PCN ratio

All KPI areas show an increase of the non-paying ratio towards Saturday. However, the KPI areas "Zuidoost", "Centrum", "Nieuw-West" show the strongest increase. We consider these as two KPI clusters because "Zuidoost" and "Centrum" have a Sunday parking regime. The visitor ratio seems to decrease on Saturday except for "Noord". In line with what we have observed seen in Figure 17, Figure 18, and Figure 20, consider "Noord" as a separate cluster. Finally, we conclude the following four KPI clusters with regards to the behavior of the ratios.

- Noord
- Nieuw-West
- Zuidoost and centrum
- West-1, West-2, Zuid-1, Zuid-2, Oost-1, Oost-2

Even though these clusters provide insights about the general behavior of a KPI area, not all neighborhoods within a KPI area behave the same. Since it is too time consuming to analyze the ratios of every single neighborhood, we think of two methods to give more insights about the different neighborhoods.

Find outliers within a KPI area

In this method, we look for neighborhood outliers and classify them in certain neighborhoods groups in order to explain their behavior with additional explanatory variables. These neighborhoods groups are:

- Nightlife
- Shopping
- Industry
- Business
- Houses
- Sports

The classification can be done with the support of EPS for a limited number of neighborhoods. In order find these outliers, we consider all neighborhood within a KPI area. Then, we write down the most noticeable outliers with regards to the ratios that cannot be explained by a small number of scans. Figure 21 shows an example, where we consider the KPI area "Centrum" and the different numbers on the x-axis show all neighborhoods that belong to "Centrum".

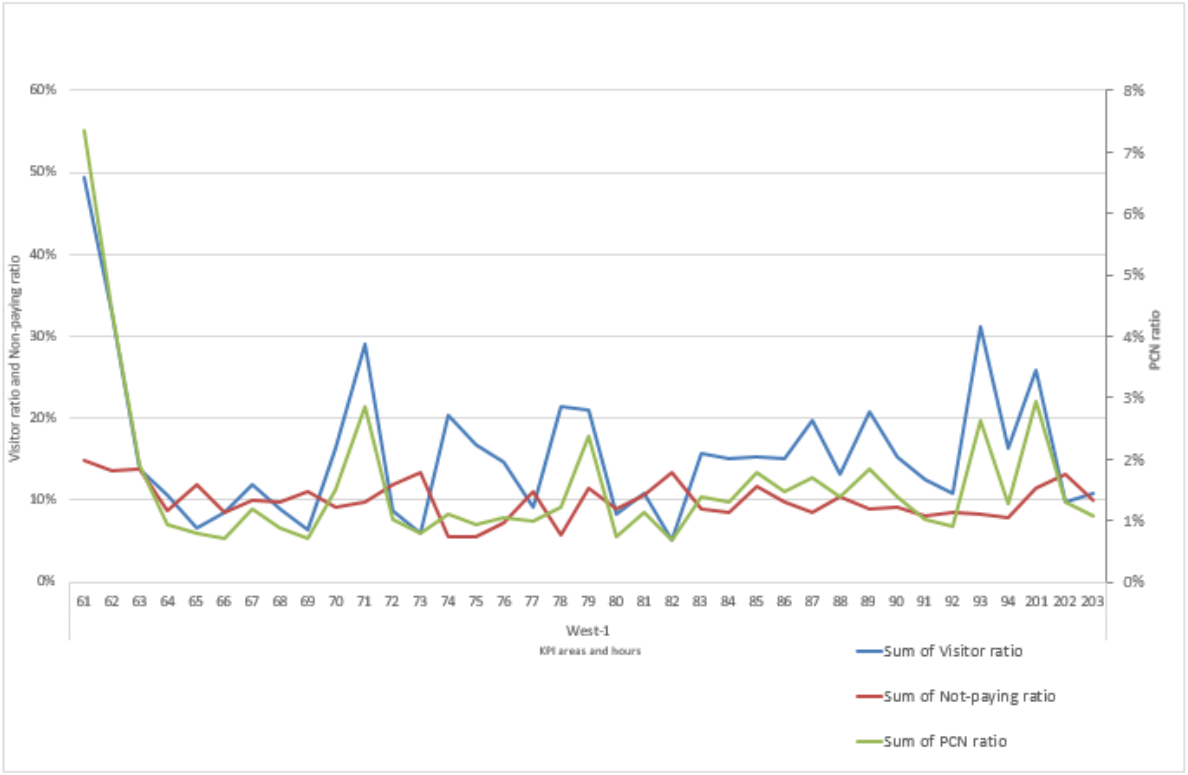


Figure 21 - KPI area and neighborhood diagram of visitor, non-paying, and PCN ratio

In this figure, we visually derive the following PCN ratio outliers: 61, 71, 79, 93, and 201. Analogously, we determine the outliers for all KPI areas. For all 10 KPI areas, we have found 43 neighborhoods that we consider outliers. The results of EPS' area classification of these 43 neighborhoods are: 6 in nightlife area, 7 in industrial area, 0 in business area, 17 in living area, 11 in shopping area, and 2 in an area with sport fields. We use the classification of these outliers later in Section 4.3.2.

PCA and K-means clustering

In this method, we investigate whether there are neighborhood clusters that behave the same without considering their location or the KPI area they belong to. With regards to the neighborhoods, we are interested in four different behaviors:

1. The behavior of the average visitor ratio regarding the hour of the day.
2. The behavior of the average non-paying ratio regarding the hour of the day.
3. The behavior of the average visitor ratio regarding the day of the week.
4. The behavior of the average non-paying ratio regarding the day of the week.

We consider the different behaviors regarding the hours and weekdays and not at the day period and weekday clusters that we distinguished before because now we want to see if there are neighborhoods that behave differently regarding the different hours and weekdays. Since we analyze the neighborhoods separately and not the average of all neighborhoods or one KPI area, it may happen that some neighborhoods have only a little number of scans. Therefore, we rank the 320 neighborhoods with an active parking regime by the number of scans, starting with the fewest. It appears that the first 11 neighborhoods have less than 362 scans within one year (1.6.2016-1.6.2017) and the 12th neighborhood has 1557. We filter out the first 11 neighborhoods because we consider the number of scans too little in order to make conclusions about a weekday or hour pattern.

Now, let us start with the behavior of the average visitor ratio regarding the hour of the day in order to explain our approach. From our data, we can retrieve the average visitor ratio for every hour of the day for every neighborhood. We exclude the night hours because only a few neighborhoods have a parking regime at that time. Thus, a neighborhood's behavior regarding the visitor ratio during the day can be modelled as a vector that starts at with hour 0 (9.00-10.00) and ends with hour 14 (23.00-24.00). The dimension of this vector is 1x15. Now, we want to compare all neighborhood vectors in order to see if there are cluster groups that behave the same and whether there are outliers that behave differently. To this end, we make use of the primary component analysis (PCA). The PCA is often used in statistics to reduce the variables of a data set to a smaller set of variables called principal components (Daszykowski,

Kaczmarek, Vander Heyden & Walczak, 2007). The principal components, which define a new coordinate system, are the outcome of maximizing the description of the data variance. The number of principal components is determined before but "usually the first few PCNs are enough to represent the data structure well" (Daszykowski et al., 2007, p.1). The PCA is often applied to combine correlated variables such that in the end the data set only exists of uncorrelated variables. In our case, we choose to reduce our data set to two principal components. This enables us to visualize the two-dimension data set in a scatter plot. But we have to check whether this is enough to represent our data structure well enough.

The goal of this approach is to assign neighborhoods to certain clusters with regards to their four behaviors, we mentioned earlier, or to define them as outliers. Figure 22 contains the coordinate system that is created in the reduction of the PCA and a scatter plot that shows all neighborhoods whose visitor ratio has a similar behavior regarding the hour of the day. The exact values of the x- and y- coordinates of this scatter plot do not matter because they contain the values of the newly created coordinate system, which do not provide additional nor useful insight for us.

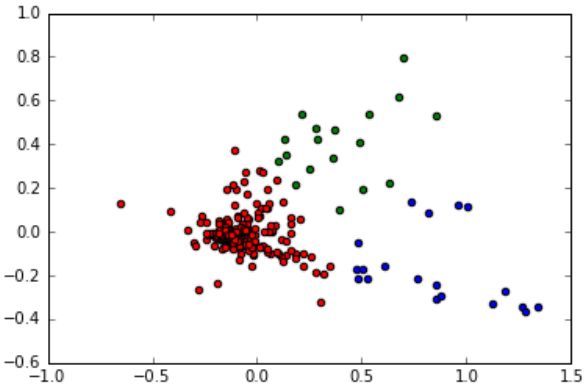


Figure 22 – Scatter plot of the neighborhoods regarding the average visitor ratio of all neighborhoods for every hour of the day with colors showing the results of the K-means clustering analysis

Figure 22 shows the neighborhood points in three different colors. These colors show three clusters, which are created by the K-means clustering method. K-means clustering requires a data set and a pre-determined number K (Anderson, 2009). The K-means clustering creates a number of so-called centroids that is equal to K. These centroids are placed randomly in the scatterplot. Every point in the data set is assigned to one of the centroids and takes the same color as the assigned centroid (in *Figure 22*: red, blue, and green). Afterwards, the location of the centroids is improved iteratively in such a way that the square of the Euclidian distance is minimized. A drawback of the K-means clustering is that it does not necessarily find the best possible outcome because its result might be a local optimum. However, for this purpose it serves well enough because we only want to find the obvious cluster that we see and can manually change them if necessary. We experiment with the value for K until we have marked the clusters that we visually derive from the scatter plot. Therefore, the number of different colors varies in the following figures. In *Figure 22*, K is equal to 3 and it marks the 309 neighborhoods that we see in the scatterplot. Figure 23 shows the original vector of the red centroid. The x-axis shows the hour of the day starting with hour 0 (9.00-10.00) and ending with hour 14 (23.00-24.00). The y-axis shows the difference between the value of the visitor ratio and the average of the day, such that 0 is the average of all neighborhoods of a day. Due to this normalization, we see immediately whether the visitor ratio of the neighborhood performs above or below its average in a certain hour.

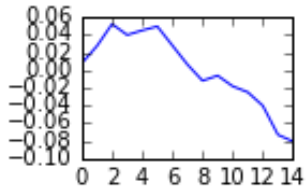


Figure 23 - Diagram showing the hour vector of the visitor ratio of the blue centroid

From Figure 23, we conclude that all 272 neighborhoods (marked in red) behave similar to the vector shown in Figure 23 and the rest do not. Since the cluster contains the majority of the neighborhoods, we expect to see this behavior in the average visitor ratio of all neighborhoods, which was shown in Figure 11. And indeed, the behavior of the visitor ratios is very much the same. We consider that as a validation

for using two principal components because the majority of coordinates in our new created coordinate system is able to represent the same effect that we have seen before. For that reason, we continue applying this method to the visitor ratio regarding the weekdays and the non-paying ratio regarding the hours and weekdays with different values for K varying between 2 and 6. The colors that are used are: red, blue, green, yellow, grey, and black.

Figure 24 shows the scatterplot of average visitor ratio of all neighborhoods regarding the day of the week in four colors. Concerning the weekdays, we do not consider Sundays because only a few neighborhood have a parking regime then.

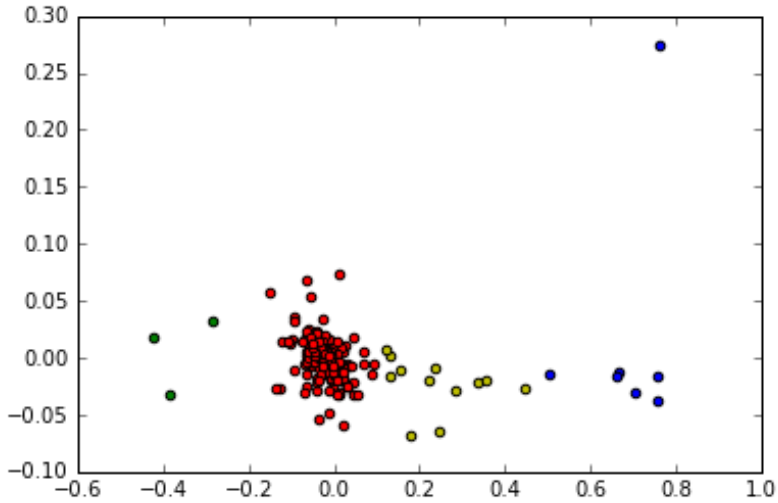


Figure 24 - Scatter plot of the neighborhoods regarding the average visitor ratio of all neighborhoods for every day of the week

We define the red points with 287 neighborhoods as one cluster. Figure 25 shows the weekday vector of the red cluster.

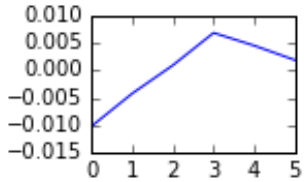


Figure 25 - Diagram showing the weekday vector of the visitor ratio of the red and blue centroid

Figure 26 shows the average non-paying ratio of all neighborhoods for every hour of the day in four colors.

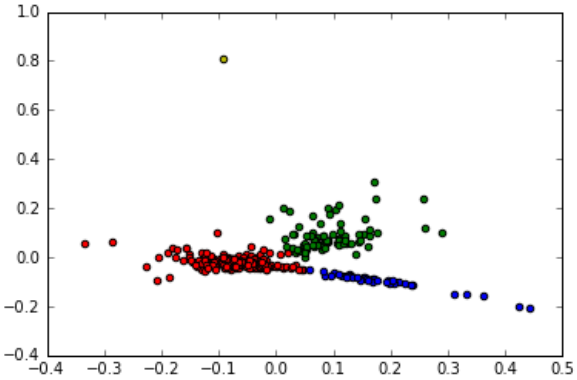


Figure 26 - Scatter plot of the neighborhoods regarding the average non-paying ratio of all neighborhoods for every hour of the week

This time, we visually derive three noticeable clusters from Figure 26: red with 193 neighborhoods, blue with 40 neighborhoods, and blue with green neighborhoods. The vectors of these clusters are shown in Figure 27.

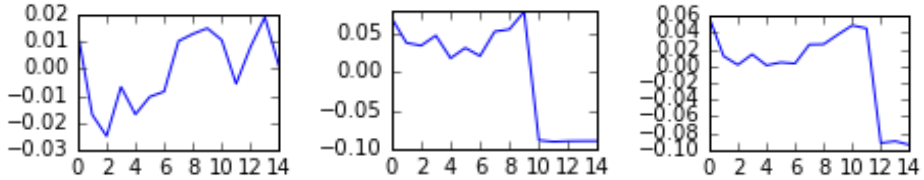


Figure 27 - Diagram showing the hour vector of the non-paying ratio of the red, green, and blue centroid

Finally, Figure 28 shows the average non-paying ratio of all neighborhoods for every day of the week in five clusters.

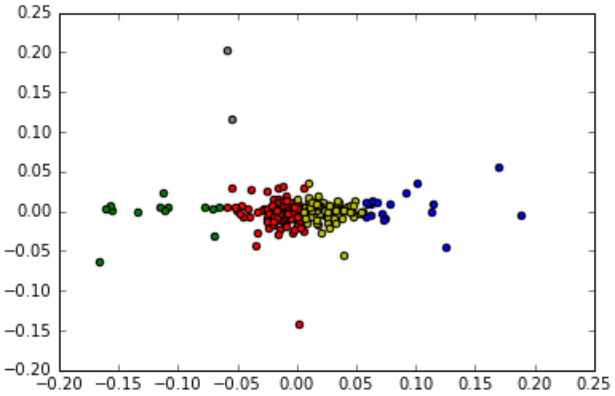


Figure 28 - Scatter plot of the neighborhoods regarding the average non-paying ratio of all neighborhoods for every day of the week

We define the red and yellow cluster to be one cluster, which has 247 neighborhoods, but we exclude the red colored outlier at the bottom of the figure. Figure 29 shows the weekday vector of the red centroid and yellow centroid.

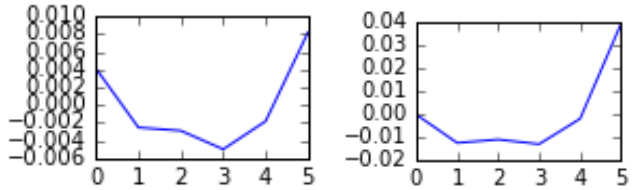


Figure 29 - Diagram showing the weekday vector of the non-paying ratio of the red centroid and yellow centroid

We have seen that the visitor ratio and non-paying ratio of the neighborhoods do not follow the same behavior regarding the hour of the day the weekday. These results will be used later in Section 4.3.2.2.

Next to time and space, we analyze the impact of the weather in Amsterdam on the ratios. We retrieve the weather data from the KNMI website (2017). Unfortunately, the weather data of the city of Amsterdam is not available in this source. That is why, we use the weather data of the airport Schiphol, which is close to Amsterdam. From the available data, we choose to analyze the average temperature, number of sun hours, the precipitation hours, and the relative humidity of a day. Figure 30 shows the influence of the temperature on the ratios.

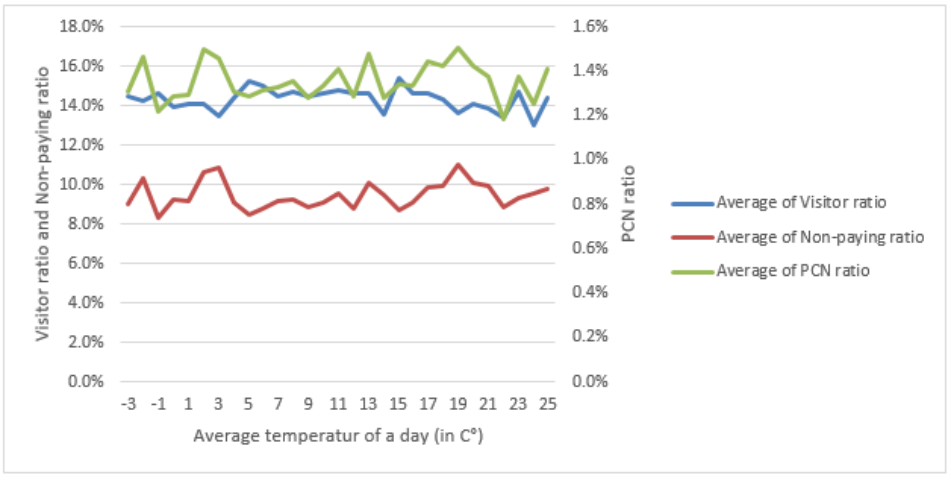


Figure 30 - Temperature diagram of visitor, non-paying, and PCN ratio

Figure 31 shows how the ratios behave regarding the number of sun hours in a day.

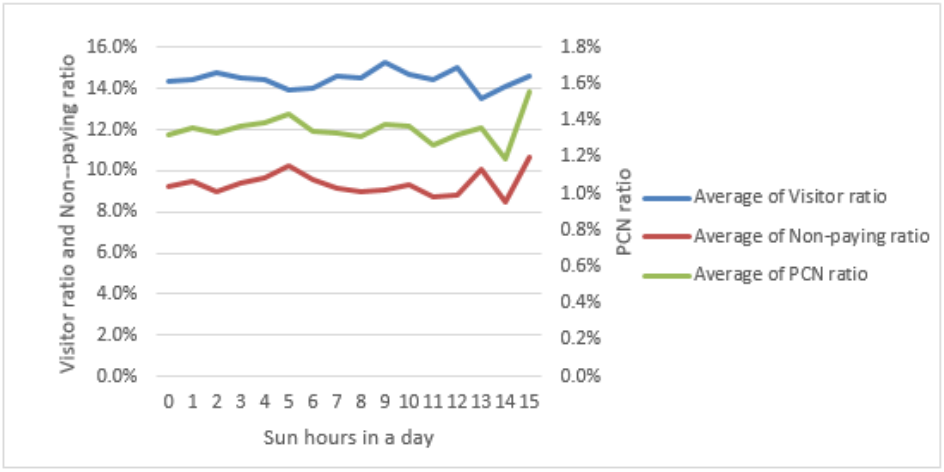


Figure 31 – Sun hour diagram of visitor, non-paying, and PCN ratio

Figure 32 presents the impact of the number of precipitation hours in a day on the ratios.

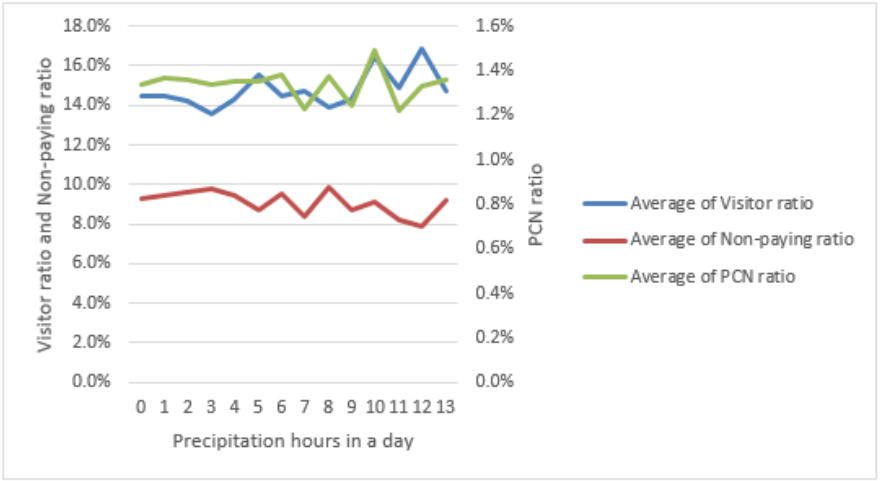


Figure 32 – Precipitation hour diagram of visitor, non-paying, and PCN ratio

Figure 33 shows how the relative humidity of the day affects the ratios.

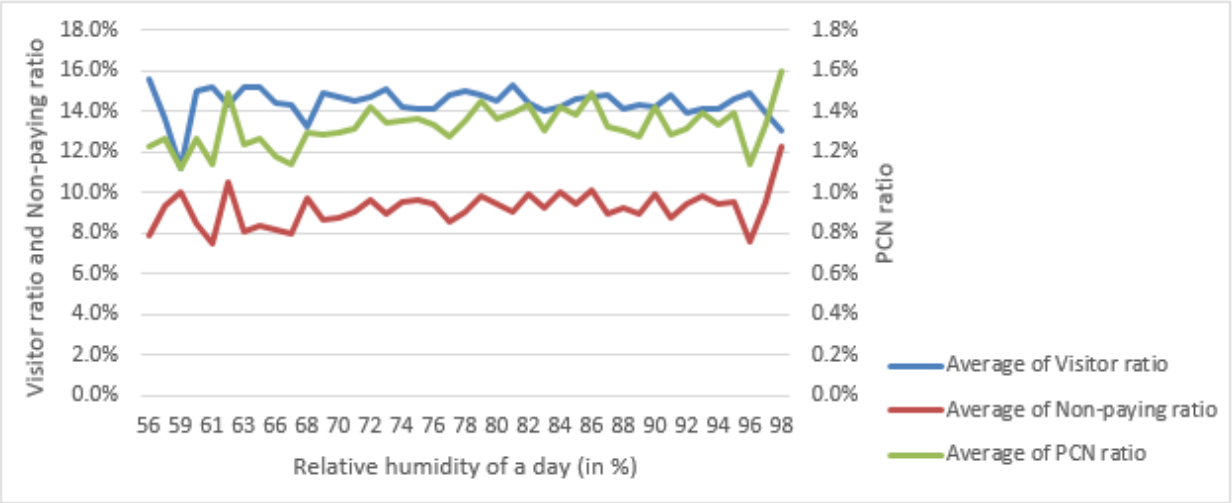


Figure 33 – Relative humidity diagram of visitor, non-paying, and PCN ratio

From these figures we cannot distinguish any patterns. One might expect that, for instance, people are less willing to pay if it is raining but this is not the case.

In the next section, we use the results of data analysis for our prediction model.

4.3.2 Prediction of the PCN ratio

This section describes our approach of predicting the PCN ratio. Section 4.3.2.1 discusses a naive prediction) that serves as a benchmark model for our main prediction model that we present in Section 4.3.2.2.

4.3.2.1 Naive prediction

This section develops a naive prediction (also denoted as Naive Forecast), which serves as a benchmark for our prediction model introduced in Section 4.3.2.2. This benchmark model is directly derived from the findings of our data analysis.

In the data analysis, we concluded that time and space is independent within four KPI areas clusters:

- Noord
- Nieuw-West
- Centrum, Zuidoost
- West-1, West-2, Zuid-1, Zuid-2, Oost-1, Oost-2

Within these clusters time and space are independent in a sense that the visitor and non-paying ratio increase and decrease the same regarding the hours of a day and days of a week. Inspired by the work of van Hal (2015), which we explained in Section 3.3.3.2, we make a naive forecast by using factors that are based on time or location.

