# Trip assignment for the retail industry

Author:
R.J. Hafkenscheid

Supervisors and committee members University of Twente:
Dr. ir. M.R.K. Mes
Dr. ir. J.M.J. Schutten

Supervisor and committee members RGB$^{+}$ Automatisering:
R. Groot Beumer

Supervisor Peter Appel Transport:
G.J. Neeft

Colloquium date:
March 24, 2017

# Preface

It is a great pleasure to present my thesis 'Trip assignment for the retail industry'. With this thesis I complete the specialization track 'Production and Logistic Management' within the master program 'Industrial Engineering and Management' at the University of Twente. I conducted my thesis at RGB$^+$ Automatisering in Raalte. RGB$^+$ Automatisering is an IT company within the OV-Software Group, specialized in logistic software. Peter Appel Transport (a large carrier, with the main office established in Middenmeer) kindly provided the opportunity to perform this project: a customer of RGB$^+$ Automatisering Transplan Transport Management System. My thesis was supervised by the department of 'Industrial Engineering and Business Information Systems' (IEBIS), of the faculty of 'Behavioural, Management and Social Sciences' (BMS) at the University of Twente.

This graduation project had a tough and slow start (I even started on another project that was unfortunately canceled halfway), since we faced a lot of troubles defining the outlines of this project. In the end we were able to overcome these troubles and made some progress. I am very happy that I can successfully finish my master program. I would like to thank my friends, family, and colleges for their moral support throughout the project.

First, I would like to thank Martijn Mes and Marco Schutten (both University of Twente) for their critical, but often useful feedback on the scientific parts of my thesis. Furthermore, Remco Groot Beumer (RGB$^+$ Automatisering) and Gert-Jan Neeft (Peter Appel Transport) were of great help with their practical view on the subject and by their feedback on the daily progress. Furthermore, I would like to mention the people working at OV-Software and Simacan (also OV-Software group), who were always willing to discuss specific issues. Finally, I would like to thank my dad, Anton Hafkenscheid, for editorial assistance. Thanks all of you!

Enjoy reading my thesis. If you have any questions or remarks left, please do not hesitate to contact me, I am ready to help you out.

Rogier Hafkenscheid
Enschede, March 2017

# Executive Summary

Within the retail industry, companies often outsource their transport movements between distribution centers and stores. Carriers (such as Peter Appel Transport; PAT) are hired to execute those movements. However, retailers still want to control parts of the logistical planning. Therefore, retailers often place *orders* for full truckloads that need to be distributed throughout a predefined sequence of *stops*: a *trip*. The duration of *trips* is not long enough to keep *drivers* and *vehicles* (*resources*) busy for an entire working day. Therefore, carriers want their *resources* to execute several *trips* subsequently. Furthermore, *trips* need to be executed by various combinations of *resources*: a *driver* and several types of *vehicles* (e.g., a *Truck* and an *Eurotrailer*). The corresponding planning process is executed by planners of the carrier.

At present, planners at PAT manually combine *resources* to form *resource-groups* and then manually assign *trips* to those *resource-groups*. Planners take the required properties of *resources* (e.g., *vehicle-type*, *drivers-license*, etc.) into account, based on the characteristics of the *trip* (e.g., addresses, volume of the load, and cooling requirements). We call the problem of combining *resources*, and scheduling, routing, and assignment of *trips* the *trip-assignment-problem*. Because the planning process is currently executed manually, both RGB$^+$ Automatisering and Peter Appel Transport have the desire to automatically solve the *trip-assignment-problem*. Therefore, the goal of this research is the following:

*Develop an algorithm for solving the trip-assignment-problem at PAT.*

We solve the problem by subsequently executing the following four steps: (*1*) define the *trip-assignment-model*; (*2*) develop a *construction-heuristic* based on a parallel scheduling method; (*3*) develop an *improvement-heuristic* based on the Simulated Annealing *metaheuristic*; and (*4*) use the heuristics to develop the Trip Assignment Solver tool, written in C# on the Microsoft .NET Framework. The choices we make are based on our literature review and the characteristics of the *trip-assignment-problem* (i.a., we use heuristics, mainly because of the problem size of $\approx 1000$ *trips* per day).

We validate our solution using the expertise of the planners at PAT and we execute several experiments using different settings for the parameters of our heuristics. One of the main targets is to perform experiments using different aggregation levels: per *department* (*Local*), per group of *departments* (*Regional*), and all *departments* at once (*National*). The *construction-heuristic* uses approximately 40 minutes to solve the problem at the *National* level (shorter at lower levels). The *total-costs* can be decreased by 1.7% and 2.5% when switching from the *Local* to *Regional* level and *Regional* to *National* level respectively. Our *improvement-heuristic* retrieves improved results for both the routing and assignment aspects, compared to the *construction-heuristic*. Strangely, our *improvement-heuristic* achieves the best results at the *Regional* aggregation level, which might be a result of our parameter selection. After the execution of the *improvement-heuristic* the average *total-cost* is decreased by 12%. However, the *runtime* of the algorithm is unrealistically large (up to 84 hours per day of scheduling). We also experiment with adjusted settings for the Simulated Annealing *metaheuristic* and we find that similar results (*total-cost* +0.3%) can

be achieved using 5 times shorter *runtimes* (up to 16 hours). We conclude that shorter *runtimes* in this case counterbalance the small increase of costs.

The main benefit of this research is that we are able to develop the *trip-assignment-model*, as well as a first step to the development of the required algorithms for solving the model. A comparison between manually and automatically created scheduled is currently missing, mainly due to the bad quality of data (both unavailable and dirty). Therefore, we cannot draw any sound conclusions related to practical scheduling efficiency (financial benefits of automatically over manually created schedules), but we can say that higher aggregation levels (*Regional* and *National*) might be beneficial for efficiency reasons. Although our model and solution are not perfect, there are possibilities for partial implementation at first. Examples are: (*1*) as a decision support system (the solution advises planners in their work); (*2*) automatically creating a base schedule, followed by manual refinements; or (*3*) scheduling only a subset of the *trips* (i.e., decreasing the problem size). Even a partial implementation requires measures to improve data quality by (*A*) consequently using the resource-availability schedules, and (*B*) gathering and storage of information regarding the properties (i.a., the required volume and *time-windows*) of *trips*. Furthermore, we recommend to start implementation to start with a partially automated scheduling procedure by reducing the problem size (thus avoiding impossibly long *runtimes*).

# Contents

# Chapter 1

# Introduction

This thesis is about road transport planning and scheduling. In road transport, *orders* are executed by a combination of *drivers* and *vehicles*. Assigning these *resources* to *orders* is a complex task that is usually supported by a software system. The Transplan Transport Management System (Transplan TMS) developed by RGB$^+$ Automatisering, is a system that intents to support planners by visualizing manually created transport schedules. It visualizes the schedule and allows the planners to work (using a computer) on the manual planning process (explained in Section 1.2.2). By now, the Transplan TMS is used by several customers of RGB$^+$ Automatisering. They use this software to manually schedule customer *orders*, trucks, trailers, *drivers*, *trips*, and charters. In collaboration with one of their customers, RGB$^+$ Automatisering has the intention to extend Transplan TMS with a module that assists planners in automatically (and, therefore, more efficiently) allocating *drivers*, *vehicles*, and *orders*.

Definition

A **schedule** is a set of *trips* (involving loads related to *orders*) that are executed by certain *resources* within a certain period of time. Each part is provided with information regarding *orders*, locations, times, and *resources*.

## 1.1 Project initiators

This section describes the stakeholders that started this project. We discuss two stakeholders: First, Section 1.1.1 is about RGB$^+$ Automatisering. Then, we introduce Peter Appel Transport in Section 1.1.2, which is the customer of RGB$^+$ Automatisering that is involved in this project.

### 1.1.1 RGB$^+$ Automatisering

RGB$^+$ Automatisering is a small (9 FTE) Dutch software company that is dedicated to design, develop, and exploit software for logistic companies. One of their products is the Transplan TMS that enables planners to visualize schedules, administrate *orders*, and manually plan the combination of *orders*, *drivers*, trucks or tractors, and trailers (*vehicles*). Most customers using their software are *retail-distribution* companies (for an explanation on *retail-distribution*, see Section 1.2.1). RGB$^+$ Automatisering is the main initiator of this project. Their aim is to add automation of transport planning to their Transplan TMS.

### 1.1.2 Peter Appel Transport

Peter Appel Transport (PAT) is a large size carrier company with approximately 1250 *drivers* employed, and owning over 800 load carrying trucks and tractor-trailer combinations. Servicing from over 40 bases, these *drivers* and *vehicles* cover an average of approximately 1000 *orders* per day. An *order* at PAT is in most cases (98%, see Section 3.4) defined as a set of *stops*, given by the customer, for a *driver*, linked to a fully loaded combination of *vehicles* (*Hire-orders* and *Trip-orders*, see Section 1.2.1). On average, their *drivers* and *vehicles* serve four *stops* per *order*, resulting in an average of approximately 4100 *stops* per day (see for example Figure 1.1). With these numbers, PAT takes the 14th place in the Dutch top 100 logistic companies (Logistiek.nl, 2015).

| | Trip-orders | Hire-orders | Network-orders |
|---|---|---|---|
| Trip start- & end-locations | • | • | |
| Trip start-time & end-time | • | •[1] | |
| Pick-up/delivery locations | • | | • |
| Pick-up/delivery time-windows | • | | •[2] |
| Full truckload (Exclusive usage) | • | • | |
| Partial truckload | | | • |

[1] End-time is an estimation

[2] Some time-windows are pretty wide (for instance an entire day).

**Table 1.1:** Properties of order-types.

Most customers of PAT are in *retail-distribution*. In *retail-distribution*, shops are replenished from a warehouse or factory. A minor part of *orders* involve goods that are transported from one fixed location to another fixed location, called *network-transport*. For our explanation on *retail-distribution*, *network-transport*, and the types of *orders* that are involved, see Section 1.2.1.

## 1.2 Motivation

As an introduction to the rationale behind this project (motivation), we start with some definitions regarding the transportation process at PAT. First, we briefly explain the transportation process in Section 1.2.1, distinguishing three kinds of *orders*. We explain the current planning process in Section 1.2.2. The motivation of stakeholders to improve the planning process is discussed in Section 1.2.3, followed by the purpose of this research in Section 1.2.4. We describe the problem in Section 1.2.5, and finally, we treat the scope of this project in Section 1.2.6.

Definition

A ***trip*** is a planned sequence of multiple predefined *stops*, containing *time-windows* and locations of *stops*.

### 1.2.1 Transportation process

Three types of *orders* can be distinguished in the process of transportation. The first type consists of a complete *trip* with a full truckload, referred to as a *Trip-order*. The second type is called *Hire-order*, defining the reservation of a *vehicle-combination* and a *driver* for an agreed-on amount of time. Typical for this type of *order* is that the start and end *stops* are defined (i.e., both time and location are known), and intermediate *stops* and tasks are unknown in advance. Furthermore, full truckloads are assumed. *Trip-orders* and *Hire-orders* are automatically transformed to *trips*. The third type is called *Network-order*, which involves transportation movements from door (pick-up *stop*) to door (delivery *stop*), with a partial truckload. Independent *Network-orders* can sometimes be combined to form a *trip*. Planners usually do this to achieve improved efficiency (improving the use of available space inside *vehicles*, decreasing the amount of driven kilometers and decreasing the numbers of *vehicles* needed for the jobs).

Note

A **full truckload** does not necessarily imply that the *vehicle-combination* is always maximally loaded literally, the customer claims the exclusive rights on a *vehicle-combination*.

Due to a large range of factors (contracts, collective labor agreements, regulation concerning driving and resting times), *drivers* can be on duty maximally 15 hours a day. Depending on different factors (distance, velocity, traffic circumstances, the number of hours a *driver* is available etc.), a *driver* can sometimes do multiple *trips* on a single working day. This subsection further explains the differences between the *order-types* (see Table 1.1) and gives corresponding examples.

#### Trip-orders

A large part of the transport *orders* involves *Trip-orders*, which can and should (due to agreements with customers) directly be converted into *trips*. This means that the sequence of *stops* within a *Trip-order* cannot be influenced: the *trip* as given should be executed

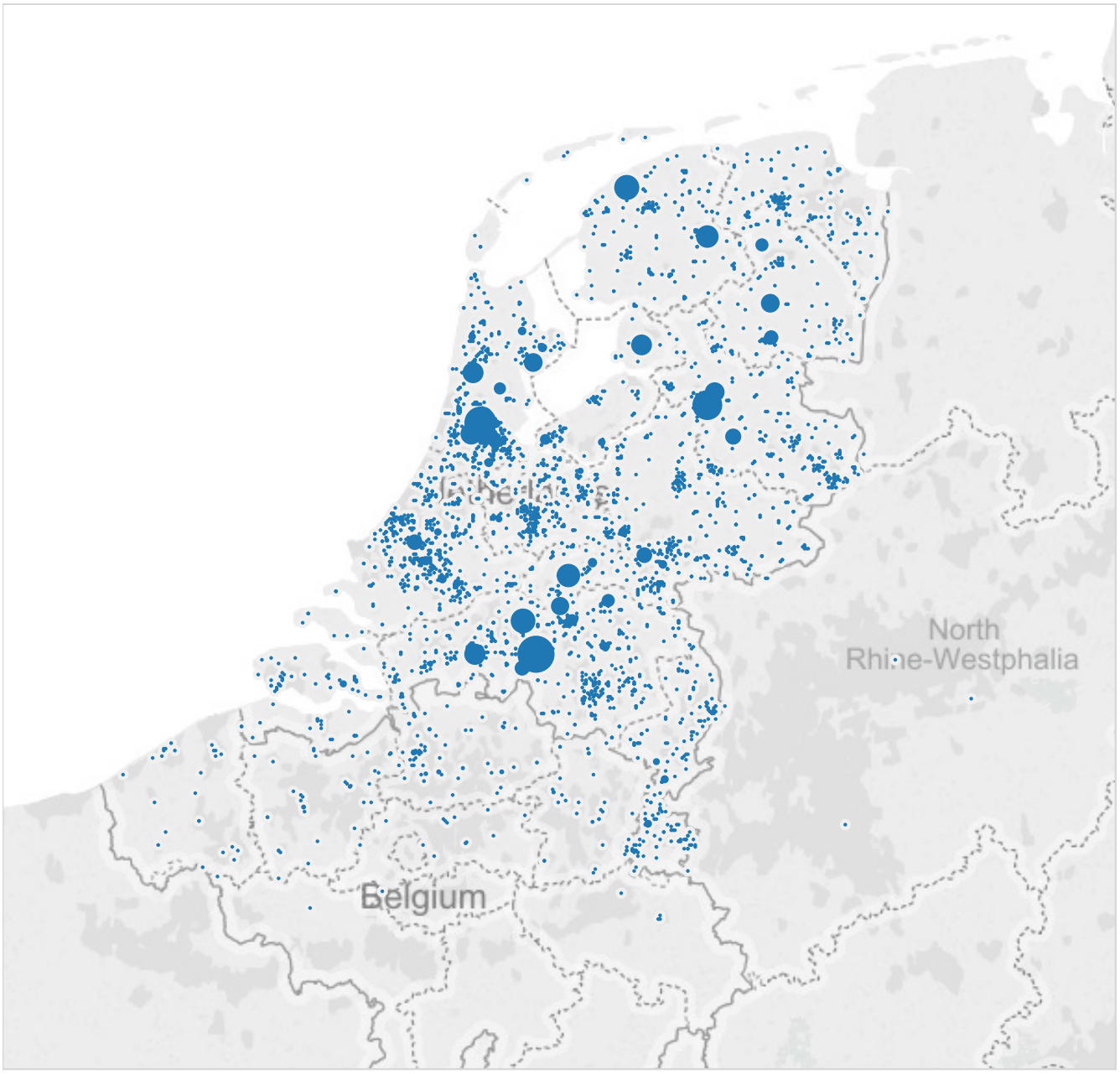
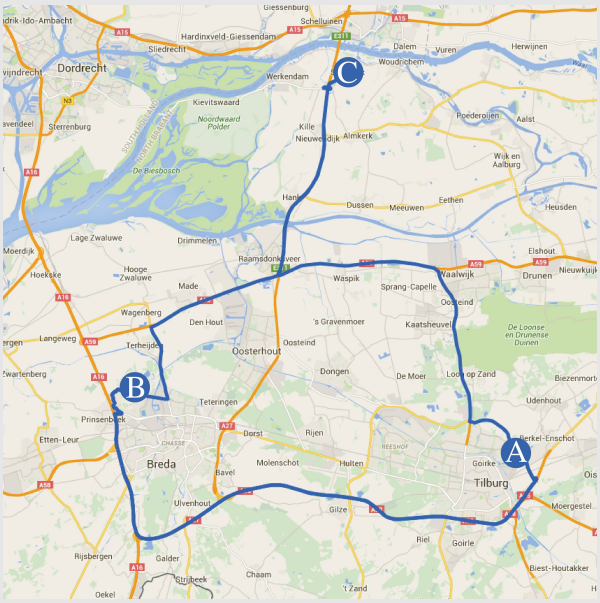



**Figure 1.1:** All stops for Hire- and Trip-orders in January 2016 at Peter Appel Transport, a larger dot means more stops at that address.

**Example 1.1:** Retail-distribution




The *Trip-order* that is used in this *retail-distribution* example has four *stops*. The first *stop* is at the distribution center (A) at 3:39 hour. After *Loading*, which takes 33 minutes, the truck departs at 4:12 hour.

Shop (B) has been assigned a *time-window* between 5:00 and 6:00 hour. Driving to (B) takes 46 minutes, so the truck arrives at 4:58 hour and then waits for 2 minutes. *Unloading* takes 41 minutes, so the truck departs at 5:41 hour for a 45 minute ride to shop (C).

Shop (C) has a *time-window* available between 6:00 and 7:00. The truck arrives at 6:26 hour at (C). *Unloading* takes 33 minutes. The truck departs at 6:59 hour for the 48 minute ride to (A) where it arrives at 7:47 hour to unload the *load-carriers*, which takes until 8:06 hour. The total *trip* takes 4 hours and 27 minutes. Truck and *driver* are, after an obligatory break, available for other work.

more or less as prescribed, usually within very strict *time-windows*. A *Trip-order* always involves the request for a full truckload and should be executed with a *vehicle-combination* of a certain type (such as *'Truck'* or *'Euro-combi'*, see Section 3.2).

*Trip-orders* are common in *retail-distribution*, where goods from warehouses or factories are delivered to retail shops. Most retailers do not own *vehicles*, and therefore have contracts with one or more transportation companies (carries such as PAT). Retailers have strict *time-windows* related to distribution center docking schedules or shelf-re-filler-crews. Therefore, the retailer is commonly charged with the responsibility for a realistic delivery schedule.

A *Trip-order* contains an entire delivery schedule consisting of a sequence of multiple *stops*. All *stop-information* is known in advance. An *order* contains a sequence of *stops* (addresses), with their corresponding *time-windows*.

Usually, an *order* starts with a *Loading stop* at the retailer's distribution center or factory. The *vehicle* is loaded with the goods that should be delivered at one or more *stops*. The *stops* should be visited in the predefined sequence. Some *vehicles* finish without any cargo after delivering the goods at the last shop. Other retailers want *vehicles* to take return goods, waste and/or empty *load-carriers*. In that case, the last *stop* of the *vehicle* is usually at the distribution center, where these return goods, waste and/or empty *load-carriers* are unloaded. The length and duration of a *Trip-order* is related to the distances between *stops*, and is attuned to the opening times of shops. See Example 1.1 for a typical *retail-distribution* situation.

Note

Historically, retailers controlled the entire transport process, using own *vehicles* and *drivers*. Now, retailers outsource *trips*, but maintain control of (part of) the planning.

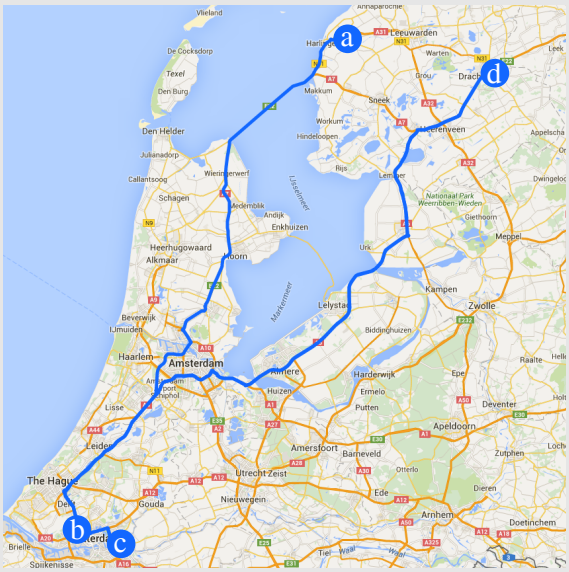
Definition

A ***load-carrier*** is a standardized unit that can carry loads, such as a bin, pallet or rolly.

### Hire-orders

Another part of the *orders* involves *Hire-orders*, which can also be directly converted into *trips*. An *Hire-order* is an *order* for the hiring of a *vehicle-combination* with a *driver* for a certain time-period. In contrast to *Trip-orders* (which contain a lot of information), *Hire-orders* state only a start-location, a start-time, an end-location, and an estimated end-time. Information regarding *activities* that happen between start and end, such as driving times and *stops* (address, *time-windows*, etc.) is not given. Similar to *Trip-orders*, a *Hire-order*

**Example 1.2:** Network-transport




The three *Network-orders* that are used in this example all involve partial truckloads, which together fit into a truck. All three *orders* share the same delivery location and are combined into a single *trip*. The first *stop* is at the first pickup address (a) at 9:00 hour. After 30 minutes of *Loading* at (a), the *driver* departs for a 3 hour ride to the next pickup address (b) at 12:30 hour where another partial truckload is loaded. This *activity* encompasses 30 minutes. It takes the *driver* 30 minutes to reach pickup address (c) where it can pick up the final partial load.

The *driver* must than first take an obligatory 45 minute break (due to *working-hour-regulations*). *Loading* can be started at 14:15 hour, and takes 30 minutes. Four hours later, the *driver* arrives at 18:45 at the unloading location (d) where 45 minutes are needed for *Unloading*.

can be seen as the request for a full truckload that should be transported with a certain type of *vehicle-combination*. Because *vehicles* and *driver* can simply be reserved for a certain amount of time, this *order-type* can be converted directly into a *trip*. Billing is usually achieved by the data collected from an on-board computer.

**Network-orders**

*Network-orders* are the third and final type. A *Network-order* is an order for the transportation of a partial truckload from a pick-up *stop* to a delivery *stop*. *Stops* usually have some *time-window*. The carrier (such as PAT) is responsible for planning the *Network-orders*. Based on agreements with the customer and characteristics of the goods, planners combine multiple *Network-orders* in a single *trip*, or even a single *vehicle-combination*. Example 1.2 illustrates how different *orders* can be combined in a single *trip*, according to the *Network-orders* logic.

### 1.2.2 Current planning process

Up to now, the planning process (see Figure 1.2) is mainly manually driven. Data needed for the planning is available in the Transplan TMS. *Orders* are either automatically loaded or manually entered into the system. The availability of the *drivers* is synchronized with the personnel administration system. The contract agreements with charters and their availability are stored in the system. *Vehicle* (truck and trailer) availability is also captured in the Transplan TMS.

*Departments* of PAT are grouped to form *regions*. Interregional transports are planned and executed by *vehicles* of the *department* closest to the first *stop* of an *order*. For each *region*, a group of planners is responsible for the integration and scheduling of the *orders*, *vehicles* and *drivers* in that *region*. The scope of the planners extend to different *departments* within that *region*. First, *Trip-orders* and *Hire-orders* are automatically converted into *trips*. Then, a planner can group *Network-orders* into *trips* where grouping is feasible and expected to be beneficial. Thereafter, *trips* are assigned to both *vehicles* and *drivers*, where planners focus on creating *shifts* for the *drivers*. To ease this process, there are default combinations of *vehicles* and *drivers*. The *stops* (if known), routes, and breaks for the

Definition

A ***shift*** is defined as the span of work of a single *driver* between departure from home and arrival to home.