- Spatial factor:
  - Neighborhood factor
- Time factors:
  - Day period factor
  - Week period factor
  - Week factor

We multiply these different factors with a base PCN ratio of 1.485%, which is the average PCN ratio of all scans between 01.06.2016 and 01.06.2017. We only consider the PCN ratio because in the end it is the product of the non-paying ratio and the visitor ratio and in Section 4.3.1 we only regarded them separately because we wanted to analyze the different behaviors of those two. The neighborhood factor adjusts the PCN ratio regarding the different neighborhood (e.g. if neighborhood 20 is below average, then the factor will be below 1). The day period factor adjusts the base line value regarding the day period clusters (Morning, Noon, Afternoon, Evening, Late Evening, and Night). For instance, the value in the morning would be above one because we expect the PCN ratio to be higher than average in the morning.

The week period factor does the same for week period clusters (During week, Saturday, and Sunday). The week factor is different from the other factors. In Section 4.3.1, we did not see any noticeable clusters regarding the weeks. However, we did see a time series effect, namely that the non-paying ratio decreases over time. Therefore, the week factor can rather be considered as a week function, which decreases over time. Finally, our naive forecast works as follows:

PCN ratio = Base PCN ratio * Neighborhood factor * Day period factor * Week period factor * Week factor

We compute the three time factors for all four KPI areas clusters that we distinguished in Section 4.3.1. First, we compute the average PCN ratio for the different week periods and day periods. Afterwards, we divide the values by their average (e.g. the PCN ratio of the weekday period "During Week" is divided by the average of all week periods "During Week", "Saturday", and "Sunday"). The result is a factor that indicates if the PCN ratio increases or decreases at that time in this KPI area. Table 5 shows the so-called "week period factor" for all KPI areas and different week periods.

KPI areas	Sunday	Saturday	During Week
Centrum and Zuidoost	1.389	1.211	0.863
Nieuw-West	0	1.198	0.964
Noord	2.476	1.095	0.762
West-1, West-2, Zuid-1, Zuid-2, Oost-1, Oost-2	4.287	1.071	0.987

Table 5 – The results of the week period factor for each KPI area cluster

The results of the hour clusters are shown in Table 6.

KPI areas	Afternoon	Evening	Late Evening	Morning	Night	Noon
Centrum and Zuidoost	1.161	1.025	0.677	1.032	1.079	1.257
Nieuw-West	0.997	1.233	2.120	0.962	0	0.902
Noord	1.085	1.114	2.484	0.739	0	0.963
West-1, West-2, Zuid-1, Zuid-2, Oost-1, Oost-2	1.091	0.906	0.664	1.282	0	1.232

Table 6 - The results of the day period factor for each KPI area cluster

Likewise, we compute the neighborhood factor. The results are shown in Appendix C.

As we have seen in Section 4.3.1, the payment rate increased every week within the year between 01.06.2016 and 01.06.2017 with 0.02% (absolute number). Therefore, the week factor is equal to 0.9998x, where x is the difference in weeks between the current week and the starting week, which is the first week of June 2016. Finally, let us consider an example. The PCN ratio of neighborhood 20 (in "Centrum") at 10.00 am on a Tuesday in the third week of June is equal to:

$$PCN\ ratio\ =\ 0.01485\ (base\ line\ PCN\ ratio) * 1.083\ (neighborhood\ factor) * 0.863\ (week\ period\ factor) * 1.032\ (day\ period\ factor) * 0.9996\ \big(week\ number\ factor: 0.9998\ *\ (3-1)\big) =\ 1.418\%.$$

4.3.2.2 Neural network

As our prediction model, we choose to apply a neural network model because neural network models are very efficient in solving different regression models. Moreover, we can add the number of weeks and years in order to include the fact that the non-paying ratio decreases over time. In addition, other techniques have problems because there is no data available for some neighborhoods at certain times. Neural networks perform well despite these gaps due to generalization as long as the data set is big enough. Building a neural network requires different design choices. The different design choices and conclusions that we made are discussed throughout this section.

4.3.2.2.1 The loss function

As a loss function, we use the absolute mean error as it is easy to interpret. When training the data set with the absolute mean error and the mean squared error in short trails, the absolute mean error outperformed the mean squared error with regards to both error measurements. This is probably due to the fact that the mean squared error has the disadvantage of weighing outliers too much. Other loss functions also have the disadvantage that they do not allow 0 as an output or require a time series model. Even though we have discussed that there is indeed a decrease of the non-paying ratio, our prediction model is mainly a regression problem.

4.3.2.2.2 The data set

As a data set, we consider the same as we used for the data analysis. This data set contains all separate hours that a neighborhood was scanned of one year (1.6.2016-1.6.2017). Initially, we had considered also to separate the data into four different data sets for every KPI area but this decreases the performance of the network; probably, because the individual data sets become too small.

In this data set, we filter out all hours that have less than 106 scans. The reasoning behind that is that we only want to keep the ratios of the separate hours (visitor ratio, non-paying ratio, and the PCN ratio) that have a margin of error of 0.05 and a confidence of 70% (the sample size formula is discussed in Section 4.2.1). If we choose a smaller margin of error or a higher confidence, the required number of scans will increase. Since we still need enough data to train our model, we choose these values and consider them acceptable to train the network. However, there is a problem with this number of scans because some neighborhoods have only a few parking spots and consequently the visits there count only a few scans. Due to this filtering out the hours with less than 106 scans, the entire neighborhoods disappear from the data set. We trust that the generalization of the network, will cope with these gaps.

As mentioned in Section 3.3.3.2, LeCun et al. (2012) states that shuffling the data set helps because the network learns faster from unexpected samples. Moreover, we do no use the full data set to train our neural network because after training the model, we want to test the model. Therefore, we split up the data into a training set (80%) and a test set (20%). Having a separate test data set, helps us to avoid overfitting, i.e., the neural network learns too much from the data set in a sense that it also learns from outliers and noise instead of creating a general applicable prediction model. The bigger the gap between the performance of the training and test set, the bigger the problem of overfitting is because the network cannot apply the model to unseen data. Due to the overfitting problem, we only consider the results of the test set in order to compare our results.

4.3.2.2.3 Choice and presentation of inputs and output

Before introducing our inputs and outputs, we introduce the different normalizations that we used for the inputs. The normalization of the variables speeds up the learning process of the neural network because the variables are presented in a way that is easier to understand for the network. We use 3 different kinds of normalizations:

- N1: Value – average of all values. This normalization returns the values in such a way that the average is 0.
- N2: Value/average of all values. This normalization only works for positive values. Returns small positives values, where the average is 1.
- N3: We apply a min max normalization for the week number. The week number is equal to: the week number of the year + 52 * the number of the year (year 2016= 1, year 2017 = 2). By applying a min max normalization, it returns a value between 0 and 1:

$$\frac{Week\ number - Minimum\ week\ number}{Maximum\ week\ number - minimum\ week\ number}.$$

For our neural network, we include the average visitor ratio and non-paying ratio of the hour of the day and day of the week. In that way, the network can learn from both of these ratios and we can apply our neighborhood clusters based on the behavior of the two ratios regarding hours and weekdays (see Section 4.3.1). As we use these neighborhood clusters, we do not consider the KPI area clusters from Section 4.3.1. As output variables, we consider the following two options:

1. We predict the PCN ratio, which is the number of PCNs divided by the number of scans.
2. We predict the absolute number of PCNs and use the absolute number of scans as an input.

We choose the second as it leads to a better performance of the network. Moreover, we add the different clusters that we observed in Section 4.3.1 by giving them a value of 0 or 1. Our final input set includes the following variables, annotated with the normalization method that is applied:

- The average visitor ratio of the neighborhood (N2)
- The average non-paying ratio of the neighborhood (N2)
- The average visitor ratio of the week period (N2)
- The average non-paying ratio of the week period (N2)
- The average visitor ratio of the day period (N2)
- The average non-paying ratio of the day period(N2)
- The week number (N3)
- The number of scans (N2)

- Clusters:
  - Morning (0,1)
  - Noon (0,1)
  - Afternoon (0,1)
  - Evening (0,1)
  - Late evening (0,1)
  - Night (0,1)
  - Weekday (0,1)
  - Saturday (0,1)
  - Sunday (0,1)
  - The seven different neighborhoods clusters with regards to their behavior during the different week periods and day periods (0,1)
  - Night regime (0,1)
  - Sunday regime (0,1)

During our experiments it appeared that the N2 normalization works slightly better than the N1 normalization. We also considered adding more variables but these variables did not improve of even decrease the performance of the prediction results. In the end, we excluded the following variables:

- Neighborhood clusters (Nightlife, Shopping, Industry, Business, Houses, and Sports)
- KPI area clusters
- The average visitor ratio of the KPI area
- The average non-paying ratio of the KPI area
- The coordinates (in order to include an additional spatial factor)

4.3.2.2.4 Number of hidden layers and nodes within a layer

The design of the network regarding the number of hidden layers and the nodes within a layer provides infinitive possibilities. White (1992) and LeCun et al. (2012) state the use of too many parameters and too many layers can lead to overfitting. And indeed, we experienced that using more than 2 hidden layers leads to better results regarding the train data set but worse results regarding the test data set. Using one hidden layer leads to slightly worse results than using two; consequently we use 2 hidden layers. The number of nodes in the hidden layers is based on experiments; in the end we choose 120 and 70 hidden nodes respectively for our two hidden layers.

4.3.2.2.5 The activation function of a layer

The activation function of a layer determines the output of the hidden layers. The performance of the activation function is based on the different input variables and the final desired output. After experimenting with different activation functions, we use an activation function called “Rectified linear units” because it shows the best performance for our data set. The definition of this function is: $F(x) = \max(x, 0)$. In our case, it simply means that the output is equal to the input because we do not have negative input variables.

4.3.2.2.6 Computing inputs

As discussed in Section 4.3.2.2.3, the output of our neural network are the expected number of PCNs, which we want to compute now for all neighborhoods, hours, weekdays, and week numbers.

While computing inputs for our routing algorithm with the proposed neural network, we notice some strange values. On the one hand, we sometimes have negative outputs, which should not be possible as the number of PCNs cannot be negative. On the other hand, we find extremely high outputs. Some of them even exceed the number of scans, which is not possible. By taking a closer look at the negative values, it seems that they occur whenever there is no parking regime since 98% of the cases happen between 0.00 and 4.00 am, when most neighborhoods do not have a parking regime. As the algorithm checks also for the parking regime times, these values would never be used and are therefore not an issue. The extremely high numbers of PCNs that we found, however, are a problem. These outcomes belong to neighborhoods that were not part of the training nor the test set, as we filtered out the hours with less than 106 scans. We hoped that the generalization of the neural network would compute reliable numbers anyways. Basheer and Hajmeer (2000) describe that it is good to have an evaluation set next to the training and test set in order to see how well the generalization of the network works. In our case, this could be done by taking some specific observations (e.g., a few neighborhoods or one specific hour) out of the original data set. These excluded observations would form the evaluation set. Afterwards, the remaining data set can be split into the test and training set. By doing so, the prediction model, which is trained on the training set, can be tested for overfitting on the test set and its generalization capabilities can be evaluated by means of the evaluation set.

If we would have done this, it might have been a better choice to keep some variables such as the GPS coordinates in the network because of the additional spatial input, which helps the neural network if some neighborhoods are unknown.

Finally, we decide to train the network again (without an evaluation set) with the PCN ratio as an output. This leads to fewer strange values but still some negative values (due to the fact that there is no parking regime) and some values still seem very high (but still within possible limits). Therefore, we keep this network but replace all negative values with zeros. Further in this research, we denote this forecast as the Neural Network A forecast. Also, we create a second version, which is denoted as the Neural Network B forecast. This forecast is the same except that the PCN ratios of the unknown neighborhoods are replaced by the PCN ratio of their neighbors.

4.3.3 Prediction of the occupancy ratio

In Section 2.1.8, we defined the occupancy ratio as the number of scanned vehicles divided by the number of parking spots. With this ratio and the number of parking spots in a neighborhood, we can predict the number of scans in a neighborhood. As explained in Section 4.1, due to the problem that in the past the neighborhoods have not been scanned entirely, it is difficult to make predictions about the occupancy ratio. If every neighborhood was visited entirely, we could divide the number of scans of a visit by the number of parking spots to determine the occupancy ratio. Since this is not possible, we use an approximation.

In the past, ARS has created a data set of all visits from 2016. From this data, we only consider those visits, where the number of scans is more than 70% of the parking spaces of the neighborhood. By doing so, we strive to exclude the visits when the neighborhood was not scanned fully. However, this approach is biased because it could be that sometimes the occupancy ratio is actually below 70%. This would lead to an overestimation of the occupancy ratio. Since ARS assumes in their algorithm that all neighborhoods have an occupancy ratio of 80% and Amsterdam is a well-known crowded city, we choose the threshold of 70%. The average occupancy ratio and standard deviation for all KPI areas based on the remaining visits are shown in Table 7. The standard deviation will be used later in Chapter 6.

KPI area	Average Occupancy ratio	Standard deviation of Occupancy ratio
Centrum	84.5%	1.60%
Nieuw-West	84.9%	1.81%
Oost-1	78.3%	0.86%
Oost-2	79.5%	1.04%
West-1	81.7%	1.03%
West-2	81.7%	1.14%
Zuid-1	81.4%	1.06%
Zuid-2	83.4%	1.15%
Noord	82.7%	1.33%
Zuidoost	82.7%	1.33%

Table 7 – Occupancy ratio of all KPI areas

In this table, we replaced the values for “Noord” and “Zuidoost” by the average ratio and standard deviation of the whole data set because at first they had illogically high values due to much fewer scans. Unlike the PCN ratio, we do not separate between different time intervals because this approach is only a rough estimation and therefore it does not make sense to make it very precise. We see that these values are close to the estimation of ARS as they have used 80% for all KPI areas. Since we assume that there are differences between the KPI areas and that our approach is more accurate than the one of ARS, we choose to use the results from Table 7 as the occupancy ratio.

4.3.4 Parking duration of non-paying visitor

This section tackles the question how long the non-paying parking visitors stay in a neighborhood. This is an essential question in order to deal with the TOP with multiple visits. As explained in Section 2.1.3, visitors can only receive a PCN once a day. If every non-paying visitor stayed at a neighborhood until the end of the day, it would never make sense to return to a neighborhood the same day. Since this is not the case, let us consider a more likely example:

We predict that in a certain neighborhood on a given day there are 6 non-paying visitors, respectively PCNs, at 9.00 and 4 at 11.00. Based on this information, we would expect 10 PCNs by visiting the neighborhood at 9.00 and 11.00. However, in reality it could happen that 3 of the 4 non-paying visitors at 11.00 have already been scanned at 9.00. Since they cannot receive a second PCN, the expected number of PCNs at 11.00 would be 1 (and 7 of both visits). In this section, we explain our approach how to deal with this problem.

It is logical that there is no data about the parking duration of non-paying visitors because visitors who do not pay are not registered. On the other hand, in theory, there is data about the parking duration of visitors that do pay. Next to the fact that the paid-for parking hours are not the same as actual parking hours, it is very difficult to make predictions about the relationship between the parking duration of paying and non-paying visitors. One might argue that non-paying visitors stay shorter because they do not want to take the risk of getting a fine. On the other hand, some non-paying visitors probably do stay all day because they estimate that the fine and the parking costs of a day a roughly the same. Anyhow, the data of the paying visitors is saved in the PRDB (see Section 1.1.2), which is, unfortunately, not accessible to us. Consequently, we have no data and therefore we develop a method, which we explain by using the example from before.

First, we compute the number of non-paying visitor that overlap according to our prediction. In this example, there are 6 visitors at 09.00 and 4 visitors at 11.00. The overlap in visitors between the two hours is 4. This gives us an estimate of the number of non-paying visitors that potentially remain in the neighborhood for this duration of 2 hours. However, we do not assume that 100% of these potential visitors remain in this neighborhood. Therefore, we apply a percentage that is based on the time difference between the two visits Δ_t, which is the difference of the finish time of the previous visit and the start time of the new visit. The finish time is chosen because it gives us an indication when the neighborhood was seen for the last time and we do not know in which sequence the streets within the neighborhood are driven. If we would know that the sequence was always the same, then we would consider the start time of the previous visit. Finally, our stability function is the following:

$$S(\Delta_t) = (Stability\ Parameter)^{\Delta_t}.$$

By performing experiments, we chose the Stability Parameter to be equal to 0.4. We explain the design of this function and the choice of the Stability Parameter by means of Figure 34, which shows the percentages of the remaining visitors and new visitors for Δ_t between 0 and 5.

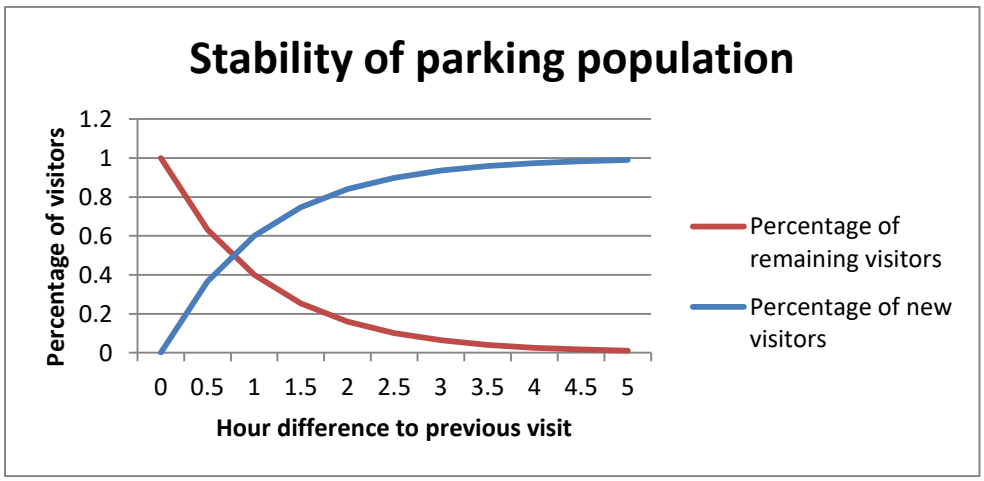


Figure 34 – Stability of the parking population for 5 hours with a Stability Parameter equal to 0.4.

With regards to Figure 34, it shows the effect that we aimed for. At the beginning the percentage of remaining visitors if high such that it is not worth to visit the neighborhood again but after 2 hours the neighborhood becomes more attractive again. This function was confirmed by the expert opinion of ARS.

Since the percentage of remaining visitors' decreases exponentially over time, this means that the time that these visitors arrive and leave actually depend on the last visit. We make the assumption here that the visitors just arrived when the neighborhood was visited previously. Even though this is obviously not true, we make this assumption in order to prevent that the same neighborhood is visited shortly afterwards.

Whenever data about the parking duration of the non-paying visitors (or at least paying visitors) is available, this function can be derived by fitting the duration hours to a certain distribution (e.g. Normal distribution). Since the probability that the visitors are still in the neighborhood at a certain time t can be expressed as $P(X \geq t)$, this probability can computed by means of the cumulative function: $1 - F_X(t)$.

4.4 Travel times

This section explains our approach to determine the travel times between the neighborhoods and break locations, which can be used in the routing algorithm.

As discussed in Section 2.1.4, we have 2 different types of GPS positions of the neighborhoods: the edge points of the neighborhood in shape of a polygon and the center point, which is the center point of the locations of all scans. To compute the travel times between 2 neighborhoods we have contemplated using the closest edge points of the neighborhoods as start/end points. This approach is unfeasible because the edge points may lie in areas that are not accessible by car, such as parks or water. Using such positions in automated route finding would likely result in errors or inefficient routes that do not actually start from a neighborhood edge. This problem almost never appears when using the center points.

This approach does have a downside that there is an overlap with the service time: the travel time includes per definition only the time between the two assigned neighborhoods and our approach includes also the time from the center point to the edge. And this occurs at both ends of the automatically calculated route. The length of this redundant distance depends on the size of the neighborhood and the streets that the PEV driver chooses and therefore it is not possible to compute exactly how big this redundant distance is.

After experimentation with the routing algorithm using this approach, it was found that a valid approximation of the redundant distance depends on too many variables. Also, it was found that actual travel times vary quite a bit. This, in combination with the usability aspect that a planning should be achievable, we decide to keep the redundant distance. This creates reasonable "upper bounds" (w.r.t. travel time) in the planning, so that the planning has extra buffer for unexpected delays, and the enforcement team is not regularly confronted with a shift in which they could not achieve the planned results.

As mentioned in Section 3.3.1, it is important to account for traffic congestion in order to have more reliable travel times but also to avoid traffic congestion. By accounting for rush hours, we strive to have shorter travel distances during the rush hours. Furthermore, the travel routes must not include highways because the PEF drivers are not allowed to enter these with the scooter. This has two consequences. It can happen that the PEF driver cannot do the follow-up work for all cars that he/she is assigned to or that the cars are already gone by the time he/she gets there. Another benefit of not taking highways is that by driving through the city, the PEVs can also scan cars while travelling. Because of limited time, we do not further investigate how many PCNs can be expected for all the travelling routes, however, it is a nice bonus to have. Moreover, we prefer to have directed speeds, as in the city of Amsterdam it does matter if the PEVs are driving from the depot to "Centrum" or the other way around.

The travel times are calculated from/to the center points (see Section 2.1.4) of each neighborhood and from/to the break locations. We have used the "HERE" database. For every 2-neighbourhood relation there are four different travel times: a journey in each direction (A->B and B->A), and for both journeys one during and one outside the rush hours. According to ANWB (2017), the time between 6.30-9.30 and 15.30-19.00 are considered to be rush hours in Amsterdam.

As stated in the introduction of this chapter, during the routing algorithm we apply interpolation when a journey overlaps with more than one time period. For example, the PEV starts driving during the rush hour at 9.20 and the travel time takes longer than 10 minutes. That means that the journey is within and outside the rush hour. For this example, we would compute the travel time by means of interpolation:

$$Travel\ time\ =\ 10\ min\ + Travel\ time\ (no\ rush\ hour) * \frac{Travel\ time\ (rush\ hour) - 10\ min}{Travel\ time\ (rush\ hour)}$$

4.5 Service times

The service time is the time that the PEV driver needs to scan one assigned neighborhood. The service time starts when the driver enters the neighborhood and ends when the driver leaves it. The entry and exit point depend on the neighborhood that was visited before, respectively afterwards and on the route that is driven inside the neighborhood. Since we do not take the street level into account, we do not determine where the entry and exit point are. As explained in Section 4.1, ARS estimated the service times by extrapolating the historical data based on the number of scans. We use a different approach that does not include extrapolating due to the reasons we discussed in Section 4.1. Our approach is based on the length of streets that the PEV drives through when visiting a neighborhood and its average speed during and outside the rush hours.

In order to compute the average speed that the PEV drives through a neighborhood within and outside the rush hours, we assume that this speed is equal to the travel time of that the PEV needs to travel from this neighborhood to the nearest neighborhood within and outside the rush hours. Therefore, we derive the speeds from the travel speeds that we calculate in Section 4.4. From the open location platform "HERE", we retrieve the total length of all streets within a neighborhood. With this information, the distance that a car needs to drive through every street of the neighborhood exactly once can be computed. However, the actual route within a neighborhood is probably more than that, as it is highly unlikely that the PEV goes through every street exactly once. In order to do so, the driver would need to solve the Chinese Postman Problem, which we discussed in Section 3.1.1. And even then, some streets needs to be driven twice if it is a dead end or the streets are so wide that the PEV cannot scan both sides at the same time. Consequently, we assume that this fact increases the actual driven distance. However, not all streets in a neighborhood contain fiscal parking spots and therefore it might not be necessary to drive through all streets. In order to account for both affects, we introduce an adjustment factor that gives an indication to what extent all streets of the neighborhood are driven.

Finally, the service time can be computed by means of the computed speeds and the length of streets within a neighborhood.

4.6 Conclusion

In this chapter, we have computed and estimated the different inputs that are required for the routing algorithm in Chapter 5, such as the margin of error for the KPI targets, the expected number of PCNs, the service times, and the travel times. We have introduced the stability function, which helps the algorithm to deal with expected number of PCNs when a neighborhood is scanned multiple times a day. Furthermore, we had to overcome different challenges of the scan data and we developed a prediction model for the number of PCNs that is based on the visitor ratio, non-paying ratio, and occupancy ratio of neighborhoods at different times. We learned from the data analysis that there are three clusters regarding the weekdays and six clusters regarding the hours of the day. Analyzing the weeks, we found no obvious seasonal pattern; however, we did see that the payment rate and the effort in terms of the number of scans and number of PCNs are increasing every week. We presented two prediction models for the PCN ratio of a visit. One approach is relatively straightforward and the other one is based on a neural network. We have shown that both approaches are more accurate than the current prediction model of ARS. However, in Chapter 6, we will show which prediction model works the best in combination with the routing algorithm. Moreover, we will use the travel time and number of scans that we computed in this chapter. We will not use our service times of the neighborhoods because we need to determine the adjustment factor to get an accurate input. The good thing about this adjustment factor is that it can be adjusted manually for every neighborhood. If in practice it seems that only half of the service time is needed, the adjustment factor can simply be reduced by 50%. However, it is very difficult to find a good value for this factor because every neighborhood is different and also the driving behavior of every PEV driver. This determination requires testing over a longer time period, which lies out of the scope of this thesis. Therefore, the determination of adjustment factor can be studied in the future and we use the duration computed by ARS for our routing algorithm. Next to that, also the stability function and the margin of error that we used for the paid-for visitor hours, and the occupancy ratio should be further studied in the future. Even though the occupancy ratio is derived from that, it should be replaced on the long-term by a more accurate approach.

5 Routing algorithm

In this chapter, we design the routing algorithm that creates the routes for the PEVs. Section 5.1 discusses the planning horizon of the algorithm, Section 5.2 explains the notation that is used throughout this chapter, Section 5.3 deals with the objective function of the algorithm in order to evaluate different solutions, and Section 5.4 describes how the algorithm works.

5.1 Planning horizon

Before choosing the best algorithm, we need to determine the planning horizon of this problem. Currently EPS always makes a planning for the next day with regards to the targets they have to reach until the end of the KPI period, which consists of 90 days. Therefore, it would be possible to create a planning that works in the same manner. However, EPS requires also an assurance whether the targets will be met at the end of the KPI period. By estimating whether the targets will be met, EPS can adjust their capacity accordingly. Whether our planning tool includes this estimation of the KPI targets depends on the planning horizon. We can think of three options for the planning horizon of our planning tool.

Option 1 creates a planning for the whole KPI period before the KPI period starts. That means that the computation time is not an issue since it happens before the KPI period has even started. This option also enables to estimate whether the KPI targets are met at the end of the three months. Unfortunately, this option has a big disadvantage due to the fact that recent data is not taken into account. Consequently, it can happen that different the actual payment rate and the actual number of PCNs are lower than expected but it would not be possible to adjust the planning.

Option 2 makes a new planning every day until the end of the KPI period. It is similar to the rolling horizon planning, where every day a new planning is created for a fixed number of days. However, the objective is not to always create routes for 90 days but only until the end of the KPI period. Meaning that at the first day it plans for 90 days and at day 40 it plans for the remaining 50 days. This brings the advantage that it is also sensitive to recent changes, plus it also enables a KPI indication. On the downside, the computation times are high as the maximum planning horizon is 90 days and the planning is done on a daily basis.

Option 3 creates a new planning every day based on recent data. The computation time would be relatively low as the planning is done daily for only one day. Due to the limited planning horizon, it is not possible to give an accurate estimation of the KPI results over the entire KPI period but it would be possible to make a poor estimation based on extrapolating historic averages.

An overview of these options is given in Table 8:

Option	**Includes recent data and changes**	**Enables KPI target estimation**	**Computation time**
Option 1: Planning for all days before KPI period	No	Yes	None (because computed before)
Option 2: Daily planning until the end of KPI period	Yes	Yes	Long
Option 3: Daily planning for a day	Yes	Poor	Short

Table 8 – Planning options

Every option has a disadvantage. For our approach, we choose option 2 because having an estimation of the KPI targets in combination with including recent data and changes is the benefit that EPS requires as it helps them in their long-term planning. However, we have to limit the computation time to 6 hours. Option 1 also enables the estimation of the KPI targets but it is not that accurate as it is not based on recent data.

5.2 Notation

This section explains the notation that we use throughout this chapter to describe the routing algorithm. Note that we consider the entire time span of one day to be from 9.00, which is equivalent to 9 am of the scheduled day, until 28.00, which is equivalent to 4 am of the following day. The routing algorithm

provides an enforcement planning that schedules visits of neighborhoods, i.e., that a neighborhood is scanned at a certain time.

Every visit is defined by the number of the neighborhood j (1…J), the number of the KPI area a (1…A), the day of the planning period n (1…N), the weekday d (1…7), the starting time t (between 9.00 am and 28.00), the finish time t_f (between 9.00 and 28.00), the week number w (1…W), the year y (1…Y), and the number of the PEV m (1…M). Therefore, every visit is denoted as $v_{j,a,n,d,t,t_f,w,y,m}$. For the purpose of simplicity, we denote the visits in this chapter as $v_{j,n,t,m}$ because it is possible to derive the KPI area a from neighborhood j and the weekday d, the week number w, and the year y from the planned day n, and the finish time t_f from the start time t. We denote this visit also as current visit because later in this chapter we also need information about the *prior visit* and the *sequential visit*. The prior visit describes the last time that neighborhood j was visited before the current visit $v_{j,n,t,m}$. The neighborhood j has to be the same in this case but the vehicle (or PEV) m and planned day n not necessarily. Likewise, the sequential visit describes the next visit of neighborhood j after the current visit. We use an additional variable x in order to denote whether we consider the current (x=0), prior (x=-1), or sequential visit (x=1). Consequently, we denote every visit as $v_{j,n,t,m,x}$. We provide an overview of this notation in Table 9.

Index	Definition	Range
j	number of the neighborhood	1…J
a	KPI area	1…A
n	day of the planning period	1…N
d	weekday	1…7
t	starting time	between 9.00 and 28.00
t_f	finish time	between 9.00 and 28.00
w	week number	1…W
y	year	1…Y
m	number of the PEV	1…M
x	visit information	-1,0,1

Table 9 – Notation table of the indices of a visit

Whenever we require the value of a certain index of a visit, we do this by putting $v_{j,n,t,m,x}$ in its index. For instance, the finish time of the visit of neighborhood 15 by PEV 3 on day 4 at 12.00 can be expressed as $t_f\, v_{15,4,12.00,3,0}$. The finish time of the visit that is prior to that one can be described as $t_f\, v_{15,4,12.00,m,-1}$. Let us say that the prior visit started at 9.00 and was done by PEV 1, then we can denote the finish time also as $t_f\, v_{15,4,9.00,1,0}$ (the prior visit can also be denoted as a current visit).

5.3 Objective function

In this section, we introduce our objective function that determines the added value of visiting a neighborhood. This objective function only considers current visits because a prior or sequential visit is always a current visit just at a different time.

In this section, we use the terms payment rate performance, PCN performance, payment rate target, and PCN target. As explained in Section 2.1.8, the problem of the KPI targets is that they are to some extent uncertain. To cope with this uncertainty, we calculated upper bounds of KPI targets in Section 4.2. By comparing the EPS' measurements of the payment rate and number of PCNs to these upper bounds, we determine the payment rate performance and PCN performance. Regarding the performance, it is important to remember (as explained in Section 2.1.8) that a KPI area can have a malus, neutral, or bonus status. This is important because of the priorities that need to be considered when choosing a neighborhood:

1. Meet either the payment rate target or the PCN target (respectively the control chance) of every KPI area (bring all KPI areas at least to a neutral status).
2. Maximize the PCN target in chosen KPI areas in order to eventually increase the payment rate and maximize the performance bonus.
3. Visit every neighborhood once a week.

As mentioned in Section 2.1.8, this priority list shows only the importance of these objectives from EPS' objective. This does not mean that first our only objective is priority number one and after this objective is met, the second priority is the new objective, and after that finally the third. We rather want an algorithm that incorporates all priorities at any time such that the priorities are met in a balanced manner even though the focus should still lie on the first priorities. Therefore, we explain our approach of quantifying this priority list in this section.

An intuitive approach would be to compute the amount of euro that is gained by visiting a neighborhood at a certain time. Let us consider three cases where the visited neighborhood lies in a KPI area with a malus, neutral, and bonus status.

Malus status:

Since both targets are not met in a KPI, EPS would receive a fine based on the difference to the PCN target. If a neighborhood within that KPI area is visited, the number of PCNs will increase and therefore the fine will decrease. For that reason, it is possible to compute the reward of a visit depending on how much the fine is decreased.

Bonus status:

In case that it has a bonus status, we could calculate the improvement of the bonus that occurs by visiting the neighborhood. However, we concluded in Section 4.3.1 that we cannot quantify the relationship between visits (number of scans) and the payment rate. Therefore, we would have to make an assumption about this relationship. However, even then, a bonus is only given if no KPI area is in a malus status, i.e., the reward is likely to be 0 for the longest time of the KPI period.

Neutral status:

There are two possibilities that a KPI area has a neutral status. The first case occurs when the number of PCNs is below the target and the payment rate is exactly equal to the target. In this rare case, the amount of euro that is gained is computed as for KPI areas with a bonus status. The second and more common case is that the PCN target is met and the payment rate is not. In this case, the reward would be 0 until the payment rate exceeds the target. However, for this purpose we require again the impact of a visit on the payment rate.

We decide not to use an approach based on the reward (positive or negative) in euro because we lack the information about the impact of a visit on the payment rate and therefore cannot compute the euro value for a visit within a bonus KPI area. Even if it would be possible, the value would be probably much lower than the value of visiting a neighborhood of a malus KPI area. Moreover, the reward for visiting a neighborhood within a neutral status would be 0 until the payment target is reached. For that reason, it is likely that neighborhoods of such a KPI area are not visited for a long time. Additionally, the third priority that a neighborhood should be visited once a week is ignored. This leads additionally to an unbalance of visited neighborhoods within the KPI areas.

Therefore, we developed an alternative approach that accounts for the three priorities by computing a priority score. The idea is to attach a weight on the expected number of PCNs of a visit. This weight incorporates the three priorities by using two factors. One factor is called the Target Factor, which includes the first two priorities and is further explained in Section 5.3.1. The other factor is called the Visit Day Factor, which incorporates priority three and is explained in Section 5.3.2. The formula of the priority score is explained in Section 5.3.3.

5.3.1 Target Factor

This section introduces the Target Factor, which is very important as it represents the need of visiting a KPI area. Every neighborhood belongs to a KPI area that has certain targets. If the targets of the KPI area are met (neutral or bonus status), the Target Factor will be between 0 and 1 in order to make this neighborhood less attractive. Whenever the target of this KPI area is not met (malus status), the Target Factor will be between 1 and 2 such that the neighborhoods within that KPI area become more attractive. We limit the factor to 2 because even though the neighborhood where the number of PCNs is achieved is important, it should not overrule the fact that the actual number of PCNs is still the most important thing.

The Target Factor is based on the current number of PCNs and the current payment rate. By dividing the current number of PCNs with the upper bound of the PCN target (see Section 4.2), we compute the PCN performance. For instance, if the upper bound of the PCN target is 1000 and the number of PCNs is 600, then the PCN performance would be 60%. Whenever a neighborhood is visited, the current number of PCNs is updated, such that the Target Factor takes the new PCN performance into account. We refer to this as updating the KPI matrix. We compute the payment rate performance the same as we do for the PCN performance. An historic example of the performances of both KPIs for all KPI areas can be seen in Figure 35.

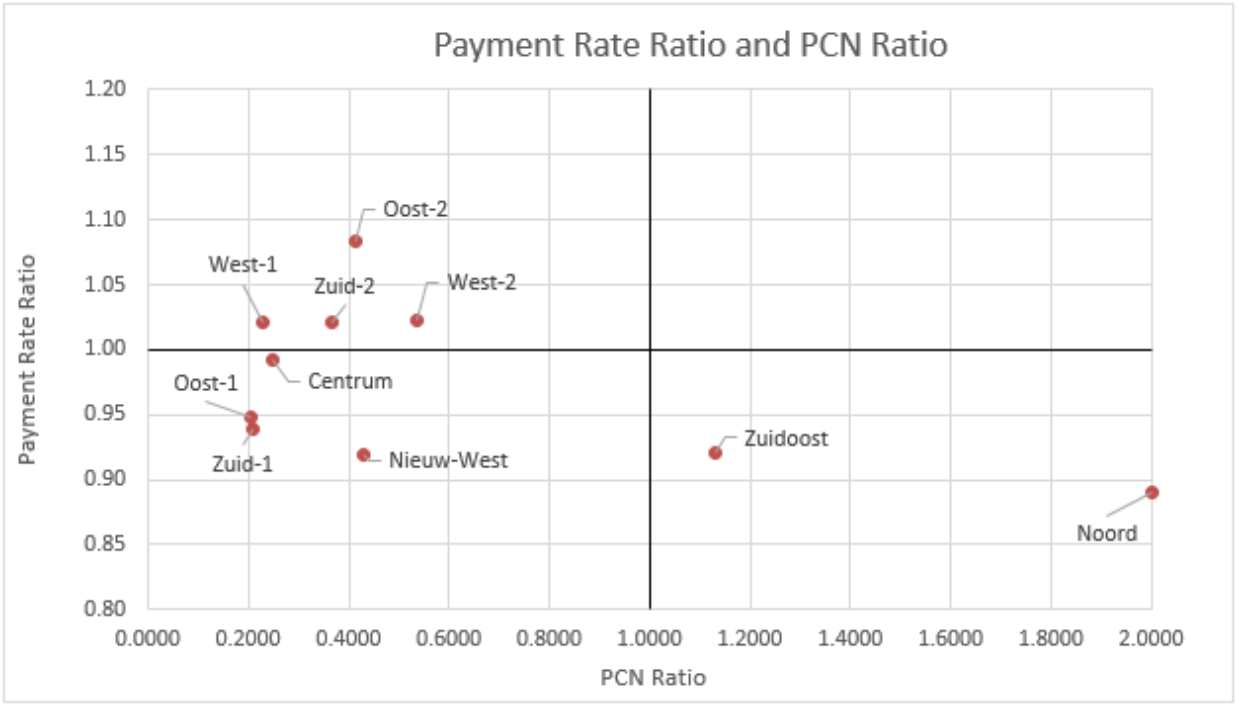


Figure 35 – Visualization of the KPI targets

Our objective is to increase the number of PCNs such that the payment rate increases eventually in the future. With regards to Figure 35 that means that the performance dots of the KPI areas first move to the right and then upwards. The payment rate performance has a usually a value around 1 because the current payment rate is in most cases 1 or 2% smaller or higher than the target. The PCN performance is quite different as it can be 0 (e.g. 0/20000) or above 1 if the target is already reached but the KPI area is still being visited since EPS wants to visit all KPI areas regularly. For small KPI areas such as "Noord" it can happen that the number of PCNs is even higher than twice the target. The Target Factor includes to what extent the target is exceeded. As one can see in this example, we limit the ratios to 2 because we decide that it does not matter if, for instance, the PCN performance is 2.1 or 2.2. Plus by using a maximum value of 2, the range of not-meeting the targets (0, 1) and meeting the targets (1, 2) have the same size, which is useful for our Target Factor computation. From Figure 35, we can derive four groups of KPI areas: those who have reached both targets, no targets, and one of the targets. These groups can be seen in Table 10:

Group	Description	Examples from Figure 35
1	Neither of the targets is met	Oost-1, Zuid-1, Nieuw-West, Centrum
2a	Only PCN target is met	Zuidoost, Noord
2b	Only Payment rate target is met	West-1, Zuid-2, West-2, Oost-2
2c	Both targets are met	None

Table 10 – KPI areas divided into priority groups

As mentioned before, the Target Factor of the KPI areas that have not met their targets (group 1) get a value above 1 and those that did (group 2a, 2b, and 2c) get a value below 1. Consequently, the objective function of the visits either decreases or increases, which makes the visits in these neighborhoods more or less attractive. We compute the Target Factor of group 1 and group 2 (a, b, and c) differently:

No target is met (group 1)

If both targets are not met, we only consider the PCN performance in order to compute the Target Factor because that is the performance that can be influenced directly. So again, by focusing on the PCN performance, the performance dot of the KPI area (see Figure 35) moves more to the right (and eventually

upwards). Since the PCN performance varies between 0 and 2, we compute the Target Factor by subtracting the PCN performance from 2:

$Target\ Factor\ =\ 2-\ PCN_{ratio}$.

At least one target met (group 2a, 2b, and 2c)

For those KPI areas that did not meet one of the targets, we want to include both targets for two reasons. Additionality to the upper bound that we use for both targets (see Section 4.2), an increase of the PCN performance increases the probability that the PCN target is reached in case the payment rate measurement of the municipality is even lower than expected in the worst case. We also want to increase the payment rate performance because this performance is the essential one for achieving a bonus. A combination of both targets could be done by a multiplication, such as:

$$Target\ Factor\ =\ (2-PCN_{ratio})\ *\ (2-PaymentRate_{ratio}).$$

However, this would mean that we multiply a number between 0 and 1 with a number between 0 and 2 because at least one target is met. Consequently, the product could be bigger than 1 (e.g.: 1.80*0.95 = 1.71), which should not happen. Therefore, we divide this number by a Target Reached Parameter, which needs to be at least 2, in order to make sure that the Target Factor is below 1. This Target Reached Parameter can also be bigger than 2 in order to make these KPI areas even less attractive in comparison with group 1. During our experiments, we found that 2 is a good value for the Target Reached Parameter, which we also discuss further in Section 6.2.1.4. Finally, we compute the Target Factor for these KPI areas, as follows:

$$Target\ Factor = \frac{(2-PCN_{ratio})*(2-PaymentRate_{ratio})}{\text{(Target Reached Parameter)}}.$$

Let us explain in the following why this formula shows the desired effect for all of the groups when using a Target Reached Parameter equal to 2. Take into account that the payment rate performance varies usually only between 0.9 and 1.1 whereas the PCN performance usually varies between 0 and 1.2.

Group 4 (payment rate performance ≥ 1 & PCN performance ≥ 1) is the least important group as already both targets are met. Therefore, we multiply two values between 0 and 1 and divide it by 2, hence the number is between 0 and 0.5.

Group 2 (payment rate performance < 1 & PCN performance ≥ 1) is less important than group 3 (payment rate performance ≥ 1 & PCN performance < 1) because both reached the target but only group 3 has the possibility to get a performance bonus. Since usually the PCN performance has a lower value (if the PCN target is not met) than the payment rate performance (if the payment rate target is not met), the Target Factor for group 3 is automatically higher.

The Target Factors for example of Figure 35 with a Target Reached Parameter equal to 2 are shown in Table 11.

KPI Area	Target Factor
Oost-1	1.794
Zuid-1	1.792
Centrum	1.750
Nieuw-West	1.571
West-1	0.866
Zuid-2	0.799
Oost-2	0.727
West-2	0.714
Zuidoost	0.468
Noord	0

Table 11 – Targets Factors of all KPI areas

As one can see "Noord" has a Target Factor of 0 in this example. As we explain later in Section 5.4.2, this means that all neighborhoods that belong to "Noord" would not be visited anymore. If the Target Factor is indeed 0, we do not see the need that neighborhoods this KPI area still have to be visited. However, in some cases the algorithm has no feasible options to go to, which is also caused by excluding KPI areas with a Target Factor of 0. For that reason, we always replace a Target Factor of 0 with a very little number, namely 0.001, in order to prevent these edge cases.

5.3.2 Visit Day Factor

The Visit Day Factor is a factor that we introduce in order to present the desire of EPS to visit every neighborhood at least once a week. As for the Target Factor, we do not want the Visit Day Factor to exceed 2 because the number of PCNs should still be the most important thing. Unlike the Target Factor, the Visit Day Factor only varies between 1 and 2. The rationale behind it is that the neighborhood will be more attractive if it has not been visited for a longer period but not less attractive if it has been visited recently, for example on the same day. EPS has no restriction policy about how many times a neighborhood may be visited on a day. Consequently, the Visit Day Factor $V_{v_{j,n,t,m,0}}$ depends on the planned day of the current visit of neighborhood j, which (as explained in Section 5.2) is defined as $n_{v_{j,n,t,m,0}}$, and the previous one $n_{v_{j,n,t,m,-1}}$. Note that the PEV of the last visit $m_{v_{j,n,t,m,-1}}$ and the current visit $m_{v_{j,n,t,m,0}}$ do not necessarily have to be the same. The Visit Day Factor measures the attractiveness of the neighborhood in terms of day difference those two visits: $\Delta_n = n_{v_{j,n,t,m,0}} - n_{v_{j,n,t,m,-1}}$, which must be a positive integer. The Visit Day Factor is 1 if the neighborhood has been visited already on the same day. Then, the Visit Day Factor increases exponentially because it is still fine if the neighborhood has not been visited in 2 or 3 days. However, when the difference approaches 7 days, the Visit Day Factor should increase faster. At day 7, the visit day reaches a value of 2 and then stops because we do not want that it overrules all other aspects and in the end it is a only soft constraint and does not have the first priority. The Visit Day Factor is determined by the following function $f(\Delta_n)$:

$$f(\Delta_n) = \min\left(2, 1 + \left(\frac{\Delta_n}{7}\right)^{(Visit\ Day\ Parameter)}\right),$$

where the Visit Day Parameter is a parameter that determines the growth of the exponential function. We have chosen a Visit Day Parameter of 5, to realize the desired effect of our function. The outcome of this function for 0 until 9 days without a visit is shown in Figure 36.

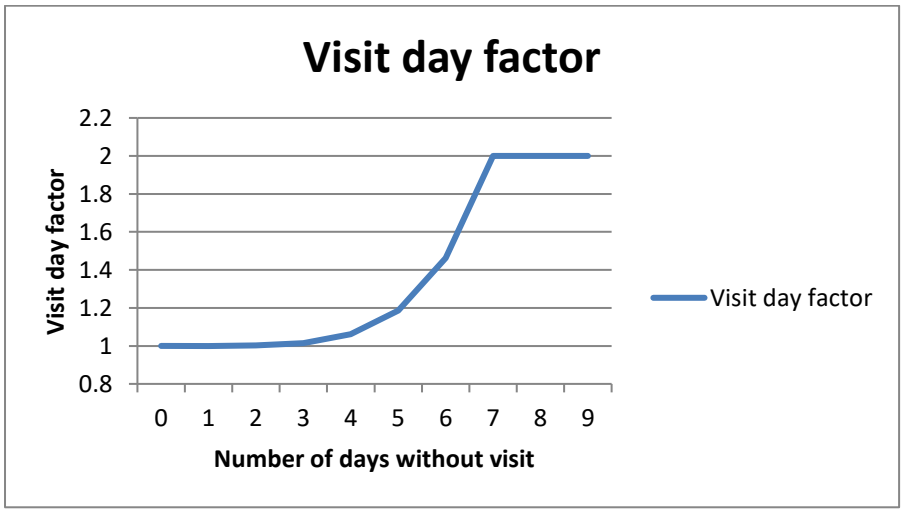


Figure 36 – Visit day function with a Visit Day Parameter equal to 5

For our analysis, we count the average number of times that a neighborhood is not visited once a week. That means that this count starts, whenever a neighborhood is not visit after one week and is increases by one if the same neighborhood is neither visited the next day the count increases further.

Apart from decreasing the number of average times that a neighborhood is not visited once a week, the Visit Day Factor also increases the variability of the route. This is beneficial for EPS because otherwise visitors might observe a pattern if some routes would be the same.

5.3.3 Priority score

Finally, we compute the priority score by multiplying the number of expected PCNs, which is denoted as $P'_{v_{j,n,t,m,0}}$ (see Section 5.4.2), by the Target Factor and Visit Day Factor. Therefore, the objective function for all days and vehicles is expressed as:

$$max \sum_{n=1}^{N} \sum_{m=1}^{M} P'_{v_{j,n,t,m,0}} * T_{v_{j,n,t,m,0}} * V_{v_{j,n,t,m,0}} .$$

This priority score will be essential for our routing algorithm.