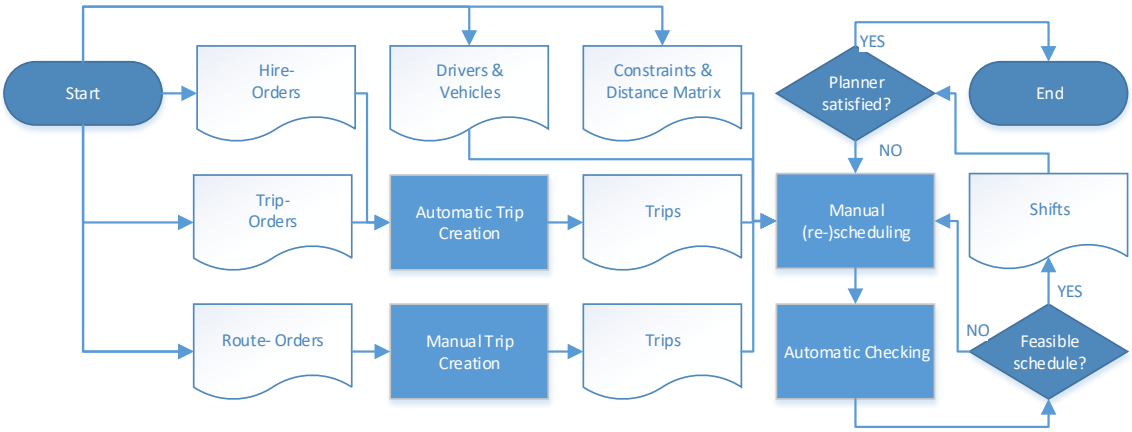



**Figure 1.2:** Current planning process

*shifts* of the *drivers* are then automatically calculated and validated, based on contracts, collective labor agreements, *working-hour-regulations*, and laws. The planner can adjust the planning or overrule the violation of constraints whenever felt necessary. If there are too few PAT *drivers* or *vehicles* available, the planners can rent *resources* or hire charters (outsourcing). The planning process is completed when planners feel satisfied. The schedules are then communicated to the *drivers*, charters and customers.

### 1.2.3 Stakeholder motivation

Stakeholders have different motives to improve the scheduling process. With regard to RGB$^+$ Automatisering, automated scheduling and planning optimization entail sales opportunities for their Transplan TMS system. At the same time, cost reduction and improved efficiency can be a main incentive for carriers (such as PAT) to automate their planning on an operational decision level. Their motivation is to ease the planning process and to reduce the costs by improving the utilization of *drivers* and *vehicles*, hiring fewer charters and driving fewer kilometers without a load. Increasing efficiency should at least not be at the expense of *drivers'* job satisfaction (minimal daily changes between night/day *shifts*, incorporating personal life circumstances, knowing routes and locations in advance, etcetera).

An automated operational scheduling system can also support carriers on the tactical and strategic decision levels. Tactical considerations can be the assignment of *resources* to bases or the acceptance of new transportation contracts. Strategic choices can involve purchasing of *vehicles* or hiring of new personnel.

### 1.2.4 Purpose

The purpose of the present project is to contribute to a solution that is committed to automate and optimize the assignment of *orders*, *drivers* and *vehicles*, for transport companies, with PAT as a case, using the available information from the Transplan TMS, resulting in a set of *shifts* that can be used in the Transplan TMS. The proposed solution should minimize the total transport costs by improving the utilization of *resources*, while maintaining the available *time-windows*, and accounting for the accompanying constraints for the optimization obtained. Furthermore, a solution is only acceptable if it accounts for the specific problems that we discuss in Section 1.2.5.

The minimization of transport costs can be achieved by balancing the minimization of the total driving *distance* and a minimum number of *resources* required. Note that this target

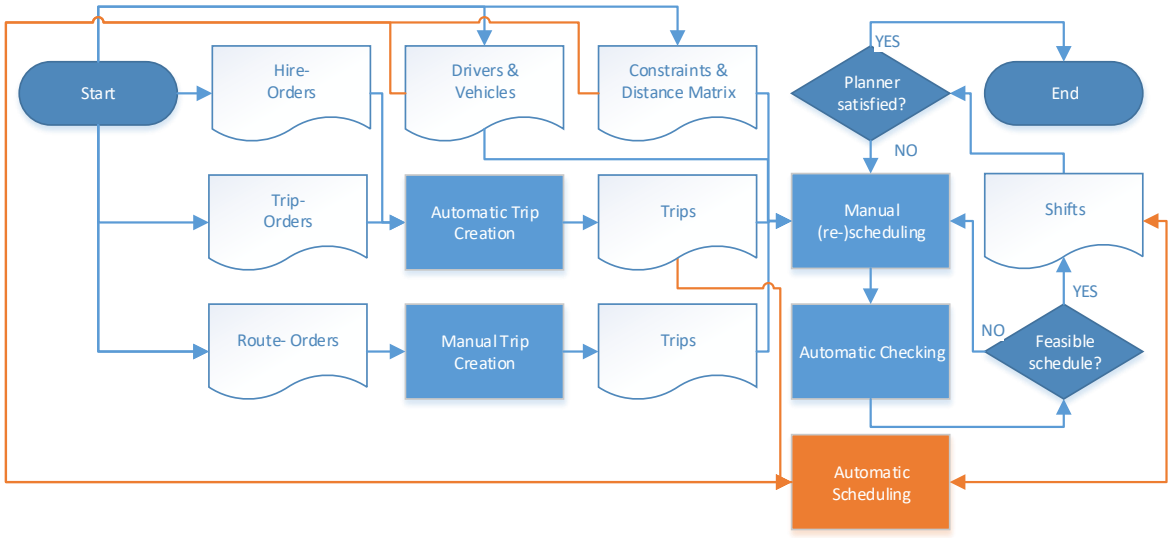



**Figure 1.3:** Targeted planning process for this research (in orange)

may only be attained by contradictory or paradoxical measures. For instance, it might be cheaper to drive more kilometers to save money on hiring charters. Planners should thus keep the freedom to manually adjust the planning and then re-optimize with manually fixed *shifts*.

In the ultimate situation, all scheduling is done automatically, based on the raw data (see Appendix A). For this research, we focus on automatically scheduling *trips* that result from *Hire-orders* and *Trip-orders* and assigning those *trips* to *resources* (both *drivers* and *vehicles*), as we visualize in orange in Figure 1.3.

### 1.2.5 Problem description

Planning *activities* by transport companies such as PAT are generally not automated so far, thus requiring lots of manual work. Manually produced schedules are vulnerable for mistakes (planners can overrule constraint violations). No optimization can be guaranteed. Therefore, it may be assumed that *resources* can be used more efficient due to imperfections such as suboptimal deployment of *drivers*, unnecessary kilometers driven, or surplus charters.

We focus on the assignment of *trips* resulting from *Trip-orders* and *Hire-orders* to *vehicles* and *drivers*. We leave out *Network-orders* because those *orders* cover only 2% of the *orders* at PAT, and require a different scheduling procedure. Therefore, there is no need to determine which *orders* need to be combined to optimize *vehicle* utilization, because we only deal with full truckloads. Because the sequence of *stops* within a *trip* is fixed, the routing aspect does not play a role during the scheduling of the *stops* of a single *trip*, however, it is important between *trips*. We aim to assign (multiple) *trips* to resources, or resources to *trips*. Once *resources* and (a sequence of) *trips* are assigned to each other, determining the *route* is an easy task, because the sequence of *stops* is then *fixed*. We refer to the combinatorial assignment of *drivers*, *vehicles*, and *trips* as the *trip-assignment-problem*.

The *trip-assignment-problem* is quite challenging: not only when planning manually, but also when algorithms are applied. In the following sections, we give four complicating factors of the *trip-assignment-problem*. First, we give an idea of the problem size; second, we detail problems involved with the scheduling characteristics of *orders*, *vehicles*, and *drivers*. Then, we continue with problem specific *working-hour-regulations*. Finally, we discuss problems arising from specifics on *vehicle-combination-types*.

### Problem size

Problem size is one of the indicators of the complexity of an assignment problem such as the *trip-assignment-problem*. The size of the problem can be determined by calculating the number of possible solutions. We illustrate this by looking at a simplification of the *trip-assignment-problem*. Using the method of Aho and Ullman (1994), we calculate the number of possible assignments and sequences of *orders* to *vehicles*. This number can become very large: the number of possible assignments (including sequence) of two *orders* to two *vehicles* is six (see Table 1.2), while the number of possible assignments and sequences of three *orders* to three *vehicles* becomes as large as sixty. In our case (1000 *orders*, 800 *vehicles*), the number of solutions (assuming no constraints, which of course includes a huge number of infeasible solutions) is approximately $10^{3102}$. See Appendix B for an explanation regarding the subject of problem size.

This huge problem size accounts for one of the main reasons why scheduling transport processes may be so complicated and time-consuming: not only for human beings (i.e., manually), but for automated systems as well. Different from automated systems, human planners are more likely to fall prey to making mistakes when working on large and complicated schedules. However, both human beings and automated systems are unlikely to schedule optimally with complicated problems to be solved.

| Vehicle 1 | Vehicle 2 |
|---|---|
| Order 1, Order 2 | none |
| Order 2, Order 1 | none |
| Order 1 | Order 2 |
| Order 2 | Order 1 |
| none | Order 1, Order 2 |
| none | Order 2, Order 1 |

**Table 1.2:** Possible assignments of two orders to two vehicles, including sequence.

### Scheduling characteristics

A specific problem in optimally scheduling transport routes is sequencing *trips* into *shifts* for *drivers*. About 67% of the *trips* takes less than eight hours (see Figure 1.4). In a twelve hour working day, some *trips* could be executed consecutively, meaning they can be succeeded using the same *resources*. For instance, a three hour *trip* can be followed by a seven hour *trip* forming a ten hour workday, or an eight hour *trip* can be followed by a twelve hour *trip* resulting in a two day *shift*, with a total of ten hours work per day.

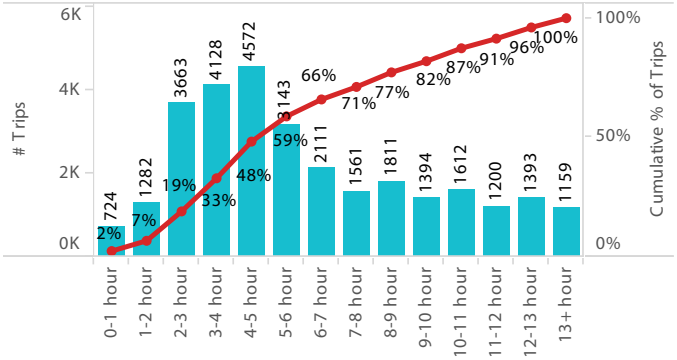



**Figure 1.4:** Frequency of route duration (January 2016). On the horizontal axis buckets of route duration in hours. Vertically, the cumulative percentage (the line) and corresponding number of routes (the bars) that fall in a bucket.

The problem is framed by location and time. *Vehicles*, *orders* and *drivers* form combinations that travel between locations. Traveling takes time, and waiting time is actually a waste of time. Besides that, a delivery location is not necessarily a pick-up location for the next *trip*. *Vehicles* and *drivers* therefore have gaps in time and/or distance between subsequent *trips*, implying that *drivers* (and their *vehicles*) are just standing still (*Waiting*), or drive their *vehicle* without a load. Therefore, both the routing and scheduling aspect play a role in the *trip-assignment-problem*.

### Working-hour-regulations

One of the specific issues in *driver* workforce scheduling is related to *working-hour-regulations*. These laws, rules and legislations define the upper limit for uninterrupted working hours, as well as the lower limit of prescribed rest hours for truck-*drivers*. Dutch national working hour laws are based on EU guidelines. *Working-hour-regulations* are period based (hourly, daily, weekly, etcetera). This means that adding an *order* to, or re-

moving it from an existing schedule might influence all *orders* following that *order*. On the road, an unexpected traffic jam can have a large impact on driving times and, therefore, the allowed schedules for the *driver's* remainder of the scheduling period.

**Vehicle-combination-types**

We know that PAT has several *vehicle-combination-types*. We choose to exclude *orders* with a request for an irregular *vehicle-combination-type* from scheduling. In Section 3.2 we explain the details on *vehicle-types*. For now we can explain that we include 12 regular *vehicle-combination-types*, which are: (a) *Tractor*; (*b*) *Truck*; (*c*) *City-Combi*; (*d*) *Bilevel–Combi*; (*e*) *Euro-Combi*; (*f*) *Axle-Combi*; (*g*) *Trailer-Combi*; (*h*) *Euro-Dolly-LHV*; (*i*) *Bilevel–Colly-LHV*; (*j*) *City-Slider-LHV*; (*k*) *Euro-Slider-LHV*; and (*l*) *Euro-Axle-LHV*. We exclude all other *vehicle-combination-types*. A *vehicle-combination* also has several properties (e.g. *Cooling*) that we include in our assignment.

### 1.2.6 Scope

For the sake of time, the focus of this project is on automatically assigning *trips* to *drivers* and *vehicles*. Furthermore, we limit the set of *orders* to comply with a fixed set of *vehicle-combination-types*. It should be noted that only data made available by PAT can be included in this research. In other words, the PAT data serves as a case. As a consequence, the validity of the results obtained in this research project only applies to PAT and we are unable to generalize our results to the branch of transport companies at large. In other words, this thesis is merely a pilot study, resulting in a proof of concept, requiring further development by RGB$^+$ Automatisering, to be fully applicable in practice.

## 1.3 Research goal and questions

The problem formulation departs from the purpose of this thesis, which was described in Section 1.2.4 as well as from the scope that was defined in Section 1.2.6. The main problem is the lack of automated planning in road transport planning and scheduling. The proposed solution should take the problem specifics into account (*order* characteristics, problem size, scheduling characteristics, *working-hour-regulations* and *vehicle-combination-types*, see Section 1.2.5). This leads to the following research goal:

*Develop an algorithm for solving the trip-assignment-problem at PAT.*

We answer the following research questions to achieve the research goal:

RQ 1. Which methods can be used to solve the *trip-assignment-problem* according to literature?

RQ 2. Which data is available for solving the *trip-assignment-problem*, and what is the current scheduling performance of PAT?

- Which data sources are currently available?
- What are the data characteristics?
- What is the current scheduling performance?

RQ 3. How can the *trip-assignment-problem* be solved using a limited time optimization algorithm?

- How can we model the *trip-assignment-problem*?
- Which algorithms are suitable for solving the model resulting from the *trip-assignment-problem*?

RQ 4. What are the consequences of implementing the automated scheduling?

- What performance can be expected from the automatically created schedules at several aggregation levels?
- What are the potential benefits on the operational level, and how can automated scheduling support decision making on tactical and strategic levels?
- Which insights can be obtained from solving the *trip-assignment-problem*?

We discuss the four research questions in the following paragraphs.

For the first research question (RQ 1) we search the existing literature for methods that might solve this problem. The starting points for this research are the Vehicle Routing Problem (VRP) and the Driver Scheduling Problem (DSP).

Definition

The ***trip-assignment-model*** is the formal description of the *trip-assignment-problem*.

To answer the second research question (RQ 2) and its subquestions, we thoroughly examine the current planning method. From this analysis, we determine which constraints are currently used and should be used in the *trip-assignment-model*. Furthermore, we collect and analyze data. We examine and prepare this data in such a way that it can be used as input and test data for constructing and reviewing the *trip-assignment-model* that is required for answering of RQ 3. Based on this data, we analyze the current performance as well. We compare the current schedules with the current execution to determine the planning performance. Furthermore, issues with the current planning method and further constraints are established.

To develop algorithms for solving the *trip-assignment-model* for the third research question (RQ 3), we use knowledge obtained from the first two research questions. Furthermore, the algorithms are tested for feasibility, speed and quality.

The fourth research question (RQ 4) is treated by examining the performance of the newly created automated schedules. We do not test our schedules in practice, so we compare the current aggregation level (*Local*) to higher aggregation-levels (*Regional* and *National*). Furthermore, we try to estimate the consequences that might be expected from implementing our heuristics. Finally, we describe the insights that we obtained from this research.

## 1.4 Conclusions

In this chapter we introduced the intentions, targets, and ambitions of the present research project. First, we described the current situation and the problems that are experienced in this situation. Based on the descriptions of several problems to be tackled, we stated our research goal, and we formulated several research questions and the selected strategy to approach our challenges for this research.

# Chapter 2

# Literature Review

Various researchers have focused on the assignment and routing of *vehicles* and *orders*. There are many real world examples (e.g., logistics, transportation, material handling systems, pick-up and delivery, dial a ride, airlines, rail-transport, school-buses, drayage) that deal with routing and the assignment of *resources*, such as we do in the *trip-assignment-problem*. Therefore, we first discuss related problems from the literature (Section 2.1). We screen the solution methods that have been proposed in the literature, with a focus on large instances (Section 2.2). Finally, we review the problem variants, the solution methods, and the applicability to the *trip-assignment-problem* (Section 2.3).

## 2.1 Related problems

We use this section to discusses problems related to the *trip-assignment-problem*. First, we discuss the Vehicle Routing Problem and its variants (Section 2.1.1), followed by the Pick-Up and Delivery Problem (Section 2.1.2). Then, we discuss the multi-Traveling Salesman Problem with *time-windows* in Section 2.1.3, followed by an overview of benchmark problems used in literature (Section 2.1.4). Finally, the applicability of the existing literature to the *trip-assignment-problem* is discussed in Section 2.1.5.

### 2.1.1 Vehicle Routing Problem and its variants

The 'Vehicle Routing Problem' (VRP) has been studied extensively (Laporte, 2009). One of the simplest approaches to routing problems is the 'Traveling Salesman Problem' (TSP). This problem is about a salesman who needs to visit an arbitrary number $n$ of locations. He starts in his base town and visits each site exactly once. The problem is defined by the minimal total distance to be traveled in order to touch each site once.

Miller, Zemlin and Tucker (1960) extended this problem by adding a maximum number of cities that the salesman is allowed to visit, before (temporary) returning to the depot. The capacity constraint transforms the 'multiple-Traveling Salesman Problem' (m-TSP) into a VRP (Stewart Jr. & Golden, 1984). There is a strong relationship between the TSP and the VRP: both cases involve a routing aspect (Bullnheimer, Hartl & Strauss, 1999).

The previously described base cases can be extended to create multiple (combinations of) variants (see e.g., Raff, 1983; Potvin, 2009; Eksioglu, Vural & Reisman, 2009; Kumar & Panneerselvam, 2012; Koç, Bektaş, Jabali & Laporte, 2015). The remaining part of this section elaborates on some known properties that can be distinguished in VRPs, using an adapted version of the structured problem variant classification by Eksioglu et al. (2009).

#### Fleet variants

Several VRP variants have been introduced in the literature, where the physical properties (e.g., capacity, length) of the *vehicles* that were used slightly differs. We discuss 'fleet heterogeneity', 'fleet size', 'external carriers', and 'time-dependency' as these are the subjects

that are applicable to the *trip-assignment-problem*.

Early literature on the VRP assumed the fleet to be homogeneous, i.e., the *vehicles* are supposed to be identical (see, e.g., Stewart Jr. & Golden, 1984; Kolen, Rinnooy Kan & Trienekens, 1987; Metters, 1996; Bullnheimer et al., 1999; Toth & Vigo, 2002; Mester & Bräysy, 2005; Hu, Ding & Wang, 2010). The assumption of a *heterogeneous fleet* of course is more realistic: *vehicles* in a fleet have different properties. For instance, capacities differ, some *vehicles* have cooling capacity (which might also be configurable) or carry a truck mounted forklift (Raff, 1983).

Another variant in the solution of the optimization problem is limitation of 'fleet size'. Most authors use a limited fleet size, while amongst others Metters (1996), Taillard (1999), Baker and Ayechew (2003), Mester and Bräysy (2005), Ropke, Cordeau and Laporte (2007), Ropke and Cordeau (2009), Goel (2009), Salhi, Wassan and Hajarat (2013), Dominguez, Juan, Barrios, Faulin and Agustin (2014) and Dayarian, Crainic, Gendreau and Rei (2015) presuppose an unlimited fleet size. The latter assumption is logically less realistic, especially in the case of a heterogeneous fleet where *vehicle-types* have different capacities.

Xu, Chen, Rajagopal and Arunapuram (2003) introduced another factor to the solution of the optimization problem: in their model, 'external carriers' (charters) can be hired. These carriers also own a fleet and employ *drivers*. These charters can be used when the own fleet falls short for the jobs to be accomplished or in case it is economically more attractive to use an external carrier over the use of *Own vehicles* and *Own drivers* (Xu et al., 2003; Zäpfel & Bögl, 2008; Potvin & Naud, 2011; Wen, Krapper, Larsen & Stidsen, 2011).

Fixed travel times (as usually assumed in VPRs), can make the solution less realistic. 'Time-dependent' travel times incorporate factors such as traffic jams during rush hours are then incorporated. Trucks drive, for instance, probably slower than average during rush hours or faster during night times, when there is less traffic on the road (B. Afshar-Nadjafi & Afshar-Nadjafi, 2014).

**Customer specific variants**

Not only fleet variants are decisive to VRPs. Individual customers can also have specific expectations or requirements. In this section we discuss *time-windows* (Solomon, 1987). A *time-window* is defined as an interval (from start- to end-time) in which service (*Loading* or *Unloading*) must start (and, depending on the definition, also be finished). Examples of *time-windows* at customer sites are morning (8.00-12.00), shop opening times (8.00-18.00) or time slots at warehouses (12.30-13.30).

**Distribution structure variants**

Another important group of variants is related to the distribution structure. In a standard VRP, several trucks each execute a single tour delivering (picking-up) goods from a single *depot*. Known variants are 'multiple depots', 'open routes', 'multiple *trips* from the *depot*', 'multi-*vehicle* task assignment', '*working-hour-regulations*' and some other variants. Many authors (e.g., Xu et al., 2003; Hollis, Forbes & Douglas, 2006; Pisinger & Ropke, 2007; Franceschelli, Rosa, Seatzu & Bullo, 2013; Bettinelli, Ceselli & Righini, 2014; B. Afshar-Nadjafi & Afshar-Nadjafi, 2014; Iori & Riera-Ledesma, 2015; Dayarian et al., 2015) described VRP variants with 'multiple depots' instead of single *depots*. Some authors used 'open routes', where *vehicles* end their *trip* at the last customer (Pisinger & Ropke, 2007; X. Li, Leung & Tian, 2012; B. Afshar-Nadjafi & Afshar-Nadjafi, 2014). In most VRP variants, a *vehicle* executes a single *trip* (start and finish at the *depot*). In the 'multi-*trips*' variant, *vehicles* might visit and revisit the *depot* for multiple *trips* (Seixas & Mendes, 2013; Lai, Crainic, Di Francesco & Zuddas, 2013; Dayarian et al., 2015).

**Working-hour-regulations**

Driver *working-hour-regulations* and laws add more complicated restrictions to *vehicle* and crew scheduling problems (Raff, 1983). Restrictions on working hours do not only constrain

assignment of tasks in terms of locations, but also in terms of time. In the planning of subsequent tours, the current location is important, but also all tours that were executed in the previous period (dependent on the applicable laws), in order to determine if the driver is allowed to execute that tour. Examples of problems are given by Zäpfel and Bögl (2008). They solve a planning problem for postal companies, Wen et al. (2011) solved a retail distribution problem where *drivers* are scheduled on a weekly basis.

The regulations for crew scheduling, combined with time constrained routing and scheduling, is relatively new and therefore research is rather limited so far (Wen et al., 2011). A further complicating factor is that laws are different across countries and continents (Goel & Vidal, 2014), making American, Asian or other Non-European research (partly) inapplicable to the Dutch situation. Authors that take *working-hour-regulations* into account are e.g. Archetti and Savelsbergh (2009), Goel and Gruhn (2006), Goel (2009), Goel and Kok (2012), Goel and Vidal (2014), Kok, Meyer, Kopfer and Schutten (2010), Prescott-Gagnon, Desaulniers, Drexl and Rousseau (2010), Wen et al. (2011), Xu et al. (2003) and Zäpfel and Bögl (2008). We explain their solution methods in Section 2.2.6.

### 2.1.2 Pickup and Delivery Problem

The 'Pickup and Delivery Problem' (PDP) is a variant of the VRP (Potvin, 2009). In this type of problem, not all deliveries are loaded at the *depot*, but rather are collected from a pickup location that needs to be visited during the tour. This type of problem creates more complex precedence relationships. Logically, the pickup location needs to be visited before the delivery location, though not necessarily in succession. Optional capacity constraints might be violated after each pickup. Examples are given by Bettinelli et al. (2014) and Qu and Bard (2015).

Smilowitz (2006) pays attention to the problem of 'drayage': transport between modalities such as ship, train, barge, truck, or storage. This multi-*resource* routing problem brings together several *resources* (trucks, *drivers*, trailers, empty and full containers) for up to 300 tasks, and is solved using an exact solution method. Caris and Janssens (2009) solved a similar problem, using heuristics.

### 2.1.3 Multi-Traveling Salesman Problem with Time-Windows

The 'multi-Traveling Salesman Problem with Time-Windows' (m-TSPTW) is also known as the 'full truckload problem'. It is an interesting special case of the VRPTW, with relaxed capacity constraints, discussed by Jula, Dessouky, Ioannou and Chassiakos (2005). These authors introduced a container movement problem: full and empty containers are transported between locations. The authors model this problem as a m-TSPTW, and also incorporate a maximum daily driving time. Due to the full truckload, pickup node and delivery node can be taken together, creating a *trip* for a single container movement. The only *time-window* that matters then is an adapted *time-window* for the start of the *trip*, which can be calculated using the pickup and delivery *time-windows* together with the travel time between pickup and delivery node. The authors concluded that their heuristic insertion method and their hybrid dynamic programming/genetic algorithm method both outperform their exact dynamic programming method (Jula et al., 2005).

### 2.1.4 Benchmark problems

As already stated in Section 1.2.5, problem size is an important factor in assignment problems. Several standard benchmark test sets of different size have been developed for VRPs. We give an overview of some commonly used benchmark problems to give an indication of the size of the *trip-assignment-problem*. In Section 2.2 we describe which of these benchmark instances have been used in literature to test certain solution methods.