5.4 Routing algorithm

We choose to construct the route planning by means of an ant colonization optimization algorithm (ACO) because the benchmark of Ke et al. (2007) (see Appendix B) shows that for the team orienteering problem the ACO algorithm shows similar or better results in terms of the objective function in comparison with other algorithms but with a better computational time. Additionality, the ACO algorithm is a constructive metaheuristic that strives to find a global optima. This is convenient for us, since local search improvement techniques are difficult to apply to our problem and without applying a metaheuristic, the algorithm would get stuck in local optima. An example of a local search technique is simulated annealing (SA) (see Section 3.2.1). SA looks for a neighborhood solution of a constructed solution. For instance, a neighborhood solution can be built by exchanging one visit with a new or already scheduled visit (also denoted as swap). In the following, we explain why such a swap is difficult to implement:

We have constructed a planning for one day for 12 PEVs. Now, we want to change the neighborhood of a certain visit of PEV 6 and let him go to another neighborhood. In that case, we have to follow these steps:

1. Recompute the travel times, service times, and expected number of PCNs for all visits after this one due to the fact that all starting times change.
2. Check whether no restrictions are violated:
    a. a neighborhood must not be visited at the same time
    b. the return to the depot, the breaks, and the shift change must happen within a certain time window
    c. a neighborhood must not be visited, if there is no parking regime
3. Recompute the expected number of PCNs of all visits that go to the same neighborhoods as those that were adjusted neighborhoods in step 1. This is due to the fact that the start and finish times of the visits change and the calculation of the remaining visitors, as we explain in detail in Section 5.4.2, is based on the finish times of prior visits.
4. Do the same as step 3 for the visits that also go to the newly added neighborhood.
5. Compute the total change of the number of PCNs and update the KPI matrix with the current number of PCNs of each KPI area (see Section 5.3.1).

Compute the change of new priority score and in case that there is no improvement, swap back.

This shows that there is a lot of recomputation and feasibility checking for considering a swap. That means that methods like simulated annealing would be possible; however, they would be very time consuming. For that reason, we choose an ACO algorithm, because, as discussed in Section 3.2.2.2, it finds one best route by constructing many possible routes and learning iteratively which routes between neighborhoods (denoted as arcs), performed well in the past. Even though we do not apply a metaheuristic improvement technique, we do apply a 2-Opt swap, similar to Verbeeck et al. (2014a) in order to prevent unnecessary zigzag routes, which probably seem illogical or inefficient to the PEV drivers. Unlike Verbeeck et al. (2014a), we apply this 2-Opt swap while constructing a solution whenever a neighborhood is scheduled. The fact that it happens during the construction phase reduces the impact on other visits and therefore the recomputation time. This 2-Opt swap exchanges the two previously scheduled visits of the current PEV, if the saved travel time is above a certain Swap Threshold. If due to the swap the reduction of the travel time in percentages is lower than the decrease of PCNs or if the time after the swap is bigger than before, we will swap back the visits. The rationale behind this is it is essential that the swap actually saves time and that we enough time is saved to earn the loss of PCNs back. We choose 1 minute for our Swap Threshold because it seems as a reasonable threshold that has to be exceeded. A lower threshold would lead to more unsuccessful swaps (visits are swapped back afterwards) and would increase the computation time. Another swap that we apply during the construction

of a solution is our break location swap. Because it is not known which neighborhood the PEV visits after going to a break location, it can happen that not the best break location is chosen in terms of travel distance. Therefore, the break location is swapped afterwards with another break location, if the travel time decreases. The benefits of these swaps are further explained in Chapter 6.

Figure 37 shows an overview of the general concept that explains how our algorithm works. We denote a day planning as one planning with all routes of all vehicles for one planned day, where M is the total number of vehicles. The algorithm creates such a day planning for every day, starting from day n until a certain day N. For every day, our algorithm performs many iterations, where the number of iterations I is a pre-determined parameter. One iteration contains one day planning. Every planned day starts by first setting the pheromone values of all arcs to a pre-determined initial value. Afterwards, I iterations are performed. At the end of each iteration, the pheromones of the arcs will be updated to create a learning effect. Since this update is a core process of our algorithm, it is highlighted in Figure 37 and will be further discussed in Section 4.3.3. For every day, we keep the iteration that led to the best result. We send our ants, which are equal to the vehicles, sequentially as proposed in the algorithm of Montemanni and Gambardella (2009) and Ke et al. (2007). Ke et al. (2007) also tried different methods but the sequential approach seemed to perform the best.

For every day in range (n,N):
- → Replace pheromone values by pre-determined initial pheromone values
- For every iteration (1, I):
  - For every ant (1,M):
    - → Create a route planning
  - → **Update Pheromones**
  - → save iteration if it is the best one

Figure 37 – General concept of our ACO routing algorithm

Before going into detail on how the route of every ant is build, we denote the following terms:

- BreakTime: is the starting time of the next break
- BreakFlexibility: is the tolerance of being early or late, which is at the moment 15 minutes
- DayStartTime: the time at which the day starts and the ant leaves the depot
- DayFinishTime: the time at which the day finishes and the ant has to return to the depot
- Neighborhood (j): the next assigned neighborhood
- TravelTimeToNeighborhood: the travel time from the current neighborhood to the new neighborhood
- TravelTimeToBreakLocation: the travel time to go back to the nearest break location
- TravelTimeToDepot: the travel time from the current neighborhood to the depot
- CurrentTime: the time of the decision
- ServiceTime: the time that is needed to scan a neighborhood
- ShiftChange: the time when the drivers switch the shift and therefore the PEV switches the driver
- Cycle: the time period between two BreakTimes, the DayStartTime and the BreakTime of the first break, or the BreakTime of the last break and the DayFinishTime
- CycleStartTime: start time of the current Cycle
- CycleFinishTime: finish time of the current Cycle

The routing algorithm takes into account all time restrictions as explained in Section 2.1.5, such as break times, start and finish time of the day, and the shift change. Therefore, the time needs to be updated after every step. As explained in Section 5.3.1, it is also required to update the KPI matrix after a visit. Figure 38 shows the route planning of an ant. This route planning is performed differently in the first iteration. We choose to build a greedy solution in the first iteration and use the ACO algorithm afterwards. In Figure 38, there are two more core decisions of our algorithm that are highlighted in Figure 38 that work differently in the first iteration. We discuss these further in Section 5.4.1 and Section 5.4.2.

For every ant:
Start
Depending on the time, the ant chooses one of the following options:
If (ShiftTime ≤ CurrentTime < ShiftTime+BreakFlexibility):
→ Ant stops for 15 minutes
→ Update Time (go back to start)
Else if (BreakTime-BreakFlexibility ≤ CurrentTime+TimeToBreakLocation < BreakTime+BreakFlexibility):
→ Ant goes to the nearest break location
→ Update Time (go back to start)
Else if (FinishTime-BreakFlexibility ≤ CurrentTime+TimeToDepot)
If ant is at break location (including depot):
→ Ant is **distributed to a KPI area** and **chooses a neighborhood** within that KPI area and adds it to the route
→ Consider break location swap
→ Update time and KPI matrix (go back to start)
Else:
→ Ant **chooses a neighborhood** out of a set of nearest neighborhoods (set of neighborhoods is explanation in Section 5.4.2.1)
If (previous two visits are no break locations and no shift changes) and (decrease of travel times is bigger than no decrease of the objective function):
→ Switch previous two neighborhoods
→ Update time and KPI matrix (go back to start)
Else:
→ Update Time and KPI matrix (go back to start)
Else:
→ Ant goes back to depot and finishes the route
→ Set time to DayStartTime (go to next ant

Figure 38 – Route planning of one ant

5.4.1 Distribution strategy

This section describes the strategy that distributes the PEVs to different KPI areas. This distribution is needed so that not all ants start at the same location and do not cross their routes all the time. This distribution occurs whenever an ant leaves the break location. Since the set of neighborhoods where the ant is allowed to go to afterwards is limited (as we explain in Section 5.4.2.1), the focus of the ant lies on the KPI area, where it is distributed to. For that reason, this distribution strategy is essential with regards to meeting the KPI area targets. One might argue that the vehicles can also be distributed to a certain neighborhood instead of a certain KPI area. We tried two approaches based on the K-means clustering techniques, as explained in Section 4.3.1. In the first approach, we determine a number of clusters, which is equal to the number of vehicles k, based on the GPS coordinates of the neighborhoods. Figure 39 shows a scatter plot with 12 different symbols that shows the result of this approach in a scatter plot. The crosses show the centroids of the different clusters.

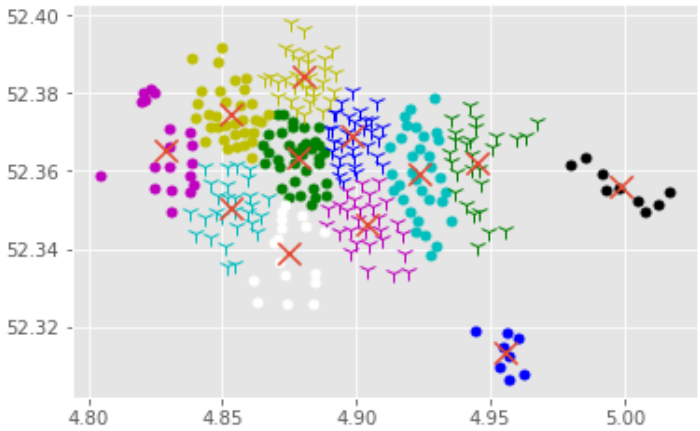


Figure 39 – Results of K-means clustering based on GPS coordinates

Even though the results seems logical, this approach only considers the GPS location. In order to generate clusters that include also the priority score of the neighborhoods, we came up with a second approach. In this approach, we create as many duplicates of the neighborhoods as the amount of their priority score (e.g. if there the priority score is 40, we make 40 duplicates of this neighborhoods). By running the K-means clustering again, we get the result as shown in Figure 40.

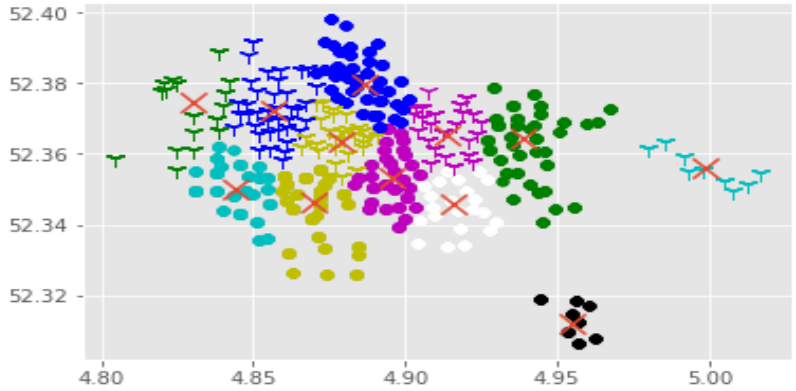


Figure 40 – Results of K-means clustering based on GPS coordinates and the priority score

Comparing the results, we can see that they very similar. In some cases, it is necessary though that more PEVs are deployed in certain KPI areas to make sure that the PCN target is reached. We conclude that this approach distributes the PEVs to the KPI areas in a manner that is too balanced such that not all PCN targets are reached in the end of the KPI period. This could be probably solved by a smarter clustering algorithm. However, we decide to use the KPI areas as pre-set clusters, which means that we "only" have to determine how many PEVs are send to every KPI area.

With our distribution approach, we want to avoid that on the one hand the distribution is too balanced. On the other hand, it is also not desirable that only the KPI areas that did not meet the targets are prioritized because this would lead to a very unbalanced routing. If, for instance, a KPI area has reached the payment rate target at the beginning of the KPI period, it would not be visited anymore. This is bad because we might never satisfy the requirement that the neighborhoods of every KPI area should be visited regularly. The target is to visit every neighborhood once a week. Additionally, by not going to a KPI area, we do not know the current payment rate of that area. This contradicts our claim that the measurements of the PEVs are the true value of the payment rate, as described in Section 4.2.1. Therefore, we choose a method that is based on the Target Factor and area size ratio, which is the ratio of the PCN target of a KPI area in comparison to the sum of all PCN targets. The area size ratio is a useful addition to the Target Factor because the Target Factor only considers the ratios but not the absolute numbers. By means of the area size ratio, we can express the need for more cars in KPI areas that have higher PCN targets (such as "Centrum"). For instance, if on day 0 the Target Factor for KPI area "Centrum" and "Noord" are both 0, we prefer to send more PEVs to "Centrum" because the absolute PCN target is much higher.

As explained in Section 5.3.1, the Target Factor can become 0.001 in case the PCN performance is twice as big as the PCN target. In this rare case, we consider it acceptable that PEVs do not get a starting point in that KPI area anymore since it is safe to say that the PCN target is met.

Our approach of distributing the ants computes a distribution array whenever an ant leaves the break location. This distribution array has the same length as the number of PEVs. When, for instance, PEV 3 leaves the break location, it goes to the KPI area that has the third index in the distribution array. In the following, we explain how this distribution array is computed. We start by creating an array with all KPI areas by adding clockwise the nearest KPI area, starting with "Centrum": "Centrum", "Oost-1", "Oost-2", "Zuidoost", "Zuid-2", "Zuid1", "Nieuw-West", "West-2", "West-1", "Noord". The reasons behind this is that it can happen that during the day a PEV is suddenly distributed to another KPI area and this KPI are should not be on the side of the city (e.g. "West-2" and "Zuidoost").

Due to two requirements of EPS, we decided to make the first PEV different from the other PEVs. One requirement is that one night shift is needed and the other is one particular neighborhood in the "Centrum" needs to be visited 6 times a day (except for Sunday). By sending the first PEV always to that specific neighborhood, including the night shift, we satisfy both requirements. This means that the number of PEVs allocated to "Centrum" has to be at least 1. The number of allocated PEVs is computed as follows:

1. For every KPI area, we multiply the Target Factor with the area size ratio to compute an adjusted Target Factor.
2. We compute a percentage for each KPI area by dividing the adjusted Target Factor of each area by the sum of all adjusted Target Factors.
3. Afterwards, the percentages are multiplied by the number of PEVs that day and are rounded to calculate the number that we want to send to this KPI area.

The results for the example of Figure 35 are shown in Table 12 for 12 PEVs.

KPI Area	Target Factor	Area size ratio	Adjusted Target Factor	Percentage	Number of allocated PEVs	Rounded number of allocated PEVs
Centrum	1.750	0.331	0.578	39.1%	4.692	5
Oost-1	1.794	0.104	0.186	12.6%	1.508	2
Oost-2	0.727	0.036	0.026	1.8%	0.215	0
Zuidoost	0.468	0.012	0.005	0.4%	0.046	0
Zuid-2	0.799	0.065	0.052	3.5%	0.425	0
Zuid-1	1.792	0.230	0.412	2.79%	3.349	3
Nieuw-West	1.547	0.052	0.081	5.5%	0.657	1
West-2	0.714	0.056	0.040	2.7%	0.327	0
West-1	0.866	0.111	0.096	6.5%	0.781	1
Noord	0.001	0.004	0	0%	0	0

Table 12 – Computation of the number of allocated PEVs with 12 PEVs

This approach does not always work in a sense that sometimes more or less PEVs are allocated than there are available due to the rounding. In case too many PEVs are allocated, this problem is solved by subtracting a PEV from KPI areas with the biggest difference between the rounded allocated number of PEVs and the real allocated number of PEVs. Considering the example in Table 12, that would be "West-2" (2-1.508 = 0.492). Whenever there are less allocated PEVs than needed, we add one PEV to the KPI with smallest difference between the rounded number of allocated PEV plus one and the real number of allocated PEVs. Considering the example in Table 12, that would be "Zuid-2" (1-0.425 = 0.575). If the number of allocated PEVs for "Centrum" is 0, one PEV has to be added, which may lead to subtracting a PEV from another KPI area.

In the end, an array is filled by going through Table 12 and add sequentially one of the KPI areas that have a number of allocated PEVs that is at least 1. Thereafter, the KPI areas with at least 2 are added. This goes on until the array is filled. In this case it happens after 2 iterations and we get the following list:

Centrum, Oost-1, Oost-2, Zuid-2, Zuid-1, Nieuw-West, West-2, West-1, Centrum, Oost-1, Zuid-2, Nieuw-West.

5.4.2 Choosing a neighborhood

The next chosen neighborhood depends on two things: the criteria of choosing a neighborhood and the set of neighborhoods from which the neighborhood is chosen. Both depend on the current iteration. In Section 5.4.2.1, we explain how the set of considered neighborhoods is determined. Section 5.4.2.2 described our desirability function and Section 5.4.2.3 the probability function for our ACO algorithm.

5.4.2.1 The set of neighborhoods

Whenever an ant chooses the next neighborhood out of a certain set of neighborhoods, there are three possible sets that can be considered:

1. All neighborhoods within a KPI area.
2. All neighborhoods that can be reached within a pre-set travel distance, which is denoted as Travel Distance Restriction.
3. All neighborhoods.

The ant chooses one of the neighborhoods of Set 1, whenever an ant leaves the break location. Set 2 is considered, whenever the ant is not at a break location. We call this set also the nearest neighborhood set. The reason for choosing neighborhoods within a pre-set travel distance is that the PEF drivers that follow the PEVs on scooters cannot follow whenever the distance gets to large. The pre-set travel distance should therefore depend on the average speed of the PEFs. We choose to set this parameter to 1500 meters because that results in routes that seem manageable for the PEF drivers. Whenever no feasible neighborhoods can be found in Set 1 or Set 2, the ant considers Set 3 as a kind of backup set.

In the first iteration, the ant chooses the neighborhood based on the result of the greedy function (see Section 5.4.2.2). Thereafter, it considers both the greedy function and the pheromone trails, as explained in Section 5.4.2.3. There is only an exception for Set 3. Whenever, the backup set is considered, the choice is only based on the greedy function, because we do not update the pheromone trails of all possible arcs to safe computation time.

5.4.2.2 The desirability function

Our desirability function is a greedy function that divides the priority score of a visit by the time that is needed to visit the neighborhood. It is an adjusted version of the one that Ke et al. (2008) are using.

Concerning the travel time from a break location b to the next neighborhood j, we use the accounted travel time, which is raw travel time reduced by a pre-set Travel Time Reduction parameter. The rationale behind it is to distribute the neighborhoods farther away from the break locations. In addition, the follow-up work for the PEF driver does not start right away, which gives him a bit spare time. We choose this value to be equal to 10 minutes, because this seems as a reasonable head start for the PEV driver. However, the accounted travel time may never exceed another pre-set parameter, namely Travel Time Restriction. The purpose of this is to restrict the travel time, whenever the ant chooses out of all possible neighborhoods (Set 3, see Section 5.4.2.1). We set this value equal to 15 minutes. If it turns out that these values, lead to routes that cannot be managed, these parameters can be adjusted easily.

However, the time does not only include the accounted travel time from neighborhood i to neighborhood j but also the service time from neighborhood j and the travel time from neighborhood j to the next break location b (or the depot depending on the shift). The latter, however, is not always important. When a shift starts, the PEV should or even must distance itself from the break location in order to avoid that it only drives close to the break location. The closer the time gets to the break time, the more important the travel time to the break location gets. Therefore, we use a progress factor $p_{v_{j,n,t,m,0}}$, which depends on the finish time and weekday of the visit. The progress factor determines how strong the travel time weighs within the desirability function and is computed as follows:

$$p_{v_{j,n,t,m,0}} = \frac{Finish\ time\ Of\ Visit\ -\ Start\ Time\ of\ Current\ Cycle}{Finish\ Time\ of\ Current\ Cycle\ -\ Start\ Time\ of\ Day\ Period}.$$

The desirability function of a visit is denoted as $\eta_{v_{j,n,t,m,0}}$ and can be denoted as:

$$\eta_{v_{j,n,t,m,0}} = \frac{PriorityScore_{v_{j,n,t,m,0}}}{c_{i,j,t,d} + s_{j,t,d} + c_{j,b_j,t,d} \ * p_{v_{j,n,t,m,0}}},$$

where $PriorityScore_{v_{j,n,t,m,0}}$ is the increase of the priority score by visit $v_{j,n,t,m,0}$, $c_{i,j,t,d}$ is the accounted travel time from neighborhood i to neighborhood j, at time t, on weekday d, $s_{j,t,d}$ is the service time at neighborhood j, at time t, on weekday d, and $c_{j,b_j,t,d}$ is the travel time from neighborhood j to the to the closest break location from neighborhood j, which is b_j.

As explained in Section 5.3.3, the priority score of a visit depends on the expected number of PCNs $P'_{v_{j,n,t,m,0}}$, the Target Factor $T_{v_{j,n,t,m,0}}$, and the Visit Day Factor of a visit $V_{v_{j,n,t,m,0}}$. We introduced the term expected number of PCNs of a visit because this number is not always equal to the forecasted number of PCNs $P_{v_{j,n,t,m,0}}$. In fact, the expected number of PCNs is unknown until all visits of the day are scheduled. The reasons for this is that it is possible that another visit is scheduled in the same neighborhood on same day but earlier. Therefore, we have to estimate the number of visitors that received a PCN during the earlier visit and still remain at the neighborhood, which we denote as $R_{v_{j,n,t,m,0}}$. The expected number of PCNs is for that reason computed as follows: $P'_{v_{j,n,t,m,0}} = P_{v_{j,n,t,m,0}} - R_{v_{j,n,t,m,0}}$. In the same way it can happen that by scheduling a visit the expected number of PCNs of a later visit in the same neighborhood on the same day can be decreased. For that reason, we introduce the current increase of the expected number of PCNs (denoted as $\Delta_{P'_{v_{j,n,t,m,0}}}$), which accounts for both phenomena: the decrease of expected number of PCNs of the current visit due to a prior visit and the decrease of the expected number of PCNs of a later visit due to the current visit. The exact computation of this increase is shown with an example in Appendix D. Note that it is obligatory that whenever a PEV driver visits a neighborhood, the driver has to scan the neighborhood fully. However, there is one exception when the visit starts in neighborhood with a parking regime but the parking regime stops during the visit. In these cases the driver is allowed to go to the neighborhood anyways until the parking regime is over. The reason behind this is that in the last minutes of the parking regime, we expect a higher PCN ratio than usual (according to EPS' experiences).

In conclusion, the desirability is finally determined as follows:

If (CurrentTime+TravelTime+ServiceTime+TimeToBreakLocation > BreakTime+BreakFlexibility) OR (CurrentTime+TravelTime+ServiceTime+TimeToDepot > FinishTime) OR (TravelTime>TravelTimeMaximum):

$$\eta_{v_{j,n,t,0}} = 0,$$

Else:

$$\eta_{v_{j,n,t,0}} = \frac{\Delta_{P'_{v_{j,n,t,m,0}}} * T_{v_{j,n,t,m,0}} * V_{v_{j,n,t,m,0}}}{c_{i,j,t,d} + s_{j,t,d} + p_{v_{j,n,t,m,0}} * c_{j,b_j,t,d}}.$$

As mentioned in Section 5.4.2.1, in the first iteration only the desirability function is used to choose the next neighborhood from Set 1 or Set 2. At the end of the first iteration, the pheromone trails are updated the first time as we describe in Section 5.4.3. Afterwards, the probability function of the ACO algorithm is applied to find better solutions than the first greedy solution as we explain in Section 5.4.2.3.

5.4.2.3 The probability function of the ACO algorithm

In the ACO approach the ant does not always choose the "most desirable" solution but goes probabilistically from neighborhood i to neighborhood j, whereby node j is from a set of considered neighborhoods j ∈ N(i), as determined in Section 5.4.2.1. The choice depends on the probability of node j, which is based on two factors: the desirability $\eta_{v_{j,n,t,0}}$ and the pheromone trails $\tau_{i,j}$. The pheromone trails depend on the pheromone value on the arc between neighborhood i and j, which is updated after every iteration as explained in Section 5.4.3. The probability p_{ij} is for all neighborhoods determined as follows:

$$p_{ij_{j \in N(i)}} = \frac{\tau_{i,j}^{\alpha} * \eta_{v_{j,n,t,0}}^{\beta}}{\sum_{l \in N(i)} \left(\tau_{i,l}^{\alpha} * \eta_{v_{l,n,t,0}}^{\beta}\right)},$$

where α and β are used to control the importance of the pheromone trails and the desirability.

All neighborhoods j ∈ N(i), are ranked according to their probabilities. Afterwards, a random number is generated with a uniform distribution between 0 and Q_0. Q_0 is a randomness parameter, which determines how big the generated numbers can be. If the random generated number is smaller than the p_{ij} of the first neighborhood in the ranked set, the first neighborhood is chosen. If this is not the case, a new number is generated and compared to p_{ij} of the second neighborhood. This continues until a neighborhood is chosen. In the very rare case that in the end no neighborhood is selected, the first ranked neighborhood is chosen. Q_0 decreases after every iteration by the decrease parameter D_{Q_0}, such that the neighborhoods with the most probability are more likely to be added to the route. The rationale behind it is that in the beginning many different solutions are explored but towards the end we want the ants to choose the best option in terms of the computed probability p_{ij}.

5.4.3 Update of pheromone trails

The first iteration of every day creates a solution of the route planning based on the greedy algorithm. The objective function of this day planning is saved as $Solution_{Greedy}$. The greedy solution serves as a benchmark solution, which we strive to exceed. The performance of the day planning of all iterations afterwards are measured by comparing it to the greedy solution:

$$Performance_{Iteration} = \frac{Solution_{Iteration}}{Solution_{Greedy}}.$$

The solution with the best performance is saved as $Solution_{Best}$ and $Performance_{Best}$. Since the greedy solution is the first created solution, it is the first best solution. We strive to maximize the best performance by updating the visited arcs with a pheromone trail after every iteration. The update of the pheromone trail is essential to the algorithm as it teaches the ants, which combination of neighborhoods worked well in the past iterations. This section introduces three different update strategies.

5.4.3.1 Update Strategy 1

Due to our generalization of the TOP that allows to visit neighborhoods multiple times and also includes other additional constraints, there is no ACO algorithm in the literature that tackles the same problem. However, we decided to derive our first updating strategy from the work of Montemanni and Gambardella (2009), who develop an ACO algorithm for the TOP with time windows. In their algorithm, only the ant that produced the best solution since the beginning of the computation, which we denoted as $Solution_{Best}$, is allowed to leave a pheromone trail. The reason behind it is that the best route is memorized, and in the future, ants will generate new (and hopefully better) solutions that are similar to this route. For that reason, we also update the pheromone of the visited arcs, only whenever a best solution is achieved. If an arc is visited more than once, it only counts as one visit. The pheromone trails are denoted as $\tau_{i,j}$, where i and j present the arc between neighborhood i and j. The initial value is denoted as τ_0. The updating rule is the following:

$$\tau_{i,j} = (1 - \rho) * \tau_{i,j} + \rho * Performance_{Best},$$

where p regulates the strengths of the pheromone that is left by the best solution. After the first iteration, this update is applied the first time. In this case, the initial value $\tau_{0_{i,j}}$ determines the attractiveness of the arcs that have not been visited by the greedy solution in comparison with the ones that have been. Since the $Performance_{Best}$ of the greedy solution is per definition equal to 1, we choose τ_0 to be smaller than 1 in order to attract more ants to the greedy solution. The exact value will be determined in Chapter 6.

Moreover, during the construction of a route, every ant decreases the pheromone trails of the arcs that is has used to prevent that arcs are visited too many times and to stimulate the exploration of new solutions. The rule is determined by:

$$\tau_{i,j} = (1 - \psi) * \tau_{i,j} + \psi * \tau_0,$$

where ψ is the evaporation parameter that regulates the decrease of the pheromone trace. We do not want the ants to visit neighborhoods many times because it decreases the exploration, plus it decreases also the performance of the solution as discussed in Section 5.4.2. If there is no improvement after the iteration, the pheromones are not updated but restored, meaning that the decrease during the solution building does not apply for the next solution.

5.4.3.2 Update Strategy 2

For this strategy, we slightly adjusted Update Strategy 1. In this strategy, the arcs are updated after every solution instead of updating the arcs only after an improvement of the $Solution_{Best}$. After every iteration all visited arcs are updated according to the formula:

$$\tau_{i,j} = (1 - \rho) * \tau_{i,j} + \rho * Performance_{Iteration}.$$

5.4.3.3 Update Strategy 3

Update Strategy 3 is another variation of Update Strategy 1. The difference is that the ants are considered separately. Not the best solution of all ants together counts but the best solution of every ant separately. The rationale behind this is that the allocation to KPI areas should be the same in most cases for all iterations. When the same ant is send to the same KPI area every iteration, we can measure the objective function of this ant. Every ant has therefore a separate performance that is compared to the greedy solution of that ant. By doing so, we also increase the pheromone trails of arcs that have been visited by a single well-performing ant even though most ants performed worse. If an arc is visited by more than one ant that achieved a personal best score, then the one with the higher score may set the trail. Regarding the update of the best solutions or all solutions, we can either use the updating rule as described in Strategy 1 or 2 for this strategy.

5.5 Conclusion

In this chapter, we presented our planning approach that is based on our desirability function and our ACO algorithm. We discussed our objective function, and how the PEV are distributed and their routes are created. Throughout this chapter, we introduced a lot of different parameters and strategies. The introduced parameters, namely the Target Reached Parameter, the Visit Day Parameter, the Swap Threshold, the Travel Distance Restriction, the Travel Time Restriction, and the Travel Time Reduction should be tested and adjusted if needed. For that reason, we call them manager decision parameters. The results of our algorithm will be presented in Chapter 6.

6 Results

In this chapter, we discuss the results of our prediction model, introduced in Chapter 4, and our routing algorithm, introduced in Chapter 5. In Section 6.1, we setup the design of our experiments. Section 6.2 discusses the accuracy of the different prediction models. Section 6.3 shows the results of our sensitivity analysis and Section 6.4 the results of our simulation. Finally, Section 6.5 demonstrates that our planning tool improves the current situation.

6.1 Design of experiment

This section introduces the experiments that are performed in this Chapter. In that regard, we first list the outputs with which we evaluate the different experiments in Section 6.1.1 and the parameter values that we introduced in Chapter 4 and 5 in Section 6.1.2. Finally, Section 6.1.3 discusses the questions that we are going to answer in this chapter and how we will answer them.

6.1.1 Outputs

In Section 5.3.3, we presented our objective function, which determines a priority score for every visit. This function can also be used to compute the priority score of all routes of one scheduled day. Next to this objective function, we compute the travel distance and count the average number of times that a neighborhood is not visited once a week, as explained in Section 5.3.2. In the end, we evaluate the results by the following outputs that are computed for every planned day:

- Objective function
- Number of PCNs
- Number of scans
- Travelled distance (in km)
- The average number of times that a neighborhoods is not visited once a week
- Algorithm running time

Regarding the number of PCNs and scans, we differ between the forecasted number and the simulated number. All outputs present the forecasted number if it is not explicitly said that they values are simulated.

6.1.2 Parameter values

In Chapter 5, we presented two algorithms: the greedy algorithm, which selects sequential visits based on the desirability alone, and the ACO algorithm, which builds on the greedy algorithm to search for a better global optimum. In that regard, we introduced various parameters throughout Chapter 4 and 5. The parameter values introduced in Chapter 4 are based our assumption and confirmed by the expert opinion of ARS. On the long-term these values should be replaced by values that are derived from data. The parameter values introduced in Chapter 5 are based on management decisions and can be adjusted by the management after evaluating the results in practice. In the following, we present the chosen parameter values that we already introduced in earlier sections:

- Parameters that are based on expert opinion:
  - Stability Parameter = 0.4 (see Section 4.3.4)
  - Margin of error for the number of paid-for visitor hours = 0.1 (see Section 4.2.2)
- Parameters that are based on a management decision:
  - Target Reached Parameter = 2 (see Section 5.3.1)
  - Visit Day Parameter = 5 (see Section 5.3.2)
  - Swap Threshold = 1 minute (see Section (5.4)
  - Travel Distance Restriction = 1500 meters (see Section 5.4.2.2)
  - Travel Time Restriction = 15 minutes (see Section 5.4.2.2)
  - Travel Time Reduction = 10 minutes (see Section 5.4.2.2)

6.1.3 Experiment questions

In this section, we design the experiments that we will perform in this chapter. Section 1.4 stated the research question that we want to show in this chapter whether our planning tool improves the current

situation. For that purpose, we divide this question into four sub question that we will answer in this chapter:

1. Which prediction model is the most accurate?

In order to answer this question, we use our prediction models and the one of ARS to forecast the number of PCNs for the same data set as in Section 4.3.2.2. We only consider the first weeks of the data set (weeks 24 to 42 2016) in order to make sure that this data is also included in the prediction model of ARS. Otherwise, the ARS forecast would have the disadvantage of not knowing the data.

2. How sensitive are the results regarding the parameter values and functions that we implemented?

This question is answered by performing a sensitivity analysis (Section 6.2), in which we analyze the impact of changes of some essential elements of our greedy and ACO algorithm on the routes and forecasted outputs.

3. Which combination of routing algorithm and prediction model lead to the best results?

By simulating the number of PCNs independently from the prediction model, we are able to compare routes that are based on a different routing algorithm and a different prediction model. The combination that leads to the best simulated results in terms of the number of PCNs will be considered as the best.

4. Does our planning tool improve the current situation?

We answer this question in Section 6.5 by answering the following two questions:

   a. To what extent does our planning tool improve the current situation at EPS?
   b. To what extent does our planning tool improve the current planning tool of ARS?

Question 4a is difficult to answer as the best scenario would be to execute our routes in practice. Even then, the results cannot be compared to results of the routes that EPS would have planned without our planning tool. Therefore, we show on average the number of PCNs that our planning tool would produce and compare it to the average number of PCNS of the historical scans of EPS.

We answer Question 4b by inserting the same inputs and compare the output of both routing algorithm to see which routing algorithm performs better with regards to the number of PCNs.

6.2 Prediction models

In Section 4.3.2.2, we discussed the benefit of the mean absolute error (MSE) that it is easy to interpret. For that reason, we compare in this section the following prediction models based on the MSE:

- Naive forecast (see Section 4.3.2.1)
- Neural Network A forecast (contains the original PCN ratios based on our neural network as presented in Section 4.3.2.2),
- Neural Network B forecast (some neighborhood predictions of the Neural Network A forecast are replaced by their nearest neighbor)
- ARS forecast

Since the same data set is used in Section 4.3.2.2, which contains only the known neighborhoods, neural network A and B are the same. The results of the three prediction models are shown in Table 13.

Prediction model	**Mean absolute error in comparison with data set**	**Improvement in comparison with ARS forecast**
ARS forecast	0.008973	0%
Naive forecast	0.008741	2.26%
Neural network forecast (A and B)	0.008615	3.99%

Table 13 – Results of forecasting methods

From these results, we conclude that our naive forecast and our neural network forecast give a more accurate prediction than the one of ARS, namely 2.26%, respectively 3.99% more accurate in terms of the MSE. Even though the absolute difference in the MSE seems very small, it has an impact for a big number of scans, with which EPS is dealing daily. However, a more accurate forecast does not necessarily mean that the obtained results in reality will be higher. This will be discussed further in our simulation study in Section 6.4.

6.3 Sensitivity analysis

In this section, we analyze the impact of some essential parts of our algorithm and discuss the results regarding the greedy and the ACO algorithm. As a forecast method, we mainly use our Neural Network A forecast in combination with our occupancy ratio (see Section 4.3.3).

6.3.1 The greedy algorithm

In the section, we experiment with the some parts of our algorithm to see how the results of the greedy algorithm is affected, such as:

- The stability function
- The Travel Distance Restriction
- The neighborhood swap and break location swap
- The Target Reached Parameter and Visit Day function

6.3.1.1 Stability function

In Section 4.3.4, we introduced the stability function, which is determined as: $S(\Delta_t) = 0.4^{\Delta_t}$ (Stability Parameter = 0.4). This function reduces exponentially over time. The lower the outcome of the function, the lower the number of remaining non-paying visitors that we still expect at a neighborhood since the previous visit. This assumption is crucial to our planning as it has a big influence on the routes and the outcomes. Figure 41 shows the route of one shift created by the greedy algorithm using the stability function, as we introduced it in Section 4.3.4.

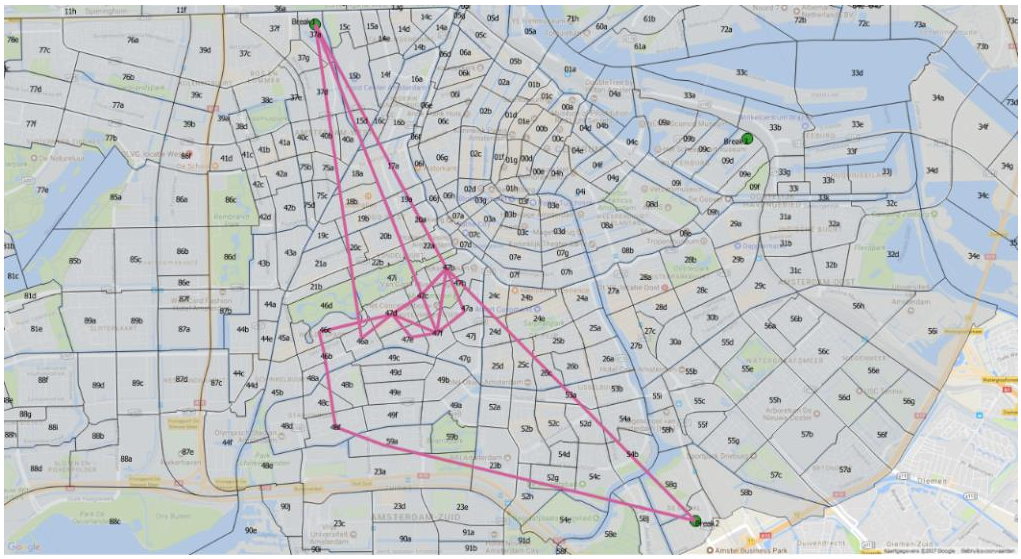


Figure 41 – Route that does take the stability function into account as determined in Section 4.3.4

Vehicle 4 starts in the morning at the depot, which is the break location at the top of the figure (break locations are marked with green points). We know that the vehicle goes to the middle of the figure and scans neighborhoods and takes a break once at break location 2 (at the bottom of the figure) and once at the depot. It is not interesting for us here how the vehicle travels exactly but to see that some neighborhoods are visited multiple times, as shown by more than 2 arcs are connected to it. By manually checking the route, we see that no neighborhood is visited twice before returning to a break location the first time.

Now, let us create a route, where we assume that there is no such thing as non-paying visitors staying at a neighborhood. By ignoring the stability function, we assume that the algorithm will send the PEV to only a few neighborhoods but multiple times a day. This assumption is confirmed in Figure 42, which shows the new route for same shift as in Figure 41.

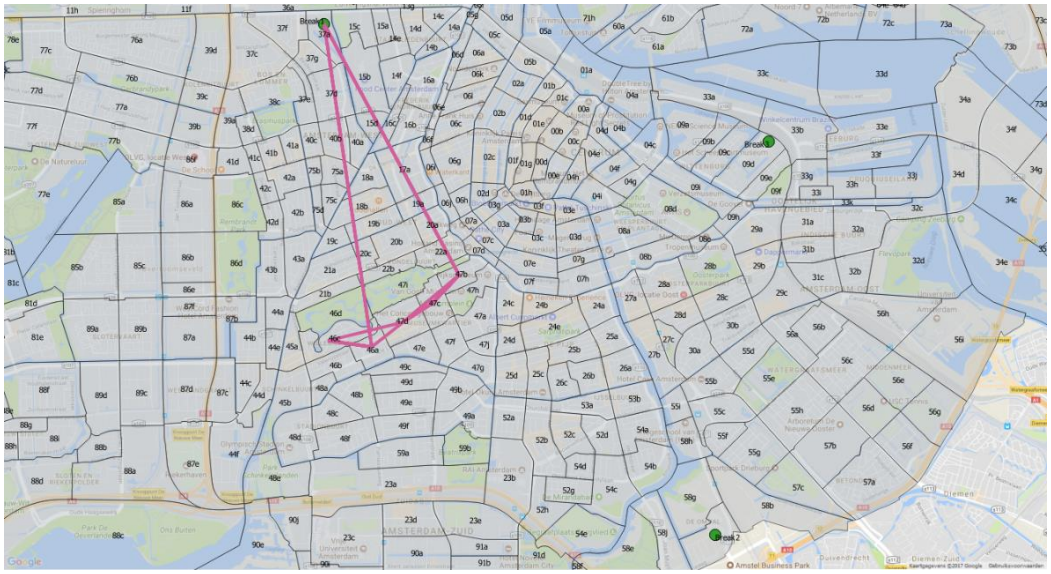


Figure 42 – Route that does not take the stability function into account

This time we see that less neighborhoods are visited within the same shift. This is due to the fact that neighborhoods are visited multiple times a day. By checking manually the route, we know that one neighborhood is even visited four times before the vehicles returns for the first break. This is the consequence of not having the stability function that we introduced. For some regions, it might be the case that the parking duration of non-paying visitors is short enough that this route would actually be the best but we assume that this is not the case. However, it is important to do further research on the parking duration of non-paying visitors.

Let us now consider what happens if the stability function results in very high values such that it is almost never attractive to go back to a visited neighborhood. As discussed in Section 6.2.2, it is not possible to set a constraint that ants are not allowed to visit a neighborhood multiple times a day as this would at some point remove all feasible options for the ant when planning a large number of routes. Therefore, we use a linear stability function that returns values close to 1: $S(\Delta_t) = \max(1 - 0.1\Delta_t, 0)$, which leads to less neighborhoods that are visited multiple times a day. The resulting route of the same shift is shown in Figure 43.

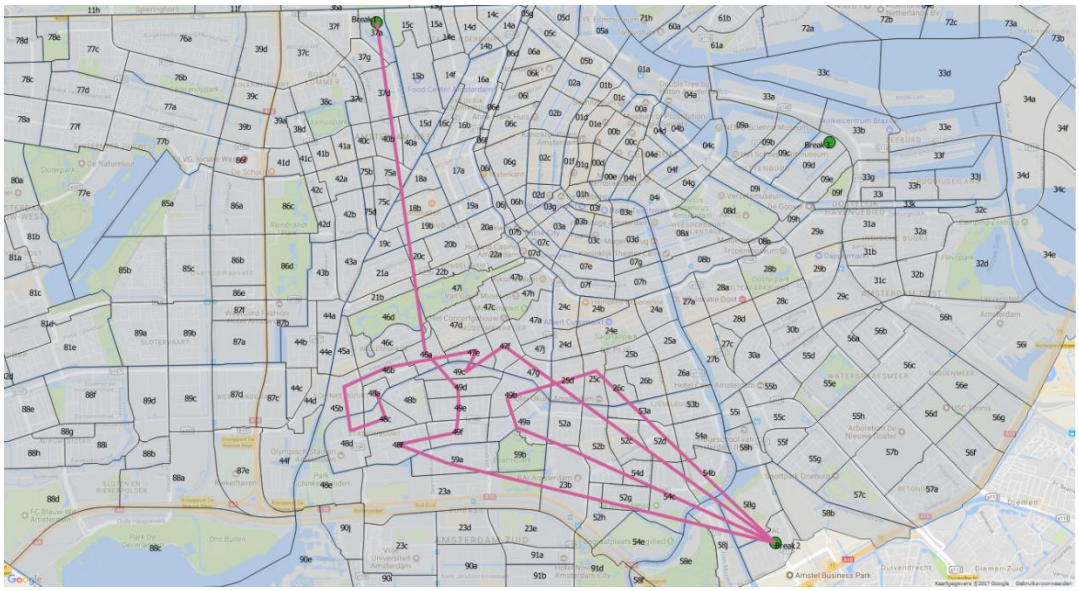


Figure 43 – Route that is based on a linear stability function

With this linear function, we see that no neighborhood is visited twice. This leads to a strong reduction in the number of PCNs.

The reason of this sensitivity analysis is only to show the impact of our exponential stability function. We conclude from this analysis that the stability function has a strong impact on the routes and therefore the number of PCNs and that this function is an important implementation to deal with the multiple visits of neighborhoods. However, further research the stability function is required by investigating how long non-paying visitors (or at least paying visitors) stay at a parking spot.

6.3.1.2 Travel Distance Restriction

In Section 5.4.2.1, we introduced the Travel Distance Restriction, which determines the size of the neighborhood set from which the ant chooses the next neighborhood. This restriction is necessary because otherwise it can happen that the distances between two neighborhoods become too large such that the PEF driver is not able to follow the PEV anymore. The consequence of this is that the PEF cannot do all the follow-up work that is needed and therefore the effective number of PCNs is reduced. We set the value to 1500 meters because that seems as a reasonable restriction. In the following, we compare the results of a route with a Travel Distance Restriction of 1500 meters to the routes with a Travel Distance Restriction of 500 and 5000 meters. Table 14 shows the results of the greedy algorithm based on the Neural Network A forecast.

Travel Distance Restriction (in meters)	Distance travelled (in km)	Number of scans	Number of PCNs	Algorithm running time (in minutes)
500	1228	195126	4385	17.76
1500	954	194277	3444	6.8
5000	1258	203303	4269	19.54

Table 14 – Sensitivity analysis of the Travel Distance Restriction (Neural Network A forecast)

Table 14 shows that the route with a Travel Distance Restriction of 5000 meters leads to a better solution with regards to the number of PCNs than the one with a restriction of 1500 meters. The reason for that is that there are less restrictions, i.e., there are more neighborhoods to choose from at each planning step. This enables the PEV to get to neighborhoods that seem to be very attractive in terms of the number of PCNs and consequently the priority score. It should be questioned whether the number of PCNs can actually be increased by such an amount if the distances become this large. A reason for that could be due to an underestimation of the travel times, which we discussed in Section 4.4. Even though one might assume that the results of the Travel Distance Restriction of 500 should be more similar to results with a restriction of 1500 meters than 5000 meters, this is not the case. The reason for this is that with a Travel Distance Restriction of 500 meters it often happens that there is no feasible available option within that distance. When that happens, the ant chooses out of all neighborhoods, as explained in Section 5.4.2.1. Choosing out of all options is similar to choosing the options within a large travel distance, such as 5000 meters, which explains the similarity between those two.

We wonder if this effect is the same when using another forecast (e.g. the ARS forecast). Logically, we cannot compare the number of PCNs and scans of two different forecasts but we can investigate whether the effect is the same. The results based on the ARS forecast are shown in Table 15.

Travel Distance Restriction (in meters)	Distance travelled (in km)	Number of scans	Number of PCNs	Algorithm running time (in minutes)
500	1235	189564	2706	8.25
1500	1043	213997	2863	4.69
5000	1377	211306	3189	10

Table 15 - Sensitivity analysis of the Travel Distance Restriction (ARS forecast)

From comparing both tables, we conclude first of all that the ARS forecast is faster than the Neural Network A forecast. This is due to the fact that the Neural Network A forecast checks the parking regime while computing. In the ARS forecast, this is already included in the forecast. This can be easily adjusted in the future by already integrating the parking regime in the forecast.

Furthermore, it appears that the Travel Distance Restriction parameter had a bigger impact on the results when using the Neural Network A forecast. We still see the difference of PCNs between the restriction of 1500 and 5000 meters but not that strong. Looking at the results of the restriction of 500 and 5000

meters, we observe that they are still similar in terms of the travel distance and the algorithm running time. However, the number of PCNs for the routing algorithm with a travel distance of 500 meters performs with the ARS forecast much worse than the one with a restriction of 5000 meters.

Generally, we conclude that a higher Travel Distance Restriction leads to more PCNs and requires more computation time. In this regard, it is important to analyze in practice which distances the PEF can handle in order to extend this restriction. When increasing the Travel Distance Restriction, it seems that both forecasts lead to longer travel distances (for the ARS forecast the distances are even longer) but that the benefit of traveling more is bigger for the Neural Network A forecast because it increases relatively more than for the ARS forecast. A reason for that could be that the PCN hotspots that cause the large travel distances, are bigger for the Neural Network A forecast.

6.3.1.3 Neighborhood and break locations swap

In our algorithm, we also introduced two swap mechanisms: a neighborhood swap and a break location swap. The idea of the neighborhood swap is to swap two neighborhoods if the saved travel time of the resulting route is above a certain threshold (in our case 1 minute). Note that a swap is not performed if the improvement of the travel time in percentages is lower than the loss in percentages of the number of PCNs. An example of a successful swap is shown in Figure 44.

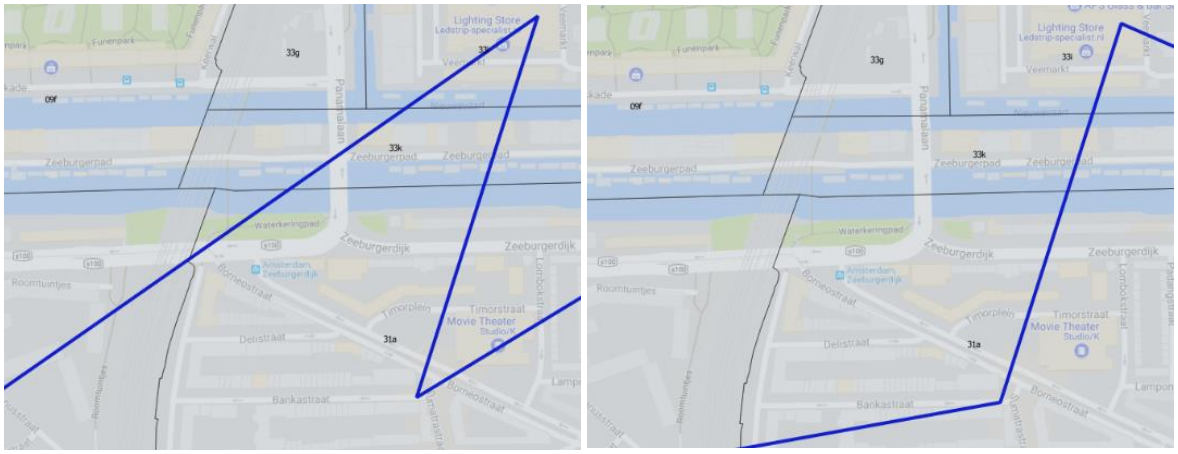


Figure 44 – Comparison of route without neighborhood swap (left) and with neighborhood swap (right)

In this figure, the PEV comes from the left side. Comparing the route without a swap and with a swap, we see that the swap is successful because the "zigzag" is removed. The idea of the break location swap is the same. The only difference is that the swap is always performed if the travel time is reduced (no matter how much). PCNs are not involved in this decision, since in our algorithm no PCNs are obtained at the break locations. In reality, longer travel times probably lead to more PCNs by scanning on the way which we do not quantify at the moment but we assume that generally it is better to have shorter travel times.