Christofides, Mingozzi and Toth (1979) developed benchmark instances, with 50 to 199 customers, as well as with differing *vehicle* capacities. Golden, Wasil, Kelly and Chao (1998)

continued by developing twenty more realistic, larger test problems, with up to 483 customers. F. Li, Golden and Wasil (2005) subsequently developed problems with up to 1200 customers. Solomon (1987) introduced *time-windows* in the benchmark problems and developed benchmark instances up to 100 customers, based on the the problems by Christofides et al. Gehring and Homberger (1999) thereafter, developed a set of benchmark problems with up to 1000 customers, for the VRPTW. Goel (2009) adapted the set of Solomon (100 customers) to be applicable to problems where *working-hour-regulations* come in sight, used by Kok et al. (2010), Prescott-Gagnon et al. (2010) and Goel and Vidal (2014). The *trip-assignment-problem* has many customers. Therefore, the fact that only a very small number of researchers use benchmark instances for problems incorporating a similar number of customers indicates that solving the *trip-assignment-problem* can be a challenge.

### 2.1.5 Applicability

As the reader may have noticed, the literature on the VRP, its variants and its possible solution methods is quit extensive. One of the major flaws is, however, that most authors presuppose a limited set of constraints by focusing on one or a few variants of the problem. Realistic applications however, usually have to deal with a larger set of constraints. The solution method for a large size problem should therefore be able to cope with such a set of constraints. For the *trip-assignment-problem*, we have to deal with a variety of constraints: (*a*) different *vehicle* capacities; (*b*) a heterogeneous fleet; (*c*) a limited fleet size; (*d*) the possibility to hire charters; (*e*) *time-windows*; (*f*) pickups and deliveries; (*g*) multiple *depots*; (*h*) multiple *trips*; (*i*) *working-hour-regulations*; and (*j*) crew scheduling.

## 2.2 Solution methods

Methods for solving problems that involve assignment and routing can be divided into exact methods (see Section 2.2.1) and heuristic methods (Bräysy & Gendreau, 2005a; Mester & Bräysy, 2005; Laporte, 2009). Three types of heuristics can be distinguished (Błocho & Czech, 2012; Mester & Bräysy, 2005): *construction-heuristics* (see Section 2.2.2), *improvement-heuristics* (see Section 2.2.3) and *metaheuristics* (see Section 2.2.4). Furthermore, we discuss parallelization (see Section 2.2.5), techniques for coping with *working-hour-regulations*, (Section 2.2.6), feasibility checking (2.2.7), and heuristic quality (2.2.8).

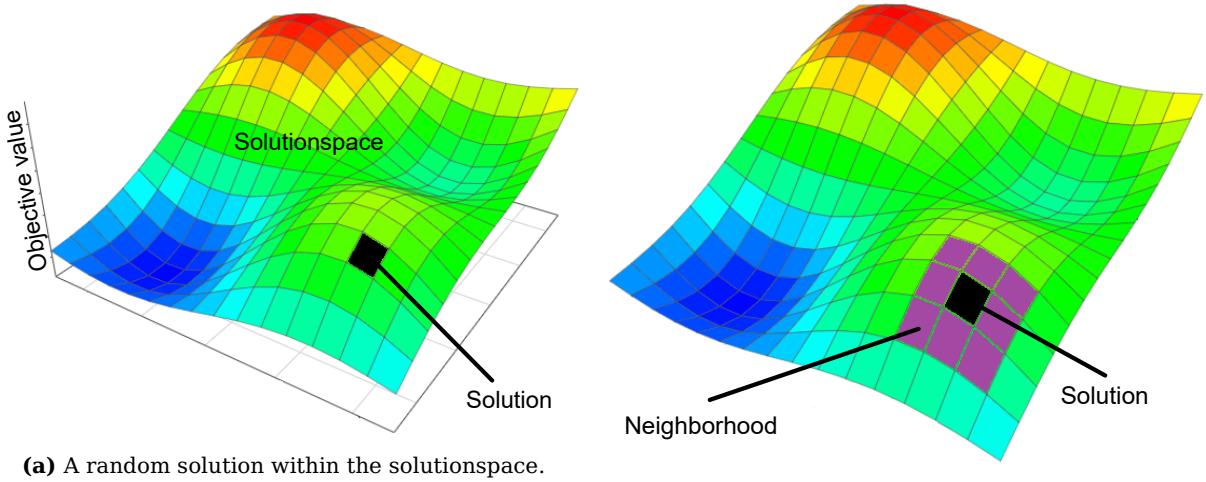
### 2.2.1 Exact Methods

As stated before (see Section 1.2.5), solving large assignment and routing problems such as the *trip-assignment-problem* might be impractical due to the extensive computation time. The best exact algorithms only solve problem instances up to approximately 50-100 customers (Cordeau, Gendreau, Laporte, Potvin & Semet, 2002; Toth & Vigo, 2002; Laporte, 2009; Kumar & Panneerselvam, 2012). Even advanced exact methods such as ‘branch and bound’, ‘branch and price’, and ‘dynamic programming’ take a lot of computational effort (Cordeau et al., 2002). Therefore, we decide not to use exact methods for solving the *trip-assignment-problem*.

### 2.2.2 Construction-heuristics

*Construction-heuristics* start building routes without any knowledge on previously built routes. Two major types of tour *construction-heuristics* can be described: first there is sequential tour construction (tours are constructed one by one), and second there are parallel tour *construction-heuristics* that are able to construct multiple tours in parallel.

‘Insertion heuristics’ are parallel tour *construction-heuristics*, which means that several tours might be constructed at once (Solomon, 1987). The algorithm searches for the best feasible position in an existing tour when scheduling a customer, if it is not found, a new tour is started. This process is repeated until every customer is scheduled. Examples of insertion procedures are (*a*) ‘nearest insertion’ (inserting the node that is closest to any current node);




**(a)** A random solution within the solutionspace. Objective value on the $z$-axis.

**(b)** Neighborhood of the solution in Figure 2.1a.

**Figure 2.1:** Solution, neighborhood, and solutionspace

(*b*) 'cheapest insertion' (inserting the node that adds the least cost); (*c*) 'arbitrary insertion' (inserting a random node); and (*d*) 'farthest insertion' (inserting the node that is farthest from any current node; Raff, 1983).

### 2.2.3 Improvement-heuristics

An *improvement-heuristic* creates new solutions (neighbors, all neighbors of a solution form the neighborhood; all possible solutions form the solutionspace; see Figure 2.1) that more or less differ from the current solution in order to try to improve upon the current solution (minimize or maximize). These heuristics are often called 'local search' heuristics (Bräysy & Gendreau, 2005a). An improved solution replaces the new current solution until no improvements can be found (see Figure 2.2).

Neighbors are evaluated on a certain acceptance criterion, where two strategies are distinguished. The first-accept strategy selects the first neighbor that satisfies the acceptance criterion, while the best-accept strategy examines all neighbors and selects the one that fits the acceptance criterion best (Osman, 1993; Bräysy & Gendreau, 2005a).

Neighborhood generators are used to create solutions that lie close to the current solution (Bräysy & Gendreau, 2005b). Intra-route neighborhood operators only affect one route, while inter-route heuristics change multiple routes (Laporte, 2009). We use the remaining part of this section to discuss some commonly used neighborhood generators, where we distinguish three types (Funke, Grünert & Irnich, 2005; Bräysy & Gendreau, 2005a): (*1*) 'node-exchange' procedures; (*2*) 'edge-exchange' procedures; and (*3*) 'destruct/construct' neighborhood generators.

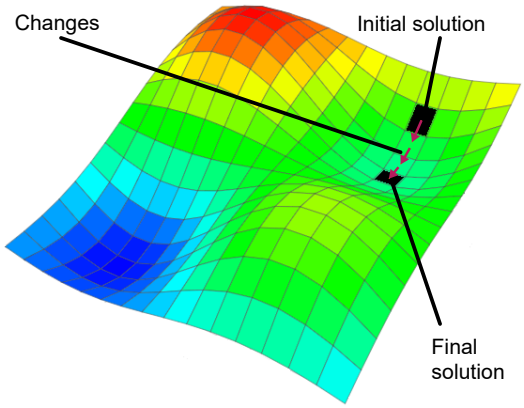



**Figure 2.2:** Local search, ending in a local minimum

#### Node-exchange

The 'node-exchange' procedure implies the exchange of customers (nodes) within the routes. The 'insertion/deletion' procedure (also known as 'relocation procedure') is the simplest neighborhood operator (Funke et al., 2005). A single customer is deleted from a route and inserted at another place: in the same or another route (Osman, 1993).

The '$\lambda$-interchange' procedure is an addition to node-exchange. This procedure interchanges $\lambda$ customers between route-segments (Bräysy & Gendreau, 2005a; Funke et al., 2005), in most cases by exchanging customers (taking each others places; (Funke et al.,

2005)). Route-segments might be on the same or different routes (Bräysy & Gendreau, 2005a). The $\lambda$-interchange with $\lambda = 1$ (1-interchange) procedure is also known as the 'exchange' or 'swap' procedure (Funke et al., 2005).

**Edge-exchange**

The '$k$-opt' (also known as '$k$-exchange') is a simple edge-exchange procedure (Lin, 1965; Raff, 1983; Savelsbergh, 1990). The general idea is that $k$ edges are removed from certain route-segments and that the remaining gaps between the resulting partial tours are filled with new edges, connecting the segments again. Most authors use $k = 2$ or $k = 3$ as parameters (Raff, 1983). Smaller values of $k$ yield weaker results than $k$-opt exchange procedures with larger values of $k$ and might end at a subordinate 'local optimum' (Raff, 1983). For values of $k \geq 4$, the computational time significantly increases, while the results do not improve analogously however (Lin, 1965). The efficiency of $k$-opt, is lessened by *time-windows*, due to feasibility checks and the fact that most $k$-opt exchanges invert some segments (and therefore the customer order) by nature (Caseau & Laburthe, 1999).

The 'Or-opt' operator (Funke et al., 2005) moves an entire chain of customers at once by replacing up to three consecutive edges with the same number of new ones without changing the orientation of the route, which makes it applicable to problems with *time-windows* (Funke et al., 2005).

**Destruct/construct**

By definition, edge-exchange and node-exchange can just produce small improvements, because adaptations are sought to create neighbors that differ minimally from the current solution. 'Destruct/construct' (also known as 'ruin and recreate' are examples of algorithms creating neighbors that are very different from the current solution (Bräysy & Gendreau, 2005a; Funke et al., 2005). Those algorithms are destroying bad parts of the current solution, while keeping the good parts intact. The algorithms then apply construction heuristics or exact methods to the remaining and removed parts, generating a neighbor solution (Deineko & Woeginger, 2000; Funke et al., 2005).

### 2.2.4 Metaheuristics

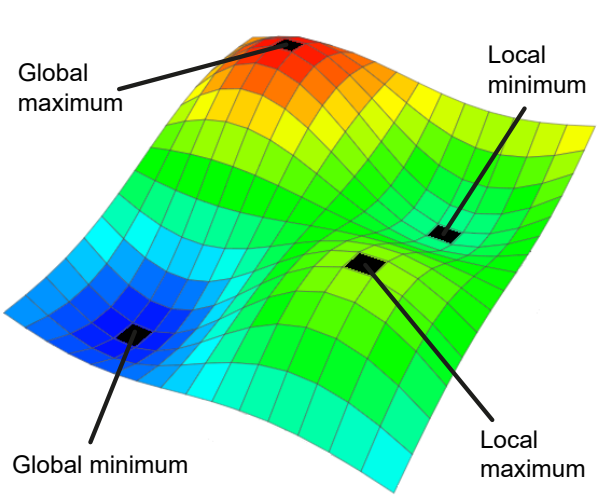
*Metaheuristics* are a special class of heuristics. There are some differences and similarities with simple *construction-* and *improvement-heuristics* (local search). Most *metaheuristics* embrace simpler heuristics by embedding and combining them within its own heuristics. *Metaheuristics* are designed to help exploring large parts of the solutionspace in order to avoid ending in a local optimum. A local optimum might occur when *improvement-heuristics* cannot find better solutions any more. Most *metaheuristics* allow intermediary solutions to get worse or even infeasible during the search process (Bräysy & Gendreau, 2005b). This principle is illustrated in figure Figure 2.3.

One of the major problems with *metaheuristics* is the increase in time needed to search for the optimum (*runtime*), especially for larger problem instances (Funke et al., 2005). A clever choice and design of the neighborhood structure and local search heuristic are ways to make *metaheuristics* run faster (Funke et al., 2005).

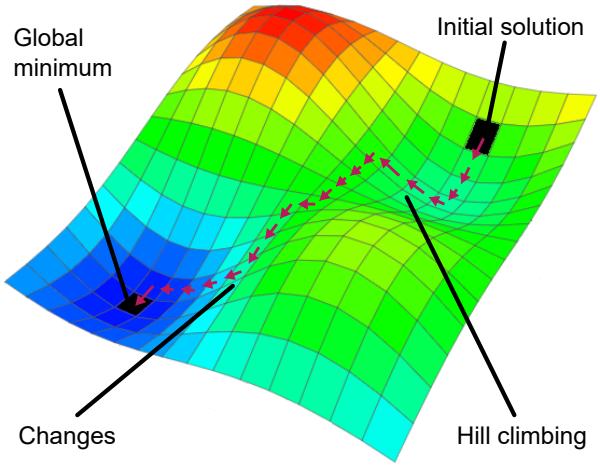
Both 'Simulated Annealing' (SA) and 'Tabu Search' (TS) are *metaheuristics* that may use *construction-* and *improvement-heuristics* as described in Section 2.2.2 and Section 2.2.3 respectively. We first describe SA and TS, which are neighbor acceptance strategies often used in *metaheuristics*. We continue with some other *metaheuristics*. Finally, we discuss the quality and applicability of these *metaheuristics* to the *trip-assignment-problem*.

**Simulated Annealing**

The SA *metaheuristic* is a neighbor acceptance strategy based on decision rules depending which neighbors will or will not be accepted. The general idea of SA is to accept many neighbor solutions in the beginning of the optimization and few neighbors in the end. Es-




**(a)** Local and global optima (minimum and maximum)




**(b)** Accepting worse solutions to escape from local minimum, ending in global minimum.

**Figure 2.3:** Optimization

pecially in the beginning, these neighbors might be worse than the current or current best solution, however, SA accepts these solutions (temporarily) in order to prevent ending in a local optimum. The probability of accepting a worse solution declines when the algorithm proceeds.

The SA algorithm is derived from physics and is based on the philosophy of the annealing process of solids, which is often applied in order to harden and strengthen metals by heating them and then slowly cooling them down (Osman, 1993). The algorithmic variant uses 'temperature' as control parameter that slowly decreases according to a certain 'cooling schedule'. If a neighbor solution appears to be better, it is always accepted. The temperature is used to calculate the chance of accepting a worse neighbor solution: a lower temperature decreases the chance of accepting it. The temperature is automatically decreased after a number of calculations. When few or no solutions are accepted in that run of calculations, the temperature is increased instead of decreased, in order to prevent ending at a local optimum again. The algorithm stops when the temperature reaches a certain threshold. We describe the SA algorithm proposed by Osman (1993) in Algorithm 1 of Appendix C.

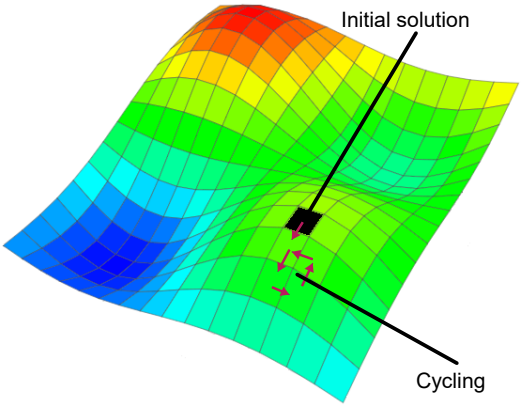



**Figure 2.4:** Cycling between solutions.

### Tabu Search

Similar to SA, the 'Tabu Search' (TS) *metaheuristic*, is a neighbor acceptance strategy departing from decision rules that state which neighbors will be accepted and which neighbors will be rejected. The general idea of TS is that it prevents previous neighbors to reoccur within a certain period of time. The main reason to do so is that algorithms that allow acceptance of worse solutions sometimes end in a loop (see Figure 2.4). To prevent this, the algorithm keeps a 'tabu'-list, which is a list of moves that are not allowed.

The algorithm is initiated by a *construction-heuristic* (see Section 2.2.2). Neighbor solutions are created by applying all possible single moves to the current solution (Prescott-Gagnon, Desaulniers & Rousseau, 2009). Finally, the best feasible and non-'tabu' neighbor solution is accepted. The move that resulted in this neighbor solution is added to the 'tabu'-

list and are hence ‘tabu’ (forbidden to be applied) for a certain number of iterations, which is implemented by a maximum length of the ‘tabu’-list (Gehring & Homberger, 1999). The algorithm continues until some stopping criterion is met (Zäpfel & Bögl, 2008). The ‘tabu’-list allows to escape from local minima (Prescott-Gagnon et al., 2009) by avoiding cycling between solutions (see Figure 2.4). In a comparison of several *metaheuristics*, Cordeau et al. (2002) concluded that TS implementations for the VRP dominate other (combinations of) heuristics. We describe the basic TS outline based on Zäpfel and Bögl (2008) and Gehring and Homberger (1999) in Appendix C (Algorithm 2).

#### Other metaheuristics

‘Adaptive Large Neighborhood Search’ (ALNS) is a general framework for *metaheuristics*, which can be used with any *metaheuristic* such as SA, TS, or other *metaheuristics* (Pisinger & Ropke, 2007). ALNS uses some smart principles, such as visiting only unvisited solutions by storing a hash key of each solution, adjusting scores during runtime for choosing a neighborhood generator, and prevention of ending in a local optimum by adding random noise (Pisinger & Ropke, 2007; Ropke & Pisinger, 2006). Pisinger and Ropke (2007) reported promising results for large scale problems, using on less vehicles the standard VRPTW instances (up to 1000 customers) from Gehring and Homberger (1999). See Algorithm 3 in Appendix C for the general outline of the ALNS.

A very different model for optimization should also be mentioned. ‘Genetic Algorithms’ (GA) (also known as ‘Evolution Strategies’ and ‘Population Search’) try to mimic natural or biological selection and evolution (Potvin, 2009). Solutions are crossed in order to generate offspring solutions with characteristics of both parent solutions (Baker & Ayechew, 2003). Baker and Ayechew (2003) concluded that a TS heuristic gives better results than the GA for the VRP. Equivalently to biological selection, strong offspring is more likely to survive (Mester & Bräysy, 2005). Potvin (2009) described that offspring is often infeasible (i.e. double and missing customers), which is a drawback of GA. We give a basic form of the GA based on the pseudocode in Potvin (2009) and Zäpfel and Bögl (2008) in Algorithm 4 of Appendix C.

Even different methods are ‘Ant Colony Optimization’ (ACO; also known as ‘Particle Swarm Optimization’. These methods are based on natural group behavior of ant colonies or swarms of bees, wasps, etcetera (Potvin, 2009). Huang, Yang and Wang (2011) stated that these solution methods are very suitable for routing problems, because the direction of the arcs is highly important. ACO is a learning algorithm, where edges are given weights, based on the behavior of ants searching for food, during that process, they lay pheromone (weights) on their paths, which over time becomes more on the shortest paths (best solutions) which are used by the largest number of ants (Laporte, 2009).

### 2.2.5 Parallelization

Gehring and Homberger (1999, 2002) and Błocho and Czech (2012) proposed parallelization of *metaheuristics* for the optimization of large VRPs. The parallel approach is motivated by the ability to solve larger problems in less time, calculating better solutions in less time or improving convergence behavior (Gehring & Homberger, 2002).

Gehring and Homberger (2002) described three types of parallelization: (*1*) parallelization of operations within an iteration of the solution method; (*2*) decomposition of problem domain or search space; and (*3*) multiple search threads with various degrees of synchronization and cooperation. The authors used the third variant where each thread strives to optimize the entire problem using GA and TS on a real world test instance: the number of vehicles is equal to the best solutions found, the travel distance is improved. However, the computation time is much longer for most instances up to 1000 customers.

### 2.2.6 Accounting for working-hour-regulations

Still another factor in the optimization of cargo transport is legislation and regulation on driver working hours. Goel and Gruhn (2006) introduced a labeling algorithm for checking and updating driver working hours during the use of optimization algorithms. The aim of this labeling algorithm is to avoid recalculation of restrictions when checking neighborhood solutions for feasibility. The proposed labels contain information on arrival time and weekly, daily and nonstop driving time, but can be extended to other information. The label values are updated on removal or insertion of customers in routes. Goel (2009) used these labels for optimizing problems using ALNS.

Kok et al. (2010) incorporated both the full European sets of driver rules and working hour rules. The problem is solved using heuristics. State dimensions are used in order to check route feasibility. The state space is restricted due to a maximum number of states and a maximum number of single state expansions. One of the assumptions of this algorithm is that truck and *driver* stay together for an entire period. Tested on the modified instances of Goel (2009), the algorithm runs in approximately one minute, giving reasonable results.

Prescott-Gagnon et al. (2010) developed a heuristic for the VRPTW with *working-hour-regulations* that has been developed for applicability to the European driving and working hour rules. It is presumed that the schedules have a six day planning horizon and that *drivers* are assigned for an entire week to the same *vehicles*. The algorithm performs a large neighborhood search, destroying part of the solution and then reconstruction the removed elements. Validation and break scheduling is accomplished using labels that are generated by a label extension algorithm, preventing recalculation of entire routes. The algorithm generates labels for all possible insertions of breaks and rests in routes. The best option is selected (e.g. it is not useful to insert a break when the daily driving time is exceeded). Depending on the parameter settings, the algorithm produces better results in comparable running times as Kok et al. (2010). Longer running times result in even higher quality results (Prescott-Gagnon et al., 2010).

The Vehicle Routing Problem with working-hour-regulations, is still in its infancy. One of the major drawbacks is that the algorithms known, have not been extensively tested yet on larger instances (such to the *trip-assignment-problem*) than that of Goel (2009, 100 customers).

Another shortcoming of the known algorithms that account for working-hour-regulations is the *planning-period*. Most authors assume that the same *driver* and *vehicle* stay together for an entire *planning-period* (usually a week), an assumption which is not suitable for the *trip-assignment-problem*. At PAT, for example, orders occur 24 hours per day, 7 days per week, while the *trip duration* is relatively short and *vehicles* are used multiple times per day, using different *drivers*: one *driver* is enjoying his daily rest period at home, while another *driver* is on the road.

### 2.2.7 Feasibility checking

According to Savelsbergh (1990), testing the feasibility of a neighbor solution is the main challenge when using *improvement-heuristics* for problems with side constraints. The author proposed a global labeling algorithm, which hold records of the possible *shifts*, waiting times, capacities and other constraints in order to avoid re-checking feasibility after every iteration. Funke et al. (2005) proposed the use of a smart feasibility function, which they call the oracle.

### 2.2.8 Heuristic Quality

In order to be able to compare different heuristics, some quality measures for heuristics might be useful. Cordeau et al. (2002) and Bräysy and Gendreau (2005a) mentioned the following 5 criteria: (*a*) '*runtime*'; (*b*) 'solution quality'; (*c*) 'ease of implementation'; (*d*) 'robustness'; and (*e*) 'flexibility'.

'Solution quality' is often given as percentage deviation from the best known solution on a particular instance (Cordeau, Gendreau, Hertz, Laporte & Sormany, 2004). Golden et al. (1998) proposed four ways to report the *runtime* of an algorithm: (*1*) time to obtain the best solution; (*2*) time to obtain a solution within a 1% to 5% range of the best-known solution; (*3*) total computation time; and (*4*) no *runtime* at all. There is obviously a trade off between *runtime* needed and solution quality (larger *runtimes* often result in higher quality solutions), which might be translated to a two-dimensional objective, where both values can be plotted in a two-dimensional space: the points where no better values exist on both dimensions are Pareto-optimal (best compromise, Bräysy & Gendreau, 2005a).

'Ease of implementation' is also very important. Differences in coding, tuning and invested effort might be huge, but usually hard to determine, because most authors only report the best results (Bräysy & Gendreau, 2005a). Many heuristics inherently contain some random components. A 'robust' heuristic however does not perform poorly on any instance and should consequently give good solutions on the same instance (Bräysy & Gendreau, 2005a). Finally, one should always be aware that 'flexibility' is important for real-world instances, where constraints, the model or even the objective function might change over time (Bräysy & Gendreau, 2005a).

## 2.3 Conclusions

In Section 2.1, the 'Vehicle Routing Problem' and its variants, the 'Pick-up and Delivery Problem' and the 'multi-Traveling Salesman Problem with Time-Windows' were explained. Then, in Section 2.2 we discussed methods that can solve the *trip-assignment-problem*.