During our experiments, we have seen that the neighborhood swap rarely improves the route in terms of the priority score or the number of PCNs. In most cases, the outcome is slightly worse (usually less than 0.1%). This is due to the fact that the saved time apparently does not bring enough time to find new PCNs. However, as we explained in Section 5.4, we prefer this setting because even though the outcome might be slightly worse, the route is more logical to the PEV driver. Furthermore, the saved time can additionally serve as a buffer, which makes the routes more likely to be managed in time. Moreover, in Section 4.4, we assumed that the travel times are underestimated. If that is the case, zigzags will be more inefficient and swaps will become more effective and save more time. Another interesting fact that we observed is that the break location swap, can actually decrease the performance of the route. This seems rather strange because, as we mentioned earlier, the break location swap only saves time and does not decrease the number of PCNs. However, sometimes this is the case because the swap does not save enough time to visit an extra neighborhood and the expected forecasts of the visits are actually lower due to an earlier arrival. When the PEV arrives earlier to the destined neighborhoods, it can happen that the obtained number of PCNs become less. Moreover, due to the fact that the swap occurs during the solution building, the time reduction can lead to a different route which does not necessarily leads to an improved route in terms of the number of PCNs.

In the end, we keep both swaps because despite of the fact that sometimes the number of PCNs is slightly decreased, they make the routes more customer friendly and save travel time.

6.3.1.4 Target Reached Parameter and Visit Day Factor function

In this section, we consider the Target Reached Parameter and the Visit Day Factor function. Additionally, we show whether our algorithm actually meets the KPI targets within 90 days based on our forecast. The Target Reached Parameter determines to what extent the KPI areas that did not reach the targets should be prioritized (see Section 5.3.1). The Visit Day Factor function determines to what extent the focus of the algorithm lies on the neighborhoods that have not been visited once a week (see Section 5.3.2).

During our experiments, we made an interesting observation. Even though the the forecasted number of PCNs is higher when using our Neural Network A forecast than when using the ARS forecast (this is not only the case for the forecasted number but also for the simulation, as we show in Section 6.4.2), we noticed that the forecasted number of PCNs in "Centrum" are on the long-term lower when we apply the Neural Network A forecast instead of the ARS forecast. Generally, we cannot compare these outputs due to fact that different forecasting methods are used. However, if the reason behind it is that the Neural Network A forecast has a lot of PCN hotspots close to "Centrum", which pull the PEVs out of this KPI area, this could be a problem because it leads to less scans in "Centrum". In this section, we will also investigate whether this is a problem.

Let us first consider the results of the routing algorithm based on the ARS forecast. When creating a planning for 90 days for the highest PCN targets that we know of, we see that the Target Factor 2 works well in that sense that all targets are met. We choose to increase these PCN targets to see if the routing can still handle these fictive targets. The results are shown in Table 16.

KPI Area	Centrum	Nieuw-West	Noord	Oost-1	Oost-2	West-1	West-2	Zuid-1	Zuid-2	Zuidoost
PCN Target	60000	7800	850	20000	6700	22000	8500	37000	11000	3700
PCN Current	61552	11413	923	27510	9508	29512	13085	50484	15301	3599
Payment Rate Target	0.81	0.88	0.88	0.87	0.8	0.81	0.81	0.92	0.86	0.81
Payment Rate Current	0.79	0.9	0.87	0.85	0.79	0.79	0.8	0.85	0.85	0.84
Average number of neighborhoods without a visit once a week: 34.39										

Table 16 – KPI matrix after a planning of 90 days with a Target Reached Parameter of 2, a limited Visit Day Factor, and the ARS forecast

Table 16 shows the basic KPI targets and not the upper bounds that we computed for our algorithm (see Section 4.2). We see that the basic PCN target of "Centrum" is met. However, the upper bound, which is 72663 (computed with a high margin of error for the number of paid-for visitor hours of 0.1), is not met in this case. Since, 60000 is a very challenging and fictive PCN target, which is 14% higher than the highest historical target for "Centrum" that we know of (52741), and in our simulation in Section 6.4.2 we show that the simulated number of PCNs is 6.8% higher than the forecasted number of PCNs based on the ARS forecast, we consider "Centrum" as a reached target. The KPI area "Zuidoost" is the only KPI area that does not meet the basic PCN target but this is due to the fact that the payment rate target is met at the beginning of the KPI period. Furthermore, we derive from this table that the average number of neighborhoods without a visit once a week is 34.39. A part of these neighborhoods belong to the KPI area "Noord", which has a quite low PCN target and therefore this KPI area does not need to be visited very often.

Furthermore, we are interested to see what happens if the Visit Day Factor function is unlimited. This means that the Visit Day Factor is not limited to 2 anymore and will increase very fast after one week without a visit. This will force the algorithm to go there and will eventually overrule all other aspects. The results are shown in Table 17.

KPI Area	Centrum	Nieuw-West	Noord	Oost-1	Oost-2	West-1	West-2	Zuid-1	Zuid-2	Zuidoost
PCN Target	60000	7800	850	20000	6700	22000	8500	37000	11000	3700
PCN Current	61436	11384	1027	27308	9627	29569	12831	50171	15245	3171
Payment Rate Target	0.81	0.88	0.88	0.87	0.8	0.81	0.81	0.92	0.86	0.81
Payment Rate Current	0.79	0.9	0.87	0.85	0.79	0.79	0.8	0.85	0.85	0.84
Average number of neighborhoods without a visit once a week: 23.88										

Table 17 - KPI matrix after a planning of 90 days with a Target Reached Parameter of 2 and an unlimited Visit Day Factor, and the ARS forecast

We see that the targets are met more or less to the same extent. This can be validated since the average forecasted number of PCNs is now 2464, which is 13 PCNs less (-0.52%) than the average that we achieved with a limited visit day factor. However, the daily average number of neighborhoods that have not been visited once a week is lower. This number decreased from 34.39 to 23.88 (-30.56%) by applying the unlimited Visit Day Factor. In the end, the management of EPS (or ARS) has to decide whether this is an improvement. In our opinion, the unlimited Visit Day Factor seems to be preferable. Even more so because it increases the variability of the routes, which is beneficial for EPS because non-paying visitors are not able to predict the routes (see Section 5.3.2).

Let us now consider the results based on the Neural Network A forecast with the unlimited Visit Day Factor in Table 18.

KPI Area	Centrum	Nieuw-West	Noord	Oost-1	Oost-2	West-1	West-2	Zuid-1	Zuid-2	Zuidoost
PCN Target	60000	7800	850	20000	6700	22000	8500	37000	11000	3700
PCN Current	55895	12327	2906	27005	11136	31454	14821	54037	17821	6766
Payment Rate Target	0.81	0.88	0.88	0.87	0.8	0.81	0.81	0.92	0.86	0.81
Payment Rate Current	0.79	0.9	0.87	0.85	0.79	0.79	0.8	0.85	0.85	0.84

Table 18 – KPI matrix after a planning of 90 days with a Target Reached Parameter of 2 and an unlimited Visit Day Factor, and the Neural Network A forecast

As mentioned in the beginning, the basic PCN target in "Centrum" is not met but the forecasted average daily number of PCNs for all KPI areas with the Neural Network A forecast is 138 higher (2602 PCNs on average). Although the basic PCN target of "Centrum" is not met in this case, we are confident that our algorithm based on the Neural Network A forecast will meet the realistic future targets of EPS, because:

- the PCN target of "Centrum" is 14% higher than the highest historical PCN target
- we know from Section 6.4.2 that the simulated number of PCNs is 9.5% more than the number of PCNs that is predicted by the Neural Network A forecast
- we know from Section 6.5.1 that we are using less shifts than EPS uses on average.

Consequently, we do not consider the observation that we made in the beginning (the less PCNs in "Centrum" with the Neural Network A forecast) to be a problem.

When increasing the Target Reached Parameter to 3, the number of forecasted PCNs in "Centrum" is increased to 57087 and therefore closer to the target. However, the average number of forecasted PCNs is

lower (2558 PCNs on average) due to the fact that more and more vehicles are forced to "Centrum" as it is the only KPI area that did not meet the upper bounds of the KPI targets. The unbalanced distribution of the PEVs leads to an oversaturation of the "Centrum", which finally leads to the smaller number of PCNs. This can also be seen in Figure 45.

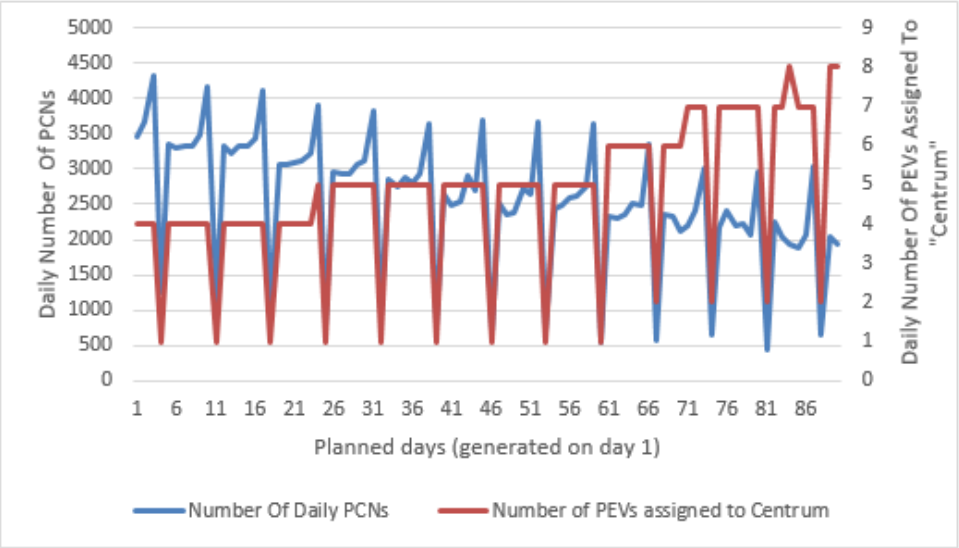


Figure 45 – A 90 day planning showing the number of daily PCNs and the number of PEVs that are assigned to "Centrum"

This figure shows for every planned day how many PEVs are assigned to "Centrum" and how many PCNs are forecasted that day. We see that towards the end of the KPI period more and more PEVs are assigned to "Centrum" as its PCN target is not met. Furthermore, we observe that the oversaturation that is caused by the many PEV drivers in "Centrum" leads to less PCNs towards the end. Note that the high number of 7 or 8 assigned PEVs to "Centrum" is an exception that is caused by the fictive high PCN target.

If the planning tool would not create a new planning every day, it would be interesting to consider assigning the PEVs on the long term. In this regard, it would be required to compute the average forecasted number of PCNs that is obtained in a KPI area by assigning X (1…12) PEVs to the KPI area. The more PEVs are assigned to one KPI area, the lower is the average number of PCNs of one PEV. Thereafter, these computations could be used to optimize the distribution of the PEVs to the different KPI areas for a certain period of time (e.g., one week or the entire KPI period).

Finally, we conclude that in our opinion the best overall performance is achieved with an unlimited Visit Day Factor and a Target Reached Parameter of 2. Even though the fictive target of 60000 in "Centrum" was not met with the Neural Network A forecast, we are confident that it will manage the realistic future targets of the municipality, which are assumed to be lower (the highest historical target that we know of was 52741). It is also important to further investigate the variability of the paid-for visitor hours because the assumption of 10% that we use now increases the PCN target by 10% (see Section 4.2.2). This especially effects the PCN target of "Centrum" because it already the highest PCN target.

6.3.2 The ACO algorithm

When making routes for 12 PEVs (also denoted as vehicles or ants), which is the usual amount of vehicles that EPS uses, with our ACO algorithm, we had some difficulties to find better solutions than the solution of the greedy algorithm (denoted as greedy solution), which is the solution of the first iteration. After trying different parameters and update strategies, we wonder whether the number of deployed PEVs has an impact on the performance of the ACO algorithm. For that reason, we considered making a route for only one vehicle, and indeed we received good results with the following parameter values (parameters are introduced in Section 5.4.2 and Section 5.4.3):

- Update Strategy 1
- $\rho = 0.90$
- $\psi = 0.95$
- $Q_0 = 0.4$
- $D_{Q_0} = 0.1$

- τ_0= 0.8
- α = 1
- β = 2
- Number of iterations = 10

The outcomes for one vehicle are shown in Table 19.

Day	Iteration	Number Of PCNs	Priority score	Improvement compared to greedy solution
1	1	254	508.12	0%
1	2	272	543.70	7%
1	3	257	513.66	1%
1	4	291	582.45	15%
1	5	278	555.44	9%
1	6	264	528.04	4%
1	7	266	531.96	5%
1	8	300	600.23	18%
1	9	282	563.56	11%
1	**10**	**306**	**612.28**	**20%**

Table 19 – Results of the ACO algorithm for one vehicle

We derive from this table that the priority score (and also the number of PCNs) increases with the number of iterations. After 10 iterations, the solution is already improved by 20%. We conclude that the ant is indeed learning and that our ACO algorithm works at least for one vehicle. Unfortunately, this learning effect seems to vanish when we create routes for more vehicles with the same parameter settings. Figure 46 shows the improvement towards the greedy algorithm for 1, 2, 3, and 7 vehicles in 10 iterations.

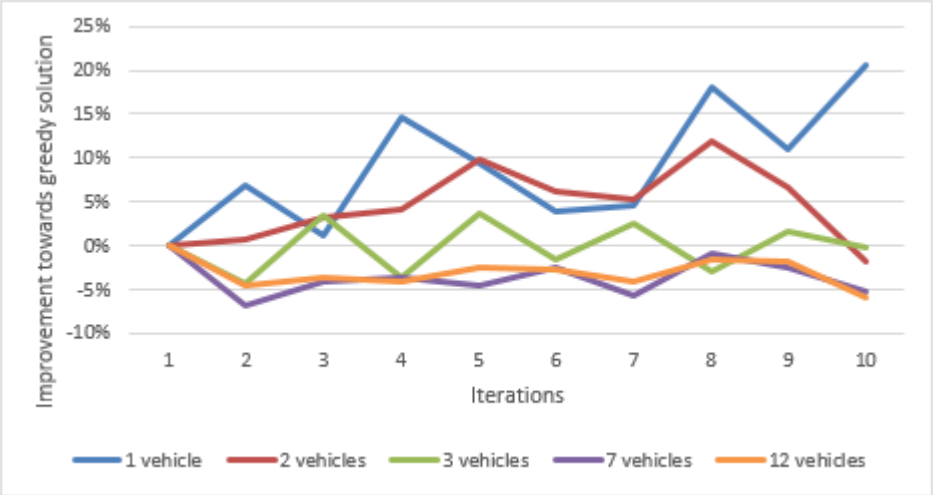


Figure 46 – Results of the ACO algorithm for 1, 2, 3, 7, and 12 vehicles

From Figure 46, we conclude that the learning effect decreases with the number of vehicles. We have observed that for 7 vehicles, the greedy solution is rarely improved. For 12 vehicles, it is sometimes possible to find an improvement; however, it takes a lot of time and only improves the solution by roughly 1%. For many vehicles the results retrieved by our Update Strategy 3 (Section 5.4.3.3), which updates the pheromones separately for all PEVs, seems to perform a bit better. Nevertheless, the results are still not good enough to significantly improve the solution. As a next step, we want to investigate what happens when (too) many vehicles are used.

One logical explain could be that with more deployed PEVs, also more PEVs are deployed to the same KPI area (for instance "Centrum"). When many ants share the same KPI area, the learning effect that always

applies for one single ant gets more complicated. For instance, in the past the first ant has learned a new route that improves the results. However, it might happen that the second ant changes its route within this iteration and visits some nodes that the first ant would have travelled to. Therefore, the learning effect of the first ant becomes in this example useless.

Furthermore, there could be another reason that leads to a decrease of the learning effect. Amsterdam has 320 neighborhoods with an active parking regime. If a neighborhood must not be visited more than once a day and 12 vehicles are scheduled, every vehicle can visit 26.67 (320/12) neighborhoods on average. With our current parameter settings and inputs, every vehicle visits around 45-50 neighborhoods a day. The first vehicle visits around 65-70 neighborhoods due to the additional night shift (see Section 5.4.1). Although ants return to visited neighborhoods due to the high desirability and not because there is no other choice, this means that it is even required that neighborhoods are visited multiple times a day. This is also the reason that the algorithm does not work when 12 vehicles are used and a neighborhood must not be visited more than once a day. If we assume that every vehicle visits on average 45 neighborhoods a day, then only 7.11 vehicles (320/45) will be required to visit all neighborhoods once. After 7 vehicles, one could say that Amsterdam gets "oversaturated" and therefore neighborhoods have to be visited at least a second time. We wonder if this is the reason for the decrease of the performance of the ACO algorithm. As discussed in Section 6.2.1.1, we know that the stability function influences the number of times that a desirable neighborhood is visited on a day. Therefore, we want to investigate whether a different stability function that leads to fewer multiple visits of a neighborhoods increases the performance of the ACO algorithm for 12 vehicles. For this purpose, we choose the following linear stability function: $S(\Delta_t) = \max(1 - 0.1\Delta_t, 0)$, which leads to less neighborhoods visited multiple times a day, as shown in Section 6.2.1.1. During our experiments with different parameters and a different update strategy, we managed to improve the greedy solution by 3% after 10 iterations with the following parameter setting:

- Update Strategy 3
- ρ = 0.95
- ψ = 0.95
- Q_0 = 0.4
- D_{Q_0} = 0.1
- τ_0= 0.8
- α = 1
- β = 2.5
- Number of iterations = 10

As we see that the ACO algorithm can improve the greedy solution when using another stability function, we conclude that the stability function has an impact on the performance of the ACO algorithm. Since there is a reason for our stability function, changing it is not an option. We assume that this improvement that occurs due to changing the stability function has the same cause that we experienced before with less PEVs, namely that there are less multiple visits of neighborhoods a day. Apparently the ACO algorithm performs better when the problem is more similar to the original TOP without multiple visits than our proposed generalization with multiple visits. It seems that the pheromones of the ACO algorithm cannot deal with the multiple visits as good as our greedy algorithm. Therefore, we conclude that the ACO algorithm might not be the best choice to deal with this new generalization. Since the entire problem especially with the multiple visits is very time-related, it might help to apply time-dependent pheromones, which was already done by Jiang, Chen, Ma, and Deng (2011). However, even though the time-dependency would be included in the learning effect, it is not guaranteed that time-dependent pheromones would really lead to an improvement regarding the multiple visits. Considering the greedy algorithm, the algorithm could be improved by adding a saturation factor that determines how many neighborhoods have been visited already in that area around the considered neighborhood at a certain time.

Finally, we conclude that even if the ACO algorithm leads to a better result than the greedy algorithm, the improvement is very little for the usual number of deployed PEVs (smaller than 1%). If we want to apply the ACO algorithm with 10 iterations, it means that the computation time for one route is more than 10 times larger (10 iterations + 10 times a pheromone update). Therefore, we choose only to consider the greedy algorithm further in this chapter.

6.4 Simulation study

In this simulation study, we want to show the impact of different forecasting methods on the performance of the routes created by the greedy algorithm. We only consider our greedy algorithm, since we have shown in Section 6.2.2 that the ACO algorithm did not significantly improve the results for 12 vehicles. The following five different prediction models are compared:

1. Poor forecast (i.e., same PCN ratio for all neighborhoods at any time)
2. ARS forecast
3. Naive forecast
4. Neural Network A forecast
5. Neural Network B forecast

Except for the ARS forecast (because it is not our forecast) all forecasts are based on the occupancy ratio, as determined in Section 4.3.3. We run the experiments for three weeks. In the data analysis (see Section 4.3.1), we have shown that the PCN prediction is not influenced by the week number, except for the very small decline of the non-paying ratio. We consider three weeks enough to get accurate averages. The structure of our simulation, will be the following:

For each of the five prediction models:

1. We create a forecast for three weeks
2. We create routes for three weeks using our greedy algorithm
3. We simulate the number of PCNs for every visit of the planned routes
4. We evaluate the results based on the simulated number of PCNs and not on the forecasted number of PCNs.

Section 6.3.1 explains how we simulate the number of PCNs and Section 6.3.2 discusses the results of this simulation study.

6.4.1 Simulation of the number of PCNs

In this section, we explain how we are going to simulate the number of PCNs. It is our general idea to simulate values for the PCN ratio and the occupancy ratio. Thereafter, the number of parking spots are multiplied by these ratios, in order to simulate the expected number of PCNs. For this purpose, we have to find a suitable distribution for both ratios.

We start with the PCN ratio, where we consider the PCN ratios of the same data set that we used in Chapter 4. The problem of finding a suitable distribution for PCN ratio of a certain hour in a certain neighborhood is that the PCN ratio of this data set depends on different factors such as time and space. For that reason, we divide the historical PCN ratios by the different factors that we determined for our naive forecast (see Section 4.3.2.1). By doing so, we normalize the historical PCN ratios such that they do not depend on time nor space anymore and we can look for a suitable distribution. We find that the normalized PCN ratios follow a mixed lognormal distribution (for more details, see Appendix E). This enables us to simulate values for the normalized PCN ratio. Since the normalized PCN ratio is not needed but the PCN ratio that depends on a specific time and neighborhood, we have to multiply these simulated normalized PCN ratios by the different factors. This process is the same as computing the naive forecast (Section 4.3.2.1), except that the base line PCN ratio is now replaced by the simulated and normalized PCN ratio.

For the occupancy ratio, we do not find a distribution but assume a normal distribution. The reason behind this is that the occupancy ratio is still more an assumption rather than a ratio retrieved from data. Therefore, it does not make sense to try to fit it to a certain distribution. Regarding the normal distribution, we use the average and standard deviation that we computed in Section 4.3.3.

In conclusion, the simulated number of PCNs is the same as the naive forecast, except for the important difference that the base line PCN ratio and the occupancy ratio are simulated and not averages.

6.4.2 Results of experiments

In this section, we present the results of our simulation study. Table 20 shows the forecasted and simulated results (average number of scans, average number of PCNs, and the average PCN ratio) of the routes that are based on the different forecasting methods. The forecasted outputs are based on the different applied forecasting method but the simulated outputs and the average distance travelled are independent. Therefore, we use the independent outputs to compare the different prediction models. As discussed in Section 6.3.1, the only difference between the number of PCNs of the simulation and the naive forecast is that the first one is based on simulated values and the latter on total averages. This means that the longer the simulation runs, the smaller the difference between the forecasted and simulated number of PCNs gets. We used this fact as a validation for the simulation accuracy after three weeks. More importantly, that means that the algorithm based on the naive forecast has an advantage because on average its forecasted number of PCNs is closer to the “truth” of the simulation. Table 20 proves that indeed the difference between the forecasted and simulated number of PCNs is the smallest for the naive forecast.

		Forecasted outputs			Simulated outputs		
Routes based on forecasting method	**Average distance travelled (in km)**	**Average number of scans**	**Average number of PCNs**	**Average PCN ratio**	**Average number of scans**	**Average number of PCNs**	**Average PCN ratio**
Naive	*815*	*186043*	*3496*	*0.0188*	*189940*	*3468*	*0.0182*
Poor	807	227685	3165	0.0139	232511	2774	0.0098
ARS	906	184315	2631	0.0142	189040	2809	0.0149
Neural Network A	839	178874	3150	0.0176	182857	3450	0.0189
Neural Network B	831	174047	3241	0.0186	178278	3334	0.0187

Table 20 – Results of our simulation study using our greedy algorithm with different prediction models

Furthermore, we observe that the algorithm based on the poor forecast, logically tries to maximize the number of scans as the PCN ratio is everywhere the same. It succeeds in having the highest number of forecasted and simulated scans but not in the number of PCNs.