Within our research, the assignment problem is far more important than the routing problem: once orders are clustered per type of *vehicle*, and being assigned to *drivers* and *vehicles*, the routing for each group of *resources* is an easy task. We therefore need to select a method that (*1*) is able to assign *orders* to *routes* (incorporating *working-hour-regulations* on a general level), based on *vehicle-type*; and (*2*) is able to assign *vehicles* and *drivers* to *routes* in order to create *shifts* (incorporating *working-hour-regulations* on a personal level). Subsequently, we determine routes for each group of *resources*, having (*1*) and (*2*) as main proviso.

Based on the instance size, exact methods are not likely to succeed. Looking at the 'Vehicle Routing Problem' and its variants, the 'Pick-Up and Delivery Problem' and the 'multi-Traveling Salesman Problem with Time-Windows', most authors conclude that Tabu Search *metaheuristics* outperform other heuristics. The major drawback on Tabu Search however, is that all neighbors must be evaluated. Due to the problem size of the *trip-assignment-problem*, the number of neighbors becomes very large. Another heuristic, such as Simulated Annealing is therefore more useful for solving the *trip-assignment-problem*.

# Chapter 3

# Data Analysis

In this chapter, we focus on the analysis of historical and real data, attained from PAT. Four data sources are available at PAT: (*a*) the Transplan TMS database; (*b*) the personnel administration system (*drivers*); (*c*) the *vehicle* administration system; and (*d*) a PTV xServer (commercial routing product). For the analysis, we take data from January 2016 (one month) from these data sources. According to the data analyst at PAT, this is an average month without peek periods such as Easter and Christmas. We extract eight historical datasets: (*1*) *orders* and *trips*; (*2*) *drivers*; (*3*) *driver* availability; (*4*) *vehicles*; (*5*) *vehicle* availability; (*6*) planned schedules; (*7*) historical realizations (what times/distances were actually driven); and (*8*) *distance-* and *duration-matrix* with driving *durations* and *distances* (using PTV xServer). We employ these data sources to structure the data analysis.

Our challenge is to schedule *orders*, *vehicles* and *drivers* in an efficient and legal way. This chapter therefore contains seven parts: we start with the company structure in Section 3.1. In Section 3.2 we analyze the *vehicles*, followed by an analysis of the *drivers* in Section 3.3. We analyze the *orders* in Section 3.4, combined with the relationship between *orders* and *resources*. We use Section 3.5 to present *working-hour-regulations*. In Section 3.6 we analyze the current performance, obtained by manual planning. Finally, in Section 3.7 we draw conclusions from the previous parts.

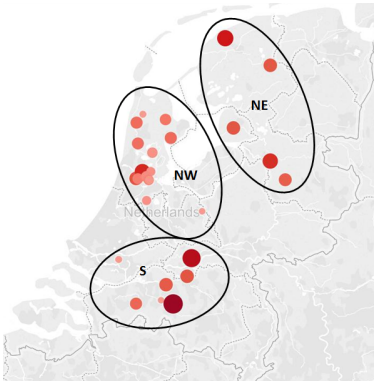



**Figure 3.1:** Departments and regions

## 3.1 Company structure

At PAT, there are several *departments*, usually located at customer distribution centers. *Trips* and *resources* are assigned to those *departments*. *Departments* are grouped to form *regions*. There are three *regions*: *South* (S), *North-West* (NW), and *North-East* (NE). The division of *departments* and *regions* is given in the map of Figure 3.1.

## 3.2 Fleet

The fleet consists of a heterogeneous set of *vehicles*. In this section we briefly describe properties that are applicable to the *vehicles* of PAT. *Vehicles* are scheduled depending on these properties. The values of properties are often hierarchical, enabling us to search for *vehicles* that are suited to be scheduled on *orders* with minimal, maximal or exact property values.

First we describe some general *vehicle-properties* in Section 3.2.1. Then, we continue with *vehicle-combinations* in Section 3.2.2, followed by the *vehicle* availability in Section 3.2.3. Finally, we discuss the implications for our model in Section 3.2.4.

### 3.2.1 Vehicle-properties

Several *vehicle-properties* are crucial in relation to *vehicles*. An *order* can require several *vehicles*, each with different properties. We describe *cooling*, *livery*, *employability-in-an-LHV*, and *ability-to-tow*. We describe the *vehicle-type* property in Section 3.2.2. Furthermore, see Appendix D for a description of some other *vehicle-properties*.

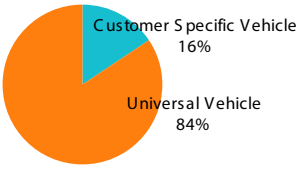



**Figure 3.2:** Vehicle livery: customer(-group) specific versus universal (PAT) vehicles

Due to the delicate properties of some loads (e.g., the risk of decay of certain foods), *cooling* is an important factor. In this respect, there are three types of *vehicles*: (*1*) those without cooling facilities; (*2*) those with one cooler; and (*3*) *vehicles* that can be compartmented into several temperature zones (*Multi-temperature*). A *Multi-temperature vehicle* is also appropriate to serve *orders* that minimally require a cooling facility. All *vehicles* (i.e., also *vehicles* equipped with cooling facilities) can be used for *orders* that do not require cooling.

Due to agreements with customers, some *vehicles* have special *livery*. This means that these *vehicles* have the branding of one, or a group of related PAT customers. In these cases, clients do not permit to visit a certain customer with a *vehicle* that has the *livery* of another customer. Some customers are connected to each other (e.g., shared by the same holding). It is allowed to share *vehicles* with different *livery* within a group of related customers. *Vehicles* without custom *livery* (blank or PAT branded) are allowed at all customers. An overview of the division between customer specific and universal *vehicles* is given in Figure 3.2.

Definition

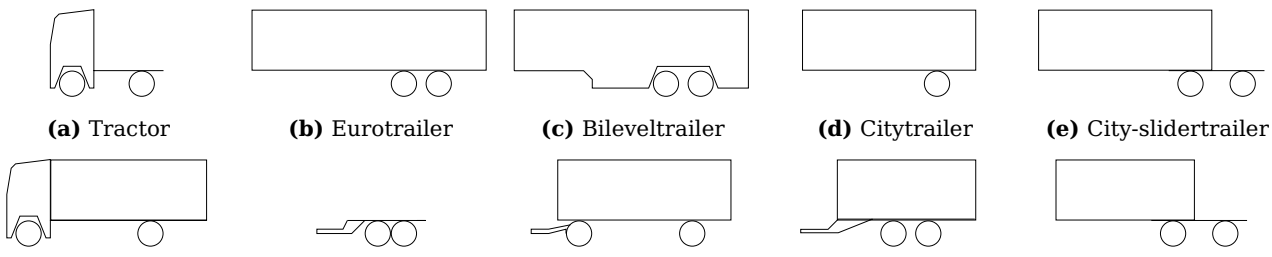
An ***LHV*** is a Long and Heavy Vehicle combination (longer than regular). LHVs are able to transport more goods per truck (economically and environmentally friendly) and require special exemptions and certificates.

The *employable-in-an-LHV* property defines if a *vehicle* has the features (e.g., a sign on the back, surplus engine power, etc.) to be employed in an LHV. A *vehicle* that has the *ability-to-tow*, is able to tow trailers. Both properties are used to distinguish *vehicles* of a certain *vehicle-type* that might be employed in an LHV from *vehicles* of the same *vehicle-type*, that might not be employed in an LHV, as well as the position of those *vehicles* in a *vehicle-combination* (see Section 3.2.2).

### 3.2.2 Vehicle-combinations

A *vehicle-combination* can be defined as the set of *vehicles* that can be assigned to a *trip*. To determine which set of *vehicles* can form a *vehicle-combination*, a relationship between *vehicles* and *vehicle-combinations* is required. We therefore define several *vehicle-combination-types*, each consisting of several *vehicle-types*. Every *vehicle* is part of a certain *vehicle-type*.

Several *vehicle-types* are defined (see Figure 3.3). Every *vehicle* belongs to exactly one *vehicle-type*. Besides *vehicle-type*, *vehicles* have several other properties, such as *cooling*, *livery*, *employability-in-an-LHV*, and *ability-to-tow* (see Section 3.2.1). Special cases are the *Citytrailer* and the *City-slidertrailer*: both trailers might be employed as a *Citytrailer*, however, the *City-slidertrailer* has the *ability-to-tow*, and can thus be used to tow other *Citytrailers*, which do not have that ability; in that case, both trailers should be *employable-*



**(a)** Tractor **(b)** Eurotrailer **(c)** Bileveltrailer **(d)** Citytrailer **(e)** City-slidertrailer

**(f)** Truck **(g)** Dolly **(h)** Trailer **(i)** Axletrailer **(j)** Euro-slidertrailer

**Figure 3.3:** Vehicle Types at PAT

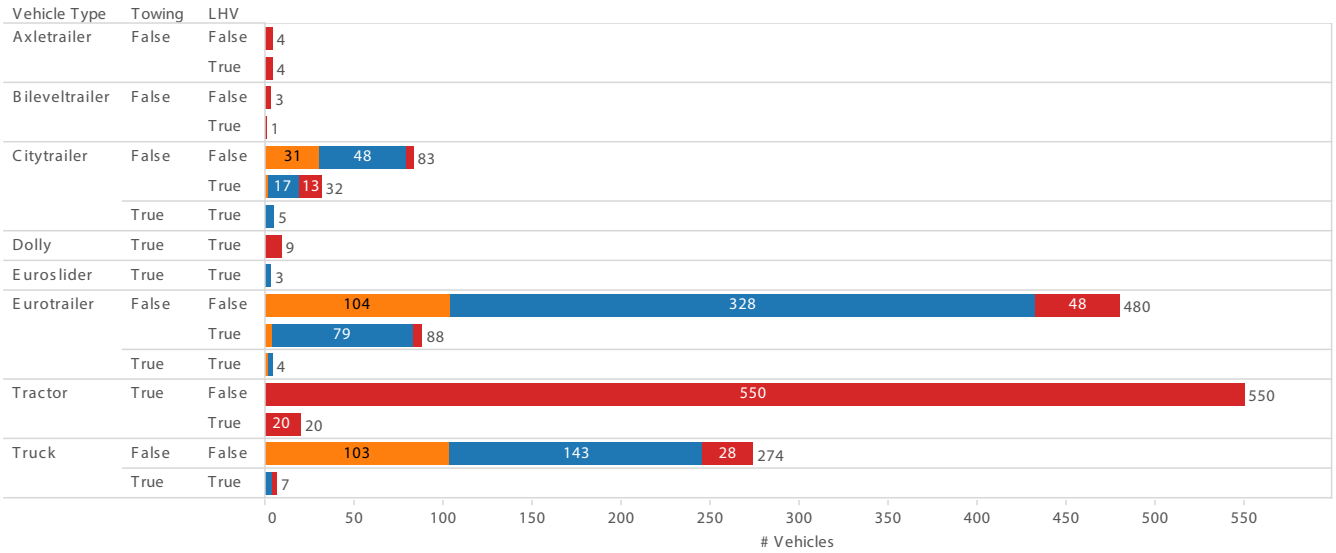



**Figure 3.4:** Number of available vehicles, per vehicle-type, ability-to-tow and employability-in-an-LHV properties, and cooling: Unregulated (red), Cooled (blue), Multi-temperature (orange).

*in-an-LHV*. A frequency distribution of *vehicles*, owned by PAT is provided in Figure 3.4, subdivided by the presence/absence of *ability-to-tow*, *employability-in-an-LHV*, and *cooling*.

Several *vehicles* can be combined to form *vehicle-combinations*, based on their *vehicle-type* and properties. Again, the *Citytrailer* and *City-slidertrailer* are in most cases interchangeable: therefore, we will now use *Citytrailer* for both *vehicle-types* and distinguish them using the *employable-in-an-LHV* and *ability-to-tow* properties. There are several *vehicle-combination-types* possible. Figure 3.5 depicts both stand-alone alternatives (Figure 3.5a & Figure 3.5b) as well as all possible combinations of *Tractor*-towed combinations and *Truck*-towed combinations (see also Appendix E).

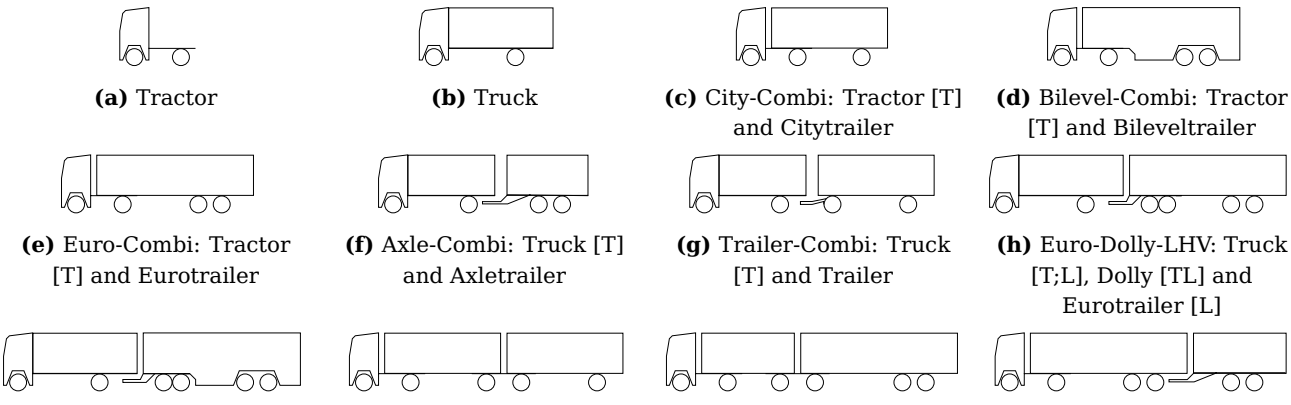
We define a set of *vehicle-combinations-types* that are requested (ordered) by the customers. The requested *vehicle-combination-type* is the required *vehicle-combination-type* for that *order*. In special cases, some *vehicle-combination-types* are not allowed to drive and the *order* must be fulfilled by one or more *vehicle-combinations* (e.g., LHVs are not allowed in bad weather conditions such as fog or hard winds, the *order* must then be split such that non-LHV *vehicle-combinations* can carry the load).

### 3.2.3 Availability

*Vehicle-availability* differs on a daily basis due to factors such as garage visits, new *vehicles*, and sold *vehicles*. In Figure 3.6 we display the daily fluctuations in *vehicle-availability* for *Eurotrailer* and *Tractor*.

### 3.2.4 Discussion

In this section we listed several *vehicle-properties* of which data is available. These properties define major restrictions in our model. Furthermore, we illustrate all possible stand-alone *vehicles* and *vehicle-combinations* in our dataset, as well as the relationship between them. Finally, we conclude that the availability of *vehicles* differs per day and that this is something to incorporate in our model.



**(a)** Tractor

**(b)** Truck

**(c)** City-Combi: Tractor [T] and Citytrailer

**(d)** Bilevel-Combi: Tractor [T] and Bileveltrailer

**(e)** Euro-Combi: Tractor [T] and Eurotrailer

**(f)** Axle-Combi: Truck [T] and Axletrailer

**(g)** Trailer-Combi: Truck [T] and Trailer

**(h)** Euro-Dolly-LHV: Truck [T;L], Dolly [TL] and Eurotrailer [L]

**(i)** Bilevel-Dolly-LHV: Truck [T;L], Dolly [T;L] and Bileveltrailer [L]

**(j)** City-Slider-LHV: Tractor [T;L], Citytrailer [T;L] (=City-slidertrailer) and Citytrailer [!T;L]

**(k)** Euro-Slider-LHV: Tractor [T;L], Euro-slidertrailer [T;L] and Eurotrailer [L]

**(l)** Euro-Axle-LHV: Tractor [T;L], Eurotrailer [T;L] and Axletrailer [L]

**Figure 3.5:** Vehicle-combination-types at PAT, in brackets ('[]') are the required properties of the individual vehicles: 'L' for employable-in-an-LHV='True' and 'T' for ability-to-tow='True'. If these letters are absent, the property is irrelevant. If the vehicle does NOT require that property, we use an exclamation mark (!).

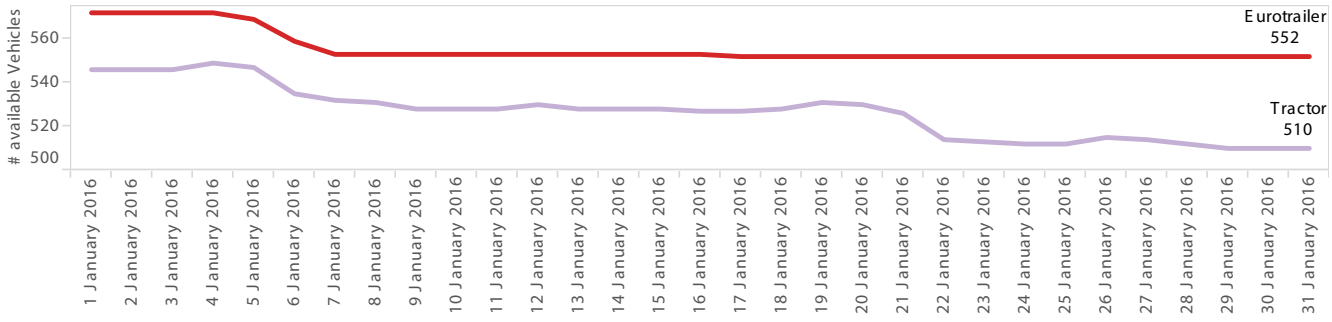



**Figure 3.6:** Daily number of available Eurotrailers (red) and Tractors (purple).

## 3.3 Drivers

In this section we approach the data in relation to human resources (*drivers*) and their availability. We describe several properties that are involved with *drivers*. First we explain the availability of personnel, then we explain the different *driver-license*-types, followed by a discussion.

*Drivers* schedule their working days (*availability*) in cooperation with their team leader. At PAT, *drivers* are often scheduled for work outside their *availability-period*. Ideally, the availability of an employee can cover multiple days (allowing overnight *shifts*). The extent to which *drivers* are actually employed in comparison to their (full or part-time) employment contracts, is also a performance measure (see Section 3.6.2).

The *drivers-license*-type defines the types of *vehicles* a *driver* is allowed to operate. The most basic type is the *C-license*, where *drivers* are allowed to drive *Trucks* and *Tractors*. The second type is the *CE-license*, where *drivers* are allowed to pull (semi)trailers. The third type is the *LHV-certificate*, which is required for driving LHVs.

In this section we describe data in terms of *drivers*, their *availability-period* and the restrictions in this sense. PAT has several preferences and demands that incorporate employee employability, such as location experience (e.g., related to damages). Unfortunately, preferences and demands cannot be supported with data, by now. If PAT wants to incorporate such wishes in the future, related data should be collected and maintained. At present, only data exists on *drivers-licenses*, therefore this is the only *driver-property* that we include.

## 3.4 Orders

Apart from *vehicle-type* and employee characteristics, *orders* also define constraints on the *trip*-assignments. In Section 3.4.1 we give an overview of the different *trip-types* encountered in the PAT data set. In Section 3.4.2 we analyze the number of *stops* related to *trips*. The analysis continues with Section 3.4.3 about the *trip*-related *vehicle-properties*. In Section 3.4.4 we analyze repeating *orders*. Finally, in Section 3.4.5 we discuss the influences on the rest of our research.

### 3.4.1 Trip-type

In Section 1.2.1, we described three types of *orders*: (*1*) *Network-orders*; (*2*) *Hire-orders*; and (*3*) *Trip-orders*. Now, a more detailed classification (the *trip-type*) related to the definitions given before is required. A *Trip-order* contains the request for a full truckload, begins at point A ('$A$'), implies visits to a given number *stops* ('$A \rightarrow$ n'), and finishes again at startingpoint A ('$A \rightarrow$ n $\rightarrow A$') or at some other point B ('$A \rightarrow$ n $\rightarrow B$'), which might be without touching any intermediate *stops* ('$A \rightarrow B$'). A *Hire-order* contains the request for a full truckload, starts at point A ('$A$'), visits a number of unknown *stops* ('$A \rightarrow$ ?') and finishes again at startingpoint A ('$A \rightarrow$ ? $\rightarrow A$') or at some other point B ('$A \rightarrow$ ? $\rightarrow B$'). Another type of *orders* that the PAT dataset reveals is *orders* that do not involve any transportation at all ('$A$') (e.g., *Loading* a truck).

| | End loc. | | # stops | | | |
|---|---|---|---|---|---|---|
| Trip-type | Shared(AA) | Different(AB) | Known(1) | Known(2) | Known(n) | Unknown(?) |
| '$A \rightarrow$ n $\rightarrow A$' | • | | | | • | |
| '$A \rightarrow$ n $\rightarrow B$' | | • | | | • | |
| '$A \rightarrow$ ? $\rightarrow A$' | • | | | | | • |
| '$A \rightarrow$ ? $\rightarrow B$' | | • | | | | • |
| '$A \rightarrow B$' | | • | | • | | |
| '$A$' | • | | • | | | |

**Table 3.1:** Trip-type by order-property

We use two variables to describe the *trip-type* (see Figure 3.7 for data):

- End-location: same location as start ('$A \rightarrow ... \rightarrow A$') or different locations ('$A \rightarrow ... \rightarrow B$'). Approximately 66% of the *orders*, start and finish at the same location. The other 34% of the *orders* end at another location.

- Number (#) of *stops*: detailed *stop-information* is known (1: '$A$'; 2: '$A \rightarrow B$'; n: '$... \rightarrow n \rightarrow ...$') or unknown ('$... \rightarrow$ ? $\rightarrow ...$'). On average for 79% of the *orders* all *stop-information* is known. The other part of the *orders* are *Hire-orders*, without *stop-information*.

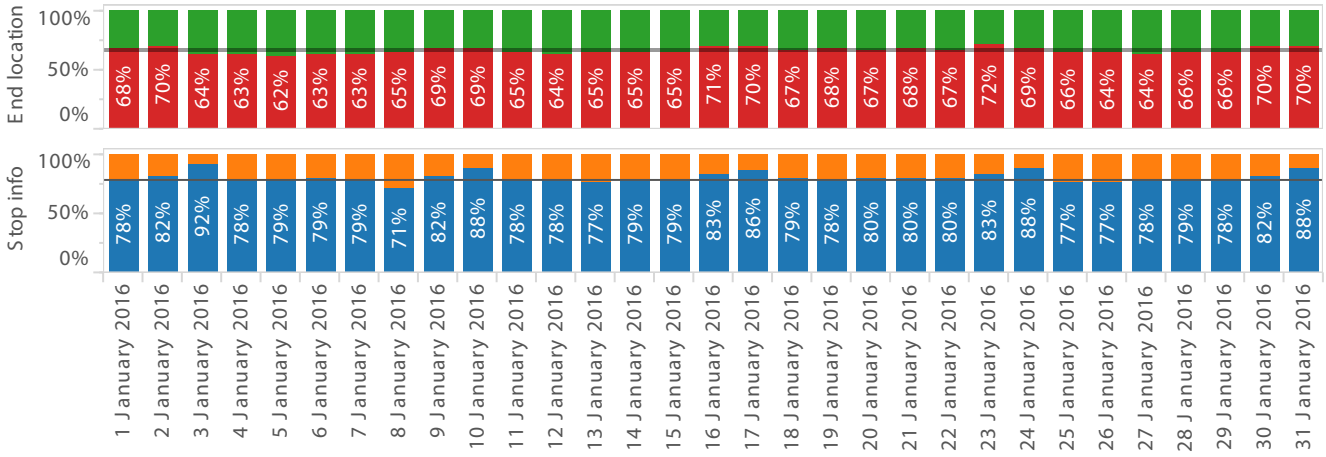



**Figure 3.7:** Daily division(%) between orders with different start-/end-locations ('$A \rightarrow ... \rightarrow B$'; green) and shared location ('$A \rightarrow ... \rightarrow A$'; red); and orders with unknown stop-information (Hire-orders; '$... \rightarrow ? \rightarrow ...$'; orange) and known stop-information ('$... \rightarrow n \rightarrow ...$'; blue).

The relationship between these order variables and *trip-type* is presented in Table 3.1. Figure 3.8 shows the division of the total amount of *orders* on a daily basis. In general, we observe that working days include more *orders* than weekend days and that Sundays contain fewer *orders* than Saturdays. In total, we found 33,013 *orders*, resulting in an average of 1065 *orders* per day.

### 3.4.2 Stops

The daily number of *stops*, as well as the average number of *stops* per *order* can be observed from Figure 3.9a. Note that the number of *stops* per *order* is quite stable. The number of *stops* differs during the day: from Figure 3.9b we derive that the number of *stops* reaches its maximum between 5.00 hour and 8.00 hour in the morning. Most *stops* are also outsourced to charters during peek hours, suggesting a proper use of charters.