Finally, we consider our proposed Neural Network A forecast as the best forecast in terms of the simulated number of PCNs and average PCN ratio. Therefore, we will use this forecast further in this chapter.

6.5 Improvement of current situation

In this section, we validate whether our routing algorithm actually improves the current situation. For this purpose, compare the number of PCNs that we expect from our planning tool with the historic number of PCNs of EPS in Section 6.4.1 and we compare our algorithm to the current algorithm of ARS in Section 6.4.2. Based on the results of Section 6.3.1.4 and Section 6.3.2, we use our greedy algorithm with an unlimited Visit Day Factor.

6.5.1 Comparison with EPS

Due to the limited time of this research, it is not possible to run our routes in practices to see if they lead to an improvement. Therefore, we compare the historical results of EPS of 90 days (01.01.2017-31.03.2017) with a planning that we create for 90 days.

During the 90 days, EPS obtained on average 2104 PCNs per day, whereas our algorithm produces 2601 PCNs per day with the Neural Network A forecast, which we showed in Section 6.2 is the most accurate. Even though this shows an improvement of 23.62%, we have to take into account that this approach has some limitations. First of all, we cannot yet determine the accuracy of, for instance, the computed travel times, service times, occupancy ratio, and the stability function because they have not been tested in practice yet. Another point is that maybe during this period, the number of paying visitors was relatively high, which led to less PCNs. The uncertainty of this approach, does not necessarily mean that the forecasted number of 2601 PCNs would be lower in practice but it could be the case. One thing, we can say more about is the number of deployed PEVs. Our algorithm schedules 12 PEVs on usual weekdays (Monday till Saturday) with one night shift. On Sunday, 3 PEVs do only the Sunday shifts, which we described in Section 2.1.4. That means, on average we use 10.7 vehicles a day for these selected days. The number of vehicles that EPS uses, changes daily but we can compute how many PEVs they schedule on average. It appears that they use on average 11.5 PEVs a day and therefore more. In spite of the limitations, we assume that EPS would perform better in terms of the number of PCNs. In addition, we have shown in Section 6.2.1.4 that our planning tool does not only lead to a high number of PCNs but also that we are certain that it will meet the future KPI targets. Even though we cannot make prediction about the increase of the payment rate in the future, which finally leads to a bigger performance bonus, achieving more PCNs and performing better with regards to the PCN target decreases the likelihood of EPS getting a fine for not meeting one of the KPI targets.

Apart from that, we experienced during the computation that it takes 6 hours and 42 seconds to make such a planning, which is more or less equal to the 6 hour constraint we had. By further optimization of the computational efficiency, we are confident that the constraint will not be exceeded.

6.5.2 Comparison with ARS routing algorithm

In this section, we compare our routing algorithm with the one of ARS by creating routes based on the same inputs (including also the same forecast). In order to have a valid comparison between the two algorithms, we compare the routes from Monday till Saturday with 12 vehicles and without a night shift. Based on the outputs we retrieved from the ARS routing algorithm for one week (24.08.-30.08.2017), we compare both routes regarding the number of PCNs. The results are shown in Table 21.

	Number of PCNs		Improvement	
Weekday	**ARS algorithm**	**Our algorithm**	**Absolute**	**in %**
Thursday	1766	2447	681	39%
Friday	2061	2411	350	17%
Saturday	2376	2981	605	25%
Monday	1619	2457	838	52%
Tuesday	1787	2411	624	35%
Wednesday	1674	2417	743	44%
Total	**11283**	**15124**	**3841**	**34%**

Table 21 - Results of the number of PCNs comparing ARS algorithm to our algorithm

We conclude that our algorithm outperforms the current ARS algorithm by 34% for the weekdays Monday until Saturday. There are several explanations for this difference. First of all ARS uses a 4-hour-no-visit constraint of the neighborhoods but we use the stability function, which allows the PEV to return to neighborhoods more often although the expected number of PCNs will be less. Replacing this hard constraint by a soft constraint gives the algorithm more flexibility. However, this is based on the assumption that our stability function is correct. Another reason can be a better distribution of the vehicles. Moreover, the routing algorithm of ARS uses no Travel Distance Restrictions but always considers the eight nearest neighborhoods. In most cases, there are more neighborhoods within 1500 meters so our algorithm considers more options, while in some cases, it considers less due to this approach. Furthermore, we introduced the two swap methods and the progress factor, which can also lead to an improvement.

The 34% improvement shows that our algorithm clearly outperforms the implementation of ARS algorithm at its current state. This finally leads to more PCNs, meeting the targets faster, and more importantly less fines and more rewards for EPS.

6.6 Conclusion

In this chapter, we have tackled four sub questions regarding our original research question whether our planning tool improves the current situation. Throughout this chapter, we have shown that our stability function and Travel Time Restriction have a big impact on the routes and its results. Therefore, it is important to do further research on the stability function and see in practice to what extent the Travel Time Restriction can be increased (regarding the speed limitations of the PEV). Furthermore, we stated that we are convinced that our algorithm will manage to meet future KPI targets and that the unlimited visit day factor is probably preferable because the average number of neighborhoods that are not visited once a week is reduced by 30.56% and the number of forecasted PCNS only by 0.52%.

Unfortunately, we experienced that the ACO algorithm has difficulties to improve the solution of the greedy algorithm, when many PEVs (for instance 12) are deployed. We assume that one reason for this is that with more deployed PEVs, ants start to “steal” visits from the desired path of other ants. Another reason is that the number of neighborhoods with multiple visits increases. Apparently the ACO algorithm, which shows good results for the original TOP in the literature and in our case for a little number of PEVs, cannot handle the multiple visits as good as the greedy algorithm.

Moreover, we have shown that our Neural Network A forecast is the most accurate forecast with regards to the PCN ratio in comparison with the presented prediction models and that it also leads to the most PCNs in our simulation scenario.

Furthermore, we are convinced that EPS benefits from our planning tool but the routes of our algorithm need to be tested in practice before we can proof it. However, we did prove (assuming that our stability function is correct) that our greedy algorithm leads to 34% more PCNs than the current ARS algorithm when the same inputs are used.

7 Conclusion and Recommendations

In Section 1.4, we stated different research questions, which we tackled within the chapters of this research. In Chapter 2, we analyzed the current situation at EPS with regards to the routing and planning process. Chapter 3 covered the literature about different routing problems, prediction models, and theories about parking and payment behavior. In Chapter 4, we developed all inputs including a neural network prediction model based on historical data. These inputs were needed for our routing algorithm, which we designed in Chapter 5. Finally, we discussed the results of our prediction model and our routing algorithm in Chapter 6.

In this chapter, we reflect about the contribution of this research to the literature in Section 7.1 and we discuss the practical conclusion for ARS and EPS, including limitations and recommendations, in Section 7.2

7.1 Contribution to the literature

We introduced this routing problem as a new research problem, namely the TD-PTOPMVMC. This problem is a new generalization of the TOP in which we have multiple constraints, time-dependent inputs, a periodic planning horizon, and nodes (in this case neighborhoods) that can be visited multiple times a day. In order to deal with multiple visits, we introduced a stability function, which determines how many non-paying visitors, who already received a PCN, stay in the parking spot after a certain amount of time. This function decreases exponentially over time. Consequently, after a certain amount of time it makes sense to return to an attractive neighborhood the same day. To the best of our knowledge, this problem is new to the literature and it is also the first research to solve the parking enforcement problem on a large scale. The research of Summerfield et al. (2015) tackles the parking enforcement problem on a street level my solving the CPP with rewards. Next to the parking enforcement, this problem (at least the TOP with multiple visits) could have different applications, such as different patrol, inspection, or collection problems. Furthermore, we can think of a salesman problem, where, for instance, an ice cream salesman visits different locations and returns to a certain neighborhood after enough time has passed.

In order to solve this problem, we presented a routing algorithm that creates a solution based on a greedy algorithm and then tries to optimize this solution by constructing new solutions based on an ACO algorithm Even though, we showed that the ACO algorithm performed well for one vehicle, it had difficulties finding better solutions than the greedy function for 12 vehicles, which is the standard number of used vehicles by EPS (from Monday to Saturday). It seems that too many vehicles lead to an oversaturation, which leads to decrease of the learning effect of the ants. One possible reason is that ants start to “steal” visits from the desired path of other ants. Another reason is that this oversaturation leads to more multiple visits of neighborhoods the same day and we concluded that the ACO algorithm works better with less multiple visits and therefore when the problems is more similar to the original TOP (without multiple visits).

Apart from that, we have shown an approach of how to perform a data analysis. In our data analysis, we split the PCN ratio in its components, namely the visitor ratio and non-paying ratio, in order to analyze different effects regarding different time-, space-, or weather-related factors towards these ratios. In our opinion, this was a useful approach to gain a lot of specific insights. In this regard, we also showed the usefulness of clustering techniques, such as K-means clustering and the principal component analysis.

Furthermore, we have shown how a neural network can be applied to such a regression problem and which variables may be interesting to consider. To train our neural network, we used a test and training set to optimize the design of our training network based on the results of the test set. We have learned that adding an evaluation set to the test and training set, can be a valuable supplement to test the generalization capacity of the network. This is also discussed by Basheer and Hajmeer (2000). In our case, this could be done by taking some specific observations (e.g., a few neighborhoods or one specific hour) out of the original data set. These excluded observations would form the evaluation set. Afterwards, the remaining data set can be split in the test and training set and results of the trained model can be tested for overfitting on the test set and the generalization capabilities of the neural network can be evaluated based on the evaluation set.

Another interesting aspect, is our method to compute the upper bounds for the KPI targets. Uncertainty concerning specific targets, has a broad field of applications, which might be valuable to the literature.

7.2 Practical conclusion

This section contains a summary of the conclusions made in this research (Section 7.2.1), our limitations (7.2.2), and recommendations (7.2.3) that are relevant for EPS.

7.2.1 Summary of conclusions

This section briefly summarizes the conclusions of this research that are relevant for EPS. For more detailed conclusions regarding the research questions, we refer to the conclusions of the specific chapters.

In this research, we tackled the routing problem of EPS who require a planning tool that plans the routes for their scanning vehicles (denoted as PEVs) for a certain KPI period. It is also required to give an indication that all KPI targets of certain KPI areas will be met until the end of this KPI period. The targets are met by visiting a neighborhood, i.e., scanning all parked vehicles in a neighborhood. The owners of parked cars who did not pay for parking, receive a fine, which is called a PCN. In this research, we developed a planning tool by designing and implementing a routing algorithm in a Python platform and computed inputs that this algorithm uses. Our most important input is our prediction model to compute the expected number of PCNs. In this regard, we came up with the idea of predicting the number of PCNs by predicting the occupancy ratio and the PCN ratio and multiplying them by the number of parking spots in a neighborhood. The advantage of this is that, for instance, the occupancy ratio can be replaced afterwards.

Next to that, we found in our data analysis that the weather has no impact on the PCN ratio. However, we did see that the hour of the day and weekday have a strong impact. Furthermore, we showed that the payment rate has increased from approximately 89% to 90% within one year (1.6.2016-1.6.2017).

Finally, we presented our results in Chapter 6. We have proven that our neural network has a more accurate prediction than the current implementation of the ARS forecast (4%) and that our routing algorithm leads to better results (34%) than the current ARS routing algorithm within the constraint of 6 hours.

7.2.2 Limitations

This section discusses the limitations of our research. First of all, we had to make some assumptions. The first example is the variability of the paid-for parking hours, which we used for determining the upper bound of our PCN target (see Section 4.2.2). The second example is more important as it concerns the stability function, which is based on our assumption the expert opinion of ARS. Preferably, this function should be derived from data and estimated separately for every neighborhood. Unlike the stability function, the occupancy ratio is derived from data but not in a very accurate manner. A more accurate and time- and space dependent estimation is desirable. Both, the stability function and occupancy ratio, have a strong impact on the number of PCNs, therefore it is important to prioritize these in the future research.

Another limitation is that the planning has not been tested in practice yet. Especially for the travel time, the service time (scan duration), and the occupancy ratio, this leads to an uncertainty in the realization of the planning that the planning tool provides.

Moreover, there are some limitations that come with the design of our planning tool. For example, the commitment to computing the routes every day, leads to the problem that the algorithm cannot make a long term planning because the next day the planning will be recomputed. This limits, for instance, the possibilities regarding the distribution of the PEVs, on which we further elaborate in Section 7.2.3. Furthermore, we designed the algorithm to create the routes on a neighborhood level. Even though this is a requirement of EPS because they want to scan entire neighborhoods in the future, this limits the capability of the algorithm. If the routing was determined on a street level, the algorithm would have more flexibility due to a bigger solution space. However, a bigger solution space would also increase the complexity and computation time of the problem. Moreover, it would have been possible to include the number of scans and PCNs that are made while travelling to a scheduled neighborhood even with operating on a neighborhood level. Such an approach would have presented the reality better but would have also increased the complexity of the problem by making it a mixed TOP, as described in Section 3.1. In the end, we did not choose such an approach because it would have required an extensive data analysis of every single street (instead of every neighborhood). Regarding our data analysis of the neighborhoods and computation of the service time and the occupancy ratio, we had to deal with another limitation, namely that the PEVs did not always scan the neighborhoods entirely in the past. This was already encountered by ARS. If this had been the case, it would have been relatively easy to make predictions about the service times and also the occupancy ratio of neighborhoods.

7.2.3 Recommendations

First of all, we recommend ARS to make use of our presented greedy algorithm (see Chapter 5) with the parameter values, as presented in Section 6.2.1, in combination with a Neural Network A forecast (see Section 4.3.2.2). Furthermore, we recommend to use an unlimited Visit Day Factor due to the results of the sensitivity analysis (see Section 6.3.1.4).

From the limitations, we derive that still more research is required to derive the occupancy ratio and stability function from data. The occupancy ratio could be derived more accurately by reconstructing the streets that the PEVs have driven in the past and then dividing the number of scans (without double scans) in a street by the number of parking spots that the street has. In order to derive the stability function from data, historic data about the parking duration of non-paying (or at least paying) visitors is needed. In Section 4.3.4, we explained how this could be done. This data about the parking duration could also be used for the control chance. In Section 2.1.8, we discussed that the control chance is not a probability but the average number of PCNs within a not-paid-for parking hour. If the not-paid-for parking hours are divided by the average parking duration of a non-paying visitor, it would be possible to estimate the probability that a non-paying visitor is actually fined with a PCN.

Furthermore, we discussed in Section 4.4, that the travel times should be further investigated because it seems that they are underestimated. In practice, it is important to keep inspecting the travel times and also the service times and make adjustments when there are unexplained significant differences. Regarding the service times, we developed a method to compute them in Section 4.5. This could be an alternative to the current method. In order to implement this, more data is required to determine the adjustment factors for each neighborhood. At least one full scan of a neighborhood would be needed as a start. Thereafter, the factor could be adjusted and improved continuously. At some point when there is enough data of the visits, in which the neighborhood is scanned fully, it makes sense to consider applying a neural network to compute the service times.

Apart from the inputs, there are other points that should be considered in the future. In order to further improve the algorithm, we have the following ideas, which do not necessarily result in better solutions. Within this research, we developed an algorithm that created routes for the vehicles sequentially due to results of our findings in the literature. However, these findings were about the TOP, whereas we introduced a new generalization of the TOP with multiple visits. Due to our sequential approach, sometimes later visits of other vehicles have to be updated. With a parallel planning, this would not be necessary and the computation time would be reduced. However, it is not clear if this would increase or decrease the performance of the routes. As mentioned in Section 6.2.2, it could be interesting to add a saturation factor to the desirability function, which takes the number of recent visited neighborhoods that surround the considered neighborhood into account. This would help the greedy algorithm to go to neighborhoods where it has more feasible options to go to afterwards. Another, possibility is to assign every vehicle to certain neighborhoods at the beginning of the day. This would require a smart clustering algorithm that finds the best neighborhoods for every vehicle. Afterwards, the routing problem would not be a team orienteering problem but an orienteering problem since the routing of every vehicle would be considered as independent problems. For that reason, no recomputation due to latter visits would be required and maybe the metaheuristics would work better. However, it could also lead to a worse performance due to the limited neighborhoods options for every vehicle. We are certain though that the computation time would be reduced. In order to improve our proposed ACO algorithm, one might consider using time-dependent pheromones, since the entire problem especially with the multiple visits is very time-related. However, even though the time-dependency would be included in the learning effect, it is not guaranteed that time-dependent pheromones would really lead to an improvement regarding the multiple visits. Next to that, EPS could also consider other constructive heuristics (e.g., adaptive search).

Regarding our distribution strategy that we presented in Section 5.4.1, one might consider it also as a separate scheduling problem. If we did not create schedules on a daily basis but on a weekly basis, then it would be possible to schedule the KPI areas that needs to be visited for one week. This would probably lead to a more balanced distribution due to the fact that the algorithm could determine not to visit one KPI area today but for example the day after tomorrow.

As an alternative to the Visit Day Factor and the Target Factor, we discussed in Section 5.3 that one might also consider that every visit leads to a reward in euros if the fine of a KPI area is decreased. We argued that it is not possible to send the vehicles only based on the amount of euros to a neighborhood because neighborhoods in a KPI area with a neural or bonus status would never be visited because we do not have a direct impact on the payment rate. However, it would be possible to make an assumption that a certain number of scans increases the payment rate by a certain amount. Additionally, a visit could decrease the margin of error of the payment rate. The more a neighborhood is scanned, the smaller should be the difference between the payment rate of EPS and the municipality (see Section 4.2.1). By reducing the

margin of error of the payment rate, the bonus would actually be increased, but the problem remains that if not all KPI areas are at least in a neural state, no bonus is given.

In the future, the rule for receiving a PCN might be changed such that visitors can receive two PCNs on one day. If this is the case, the stability function needs to be adjusted because visiting neighborhoods a second time would not be a bad thing. Even more so, because neighborhoods that resulted in high number of PCNs during the first visit, would become even more attractive for a second visit. This, however, would preferable require an online algorithm that takes the actual number of PCNs of the first visit into account. This would mean that when the PEV drives through a neighborhoods and knows this visit resulted in an extraordinary high number of PCNs, that it would make sense to visit that neighborhood immediately afterwards again.

In Section 3.1, we presented also another idea for such an online application. In the future, one might consider to plan also the routes for the PEV within a scheduled neighborhood. For that reason, an exact algorithm for the Chinese postman problem could be applied. Since this problem is small enough to be solved to optimality in reasonable time, this could be implemented as an online or offline application. With an online application, the algorithm could take unexpected problems on the way into account and therefore adjust the route within the neighborhood in order be on time for the next scheduled neighborhood. Furthermore, we mentioned the mixed TOP, which does not only assign weights to nodes (neighborhoods) but also to the arcs (travel route between two neighborhoods). If the parking spots that are scanned by using a certain travel route are known, then it would be possible to make a prediction or estimation about the expected number of PCNs in this route. However, the occupancy ratio and PCN ratio of these parking spots, would require an additional analysis.

8 References

Adiv, A., & Wang, W. (1987). *On-street parking meter behavior* (No. UMTRI-86-37).

Aikoh, T., Abe, R., Kohsaka, R., Iwata, M., & Shoji, Y. (2012). Factors influencing visitors to suburban open space areas near a northern Japanese city. *Forests*, *3*(2), 155-165.

Amsterdam. (2017, July 6). About parking regime in Amsterdam. Retrieved from Amsterdam Web site:

https://www.amsterdam.nl/parkeren-verkeer/parkeertarieven

Anderson, T. K. (2009). Kernel density estimation and K-means clustering to profile road accident hotspots. *Accident Analysis & Prevention*, *41*(3), 359-364.

ANWB. (2017, July 6). About traffic congestion in the Netherlands. Retrieved from ANWB Web site: https://www.anwb.nl/verkeer/nederland/verkeersinformatie/dagelijkse-drukke-trajecten

Archetti, C., Hertz, A., & Speranza, M. G. (2007). Metaheuristics for the team orienteering problem. *Journal of Heuristics*, *13*(1), 49-76.

ARS T&TT. (2017, February 1). About ARS. Retrieved from ARS T&TT Web site: http//www.ars-traffic.com/about-ars

Awerbuch, B., Azar, Y., Blum, A., & Vempala, S. (1998). New approximation guarantees for minimum-weight k-trees and prize-collecting salesmen. *SIAM Journal on Computing*, *28*(1), 254-262.

Basheer, I. A., & Hajmeer, M. (2000). Artificial neural networks: fundamentals, computing, design, and application. *Journal of microbiological methods*, *43*(1), 3-31.

Bektas, T. (2006). The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega*, *34*(3), 209-219.

Boussier, S., Feillet, D., & Gendreau, M. (2007). An exact algorithm for team orienteering problems. *4OR: A Quarterly Journal of Operations Research*, *5*(3), 211-230.

Braekers, K., Ramaekers, K., & Van Nieuwenhuyse, I. (2016). The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering*, *99*, 300-313.

Butt, S. E., & Ryan, D. M. (1999). An optimal solution procedure for the multiple tour maximum collection problem using column generation. *Computers & Operations Research*, *26*(4), 427-441.

Chao, I., Golden, B. L., & Wasil, E. A. (1996). The team orienteering problem. European Journal of Operational Research, 88(3), 464-474.

Claassen, G. D. H., & Hendriks, T. H. (2007). An application of special ordered sets to a periodic milk collection problem. *European Journal of Operational Research*, *180*(2), 754-769.

Cordeau, J. F., Gendreau, M., Hertz, A., Laporte, G., & Sormany, J. S. (2005). New heuristics for the vehicle routing problem. In *Logistics systems: design and optimization* (pp. 279-297). Springer US.

M. Daszykowski, K. Kaczmarek, Y. Vander Heyden, B. Walczak (2007), Robust statistics in data analysis - A review, *Chemometrics and Intelligent Laboratory Systems, 85(2), 203-219.*

Eiselt, H. A., Gendreau, M., & Laporte, G. (1995). Arc routing problems, part I: The Chinese postman problem. *Operations Research*, *43*(2), 231-242.

Elliot, J. R., & Wright, C. C. (1982). The collapse of parking enforcement in large towns: some causes and solutions. *Traffic Engineering & Control*, *23*(HS-033 448).

Feillet, D., Dejax, P., & Gendreau, M. (2005). Traveling salesman problems with profits. *Transportation science*, *39*(2), 188-205.

Gavalas, D., Konstantopoulos, C., Mastakas, K., & Pantziou, G. (2014). A survey on algorithmic approaches for solving tourist trip design problems. *Journal of Heuristics*, *20*(3), 291-328.

Gavalas, D., Konstantopoulos, C., Mastakas, K., Pantziou, G., & Vathis, N. (2015). Heuristics for the time dependent team orienteering problem: Application to tourist route planning. *Computers & Operations Research*, *62*, 36-50.

Gendreau, M., Laporte, G., & Semet, F. (1998). A tabu search heuristic for the undirected selective travelling salesman problem. *European Journal of Operational Research*, *106*(2-3), 539-545.

Golden, B. L., Laporte, G., & Taillard, É. D. (1997). An adaptive memory heuristic for a class of vehicle routing problems with minmax objective. *Computers & Operations Research*, *24*(5), 445-452.

Graham, S. M., Joshi, A., & Pizlo, Z. (2000). The traveling salesman problem: A hierarchical model. *Memory & cognition*, *28*(7), 1191-1204.

Hochbaum, D. S. (1995). A nonlinear knapsack problem. *Operations Research Letters*, *17*(3), 103-110.

Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data clustering: a review. *ACM computing surveys (CSUR)*, *31*(3), 264-323.

Jiang, B. B., Chen, H. M., Ma, L. N., & Deng, L. (2011). Time-dependent pheromones and electric-field model: a new ACO algorithm for dynamic traffic routing. *International Journal of Modelling, Identification and Control*, *12*(1-2), 29-35.

Ke, L., Archetti, C., & Feng, Z. (2008). Ants can solve the team orienteering problem. *Computers & Industrial Engineering*, *54*(3), 648-665.

KNMI. (2017, June 22). About weather data of specific days. Retrieved from KNMI Web site: http://projects.knmi.nl/klimatologie/daggegevens/

Kok, A. L., Hans, E. W., Schutten, J. M. J., & Zijm, W. H. M. (2010). Vehicle routing with traffic congestion and drivers' driving and working rules.

Kok, A. L., Hans, E. W., & Schutten, J. M. J. (2012). Vehicle routing under time-dependent travel times: the impact of congestion avoidance. *Computers & operations research*, *39*(5), 910-918.