### 3.4.3 Requested-vehicle-properties

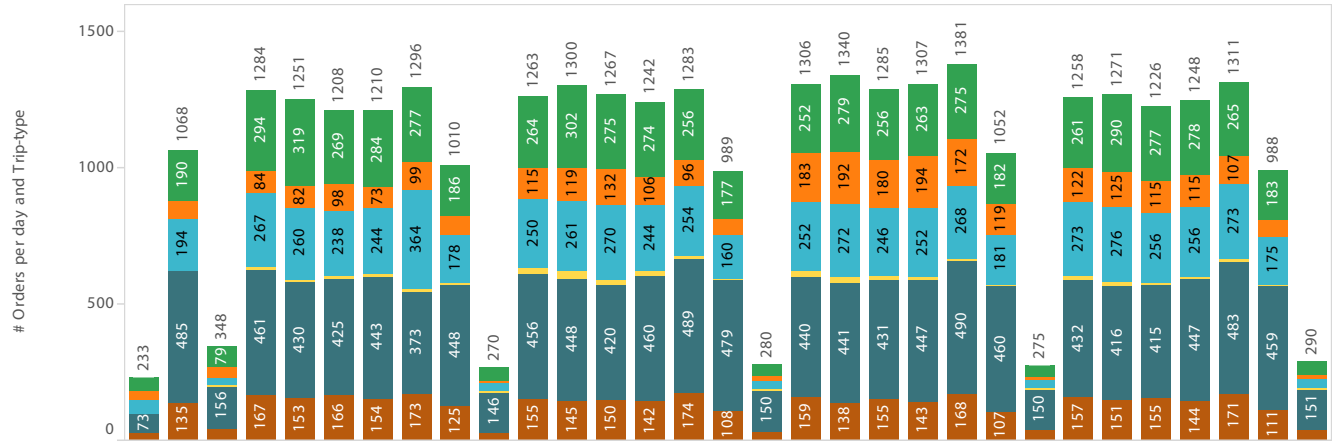
| | Orders | | Routes | |
|---|---|---|---|---|
| # | Time | Stops | Route | Frequency |
| 1 | 10.00 | A-B-C | 10.00:A-B-C | 2 |
| 2 | 10.00 | A-C-B | 10.00:A-C-B | 3 |
| 3 | 11.00 | A-B-C | 11.00:A-B-C | 1 |
| 4 | 10.00 | A-C-B | 11.00:A-?-A | 1 |
| 5 | 11.00 | A-?-A | | |
| 6 | 10.00 | A-B-C | | |
| 7 | 10.00 | A-C-B | | |

**Table 3.2:** Repeating orders

Each *order* reflects a certain load. Based on the characteristics of that load, an *order* requires a *vehicle* with certain properties. These properties are delegated to the *trips* that service certain *orders*. Each *trip* then has to be accomplished by a *vehicle* that complies with the *requested-vehicle-properties*. These properties were treated in Section 3.2. Figure 3.10 shows the division of *orders* requiring certain *vehicle-combinations* and certain *cooling*.

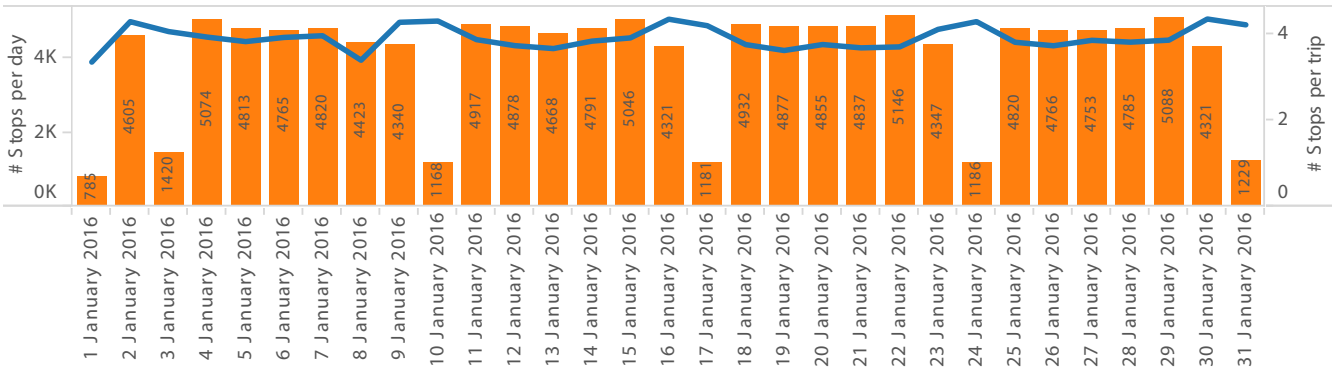
### 3.4.4 Repeating orders

We explored the nature of *orders*. Some *trips* are ordered more than once. In this case, we define a unique route as being a combination of start-time and the ordered *stops*: if the start-time, and the sequence of *stops* are identical for two *orders*, these *orders* belong to the same route (see Table 3.2 for examples). Figure 3.11 details the relationship between *orders* and the frequency of repeated *orders*. If most routes had a high frequency, this means that the *orders* are often identical and this might be useful for in advance (repeated) calculations within solution algorithms. However, most *orders* appear to occur only once or only a few times within the entire month. Hence, we speculate that historic schedules are of limited value as a source of information for future *trip*-assignments.

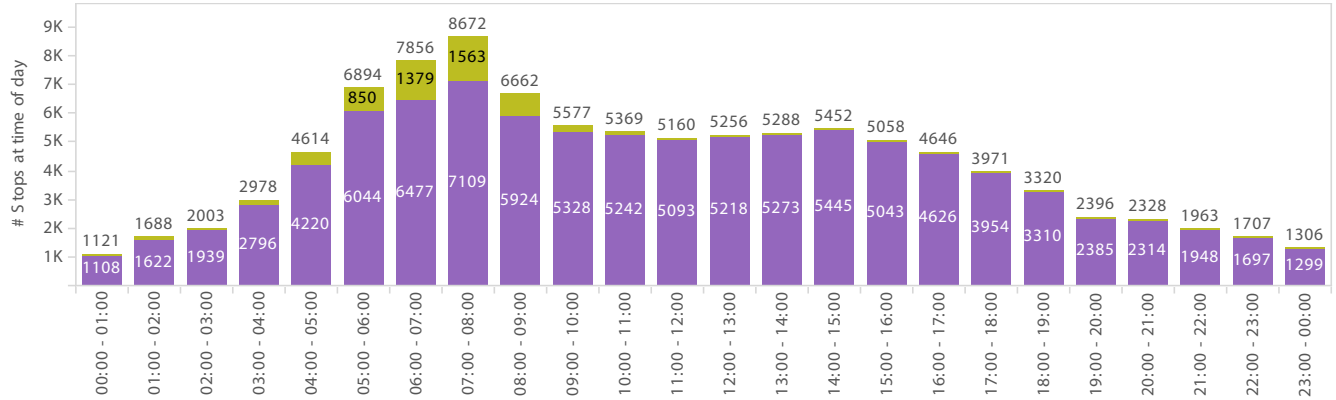



| Trip-type | 1 January 2016 | 2 January 2016 | 3 January 2016 | 4 January 2016 | 5 January 2016 | 6 January 2016 | 7 January 2016 | 8 January 2016 | 9 January 2016 | 10 January 2016 | 11 January 2016 | 12 January 2016 | 13 January 2016 | 14 January 2016 | 15 January 2016 | 16 January 2016 | 17 January 2016 | 18 January 2016 | 19 January 2016 | 20 January 2016 | 21 January 2016 | 22 January 2016 | 23 January 2016 | 24 January 2016 | 25 January 2016 | 26 January 2016 | 27 January 2016 | 28 January 2016 | 29 January 2016 | 30 January 2016 | 31 January 2016 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A->B | 49 | 190 | 79 | 294 | 319 | 269 | 284 | 277 | 186 | 48 | 264 | 302 | 275 | 274 | 256 | 177 | 45 | 252 | 279 | 256 | 263 | 275 | 182 | 45 | 261 | 290 | 277 | 278 | 265 | 183 | 48 |
| A | 34 | 64 | 39 | 84 | 82 | 98 | 73 | 99 | 70 | 14 | 115 | 119 | 132 | 106 | 96 | 61 | 16 | 183 | 192 | 180 | 194 | 172 | 119 | 12 | 122 | 125 | 115 | 115 | 107 | 57 | 19 |
| A->?->A | 52 | 194 | 28 | 267 | 260 | 238 | 244 | 364 | 178 | 26 | 250 | 261 | 270 | 244 | 254 | 160 | 30 | 252 | 272 | 246 | 252 | 268 | 181 | 27 | 273 | 276 | 256 | 256 | 273 | 175 | 32 |
| A->?->B | | | 1 | 11 | 7 | 12 | 12 | 10 | 3 | 7 | 23 | 25 | 20 | 16 | 14 | 4 | 8 | 20 | 18 | 17 | 8 | 8 | 3 | 5 | 13 | 13 | 8 | 8 | 12 | 3 | 3 |
| A->n->A | 73 | 485 | 156 | 461 | 430 | 425 | 443 | 373 | 448 | 146 | 456 | 448 | 420 | 460 | 489 | 479 | 150 | 440 | 441 | 431 | 447 | 490 | 460 | 150 | 432 | 416 | 415 | 447 | 483 | 459 | 151 |
| A->n->B | 25 | 135 | 45 | 167 | 153 | 166 | 154 | 173 | 125 | 29 | 155 | 145 | 150 | 142 | 174 | 108 | 31 | 159 | 138 | 155 | 143 | 168 | 107 | 36 | 157 | 151 | 155 | 144 | 171 | 111 | 37 |

**Figure 3.8:** Number of orders per day (extracted between January, 1, 2016 and January, 31, 2016), based on trip-type: '$A \to B$' (green); '$A$' (orange); '$A \to ? \to A$' (lightblue); '$A \to ? \to B$' (yellow); '$A \to n \to A$' (darkblue); '$A \to n \to B$' (brown). Total on top of the graph.




**(a)** Total number of stops per day (bars; orange) and average number of stops per trip (line; blue).




**(b)** Partition per hour of orders that were executed by a charter in yellow and by own resources in purple. The total hourly stops (entire month) on top.

**Figure 3.9:** Number of stops per day and at time of the day

| Veh Combi Type | 1 January 2016 | 2 January 2016 | 3 January 2016 | 4 January 2016 | 5 January 2016 | 6 January 2016 | 7 January 2016 | 8 January 2016 | 9 January 2016 | 10 January 2016 | 11 January 2016 | 12 January 2016 | 13 January 2016 | 14 January 2016 | 15 January 2016 | 16 January 2016 | 17 January 2016 | 18 January 2016 | 19 January 2016 | 20 January 2016 | 21 January 2016 | 22 January 2016 | 23 January 2016 | 24 January 2016 | 25 January 2016 | 26 January 2016 | 27 January 2016 | 28 January 2016 | 29 January 2016 | 30 January 2016 | 31 January 2016 | Average |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 233 | 1068 | 348 | 1284 | 1251 | 1208 | 1210 | 1296 | 1010 | 270 | 1263 | 1300 | 1267 | 1242 | 1283 | 989 | 280 | 1306 | 1340 | 1285 | 1307 | 1381 | 1052 | 275 | 1258 | 1271 | 1226 | 1248 | 1311 | 988 | 290 | |
| Euro-Combi | 99 | 704 | 231 | 794 | 775 | 773 | 781 | 846 | 663 | 176 | 783 | 805 | 792 | 789 | 838 | 644 | 181 | 762 | 771 | 753 | 776 | 840 | 642 | 173 | 763 | 785 | 748 | 770 | 833 | 635 | 172 | 648 |
| Truck | 23 | 156 | 36 | 234 | 224 | 223 | 209 | 231 | 151 | 31 | 243 | 235 | 229 | 228 | 229 | 147 | 31 | 241 | 241 | 229 | 233 | 232 | 152 | 27 | 240 | 236 | 226 | 233 | 230 | 158 | 28 | 180 |
| City-Combi | 11 | 104 | 43 | 120 | 124 | 110 | 117 | 117 | 109 | 28 | 121 | 121 | 115 | 108 | 113 | 112 | 26 | 116 | 116 | 116 | 107 | 114 | 109 | 33 | 121 | 110 | 101 | 95 | 108 | 104 | 34 | 96 |
| City-Slider-LHV | 2 | 23 | | 13 | 13 | 11 | 12 | 12 | 17 | 2 | 20 | 15 | 15 | 19 | 18 | 20 | 1 | 18 | 19 | 18 | 19 | 19 | 21 | 2 | 16 | 19 | 22 | 22 | 21 | 16 | 2 | 15 |
| Euro-Dolly-LHV | | 3 | | 3 | 3 | 2 | 2 | 4 | 2 | | 3 | 3 | 4 | 3 | 4 | 2 | | 1 | 2 | 1 | 2 | 2 | 2 | | 3 | 4 | 6 | 3 | 5 | 4 | | 3 |
| Tractor | | | | 5 | 4 | 2 | 1 | 3 | | | 4 | 5 | 7 | 3 | 2 | 1 | 1 | 2 | 3 | 3 | 4 | | | 1 | 3 | 4 | 6 | 3 | 2 | | | 3 |
| Axle-Combi | | | | 2 | 1 | 1 | 1 | | | | 3 | 2 | | 3 | 2 | 2 | | 1 | 2 | 2 | 1 | 2 | 1 | | 1 | | 2 | 1 | 2 | 1 | | 2 |
| Other | 98 | 78 | 38 | 113 | 107 | 86 | 87 | 83 | 68 | 33 | 86 | 114 | 105 | 89 | 77 | 61 | 40 | 165 | 186 | 163 | 165 | 172 | 125 | 39 | 111 | 113 | 115 | 121 | 110 | 70 | 54 | 99 |

| Veh Combi Cooling | 1 January 2016 | 2 January 2016 | 3 January 2016 | 4 January 2016 | 5 January 2016 | 6 January 2016 | 7 January 2016 | 8 January 2016 | 9 January 2016 | 10 January 2016 | 11 January 2016 | 12 January 2016 | 13 January 2016 | 14 January 2016 | 15 January 2016 | 16 January 2016 | 17 January 2016 | 18 January 2016 | 19 January 2016 | 20 January 2016 | 21 January 2016 | 22 January 2016 | 23 January 2016 | 24 January 2016 | 25 January 2016 | 26 January 2016 | 27 January 2016 | 28 January 2016 | 29 January 2016 | 30 January 2016 | 31 January 2016 | Average |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 233 | 1068 | 348 | 1284 | 1251 | 1208 | 1210 | 1296 | 1010 | 270 | 1263 | 1300 | 1267 | 1242 | 1283 | 989 | 280 | 1306 | 1340 | 1285 | 1307 | 1381 | 1052 | 275 | 1258 | 1271 | 1226 | 1248 | 1311 | 988 | 290 | |
| Unregulated | 5 | 46 | 35 | 95 | 91 | 91 | 78 | 99 | 43 | 6 | 91 | 92 | 102 | 81 | 88 | 36 | 8 | 86 | 103 | 91 | 91 | 87 | 45 | 8 | 90 | 94 | 89 | 77 | 83 | 33 | 7 | 67 |
| Cooled | 119 | 762 | 239 | 836 | 812 | 829 | 811 | 881 | 716 | 215 | 852 | 854 | 828 | 843 | 889 | 724 | 217 | 825 | 807 | 815 | 814 | 878 | 702 | 214 | 819 | 803 | 791 | 801 | 877 | 711 | 211 | 693 |
| Multi-Temperature | 11 | 182 | 36 | 240 | 241 | 202 | 234 | 233 | 183 | 16 | 234 | 240 | 232 | 229 | 229 | 168 | 15 | 230 | 244 | 216 | 237 | 244 | 180 | 14 | 238 | 261 | 231 | 249 | 241 | 174 | 18 | 184 |
| Other | 98 | 78 | 38 | 113 | 107 | 86 | 87 | 83 | 68 | 33 | 86 | 114 | 105 | 89 | 77 | 61 | 40 | 165 | 186 | 163 | 165 | 172 | 125 | 39 | 111 | 113 | 115 | 121 | 110 | 70 | 54 | 99 |

**Figure 3.10:** Number of orders that should be executed by a certain vehicle-combination-type, and cooling.

| Route Frequency | # of orders | % of orders | # of routes | % of routes |
|---|---|---|---|---|
| 1 | 2364 | 7.148% | 2364 | 40.25% |
| 2 | 1990 | 6.018% | 995 | 16.94% |
| 3 | 1416 | 4.282% | 472 | 8.04% |
| 4 | 2616 | 7.910% | 654 | 11.13% |
| 5 | 1510 | 4.566% | 302 | 5.14% |
| 6-7 | 1390 | 4.203% | 217 | 3.69% |
| 8-9 | 1686 | 5.098% | 201 | 3.42% |
| 10-12 | 1535 | 4.642% | 140 | 2.38% |
| 13-16 | 1783 | 5.392% | 121 | 2.06% |
| 17-20 | 2081 | 6.293% | 109 | 1.86% |
| 21-25 | 2719 | 8.222% | 115 | 1.96% |
| 26-30 | 1653 | 4.998% | 61 | 1.04% |
| 31-50 | 2160 | 6.532% | 55 | 0.94% |
| 51-100 | 2784 | 8.419% | 42 | 0.72% |
| 101-200 | 2339 | 7.073% | 18 | 0.31% |
| 201+ | 3044 | 9.205% | 8 | 0.14% |

**Figure 3.11:** Repeating orders. For example: 2616 orders are not unique and have a route frequency of four, meaning that there are 654 unique routes which each are repeated four times within the dataset (January 2016).

### 3.4.5 Discussion

In this section we discussed and analyzed the *orders* in the PAT dataset in terms of *trip-types*, *stops*, and repetition. Unfortunately, the information related to the required *vehicles* is scarce. We therefore have to limit our set of *vehicle-property* requirements to *vehicle-combination-type*, *livery*, and *cooling*. Most *orders* enclose lots of *stop-information*, enabling us to properly schedule breaks and rests.

## 3.5 Working-hour-regulations

*Working-hour-regulations* define a *shift*, and each *shift* (or combination of *shifts*) should obey the *working-hour-regulations*. This section elaborates on the legislation and regulation of driving and working hours within the European Union as described by Inspectie Leefomgeving en Transport (2009, 2010a, 2010b), Rijksoverheid (2016a, 2016b) and Ministerie van Sociale Zaken en Werkgelegenheid (2011).

Because the rules for driving and working slightly differ, one should distinguish between driving- and working-time. Driving-time is the time the *driver* is actually *Driving* the *vehicle* (including rush hours and maneuvering). Working-time is all the time the *driver* is not free to spend his/her time (including *Loading*, *Unloading*, and *Waiting*). Furthermore, there are rules that apply to different time-periods. These rules prescribe maximum driving- and working-times within that period, which, by law, must be followed by breaks or rests. The following paragraphs describe these rules for different periods of time.

'Uninterrupted working- and driving-times' are the accumulated working- or driving-times without any breaks longer than the legally required breaks. The upper limit for uninterrupted driving is at most 4.5 hours. The upper limit for uninterrupted working is set at 6 hours. After a driving period of 4.5 hours, a 45 minute break must be taken. This duration

of this break can be reduced to 30 minutes, if the 4.5 hour driving period was split by a break of at least 15 minutes.

1. Uninterrupted driving: at most 4.5 hours;
2. Uninterrupted working: at most 6 hours;
3. After a driving period of 4.5 hours, a 45 minute break must be taken. This break can be 30 minutes, if the 4.5 hour driving period was split by a break of at least 15 minutes.

'Daily working- and driving-times' are the accumulated working- or driving-times between any daily or weekly rests. The following rules apply:

4. Regular daily driving may be at most 9 hours;
5. Extended daily driving may be at most 10 hours;
6. Extended daily driving may occur at most two times within each calendar week;
7. If the daily working-time is between 6 and 9 hours, at least 30 minutes of break time must be scheduled in blocks of at least 15 minutes;
8. If the daily working time is more than 9 hours, at least 45 minutes of break time must be taken in blocks of at least 15 minutes.

Workdays are always followed by a 'daily rest'. The following rules apply:

9. Within 24 hours after ending the last daily or weekly rest, a new (daily or weekly) rest must have been taken;
10. Regular daily rest: at least 11 consecutive hours;
11. Split daily rest: at least 3 consecutive hours + at least 9 consecutive hours;
12. Short daily rest: at least 9 consecutive hours;
13. Short daily rests can be taken at most 3 times between two weekly rests;
14. A weekly rest also counts as a daily rest.

The 'daily rest rules' result in a maximum length of the working day including breaks, which can be at most 15 hours in case of a short daily rest, or 13 hours in case of a regular daily rest. This working day may include at most 12 hours of *Working* and 10 hours of *Driving*.

There are also limits on the accumulated 'weekly working- and driving-times'. The following rules apply:

15. Weekly driving: at most 56 hours;
16. Weekly working: at most 60 hours;
17. Within 144 hours after ending the last weekly rest, a new weekly rest must have been started.

The 'weekly rest periods' must comply to the following rules:

18. Regular weekly rest: at least 45 hours;
19. Short weekly rest: at least 24 hours;
20. A short weekly rest must always be followed by a regular weekly rest at the end of the next week.

Finally there is a two weekly driving rule, which states that the following:

21. Two weekly driving must be at most 90 hours.

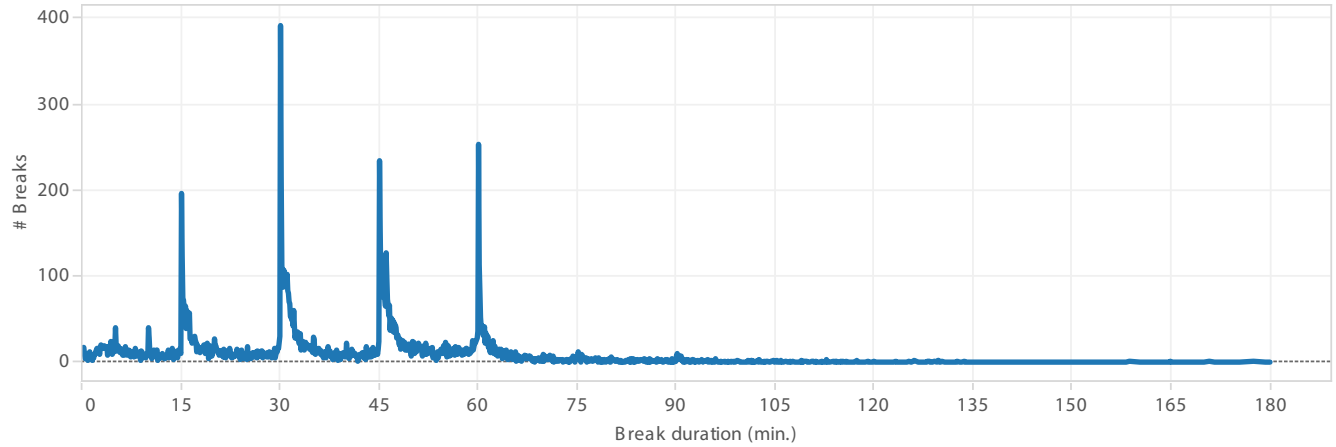



**Figure 3.12:** Frequency of the occurrence of the duration of breaks, in minutes.

## 3.6 Performance

The present section focuses on the current performance. We describe the performance of activities in Section 3.6.1. The subject of Section 3.6.2 is the effectiveness and efficiency of available resources.

### 3.6.1 Activity duration

We compare realization data (administered by the *drivers* in their onboard computers) with planned schedules. The realization data is grouped by *shift* and features all activities that the *driver* and *vehicle* performed for each *trip* within that *shift*. This data is linked by the Transplan TMS system with the *orders* and orderlines (*stops*). We are able to relate planned data to execution data in only 42% of the cases, because of (partly) missing links between datasets. Therefore this part of our analysis is probably not representative for the entire dataset. The causes of the mismatch for a large part of the *orders* are threefold: (*1*) the *order* was executed by a chartered *vehicle* and *driver*; (*2*) the *vehicle* that executed the *order* might not have a (working) onboard computer; or (*3*) the *order* execution data might be manually corrected. We look at the performance of three types of *activities*: (*1*) breaks; (*2*) stops; and (*3*) driving.

*Drivers* are required to take breaks with some minimum duration at regular intervals (see Section 3.5). It is interesting to analyze the current duration of breaks for each *driver*. As we see from Figure 3.12, breaks have several durations, there are however (positive skewed) peaks around every 15, 30, 45 and 60 minutes. Because *drivers* are able to manually 'correct' their breaks, we cannot be sure if this is due to these manual corrections or the fact that *drivers* do actually take breaks in exact blocks of 15 minutes. Legally (see Section 3.5), there are options for breaks for at least 15, 30 or 45 minutes, which does not explain the peak at 60 minutes. We conclude from this that *drivers* take the executed duration of their planned breaks quit accurately and that there is no need to take longer than legally required breaks into account.

We also analyze the differences between planned and actual *stop* durations. Logically, due to fluctuations, *stop* durations are not always exactly as long as planned in advance. There are a lot of extreme values (between 40-1428 minutes shorter and 60-1390 minutes longer), which are excluded from the analysis because these outliers are expected to be unrealistic and are usually a result of faulty administration. We divide the *stop* durations in bins of 5 minutes each. In Figure 3.13 we see the binned non out-lier differences. On average, realized *stops* are 1 minute longer than planned, with a standard deviation of 15 minutes. Due to the fact that most customers of PAT provide planned *stop* durations in

their *orders*, the differences in *stop* duration are related to individual customers, addresses, *vehicle-combination-types* and employees (see Appendix F).

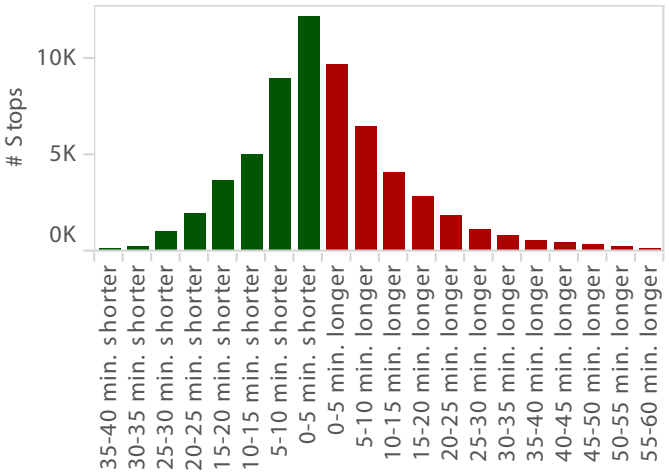



**Figure 3.13:** Difference (binned) between planned and realized stop durations, in minutes.

To automate the scheduling process, we need *distance-* and *duration-matrices* between addresses. We decide to use a commercial service (PTV xServer) that is also currently used in the transplan TMS for calculating distances and times. We compare the realized *Driving activities* with their planned equivalents (see Figure 3.13). We calculate the relationship between speed and time of the day: lower speeds during rush hours and higher during night time, however looking at the scatter plot (see Figure 3.14b) for driving speed related to time of the day, no real relationships can be seen.

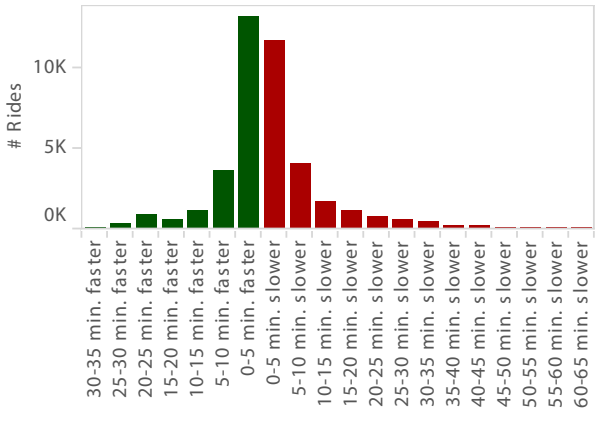
### 3.6.2 Efficient use of resources

We analyze the proportion of *orders* that is outsourced, we compare the availability of the *resources* with the execution of those *resources*, and we analyze the loss of efficient driving hours by overhead (useless mobilization of resources).

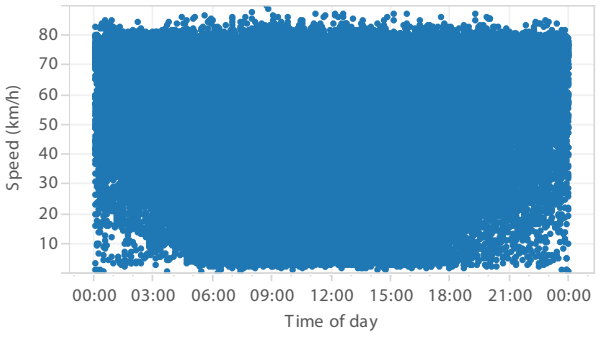
When there are no suitable *resources* available, PAT can choose to outsource certain *orders* to a charter company (chartered). This means that no *Own resources* (neither *vehicles* nor *drivers*) are used, and the *order* is completely executed by another company. Every day, about 9% of the *orders* is chartered. In Section 3.4.2 we show that most *stops* are chartered in peak hours.

We define overhead as the useless employment of *resources* (i.e. driving without a load, or *pre-trips* and *return-trips*). From Figure 3.15 we conclude that, on average about 10% time and 11% distance due to overhead is added on each *order*.

We compare the availability of *resources* with the requirements for those *resources*. We consider *vehicle-availability*, *employee-availability*, and we look at the *total-duration* of *order* (overhead excluded). For the calculation of the daily *vehicle-availability*, we took the sum of the daily available *vehicles* that need a *driver*, known *Tractors* and *Trucks*.




**(a)** Difference (binned) between planned and realized drive durations, in minutes.




**(b)** Speed at different times of the day.

**Figure 3.14:** Driving duration performance

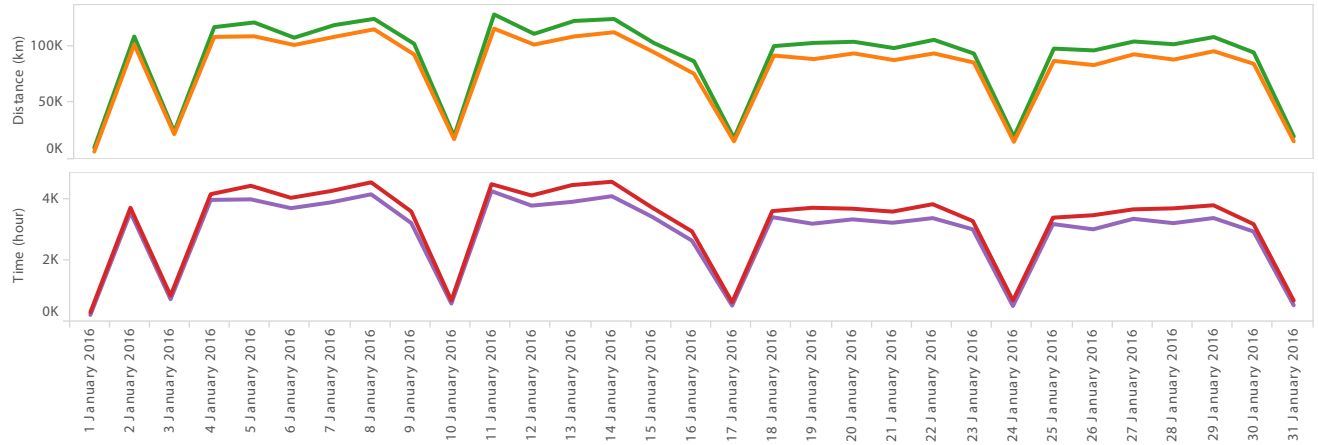
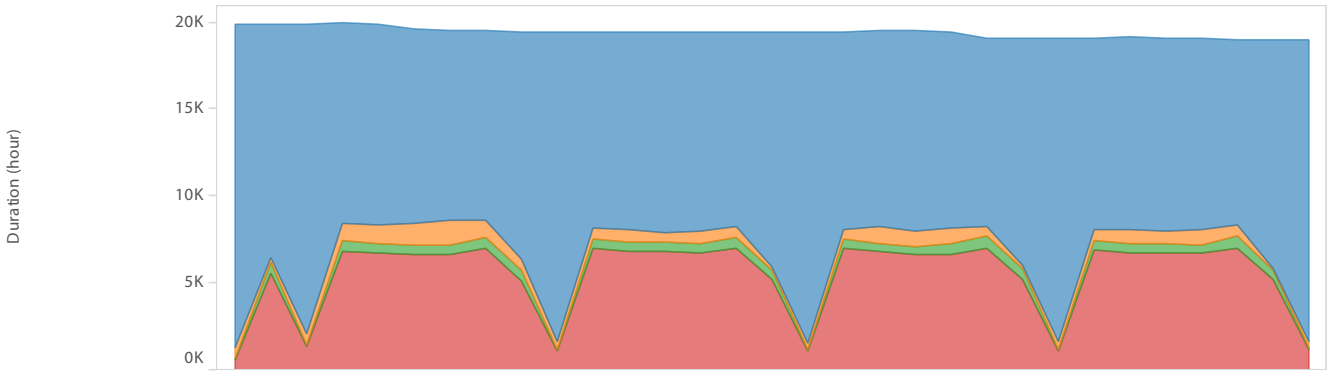



**Figure 3.15:** Distance and duration due to overhead: distance (kilometers) with (green) and without (orange) overhead, and duration (hours) with (red) and without (purple) overhead.

The *vehicle-availability* is calculated by multiplying the daily availability of *vehicles* by 24 hours (number of hours in a day). For the calculation of the total *driver-availability*, we summed each *drivers* daily availability. Due to *working-hour-regulations*, we decide to set the maximum daily availability at 12 hours per driver (60 hours per week, 5 days). Then we calculate the sum of intervals between the start and finish of each *order* delivered to a client. We refined this by dividing it into *orders* that were chartered to be executed by third parties and *orders* that were executed by PAT *vehicles* and *drivers*. Figure 3.16 provides a view of the available and used *resources*. The figure reveals that in particular *vehicles* are a source of excess availability, standing still at a parking place during large parts of the day, while charters are hired (mostly during peak hours) at the same time.

## 3.7 Conclusions

In this chapter, we analyzed the data made available by PAT over a particular representative month (January 2016). We discussed *vehicles*, *orders*, *trips*, *drivers*, *working-hour-regulations*, and the current performance of PAT. Most *resource*- and *order*-related information is currently captured in the planners' heads. Due to the limited amount of information regarding *resource*-related constraints in the dataset, the target is to develop a model which only includes the information that is available (i.e., we only supply our models with available information). Our final target is to improve the current performance of PAT and to make the current manual planning method more simple, convenient and effective.




| | Average | 1 January 2016 | 2 January 2016 | 3 January 2016 | 4 January 2016 | 5 January 2016 | 6 January 2016 | 7 January 2016 | 8 January 2016 | 9 January 2016 | 10 January 2016 | 11 January 2016 | 12 January 2016 | 13 January 2016 | 14 January 2016 | 15 January 2016 | 16 January 2016 | 17 January 2016 | 18 January 2016 | 19 January 2016 | 20 January 2016 | 21 January 2016 | 22 January 2016 | 23 January 2016 | 24 January 2016 | 25 January 2016 | 26 January 2016 | 27 January 2016 | 28 January 2016 | 29 January 2016 | 30 January 2016 | 31 January 2016 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Total Available Vehicle Time (hour) | 19392 | 19872 | 19872 | 19872 | 19944 | 19872 | 19584 | 19512 | 19488 | 19416 | 19416 | 19416 | 19464 | 19416 | 19416 | 19416 | 19416 | 19416 | 19440 | 19512 | 19488 | 19392 | 19104 | 19080 | 19056 | 19056 | 19128 | 19104 | 19032 | 18984 | 18984 |
| Available Employee Time (hour) | 8106 | 1860 | 8071 | 2716 | 10438 | 10323 | 10520 | 10767 | 10732 | 7857 | 2126 | 9982 | 9952 | 9745 | 9883 | 10090 | 7155 | 2008 | 9753 | 9997 | 9755 | 10090 | 10041 | 7241 | 2011 | 9724 | 9787 | 9639 | 9876 | 10103 | 7027 | 2004 |
| Available Employee Time (hour), max 12 hours | 6588 | 1272 | 6455 | 2046 | 8453 | 8346 | 8455 | 8605 | 8624 | 6364 | 1593 | 8127 | 8046 | 7919 | 8009 | 8217 | 5918 | 1554 | 8027 | 8211 | 7997 | 8188 | 8238 | 6033 | 1616 | 8041 | 8055 | 7957 | 8093 | 8285 | 5857 | 1631 |
| Order Execution Duration (hour) | 5906 | 609 | 6173 | 1474 | 7386 | 7224 | 7124 | 7176 | 7620 | 5731 | 1169 | 7478 | 7334 | 7328 | 7252 | 7636 | 5690 | 1145 | 7500 | 7293 | 7101 | 7230 | 7696 | 5789 | 1164 | 7443 | 7258 | 7241 | 7204 | 7653 | 5753 | 1228 |
| Chartered Order Duration (hour) | 459 | 70 | 644 | 110 | 552 | 468 | 503 | 521 | 611 | 585 | 96 | 482 | 493 | 514 | 513 | 654 | 536 | 109 | 525 | 477 | 478 | 612 | 678 | 557 | 88 | 561 | 509 | 509 | 535 | 628 | 535 | 68 |
| PAT Order Duration (hour) | 5448 | 539 | 5529 | 1363 | 6834 | 6756 | 6621 | 6655 | 7008 | 5146 | 1073 | 6996 | 6842 | 6814 | 6738 | 6983 | 5154 | 1036 | 6976 | 6817 | 6624 | 6618 | 7018 | 5231 | 1076 | 6882 | 6749 | 6732 | 6669 | 7025 | 5219 | 1159 |

**Figure 3.16:** Available and Executed hours: Total Available Vehicle Time (blue); Available Employee Time (not in graph); Available Employee Time, max 12 hours (orange); Total Executed Order Duration (red + green); Chartered Order Duration (green); PAT Order Duration (red).

# Chapter 4

# Trip-assignment-model

In this chapter we describe our model. Using the problem definition described in Chapter 1 and the data that we analyzed in Chapter 3, we construct a model based on the PAT case. In Section 4.1 define the terminology we use throughout this chapter. The terminology is further explained in later sections. In Section 4.2 we describe the decisions made in our solution. We describe the input of our model involving *resources* and *trips* in Section 4.3 and Section 4.4 respectively. In Section 4.5, Section 4.6 and Section 4.7 we describe the concepts of *resource-groups*, cost, and the *duration-* and *distance-matrices* respectively. In Section 4.8 we define the *planning-period*. Finally, we describe the constraints that are used in our model in Section 4.10, followed by the conclusions in Section 4.11.

## 4.1 Terminology

In this section we list the definitions for the terminology used throughout the chapter. The structure of this section is intended to support the reader in understanding the relationships between them. We explain definitions that require further understanding in later sections.

- **Resource**: A *driver* or *vehicle* used to execute an *activity*.
- **Own resource**: A *resource* that is owned or employed by PAT.
- **Foreign resource**: A *resource* that is not owned or employed by PAT.
- **Resource-group**: A group of *resources* meeting the requirements of the *requested-resources* of the *trips* that are assigned to the *resource-group*.
- The **resource-group-type** defines the nature of the *resource-group*, based on the ownership of the *resources* within the *resource-group*. We distinguish three different *resource-group-types*:
  1. **Own resource-group**: A *resource-group* consisting solely of *Own resources*.
  2. **Rented resource-group**: A *resource-group* consisting of one *Foreign resource* and for the rest *Own resources*.
  3. **Chartered resource-group**: A *resource-group* consisting solely of *Foreign resources*.
- We distinguish three **general-resource-properties**:
  1. **Start-location**: The location where the *resource* is at the beginning of its *availability-period*. This location is not necessarily a *depot*, because *resources* might just be finished working on a previous day *trip*.

2. **Return-depot**: The location where the *resource* should finish at the end of its *availability-period*.
3. **Availability-period**: Defines the start-time and end-time of the availability of a *resource*.

- **Driver**: A specific *resource* that can drive *vehicles* loaded with goods. A *driver* is identified by its *name*, *availability-period*, *start-location* and *return-depot*.
- Besides the *general-resource-properties*, a *driver* has two **driver-properties**:
  - **Drivers-license property**: Defines which *vehicle-combinations* a *driver* is allowed to drive. We distinguish three licenses:
    1. **C-license**: The *driver* is allowed to drive *vehicle-combinations* that require *drivers-license* 'C'.
    2. **CE-license**: The *driver* is allowed to drive *vehicle-combinations* that require *drivers-license* 'C' or 'CE'
    3. **LHV-license**: The *driver* is allowed to drive *vehicle-combinations* that require *drivers-license* 'C', 'CE', or 'LHV'.
  - **DWH-status**: The Driver-Working-Hour-status (*DWH-status*) determines the history of *working*, *driving* and *resting* activities of the *driver* per *period* and per *day*:
    * **Period**: The time interval of a series of consecutive *activities* that is not interrupted by a *Regular-break* or *Daily-rest*.
    * We use the *period* to measure three **period durations**:
      1. **Period-working-duration**: The sum of durations of *activities* in the *working* activity category.
      2. **Period-driving-duration**: The sum of durations of *activities* in the *driving* activity category.
    * **Resting-duration**: The sum of durations of consecutive *activities* in the *resting* activity category.
    * **Day**: The time interval of a series of consecutive *activities* that is not interrupted by a *daily-rest* activity.
    * We use the *day* to measure two **daily durations**:
      1. **Daily-working-duration**: The sum of durations of *activities* in the *working* activity category.
      2. **Daily-driving-duration**: The sum of durations of *activities* in the *driving* activity category.
- **Vehicle**: A specific *resource* that is used to carry goods and/or tow other *vehicles*. A *vehicle* is identified by its *license-plate*, *availability-period*, *start-location* and *return-depot*.
- **Vehicle-combination**: Several *vehicles* combined together within the *resource-group* as defined in Section 3.2.2.
- Besides the *general-resource-properties*, a *vehicle* has five **vehicle-properties**:
  - **Vehicle-type property**: Determines the type of the *vehicle*. We distinguish eight *vehicle-types* as defined in Section 3.2.2.
  - **Ability-to-tow property**: Determines if a *vehicle* is able to tow another *vehicle*.
  - **Employable-in-an-LHV property**: Determines if a *vehicle* is allowed to be employed in a Long-and-Heavy-Vehicle (LHV).

- **Cooling property**: Determines the extent to which *vehicles* are able to cool loads. We distinguish three **cooling types**:
  1. **Unregulated**: No cooling at all.
  2. **Cooled**: A single cooled temperature zone.
  3. **Multi-temperature**: Multiple cooled temperature zones.
- **Livery property**: Determines if a *vehicle* is universally employable or just for a single *customer*. If the *vehicle* is employable for a single *customer*, this specific *customer* is defined.

- **Trip**: A set of successive *activities* that should be executed by a set of *resources*.
- **Pre-trip**: The traveling of a *resource-group* from the last administered location (e.g., *depot* or *end-location*) of previous *trip*) to the first location of the first *activity* of a *trip*.
- **Return-trip**: The traveling of a *resource-group* to the *return-depot*.
- **Activity**: the execution of a task within a *trip*.
- We distinguish three **activity-categories**:
  1. **Working**: A *resource-group* performing a *Loading*, *Unloading*, *Driving*, *Transporting* or *Waiting* activity.
  2. **Driving**: A *resource-group* performing a *Driving* or *Transporting* activity.
  3. **Resting**: A *resource-group* performing a *Resting* activity.
- **Activity-properties**: An *activity* has several *activity-properties*:
  1. **Activity-type**: A particular type of the *activity*. We distinguish six different **activity-types** which fall in one or more of the previously mentioned *activity-categories*:
     (a) **Loading**: Adding goods to the *vehicle-combination*.
     (b) **Unloading**: Removing goods from the *vehicle-combination*.
     (c) **Driving**: Moving the *resource-group* from point to point.
     (d) **Transporting**: An unknown and probably mixed combination of *Loading*, *Unloading*, *Driving* and/or *Waiting activities*.
     (e) **Waiting**: The *resource-group* waiting for the next *activity* to start.
     (f) **Resting**: The *drivers* in the *resource-group* taking a break or rest:
        i. **Regular-break**: A break of 45 minutes.
        ii. **Daily-rest**: A rest of 11 hours.
  2. **Time-window**: Describes the earliest and latest allowed start-times of an *activity*.
  3. **Duration**: The regular duration of the *activity*.
  4. **Distance**: The distance traveled during an *activity* (only relevant for driving and transporting activities).
  5. **Start-location**: The location where the *activity* starts.
  6. **End-location**: The location where the *activity* ends.
- **Trip-properties**: The characteristics that describe the *trip*. Besides the *activities*, we distinguish several *trip-properties*:
  - **Customer**: The *customer* that ordered the *trip*.

- **Requested-resources**: A *requested-resource* describes the properties of the *resource*, required to execute a *trip*. The *requested-resource* might have several of the following ***requested-resource* properties**:
  1. **Minimum requested *drivers-license* property**: The minimum *drivers-license* property that a *driver* needs in order to be allowed to execute a *trip*.
  2. **Requested *vehicle-type* property**: The *vehicle-type* property that a *vehicle* needs in order to be allowed to execute a *trip*.
  3. **Requested *ability-to-tow* property**: The *ability-to-tow* property that a *vehicle* needs in order to be allowed to execute a *trip*.
  4. **Requested *employability-in-an-LHV* property**: The *employability-in-an-LHV* property that a *vehicle* needs in order to be allowed to execute a *trip*.
  5. **Minimum requested *cooling* property** The minimum *cooling* property that a *vehicle* needs in order to be allowed to execute a *trip*.

- **Planning-period**: The time frame in which we schedule *trips*.
- **Scheduling-moment**: The moment when the *trips* that fall within the *planning-period* are scheduled.
- **Depot**: The location where *vehicles* are stored, *drivers* start and finish their *shifts* and *resource-groups* are formed.

## 4.2 Decisions made in solution

The employment of *resources* is generally inter-dependent. For example: a *Truck* cannot drive to another location without a *driver*, an *Eurotrailer* cannot move without a *Tractor*, and a *driver* cannot move without a *Truck* or *Tractor*. For that reason we suggest the concept of *resource-groups*: a group of *resources* that stays together for a certain amount of time. The solution of the model should construct *Own*, *Rented*, and *Chartered resource-groups* and assign *trips* to those groups. Therefore we take two types of decisions while solving our model: (*1*) the assignment of *resources* to generate *resource-groups* and (*2*) the assignment of *trips* to those *resource-groups*. Example 4.1 illustrates the kind of decisions to be taken.

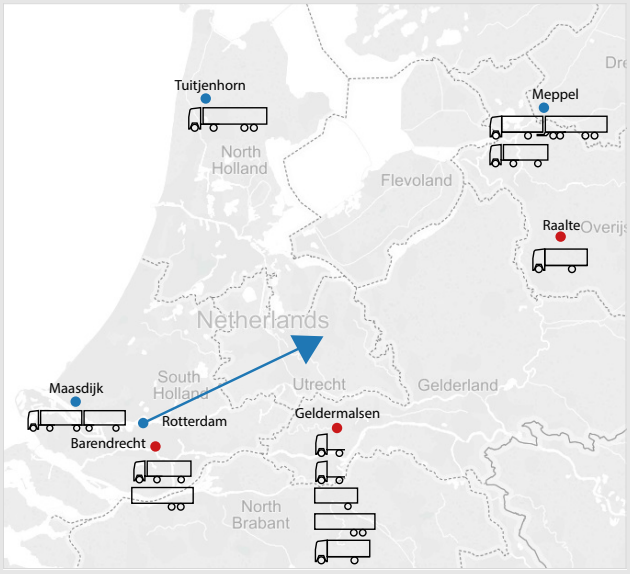
## 4.3 Resources

Each *trip* requires several *resources*. The *requested-resource* properties are specified based on the nature of the *trip*. We describe the properties of the *resources* in this section. First we describe general resource-properties (applicable to both *drivers* and *vehicles*), properties that are applicable to *drivers* only, and properties that are specifically applicable to *vehicles*.

We distinguish three *general-resource-properties*: (*1*) *availability-period*; (*2*) *start-location*; and (*3*) *return-depot*. The *availability-period* of a *resource* is similar to the time that a *resource* is available, usually related to the agenda of the *resource*. The *availability-period* of *vehicles* is mostly depending on maintenance. The *availability-period* of *drivers* is depending on agreements with employees: they are scheduled to work in different shifts, where the start-time, end-time and duration of the shift differ (examples are day, night or multi-day shifts). The *start-location* defines the location where the *resource* starts within our *planning-period*, and the *return-depot* defines the *depot* where the *resource* should end before the end of its *availability-period*.

We distinguish two *driver-properties*: (*1*) *drivers-license*; and (*2*) *DWH-status*. The required *drivers-license* property is logically dependent on the *vehicle-combination* that is employed. The *C-license* suffices for *Trucks* and *Tractors* without trailer, the *CE-license* is applicable to *Trucks* and *Tractors* with a single trailer and the *LHV-license* is mandated to

**Example 4.1:** Decisions




In this example, we ignore the *drivers* and focus only on *vehicles*. We examine a *trip* from Rotterdam to an (for this example) irrelevant location. The *trip* requires a *Tractor* and an *Eurotrailer*. There are three *depots* (red dots) which hold some *resoure-groups* and some non-grouped *resources*:

- Raalte: not usable.
- Geldermalsen: two possible *resource-groups*: (1) first *Tractor* with the *Eurotrailer* or (2) the second *Tractor* with the *Eurotrailer*.
- Barendrecht: reallocation using *Eurotrailer* and *Tractor* from *City-Combi*.

Furthermore there are three other locations (blue dots) which hold currently available *resource-groups*:

- Maasdijk: the *City-Slider-LHV* is not sufficient for the *trip*.
- Tuitjenhorn: usable.
- Meppel: individual *resources* sufficient, however assigned to *resource-groups* (*Euro-Dolly-LHV* and *City-Combi*). No *depot*, so no reallocation allowed.

There are now four possible *resource-groups* that can execute the *trip* from Rotterdam: from Tuitjenhorn, two from Geldermalsen and Barendrecht. The decision is now related to the choice which of those four *resource-groups* should execute the *trip*.

drive *Trucks* and *Tractors* with two trailers. *Drivers-licenses* are hierarchical: the *C-license* is lowest in rank. *Drivers* with the *CE-license* are also allowed to drive *vehicles* where the *C-license* is required and *drivers* with the *LHV-license* are allowed to drive all *vehicles*. The *DWH-status* is the status of a *driver* at the beginning of the *availability-period*. We use this status to determine if the *driver* is legally allowed to do *activities* from the *driving* or *working* category. Details of the status were already outlined in Section 4.1.