Kumar, M., Husian, M., Upreti, N., & Gupta, D. (2010). Genetic algorithm: Review and application. *International Journal of Information Technology and Knowledge Management*, *2*(2), 451-454.

Larsen, R. J., & Marx, M. L. (2012). *An introduction to mathematical statistics and its applications* (Vol. 5). Englewood Cliffs, NJ: Prentice-Hall.

LeCun, Y. A., Bottou, L., Orr, G. B., & Müller, K. R. (2012). Efficient backprop. In *Neural networks: Tricks of the trade* (pp. 9-48). Springer Berlin Heidelberg.

Liao, T. W. (2005). Clustering of time series data—a survey. *Pattern recognition*, *38*(11), 1857-1874.

Lin, S. W., & Vincent, F. Y. (2015). A simulated annealing heuristic for the multiconstraint team orienteering problem with multiple time windows. *Applied Soft Computing*, *37*, 632-642.

Mair, C., Kadoda, G., Lefley, M., Phalp, K., Schofield, C., Shepperd, M., & Webster, S. (2000). An investigation of machine learning based prediction systems. *Journal of Systems and Software*, *53*(1), 23-29.

Mladenović, N., & Hansen, P. (1997). Variable neighborhood search. *Computers & operations research*, *24*(11), 1097-1100.

Montemanni, R., & Gambardella, L. M. (2009). An ant colony system for team orienteering problems with time windows. *Foundation Of Computing And Decision Sciences*, *34*(4), 287.

Petiot, R. (2004). Parking enforcement and travel demand management. *Transport Policy*, *11*(4), 399-411.

Rochat, Y., & Taillard, É. D. (1995). Probabilistic diversification and intensification in local search for vehicle routing. *Journal of heuristics*, *1*(1), 147-167.

Sarkar, K., Ghalia, M. B., Wu, Z., & Bose, S. C. (2009). A neural network model for the numerical prediction of the diameter of electro-spun polyethylene oxide nanofibers. *Journal of materials processing technology*, *209*(7), 3156-3165.

Souffriau, W., Vansteenwegen, P., Berghe, G. V., & Van Oudheusden, D. (2010). A path relinking approach for the team orienteering problem. *Computers & operations research*, *37*(11), 1853-1859.

Summerfield, N. S., Dror, M., & Cohen, M. A. (2015). City streets parking enforcement inspection decisions: The Chinese postman's perspective. *European Journal of Operational Research*, *242*(1), 149-160.

Tang, H., & Miller-Hooks, E. (2005). A tabu search heuristic for the team orienteering problem. *Computers & Operations Research*, *32*(6), 1379-1407.

Tang, H., Miller-Hooks, E., & Tomastik, R. (2007). Scheduling technicians for planned maintenance of geographically distributed equipment. *Transportation Research Part E: Logistics and Transportation Review*, *43*(5), 591-609.

Thimbleby, H. (2003). The directed Chinese postman problem. *Software: Practice and Experience*, *33*(11), 1081-1096.

Tricoire, F., Romauch, M., Doerner, K. F., & Hartl, R. F. (2010). Heuristics for the multi-period orienteering problem with multiple time windows. *Computers & Operations Research*, *37*(2), 351-367.

Van der Heijden, M.C., van der Wegen, L.L.M. (September 2011). *Logistiek Management* [lecture guide]. Department of Industrial Engineering and Management, University of Twente, Enschede.

Van Hal, K. (February 2015). When and Where to Fly and Stand by (Unpublished master thesis). Department of Industrial Engineering and Management, University of Twente, Enschede.

Vansteenwegen, P. (2009a). Planning in tourism and public transportation. *4OR: A Quarterly Journal of Operations Research*, *7*(3), 293-296.

Vansteenwegen, P., Souffriau, W., Berghe, G. V., & Van Oudheusden, D. (2009b). Iterated local search for the team orienteering problem with time windows. *Computers & Operations Research*, *36*(12), 3281-3290.

Vansteenwegen, P., Souffriau, W., Berghe, G. V., & Van Oudheusden, D. (2009c). Metaheuristics for tourist trip planning. In *Metaheuristics in the service industry* (pp. 15-31). Springer Berlin Heidelberg.

Vansteenwegen, P., Souffriau, W., & Van Oudheusden, D. (2011). The orienteering problem: A survey. *European Journal of Operational Research*, *209*(1), 1-10.

Van Urk, R., Mes, M. R., & Hans, E. W. (2013). Anticipatory routing of police helicopters. *Expert systems with applications*, *40*(17), 6938-6947.

Verbeeck, C., Sörensen, K., Aghezzaf, E. H., & Vansteenwegen, P. (2014a). A fast solution method for the time-dependent orienteering problem. *European Journal of Operational Research*, *236*(2), 419-432.

Verbeeck, C., Aghezzaf, E. H., & Vansteenwegen, P. (2014b, November). Solving the Stochastic Time-Dependent Orienteering Problem. In *MOSIM 2014, 10ème Conférence Francophone de Modélisation, Optimisation et Simulation*.

Wang, J., Yan, R., Hollister, K., & Zhu, D. (2008). A historic review of management science research in China. *Omega*, *36*(6), 919-932.

White, H. (1992). *Artificial neural networks: approximation and learning theory*. Blackwell Publishers, Inc.

Zhang, J. (2011, August). Modeling and Solution for Multiple Chinese Postman Problems. In *International Conference on Computer Science, Environment, Ecoinformatics, and Education* (pp. 520-525). Springer Berlin Heidelberg.

9 Appendix

9.1 Appendix A – Research method of problem review

In scopus, we look for "street parking enforcement" and only found two articles. Only one of these articles seems interesting and therefore we extend the search by deleting the word "street". By doing so we find 48 articles of which only one article seems to be related to routing problems. This becomes clear when we add the search word "routing" ("parking enforcement"AND"routing") because only that one remains, which is the same as we found previously. It seems that there is only one article regarding this problem, which is called "City streets parking enforcement inspection decisions: The Chinese postman's perspective" (Summerfield, Dror & Cohen, 2015).

9.2 Appendix B – Algorithm benchmark of Ke et al. (2008)

The rewards obtained by the seven algorithms

Group	ACO–TOP	CGW	TMH	GTP	GTF	FVF	SVF
1.2	**149.1**	148.5	148.8	**149.1**	**149.1**	**149.1**	**149.1**
1.3	125.0	**125.6**	124.7	125.0	125.0	125.0	125.0
1.4	**101.0**	99.3	**101.0**	**101.0**	**101.0**	**101.0**	**101.0**
2.2	**190.5**	190.0	190.0	**190.5**	**190.5**	**190.5**	**190.5**
2.3	**136.4**	135.9	135.9	**136.4**	**136.4**	**136.4**	**136.4**
2.4	**94.5**	**94.5**	**94.5**	**94.5**	**94.5**	**94.5**	**94.5**
3.2	**496.0**	488.5	492.0	494.5	**496.0**	**496.0**	**496.0**
3.3	**411.5**	403.0	408.0	**411.5**	**411.5**	**411.5**	**411.5**
3.4	**336.5**	332.5	335.0	**336.5**	**336.5**	**336.5**	**336.5**
4.2	915.6	875.7	895.1	904.9	908.5	914.0	**916.2**
4.3	853.8	815.1	844.3	845.5	852.5	853.0	**855.6**
4.4	798.1	766.1	784.6	800.1	802.3	801.7	**803.2**
5.2	**897.6**	890.6	886.8	892.6	897.4	895.8	897.0
5.3	783.4	776.6	775.8	781.4	**783.6**	**783.6**	**783.6**
5.4	**708.8**	696.0	699.0	707.5	**708.8**	**708.8**	**708.8**
6.2	**819.3**	814.9	818.2	813.8	818.7	**819.3**	**819.3**
6.3	**792.8**	787.5	783.0	**792.8**	**792.8**	**792.8**	**792.8**
6.4	714.0	**716.4**	712.8	714.0	714.0	714.0	714.0
7.2	**642.7**	633.9	633.5	639.6	641.4	640.6	642.5
7.3	**599.9**	585.5	592.5	596.7	597.7	597.1	599.3
7.4	**519.1**	497.4	514.6	517.2	516.9	516.9	518.9

Table 22 – Benchmark of Ke et al. (2008) showing the rewards obtained by different algorithms

The maximal computational times of the seven algorithms

	ACO–TOP	CGW	TMH	GTP	GTF	FVF	SVF
Set 1	7.9	15.4	NA	10.0	5.0	1.0	22.0
Set 2	3.8	0.9	NA	0.0	0.0	0.0	1.0
Set 3	8.5	15.4	NA	10.0	9.0	1.0	19.0
Set 4	51.1	934.8	796.7	612.0	324.0	121.0	1118.0
Set 5	25.2	193.7	71.3	147.0	105.0	30.0	394.0
Set 6	20.3	150.1	45.7	96.0	48.0	20.0	310.0
Set 7	44.7	841.4	432.6	582.0	514.0	90.0	911.0

Table 23 - Benchmark of Ke et al. (2008) showing the computation times of different algorithms

9.3 Appdenix C – Neighborhood factor

GeoId	Neighborhood factor
2	2.895
3	2.419
4	1.622
5	2.870
6	1.996
7	2.643
8	1.579
9	2.157
10	0.764
11	1.094
12	1.021
13	0.803
15	1.186
16	1.801
17	1.306
18	0.911
19	2.538
20	1.083
21	1.406
22	1.354
23	0.983
24	0.862
25	0.858
26	1.242
27	0.961
28	2.132
29	1.660
30	0.812
31	0.515
33	0.431
35	0.584
36	0.364
37	0.471
38	0.886
39	0.890
40	4.025
41	1.277
42	2.083
43	3.439
44	1.830
45	1.782
46	0.781

47	0.684
48	0.664
49	0.803
50	0.736
51	1.078
53	1.887
54	1.243
55	0.642
56	0.856
57	1.021
58	0.970
60	0.524
61	4.948
62	2.978
63	1.285
64	0.624
65	0.528
66	0.488
67	0.810
68	0.595
69	0.481
70	1.008
71	1.927
72	0.693
73	0.529
74	0.743
75	0.622
76	0.706
77	0.675
78	0.826
79	1.609
80	0.493
81	0.767
82	0.453
83	0.939
84	0.873
85	1.201
86	0.992
87	1.135
88	0.934
89	1.237
90	0.942
91	0.687
92	0.617

93	1.763
94	0.861
95	1.509
96	0.641
98	0.923
99	0.876
100	0.839
101	1.180
102	1.023
103	0.872
104	0.879
105	0.724
106	0.755
107	0.476
108	0.578
109	0.495
110	0.443
111	0.551
112	0.543
113	1.800
114	1.070
115	1.436
116	0.694
117	0.622
118	0.765
120	0.806
121	2.173
122	0.665
123	0.695
124	0.574
125	0.553
126	0.274
127	1.767
128	1.909
129	0.570
130	0.803
131	0.698
132	1.408
133	1.914
134	0.664
135	1.287
136	1.094
137	2.018
138	0.692

139	0.612
140	0.474
142	0.598
143	0.918
144	0.789
145	0.583
147	0.825
148	0.679
149	0.841
150	0.542
151	0.813
152	0.703
153	1.365
154	1.178
155	0.990
156	0.767
157	0.820
158	0.751
159	0.872
160	0.941
161	1.316
162	0.735
163	0.836
164	0.546
165	0.568
166	0.982
167	0.569
168	1.813
169	3.925
170	2.199
171	2.312
172	1.372
173	1.647
175	2.124
176	2.573
177	0.757
180	3.009
181	1.541
182	0.000
183	2.185
184	1.141
186	3.515
187	0.404
201	1.988

202	0.876
203	0.732
204	1.646
206	1.070
207	0.399
208	0.662
211	1.669
212	4.435
213	2.428
214	1.799
215	0.444
216	0.555
217	0.767
222	0.900
223	0.965
224	0.664
225	3.156
226	0.959
227	1.032
228	0.653
229	0.753
230	0.396
233	5.957
234	2.086
235	2.791
236	12.684
237	4.144
238	5.979
253	3.083
266	0.515
267	1.248
268	1.009
273	0.946
274	1.597
275	3.543
276	2.215
277	0.805
278	1.867
279	1.488
280	2.035
281	1.458
282	1.673
283	6.002
284	2.671

294	1.039
295	0.815
297	1.342
298	2.140
301	1.593
303	1.669
304	1.366
305	2.461
309	1.934
310	1.041
311	1.654
312	0.758
314	0.943
315	0.742
316	1.477
317	0.943
318	1.343
319	0.629
320	0.785
321	1.336
322	1.960
323	0.864
324	0.961
325	0.680
328	11.295
329	3.115
330	5.540
334	1.977
335	0.523
336	0.909
337	1.407
338	0.984
339	0.686
340	1.492
341	0.647
342	0.538
343	2.955
344	3.564
345	5.615
346	1.304
347	9.099
349	0.967
351	4.535
353	2.660

398	1.796
406	0.000
433	1.440
435	2.764
436	1.994
437	0.660
438	3.864
439	2.768
440	2.747
441	1.605
442	1.186
475	4.192
476	1.117
477	0.667
478	0.866
479	0.538
480	0.501
481	1.105
482	0.447
483	0.419
485	2.380
486	0.460
487	0.544
488	1.656
489	0.461
490	1.061
491	1.029
492	0.866
493	0.521
498	1.701
502	2.272
503	1.852
504	1.441
506	3.875
507	1.443
508	2.823
509	3.860
510	1.820
511	0.678
512	1.427
513	2.438
514	2.258
515	1.681
516	4.428

518	2.687
519	2.082
523	2.104
526	0.744
527	1.559
528	3.112
529	1.288
530	1.148
531	1.903
534	2.858
535	1.608
536	1.505
537	1.644
539	6.501

Table 24 – Table of the neighborhood factor

9.4 Appendix D – Computation of the increase of the expected number of PCNs

Let us consider an example to explain how the current increase of the expected number of PCNs due to a visit is computed:

Neighborhood 15 has at any time t a service time of 30 minutes ($s_{15,t,d} = 30\text{min}$) and the number of PCNs is always predicted to be 10 ($P_{v_{15,3,t,m,0}} = 10$). On a given day, neighborhood 15 is scanned between 10:30 and 11:00 by PEV 1 ($v_{15,3,10.30,1,0}$). We consider to visit neighborhood 15 another time with PEV 2 starting at 14:00 ($v_{15,3,14.00,2,0}$). For the stability function, we need to determine the visit time between these two visits. As we explained in Section 5.2, the difference of the start of the visit at 14:00 and finish time of the prior one, can be denoted as: $t_{v_{15,3,14.00,m,0}} - t_{f_{v_{15,3,14.00,m,-1}}}$. The time between the two visits is 3 hours. As a next step, we need to estimate how many visitors that received a PCN between 10:30 and 11:00 are still at the neighborhood, since they do not get a second PCN and therefore we have to subtract these. We denote the number of visitors that remain until $v_{v_{j,n,t,0}}$ as $R_{v_{j,n,t,m,0}}$. Let us say that $R_{v_{15,3,14.00,m,0}}$ is 2 in this example. Therefore, we can only expect 8 PCNs between 14:00 and 14:30. We denote this updated value as $P'_{v_{15,3,14.00,m,0}}$, which is equal to $P_{v_{15,3,14.00,m,0}} - R_{v_{15,3,14.00,m,0}}$. If we want now to add another visit of another PEV at 13:00, it gets more complex, because not only do we have to estimate $R_{v_{15,3,13.00,m,0}}$, but also the decrease of PCNs for the visit that is scheduled afterwards at 14:00. This decrease of PCNs of the visit at 14.00 is expressed as $R_{v_{15,3,13.00,m,1}}$. If we do visit the neighborhood at 13:00, we have to update the number of PCNs of the latter visit at 14:00 by applying: $P'_{v_{15,3,13.00,m,1}} = P_{v_{15,3,13.00,m,1}} - R_{v_{15,3,13.00,m,1}}$.

As discussed in Section 4.3.4, we compute $R_{v_{j,n,t,0}}$ by using the overlap of PCNs of these two visits and stability function $S(\Delta_t)$ to compute the fraction of non-paying visitors that still remain. Since Δ_t is equal to $t_{v_{j,n,t,m,x}} - t_{f_{v_{j,n,t,m,x-1}}}$, $S(\Delta_t)$ is computed as follows:

$$S(\Delta_t) = 1 - 0.4^{\left(t_{v_{j,n,t,m,x}} - t_{f_{v_{j,n,t,m,x-1}}}\right)}$$

$O_{v_{j,n,t,x}}$ computes the overlap Of PCNs between the current visit and the previous visit. This is important because the potential number of non-paying visitors that remains depends on the forecasted number of PCNs of both visits. If we forecast only 1 PCN for the prior visit and 5 for second, then there is only 1 non-paying visitor who might still remain. The overlap is computed as follows:

$$O_{v_{j,n,t,x}} = \max(P_{v_{j,n,t,m,x}} - P_{f_{v_{j,n,t,m,x-1}}}, 0)$$

Finally, the number of remaining visitors for the later visit $R_{v_{j,n,t,m,0}}$ is:

$$R_{v_{j,n,t,x}} = O_{v_{j,n,t,m,x}} * S(t_{v_{j,n,t,m,x}} - t_{f_{v_{j,n,t,m,x-1}}})$$

So, whenever a neighborhood is considered, the ant uses the updated value of PCNs, which is the forecasted number of PCNs minus the remaining visitors from the prior visit.

$P'_{v_{j,n,t,m,x}} = P_{v_{j,n,t,m,x}} - R_{v_{j,n,t,m,x}}$.

If the neighborhood has been visited afterwards that day, we need to add the decrease due to the remaining number of non-paying visitors that influences the visit afterwards:

$\eta_{v_{j,n,t,0}} = \frac{(P'_{v_{j,n,t,0}} - R_{v_{j,n,t,1}}) * T_{v_{j,n,t,0}} * V_{v_{j,n,t,0}}}{c_{i,j,t,d} + s_{j,t,d} + p_{v_{j,n,t,0}} * c_{j,b_{j,t,d}}}$.

However, it can also happen that a visit has already been decreased before. In that case we only have to subtract the difference of the old updated and new updated PCNs, which is:

$\Delta_{Update_{v_{j,n,t,m,1}}} = P'_{old_{v_{j,n,t,m,1}}} - P'_{new_{v_{j,n,t,m,1}}}$.

Therefore, $\Delta_{Update_{v_{j,n,t,m,1}}}$ is either 0 or a positive number. Finally, the increase of expected number of PCNs is determined as follows:

$$\Delta_{P'_{v_{j,n,t,m,0}}} = P'_{v_{j,n,t,m,0}} - \Delta_{Update_{v_{j,n,t,m,1}}}$$

9.5 Appendix E – Finding a distribution for the normalized data of the PCN ratios

After having normalized all historical PCN ratios of the data set and filtering out the excluding all scans below 106 scans due to the same reasoning as in Section 4.3.2.2, we plot the frequencies of each PCN ratio, i.e., how many times a certain PCN ratio occurs in the data set. This plot is shown in Figure 47.

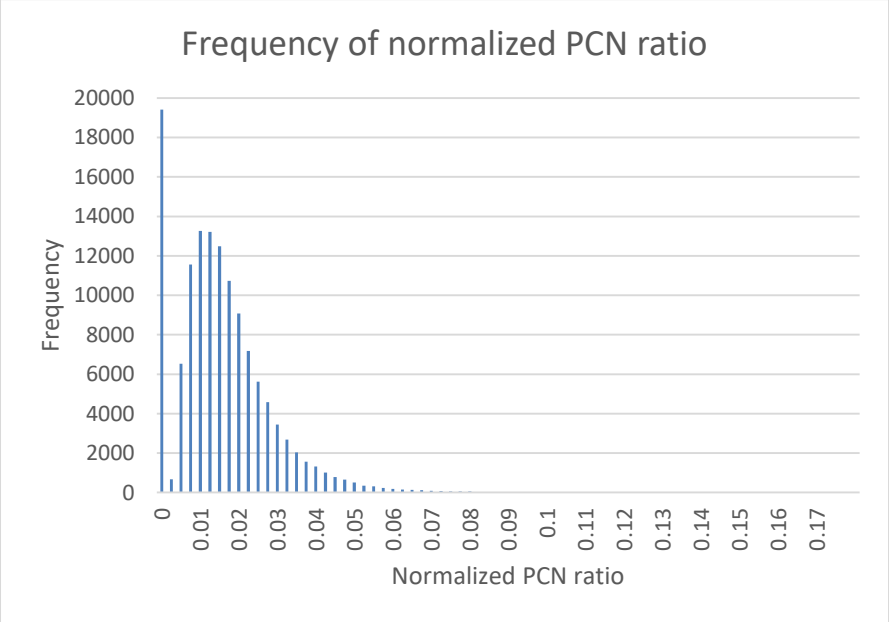


Figure 47 – Frequency of the normalized PCN ratio in the data set

From this figure, we conclude two things. Apparently 14.89% of all observations have a PCN ratio of 0. It does not seem that these zeros belong to a certain group of neighborhoods or time period. Not regarding the zeros, the distribution seems to follow a lognormal distribution. Therefore, we say that 14.89% of the simulated PCN ratios will have a ratio equal to 0. For the other 85.11%, we investigate whether they are indeed distributed according to a lognormal distribution. This will be the case if the lognormal function of the PCN ratios is normally distributed. Figure 48 shows the outcome of this investigation.

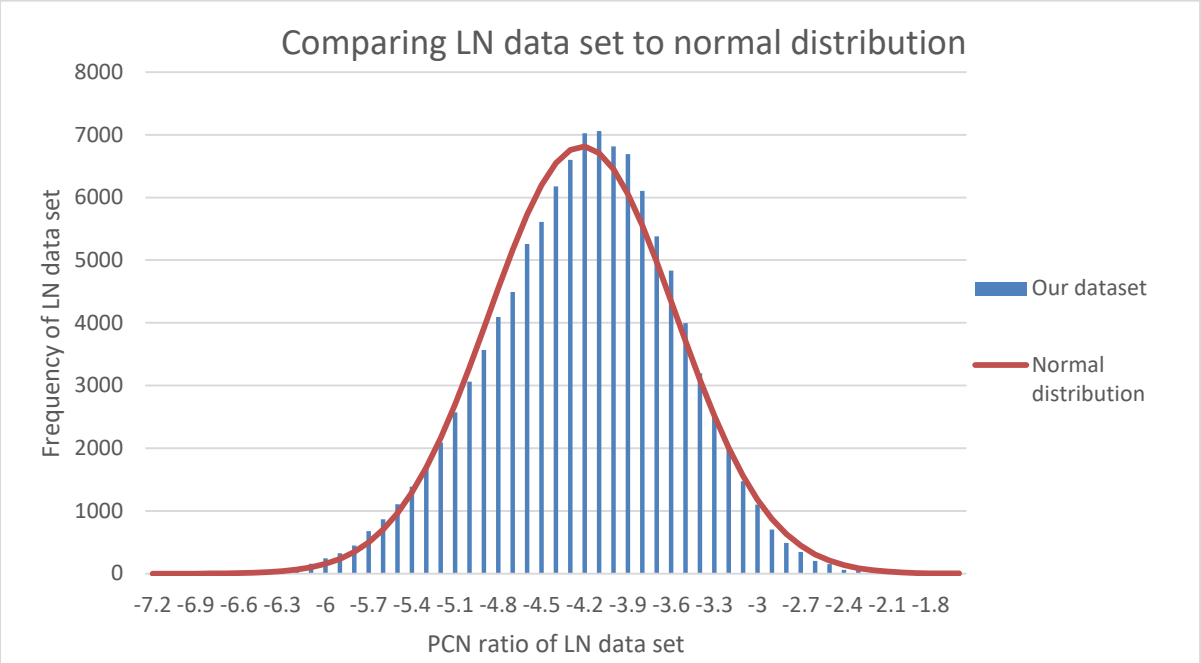


Figure 48 – The lognormal values of the normalized value in comparison with a normal distribution

Figure 48 shows the lognormal values of the distribution of our normalized data set and compares it to a normal distribution with a mean of -4.266 and standard deviation of 0.649. It seems that indeed the lognormal values are normally distributed even though it is slightly skewed to the right. In addition with the fact that 14.89% of the ratios will be zeros, we conclude that our normalized data set is follows a mixed lognormal distribution.

Finally, our simulation works as follows:

We generate a random number between 0 and 1. If the number is below 0.1489, the PCN ratio is 0. Otherwise, we generate a new number. For this value, we apply an inverse lognormal function. This gives us a random value that is distributed according to our found distribution.