We distinguish five *vehicle-properties*: (*1*) *vehicle-type*; (*2*) *employability-in-an- LHV*; (*3*) *ability-to-tow*; (*4*) *Cooling*; and (*5*) *Livery*. The *vehicle-type* defines the type of *vehicle*. The known *vehicle-types* were clarified in Section 3.2.2. *Vehicles* with different *vehicle-types* are not interchangeable. The *employability-in-an-LHV* property establishes whether a *vehicle* has the features (e.g., a sign on the back, surplus engine power, etc.) to be employed in a LHV, the property can be *'True'* or *'False'*. The property is hierarchical, meaning that *vehicles* that are employable in LHVs (*True*) can also be scheduled for *trips* that do not require the property. A *vehicle* that has the *ability-to-tow*, is able to tow trailers. The property is binary, it can be *'True'* or *'False'*. The property is not hierarchical: *vehicles* that can tow, cannot always be employed as non-towing *vehicles*. The *ability-to-tow* property is a simplification of reality: for our model we assume that other properties (e.g., the differences between a fifth wheel and a draw-bar, the height of tow-bar, etc.) are not relevant. The *cooling* property is also a simplification of reality. For our model, we ignore the layout and the capacity of the compartments and only include the number of temperature zones, we assume that any temperature zone can be set up to cool to any temperature. We only differentiate between the number of temperature zones (*Unregulated*: none; *Cooled*: one; *Multi-temperature*: more than one). The property is hierarchical: *trips* that require *Unregulated vehicles* can also be carried out by *Cooled* or *Multi-temperature vehicles*, and *trips* that require *Cooled vehicles* can also be executed by *Multi-temperature vehicles*. The other way round is logically impossible. Some *vehicles* carry customer specific *livery*, meaning that these *vehicles* can only be applied at these customers, while unbranded *vehicles* are universally employable (including customers that have branded *vehicles*). The *livery* property can have an *'Unbranded'* value for universally employable *vehicles* and a specific value for *vehicles* that are only employable for a certain customer (e.g., *'AH'*, *'Sligro'* or *'Bakkersland'*).

## 4.4 Trips

| Activity-type | Activity-category: Driving | Working | Resting |
|---|---|---|---|
| Transporting | • | • | • |
| Loading | | • | |
| Unloading | | • | |
| Driving | • | • | |
| Resting | | | • |

**Table 4.1:** The relationship between *activity-category* (horizontal) and *activity-type* (vertical).

A *trip* is a set of successive *activities* that should be executed by a set of *resources* that fulfill the properties of the *requested-resources*. An inherent characteristic of a *trip* is that a single customer is served. We now explain the concepts and properties of *activities*, *requested resources*, and the specifics of *pre-trips*, *return-trips*, *breaks*, and *rests*.

An *activity* is a certain type of action within a *trip*. The categories, types, and properties of *activities* were outlined in Section 4.1. The relationship between the *activity-category* and *activity-type* is given in Table 4.1. We use this relationship to determine the *DWH-status* of a certain *driver*.

The *Transporting activity* is a special case, which is applicable to *trips* that belong to the *trip-types* that we earlier classified as '$A \rightarrow ? \rightarrow B$' and '$A \rightarrow ? \rightarrow A$' (see Section 3.4). The *Transporting activity* is an *activity* that might contain *Loading*, *Unloading*, *Waiting*, or *Driving*: we do not exactly know in advance what the *resources* are doing during the *trip*. The *distance* and *duration* of the *Transporting activity* should be given, because it cannot be determined by calculating the *distance* and *duration* between the start- and end-locations (which we do for *Driving activities*).

One of the properties of an *activity* is its *time-window*. A *time-window* describes the

| | Own resources | Foreign resources | cost components | creation/destroy location | initial cost parameter |
|---|---|---|---|---|---|
| Own | all | none | pre-trips, trips, return-trip | depot | 1.0 |
| Rented | all - one(1) | one(1) | pre-trips, trips, return-trip | depot | 1.5 |
| Chartered | none | all | trip | everywhere | 2.0 |

**Table 4.2:** Resource-groups

earliest and latest start-times of an activity. For example: an *activity* (duration 30 minutes) has a *time-window* with a span of two hours (from 14.00 to 16.00). The *activity* can therefore be scheduled from 14.00 to 14.30 hour at the earliest, from 16.00 to 16.30 at the latest, or anywhere in between.

A *trip* requires a set of *resources*, according to the specifications given by the *order*. An example of a set of *requested-resources* for a cooled *Euro-Combi* are: (*1*) a *driver* that owns appropriate *CE-license*; (*2*) a *Tractor* that can tow; and (*3*) an *Eurotrailer* that has at least cooling facilities.

To enable possible *resources* to start a *trip* at a certain location, we need to ensure resources to travel between *trips*, using so-called *pre-trips*. We choose to generate these *pre-trips* on the fly and then assigning them to a *resource-group*. *Regular-breaks* and *Daily-rests* are *activities* that are generated during the scheduling and can be scheduled between other *activities*. We do not split *activities* to be interrupted by a *Regular-break* or *Daily-rest* within our solution. If necessary, we split long *activities* beforehand. A *return-trip* is the final *trip* of the *resource-group*, taking the *resource-group* to the *return-depot*.

## 4.5 Resource-groups

A *resource-group* is a group of *resources* operational for a certain amount of time. We distinguish three types of *resource-groups*: (*1*) *Own resource-groups*; (*2*) *Rented resource–groups*; and (*3*) *Chartered resource-groups*. An *Own resource-group* consists of only *Own resources*, thus all *resources* within the group are actually linked to a physical *resource*. Within a *Rented resource-group*, one of the *resources* is *Foreign*. All other *resources* must be owned. For *Chartered resource-group*, the entire group consists of *Foreign resources*.

We can generate a *Rented resource-group* if there are insufficient *Own resources*. We then choose to rent *Foreign resources*. This is usually more expensive than using *Own resources*, but might be cheaper when compared to chartering (i.a., using a *Chartered resource-group*) or driving a long distance *pre-trip*. Every *requested-resource* can be rented and we assume unlimited renting capacity regardless of the *resource-properties*. To prevent unnecessary complications, we simply assume that only one *resource* can be rented per *resource-group* and thus, that a *Rented resource-group* consists of one *Rented resource*, whereas all other *resources* are owned. We assume that *resources* can only be rented at *depots*, because we want to avoid that we need to pick up *Rented resources*.

A *Chartered resource-group* is in fact hiring a charter or outsourcing the entire *trip*. Logically, hiring a charter should be more expensive than using *Own resources*. A *Chartered resource-group* includes *Foreign resources* that fulfill all *requested-resources*. We make the following assumptions regarding charters: (*1*) charters are responsible for their own planning; (*2*) break and rest scheduling is done by the charters; (*3*) they do not require scheduled *pre-trips* nor *return-trips*; and (*4*) there is unlimited charter capacity.

## 4.6 Cost

We give three examples of considerations regarding the costs at PAT: (*1*) different costs for *drivers* based on their seniority or skills; (*2*) different costs for *vehicles* of different type, age, or other properties; and (*3*) different prices of charters and *Rented resources*, based on contracts and availability. To maintain simplicity of our model, we use a basic cost structure. Decisions about the type and number of *resources* we need to schedule (e.g., *Euro-Combi* or

| Resource Group | Kilometer | Hour | Kilometer Cost | Hour Cost | Multiplication factor | Cost |
|---|---|---|---|---|---|---|
| Own | 400 | 8 | 0.8 | 37 | 1.0 | € 616 |
| Rented | 400 | 8 | 0.8 | 37 | 1.5 | € 924 |
| Chartered | 400 | 8 | 0.8 | 37 | 2.0 | € 1232 |

**Table 4.3:** Cost example

*City-Combi*) are beyond our control (i.e., provided by the customers), therefore we assume that each *resource-group* is equally expensive, irrespectively of the properties and number of *resources* that a *trip* requires. Furthermore, contracts with *drivers*, charter companies, car rental companies, and employment agencies are not always very transparent. Most of the times these contracts are complex the differences in structure differs per contractor. We therefore prefer to implement a simple cost structure for the calculation of cost over the use of a sophisticated cost structure. The basic parameters of our cost calculation are based on prices per unit of *distance* (kilometer) and *duration* (hour).

Logically, renting or chartering *resources* is more expensive than using *Own resources*. Therefore, using *Foreign resources* should be counterbalanced by punishment (i.e., giving penalties). We model this by the multiplication of the *trip*-prices with a predetermined cost factor in cases where *resource-groups* are *Rented* or *Chartered*. We assume that there is no difference in price (for the *resource-group* in total) between renting a *driver* or a *vehicle*, and that there are no differences in price between *vehicles* of different types, *resources* with different properties, different rental companies, etcetera. We also do not distinguish prices between different charter companies nor different combinations of *resource-types*. *Own* and *Rented resource-groups* can only be created and destroyed at *depots*, therefore, the costs of those *resource-groups* consist of *trips*, *pre-trips*, and *return-trips*. *Chartered resource-groups* can be created and destroyed at any location, therefore, for *Chartered resource-groups* the *pre-trips* and *return-trips* are omitted. Individual *trips* with *Own*, *Rented*, and *Chartered resource-groups* are equally expensive, however we differentiate between *Own*, *Rented*, and *Chartered resource-groups* by using a multiplication factor for those *resource-groups*. Based on the experience at PAT, we use a multiplication factor of 1.5 for *Rented resource-groups* (i.e., using *Rented resources* is 50% more expensive than using *Own resources*) and 2.0 for *Chartered resource-groups* (i.e., using *Chartered resources* is 100% more expensive than using *Own resources*).

With these limitations set in advance, costs are now only depending on two factors: (*1*) the total set of *activities* (with *duration* and *distance*) that is planned for a *resource–group* and (*2*) the nature of the *resource-group* (*Own*, *Rented*, or *Chartered*). Initially, we use the prices of 0.8 per kilometer and 37 per hour. We present these parameters in Table 4.3 to give an example of a cost calculation for different *resource-groups* that can execute a *trip* that takes 8 hours and where a total of 400 kilometers should be driven. Although we are aware that this is a simplified cost structure, it facilitates the assignment of value to both the current manually and future automatically created schedules. We are therefore able to value and compare schedules based on cost.

## 4.7 Duration- and distance-matrices

Both duration and distance are variables needed for scheduling and cost calculation. We cannot use distance alone, because we deal with *activities* that lack a distance (e.g., *Waiting*). We cannot use duration on its own either, because duration and distance are not necessarily linearly related. Therefore, both a *duration-* and *distance-matrix* for all *activities* is pertinent. The matrices should be composed of driving durations and driving distances between all known locations. We assume that these durations and distances are time-independent (e.g., no rush hours). The *duration-* and *distances-matrices* should also be used to schedule *pre-trips* and *return-trips*.

## 4.8 Planning-period

Because PAT runs a 24/7 business, there are always *resources* working and *trips* being executed. Because we do not focus on continuous (online) planning, we incorporate a *planning-period*. The *planning-period* is the time-frame that defines which *trips* (and *activities* that make up the *trip*) are scheduled during a scheduling run and which *trips* are not. Scheduling runs are executed at the *Scheduling-moment* [s] before the start of the *Planning-period* [t]. In order to be able to keep our scheduling within limits of practical size, we choose a period of 24 hours. We do not add or adapt *trips* or *resources* during the scheduling run: scheduling cannot be interrupted and schedules cannot be adapted. If there are changes (e.g., new *trips*, changed *availability-periods*), these changes can be processed manually, or a new scheduling run should be executed. *Trips* that already started cannot be rescheduled automatically.

*Trips* are scheduled collectively (i.e., within the same *planning-period*) based on the earliest possible start-times of those *trips*. This means that *trips* falling within the *planning-period* [t] are scheduled within a single run. This run is executed on the *scheduling-moment* [s], before the start of [t]. Because the *scheduling-moments* of *trips* are based on start-times of *trips*, *trips* falling in *planning-period* [t] might overlap the next *planning-period* [t+1]. Figure 4.1a illustrates this principle.

Because *trips* can now cover both the current *planning-period* [t], as well as part of the next *planning-period* [t+1], we require *resources* that cover multiple *planning-periods*. We include all *resources* which *availability-period* covers (part of) [t]. Figure 4.1b illustrates this principle.

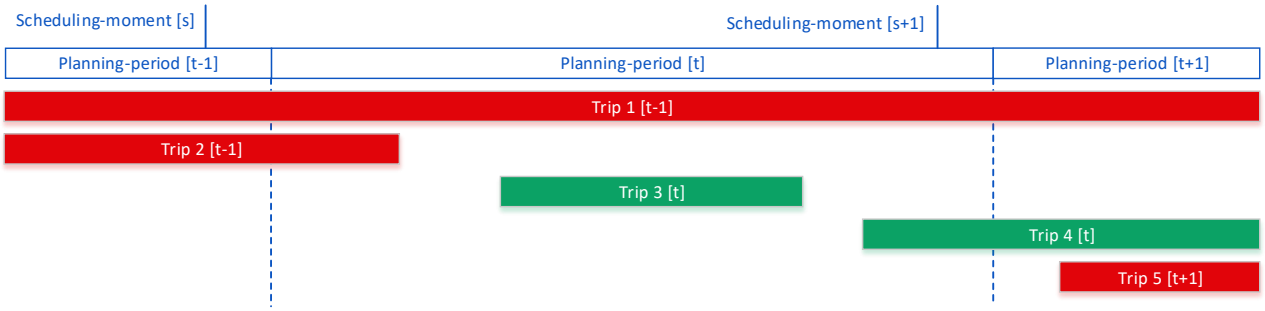
As we can see, *planning-periods* are overlapping. We now establish the method we use to determine how to deal with overlapping *planning-periods*. As stated earlier, *trips* fall in a *planning-period* based on the earliest possible start-time. A *trip* falling in *planning-period* [t-1] might be scheduled (at *scheduling-moment* [s-1]) using a *resource* having an *availability-period* covering both *planning-period* [t-1] and *planning-period* [t] (partially). At *scheduling-moment* [s] (when scheduling *planning-period* [t]) we need to realize that the *availability-period* of that *resource* depends on *planning-period* [t-1]. The *trip* already started at *scheduling-moment* [s], so we cannot reschedule the *trip*. The solution for this problem is that the *availability-period* of the *resource* for *planning-period* [t] requires adjustments, based on *planning-period* [t-1], as is illustrated by Figure 4.1c.

## 4.9 Target, objective and output

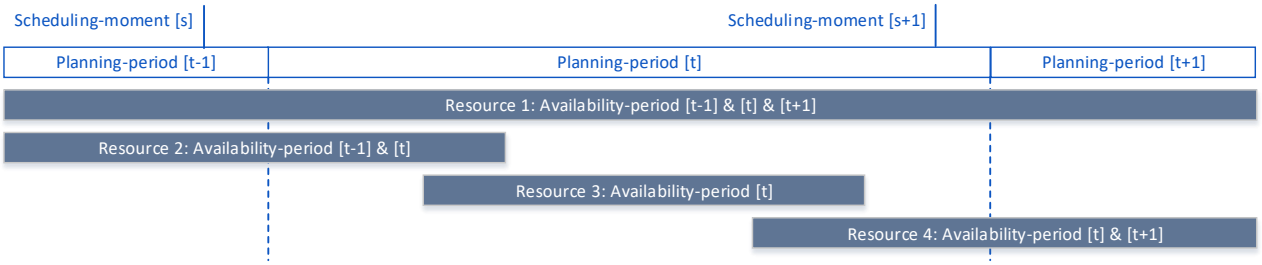
The target of our efforts is to schedule all *trips* that start within the *planning-period* (see Section 4.8) for the lowest cost possible. Within this schedule, each *trip* should be assigned to a *resource-group* and the schedule should be feasible. *Resource-groups* contain a mix of *Own* and/or *Foreign resources*, depending on their nature (*Own*, *Rented*, or *Chartered*), each with its own multiplication factor for the costs. As stated before, we make decisions regarding the assignment of all given *trips* to *resource-groups*. We also decide on the assignment of *resources* to those *resource-groups*. With the assignment of *trips* to *resource-groups*, we create a schedule in which the total cost of all *trips* (including *pre-trips* and *return-trips*) are as low as possible. These total costs are influenced by the *duration* and *distance* of *pre-trips* and *return-trips*, as well as the composition of *resource-groups* (*Own*, *Rented*, or *Chartered*).

## 4.10 Constraints

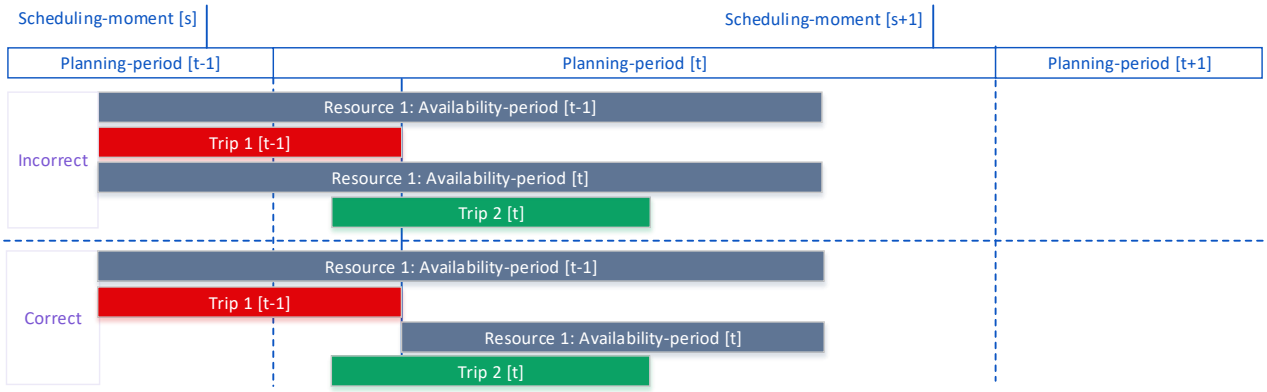
In this section we discuss the constraints in our model. We describe the *driver-working-hour*-constraints in Section 4.10.1, the *requested-resource*-constraints in Section 4.10.2, followed by the *resource-group*-constraints in Section 4.10.3 and *activity*-constraints in Section 4.10.4.




**(a)** Planning-period and trips: based on earliest possible start-times, Trip 1 and Trip 2 fall in Planning-period [t-1], Trip 3 and Trip 4 fall in Planning-period [t], and Trip 5 falls in Planning-period [t+1]. Trip 1 and Trip 2 are therefore scheduled at Scheduling-moment [s-1] (not in image), Trip 3 and Trip 4 are scheduled at Scheduling-moment [s], and Trip 5 is scheduled at Scheduling-moment [s+1]




**(b)** Planning-period and availability-period: Resource 1 is available during (parts of) Planning-period [t-1], [t] and [t+1], Resource 2 is available during the previous ([t-1]) and the current ([t]) Planning-period, Resource 3 is available during the current Planning-period and Resource 4 is available during the current and next ([t+1]) Planning-period.




**(c)** Planning-period, availability-period and trips: Resource 1 is available during Planning-periods [t-1] and [t]. During [t-1], Resource 1 is executing Trip 1. For Planning-period [t] it is now incorrect to assume that the *Availability-period* of Resource 1 is exactly the same for Planning-periods [t-1] and [t]: Trip 2 cannot be scheduled on Resource 1 because of Trip 1. The *Availability-period* of Resource 1 in [t] should now be adjusted according to [t-1] to get a correct situation (in which Trip 2 cannot be scheduled on Resource 1).

**Figure 4.1:** Planning-period

### 4.10.1 Driver-working-hours

*Drivers* are obliged to comply with *driver-working-hour*-constraints. The following rules apply within our model:

- Breaks or rests can only be scheduled between other *activities*: *activities* cannot be split by breaks or rests.
- *Drivers* can *drive* (*activity-category*) maximally 4.5 hours per *period* and 9 hours per *day*.
- *Drivers* can *work* (*activity-category*)maximally 6 hours per *period* and 12 hours per *day*.
- After a *period* of *driving* or *working*, the *driver* is mandated to take a *Regular-break* (45 minutes).
- After a *day* of *driving* or *working*, a *Daily-rest* (11 hours) is legally demanded.
- Weekly rests must be included in the *availability-period* and we assume that they are not relevant for daily scheduling.
- Extended driving, short rests, etcetera (see Section 3.5), are ignored in our model, because we assume the legislators' intention is to use those in exceptional situations which might occur in the execution of our schedules.

### 4.10.2 Requested-resources

We determine the physical *resources* that can be assigned to a specific *resource-group*, based on the *trips* currently assigned to that specific *resource-group*. Each *trip* is linked to *requested-resources*, having properties that should match the properties of physical *resources*. We distinguish *vehicles* and *drivers*, both with their own sets of constraints. In Table 4.4 we give an overview of the relationships between *resource-properties* and properties of *requested-resources*. Furthermore, we give a brief explanation below:

- The *vehicle-type* property of a *requested-vehicle* must match the *vehicle-type* property of the *vehicle* that is scheduled on that *trip* (e.g.: if a *trip* requires a *vehicle* with the *vehicle-type 'Truck'*, then only a *vehicle* of the *vehicle-type 'Truck'* can be scheduled).
- The *cooling* property of a *requested-vehicle* must at least match the *cooling* facility of the *vehicle* that is scheduled on that *trip* (e.g.: if a *trip* requires a *vehicle* with the *cooling* property *'Cooled'*, then a *vehicle* with the *cooling* property *'Cooled'* or *'Multi-Temperature'* is required). If the *cooling* property of the *requested-vehicle* is undefined, then the *cooling* property of the *vehicle* does not matter.
- The *ability-to-tow* property of a *requested-vehicle* must match the *ability-to-tow* property of the *vehicle* that is scheduled on that *trip* (e.g.: if a *trip* requires a *vehicle* with the *ability-to-tow* property *'False'*, then only a *vehicle* with the *ability-to-tow* property *'False'* can be scheduled). If the *ability-to-tow* property of the *requested-vehicle* is undefined, then the *ability-to-tow* property of the *vehicle* is irrelevant.
- The *employability-in-an-LHV* property of a *requested-vehicle* must match the *employability-in-an-LHV* property of the *vehicle* that is scheduled on that *trip* (e.g.: if a *trip* requires a *vehicle* with the *employability-in-an-LHV* property *'True'*, then only a *vehicle* with the *employability-in-an-LHV* property *'True'* can be scheduled). If the *employability-in-an-LHV* property of the *requested-vehicle* is undefined, then the *employability-in-an-LHV* property of the *vehicle* does not matter.

- The *customer* property of a *trip* must match the *livery* property of the *vehicles* that are scheduled on that *trip*. Furthermore, *Unbranded vehicles* can always be scheduled. (e.g.: if the *customer* of a *trip* is *'Albert Heijn'*, then only *vehicles* which *livery* property is *'Unbranded'* or *'Albert Heijn'* can be scheduled)
- The *drivers-license* property of a *requested-driver* must at least match the *drivers-license* property of the *driver* that is scheduled on that *trip* (e.g. if a *trip* requires a *driver* with the *drivers-license 'CE'*, then only *drivers* with the *drivers-license 'CE'* or *'LHV'* can be scheduled).

### 4.10.3 Resource-groups

- All *resources* within the *resource-group* must be available.
- In a *Rented resource-group*, only one *resource* can be rented.
- A *resource-group* should return to the *return-depot* using a *return-trip*. This *return-trip* should end before the end of the first ending *availability-period* of the *resources* in the group.
- *Own* and *Rented resource-groups* can only be created and eliminated at the *return-depot* of the *resources*. *Resources* within a group have thus always the same *return-depot* (*Chartered resource-groups* can be created and destroyed at any location).
- *Recourse-groups* cannot be split up at locations other than a *depot*.

### 4.10.4 Activities

- The scheduled start-time of an *activity* must fall between the lower and upper limits of its *time-window*.

## 4.11 Conclusions

In this chapter we developed a simplified model to approach our problem. First, we formulated the definitions used throughout our model. Then we defined the decisions that our model supports, followed by the description of the *resource* and *trip* inputs of our model. Furthermore, we defined that our model should generate *pre-trips*, breaks, rests and *return-trips*. We explained the terms *resource-groups*, *duration-* and *distance-matrices*, and *planning-period*. Then we defined the target and objectives of our model, followed by the model constraints. In the next chapter we detail the method for solving this model.

| Requested-vehicle: vehicle-type | Vehicle: vehicle-type | | |
|---|---|---|---|
| | Truck | Eurotrailer | ... |
| Truck | ✓ | | |
| Eurotrailer | | ✓ | |
| ... | | | ✓ |

| Requested-vehicle: cooling | Vehicle: cooling | | |
|---|---|---|---|
| | Unregulated | Cooled | Multi-temperature |
| Unregulated | ✓ | ✓ | ✓ |
| Cooled | | ✓ | ✓ |
| Multi-temperature | | | ✓ |
| Undefined | ✓ | ✓ | ✓ |

| Requested-vehicle: ability-to-tow | Vehicle: ability-to-tow | |
|---|---|---|
| | Yes | No |
| Yes | ✓ | |
| No | | ✓ |
| Undefined | ✓ | ✓ |

| Requested-vehicle: employability-in-an-LHV | Vehicle: employability-in-an-LHV | |
|---|---|---|
| | Yes | No |
| Yes | ✓ | |
| No | | ✓ |
| Undefined | ✓ | ✓ |

| Trip: customer | Vehicle: livery | | |
|---|---|---|---|
| | Unbranded | Albert Heijn | ... |
| Undefined | ✓ | | |
| Albert Heijn | ✓ | ✓ | |
| ... | ✓ | | ✓ |

| Requested-driver: drivers-license | Driver: drivers-license | | |
|---|---|---|---|
| | C | CE | LHV |
| C | ✓ | ✓ | ✓ |
| CE | | ✓ | ✓ |
| LHV | | | ✓ |

**Table 4.4:** Resource constraints: In this table we specify the relationship between the properties of the requested-resources (vehicle or driver) of a trip and properties of resources (vehicle or driver) which could execute the trip. Based on the property of the requested-resource (left), the table shows (using checkmarks) which property a resource should have to allowed to fulfill the requirements of the trip.

# Chapter 5

# Solution

In this chapter, we elaborate on a method for solving the *trip-assignment-model*. First we describe some considerations regarding the solution in Section 5.1. Then, in Section 5.2, we discuss our approach to solve our model. In Section 5.3 we describe the tool we develop for solving our model, followed by some conclusions in Section 5.4.

## 5.1 Considerations

There are several challenges in solving the model defined in Chapter 4. We discuss these challenges in Section 5.1.1. Furthermore, we discuss the considerations between parallel and sequential scheduling in Section 5.1.2. We finish drawing conclusions (Section 5.1.3) to be able to describe our approach.

### 5.1.1 Challenges

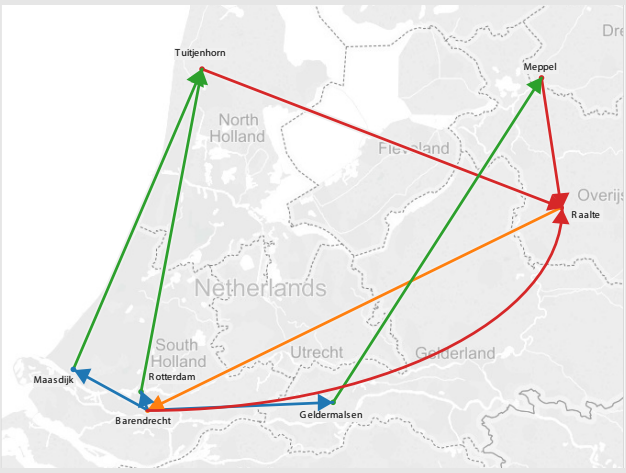
The challenges discussed in this section include: (*1*) problem size; (*2*) assignment of multiple dependent *resources*; (*3*) *time-windows*; and (*4*) *working-hour-regulations*.

As described in Chapter 4, we intend to use our model for one day of input data. In Chapter 3, we demonstrated that this results in an average of 1065 *trips* per day. In Chapter 2 we reviewed several methods that are available to solve assignment problems. We arrived at a threefold conclusion: (*1*) the solution space is very large; (*2*) exact methods are not likely to solve the problem in a reasonable amount of time; and (*3*) heuristics that make small adjustments and explore few neighbors are more useful than heuristics that explore a large number of solutions, due to the large solution space. Based on this threefold conclusion we choose Simulated Annealing as the *metaheuristic* for our solution and keep our heuristics simple and clear. Therefore, we develop a *construction*- and an *improvement-heuristic*.

Another challenge is the assignment of multiple *resources*. As described in Chapter 4, *resources* generally depend on each other (i.e., *resources* cannot execute *trips* independently of each other). We therefore introduced *resource-groups*. With the introduction of those *resource-groups*, we induce two kinds of assignments: first, we need to assign *resources* to *resource-groups*, and second we need to assign *trips* to those *resource-groups*. This double assignment also results in checking constraints on multiple levels: we need to check if there is an existing *resource-group* able to execute the *trip* (based on the *resources* in that group and other *trips* assigned to that group), and we need to check if we can create new *resource-groups* for executing the *trip* (using unassigned *resources* or *resources* that become available by taking another *resource-group* apart).

Among others, *time-windows* are factors that we need to take care of. In our model, we incorporate a *time-window* for each *activity*. As stated in Chapter 2, Jula et al. (2005) developed a model for aggregating *time-windows* into a single *trip*-based *time-window*. Their approach is used in our solution.

**Example 5.1:** Sequential scheduling




We display four types of lines: (*1*) the orange arrow represents a *trip* from Raalte to Barendrecht that we assume to be currently finished; (*2*) green arrows show *trips* that still have to be run; (*3*) blue arrows display movements that finished *resources* stored in Barendrecht should make before being able to execute one of the green *trips*; and (*4*) red arrows represent movements that enable *resources* to return to the *depot* in Raalte after finishing a *trip*.

Because the orange *trip* just finished in this hypothetical example, the *resource-group* is now available in Barendrecht. We should decide which *trip* has to be executed next by the *resource-group*.

**Example 5.2:** Parallel scheduling


The figure contains two types of lines: (*1*) the green arrow shows a *trip* that still has to be executed; and (*2*) blue arrows display movements that available *resource-groups* should make before they can execute the green *trip* from Rotterdam to Tuitjenhorn.

There are *resource-groups* available in Maasdijk, Barendrecht, Geldermalsen, Raalte, and Meppel. Now, the planner has to single out the *resource-group* intended for the *trip* from Rotterdam to Tuitjenhorn.

Furthermore, we incorporate a subset of *working-hour-regulations* in our model (see Section 4.10.1). To keep track of the *DWH-status*, we adopted the driver working hour labels introduced by Goel and Gruhn (2006). Furthermore, we developed an algorithm for scheduling breaks and rests based on the ideas of Kok et al. (2010).

### 5.1.2 Parallel versus sequential scheduling

An important consideration regarding the *construction-heuristic* is the use of a parallel or sequential scheduling approach. The sequential scheduling method creates a *resource-group* and assigns *trips* to that *resource-group* until no more *trips* fit in that *resource-group*. After that, a new *resource-group* is created and the process repeats. We use Example 5.1 to illustrate this principle. The parallel scheduling method can schedule multiple *resource-groups* at the same time by generating the required *resource-groups* in advance. Example 5.2 illustrates this method.

A complicating factor for the comparison between parallel and sequential scheduling is that there is a second decision to be reached when generating *resource-groups*: which

*resources* are assigned to the *resource-groups*, taking into consideration that we can use *Rented resources* and *Chartered resource-groups*, rather than *Own resources*, making the number of options tremendous. Therefore, it is impracticable to generate all *resource-groups* in advance (as is common in the parallel scheduling method). Due to the use of *resource-groups* we cannot use the sequential scheduling method: it is impossible to decide which *resources* to combine within a *resource-group*, without looking at the *trips*. We therefore use a parallel scheduling method, however, we generate *resource-groups* on the fly, based on the characteristics of the *trip* we are scheduling, avoiding that we become short-sighted by looking only at one *resource-group* at the time: there might be multiple possible *resource-groups* after all. In our algorithm, we look at the characteristics of a *trip*, then we check if there are any existing *resource-groups* which can feasibly execute this *trip* and determine the cost of this insertion (including changes in *pre-* and *return-trips*). Besides that, we generate a number of new *resource-groups* which can feasibly execute the *trip* and calculate the costs of using those *resource-groups* (again including *pre-* and *return-trips*). Finally, the *trip* is assigned to that particular *resource-group* which executes the *trip* at the lowest cost.

### 5.1.3 Conclusion

In this section we described some considerations regarding the development of a solution for solving the model described earlier. We decide to solve the model heuristically, using both a *construction-* and an *improvement-heuristic*. Furthermore, we adopt some techniques proposed in the literature, regarding *time-windows* and *working-hour-regulations*. Finally, we discussed the parallel and sequential scheduling methods and selected a parallel scheduling method for our *construction-heuristic*. Furthermore, we selected Simulated Annealing as the *metaheuristic* for our *improvement-heuristic*. We continue describing the approach of solving the model pursuing the methods discussed before.

## 5.2 Approach

Based on the considerations from Section 5.1, we design heuristics to solve the model from Chapter 4. First, we develop a data model (see Section 5.2.1) for the objects needed to solve the model. Then we discuss the functions that evaluate and adapt our solutions in Section 5.2.2. Thereafter, we check the constraints in Section 5.2.3, followed by our *construction-heuristic* (Section 5.2.4) and *improvement-heuristic* (Section 5.2.5).

### 5.2.1 Data Model

Our data model encompasses seven main objects: *trips*, *activities*, *requested-drivers* and *requested-vehicles* on the one hand, and *drivers* and *vehicles* on the other hand. We use *resource-groups* to link them all together. The objects and their relationships are represented in Figure 5.1. A full version including attributes is detailed in Appendix G. All attributes have been discussed earlier in Chapter 4. We continue with explaining the outstanding properties of our data model.

We distinguish between *trips* and *activities* that are input of our model (i.e., provided by PAT), and *trips* and *activities* that are generated by our heuristics (*pre-trips* and *return-trips*). Within *trips* that are input, *Driving* and *Unloading* are examples of *activities* that are pre-determined by the demands and restrictions put by the client (i.e., PAT), while *Waiting* and *Resting* are *activities* that are generated. Obviously, *activities* within a generated *trip* are also generated *activities*.

There is a two-way relationship between both *drivers* and *vehicles* with *resource-groups*. This relationship is used to determine which *drivers* and *vehicles* are allocated to which *resource-group*, while, the other way around, it is used to determine if *drivers* and *vehicles* are available (thus not busy on *trips* in other *resource-groups*).

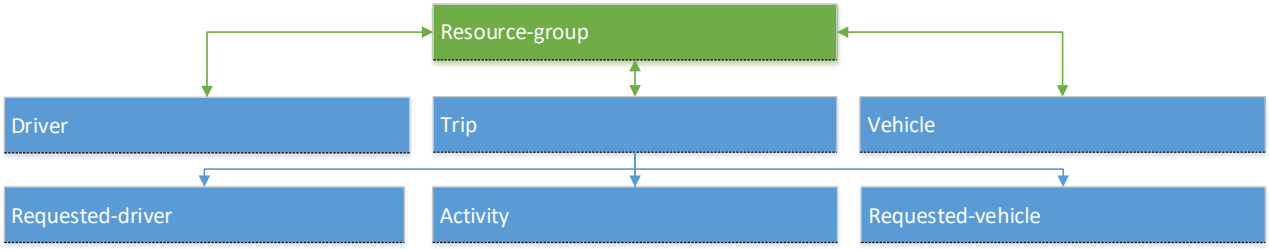



**Figure 5.1:** Data model (simplified)

### 5.2.2 Resource-group- and trip-evaluation

For each change in the assignment of *trips* or *resources* to *resource-groups*, there may be three possible consequences: (*1*) we might need different *pre-trips* and *return-trips*; (*2*) arrival times at successive *stops* might change (resulting in *Waiting* or *time-window* violation); and (*3*) the number and duration of breaks and rests that each *driver* has to take might change. Therefore we evaluate the *resource-group* and each *trip* assigned to that *resource-group* after every change to that *resource-group*. In the evaluation of a *resource-group*, we first remove all generated *trips*. Then, for each *trip*, we determine if that *trip* requires a *pre-trip* and, if so, insert one. Then we insert the *trip* under investigation. After the final *trip* we determine if a *return-trip* is required and, if so, insert one. The resource-group-evaluation-procedure is depicted as a flowchart in Figure 5.2a.

Meanwhile, we also evaluate all *pre-trips*, *trips*, and *return-trips*. We remove all *activities* from the *trip* that were not provided by the customer. Then, we determine if the next *activity* can be executed without taking a break. If this is not the case, a break or rest is scheduled before scheduling the *activity* under investigation. If the next *activity* has a *time-window* such that we have to wait, we first check if it is possible to schedule a break or rest without violating the *time-window*. If this is the case, we schedule a *Regular-break* or *Daily-rest*. Then we check if waiting is still required. If this is the case, we insert a *Waiting activity* before we schedule the *activity* under investigation. Figure 5.2b depicts a flowchart of trip-evaluation-procedure.

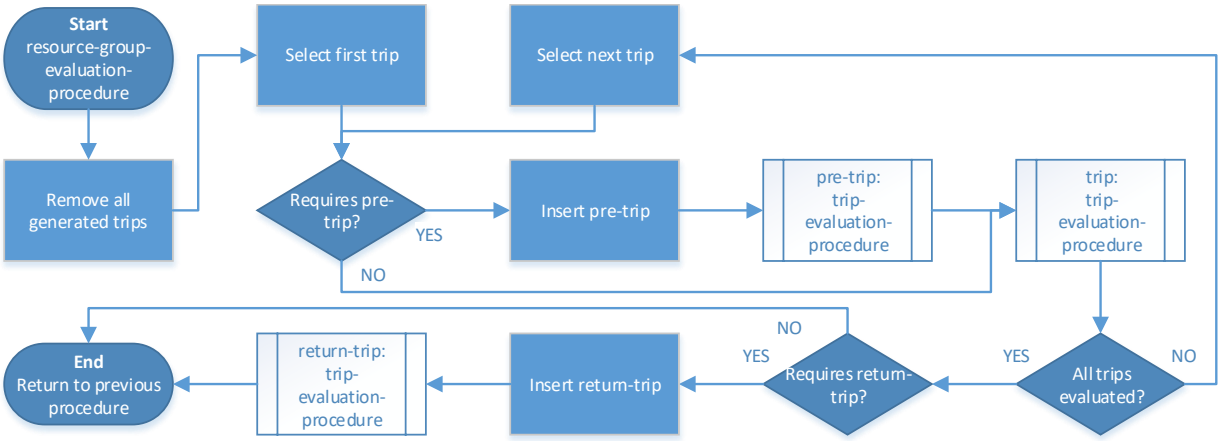
### 5.2.3 Checking constraints

In Chapter 4 we explained the constraints of our model. We distinguish *working-hour-regulation* constraints, *requested-resource* constraints, *resource-group* constraints, and *activity* constraints. We routinely check the constraints at several points in the algorithm: (*1*) while searching *resources* or *resource-groups*, the impacts of *resource-group* constraints and *requested-resource* constraints are closely scrutinized; and (*2*) the *resource-group-evaluation-procedure* and the *trip-evaluation-procedure* (see Section 5.2.2) check the *working-hour-regulation* constraints and the *activity* constraints.

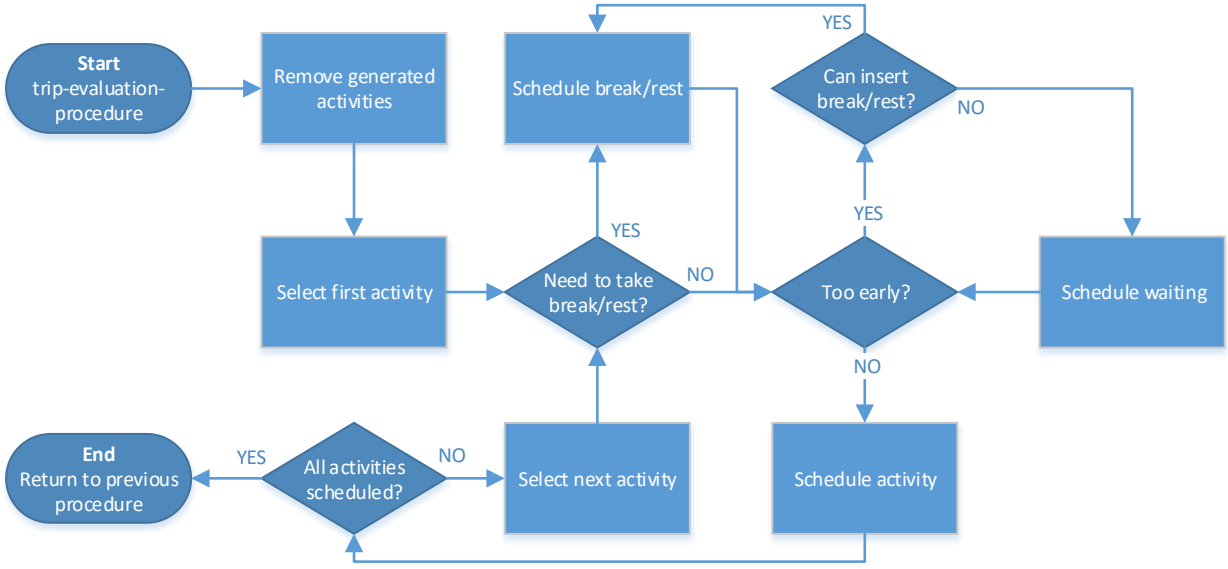
### 5.2.4 Construction-heuristic

The first step of our heuristic is a construction phase. We earlier introduced the considerations regarding our scheduling approach in Section 5.1, discussed the data model in Section 5.2.1, elaborated on the evaluation procedures in Section 5.2.2 and the checked constraints in Section 5.2.3. In this section we use that information to focus further on our *construction-heuristic*.

The *construction-heuristic* schedules *trips* one by one: we schedule the earliest possible starting *trip* first and then continue scheduling *trips* with later possible start-times.




**(a)** Resource-group-evaluation-procedure




**(b)** Trip-evaluation-procedure

**Figure 5.2:** Evaluation procedures

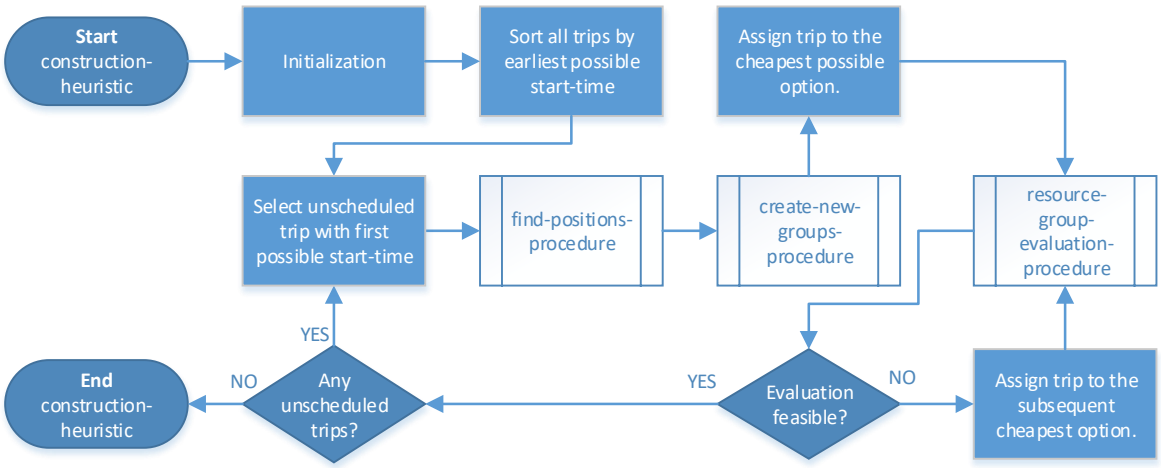



**Figure 5.3:** Construction-heuristic

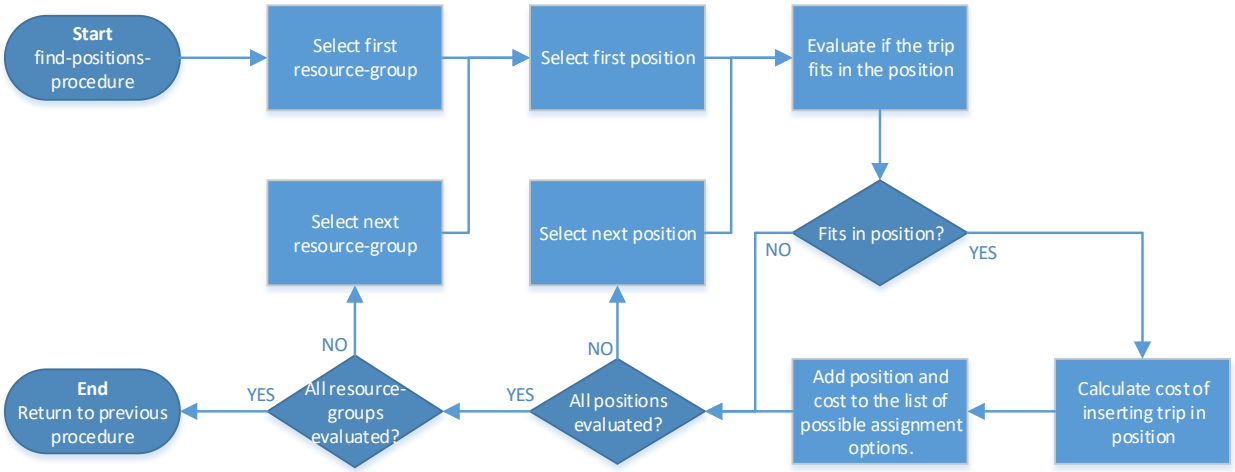


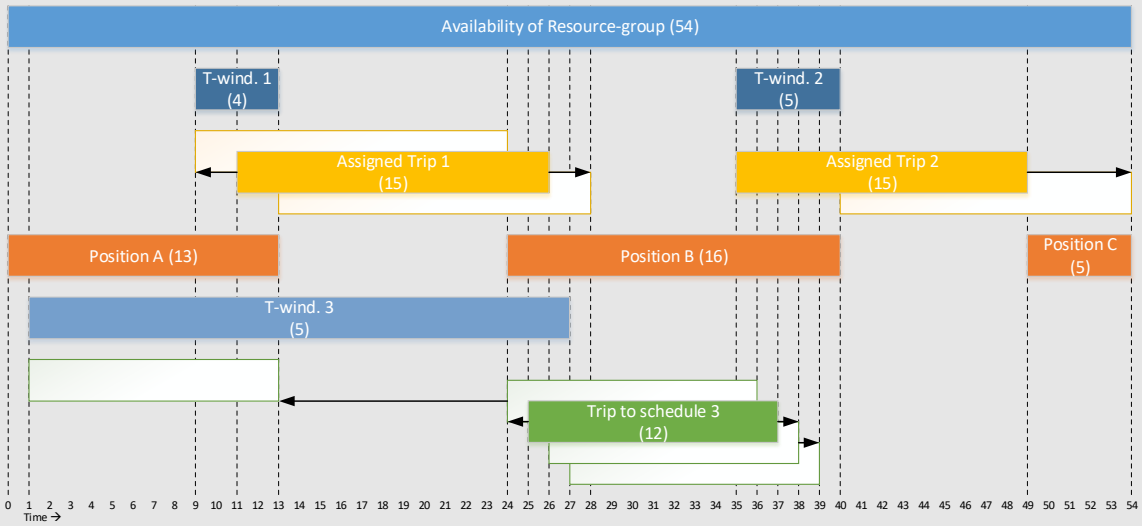

**Figure 5.4:** Find-positions-procedure

For each *trip* we search for positions within already existing *resource-groups* using the find-positions-procedure (existing groups might be feasible for scheduling the *trip*) and create new *resource-groups* using the create-new-groups-procedure (it might be cheaper to start a new group than assigning the *trip* to an existing group). Then, we assign the *trip* to the *resource-group* that can execute the *trip* as the cheapest, and finally, we evaluate the *resource-group* (generating *pre-trips*, *return-trips*, waiting *activities*, and breaks). Figure 5.3 shows a flowchart of our *construction-heuristic*.

Within the find-positions-procedure, we search for existing *resource-groups* and we look for *positions* (see Example 5.3) where the current *trip* can 'fit' in that *resource-group*. We check also check the constraints. A *position* is defined as an unoccupied time-slot in a *resource-group*, related to both the *trips* that are already assigned to that *resource-group*, and the availability of the *resources* in that *resource-group*. The find-positions-procedure is given in Figure 5.4.

The next phase of the *construction-heuristic* is to create new *resource-groups* using the create-new-groups-procedure. In that procedure we first find *resources* that obey the

**Example 5.3:** Positions




Example 5.3 illustrates the principle of *positions* within an existing *resource-group*. We see (top down, with duration in brackets): (*1*) the availability of the *resource-group* under investigation (blue); (*2*) the respective *time-windows* (T-wind. 1 and T-wind. 2; blue) of two assigned *trips* (Trip 1 and Trip 2; yellow); (*3*) three *positions* (Position A, B, and C; orange); and (*4*) the *time-window* (T-wind. 3; blue) of the *trip* to schedule (Trip 3; green).

We check the alternatives to schedule 'Trip 3' on the *resource-group* of investigation. We assume that all *resource* constraints are met. Based on the *time-windows* of the three *trips* and the availability of the *resources-group*, there are three *positions* available (Position A, B, and C). The *trip* takes a duration of 12 units of time and can only fit in Position A or B in the following ways:

- Start Trip 3 as early as possible while delaying Trip 1;
- Expedite Trip 1, start Trip 3 right after and delay Trip 2; or
- Start Trip 3 right after Trip 1 and delay Trip 2.

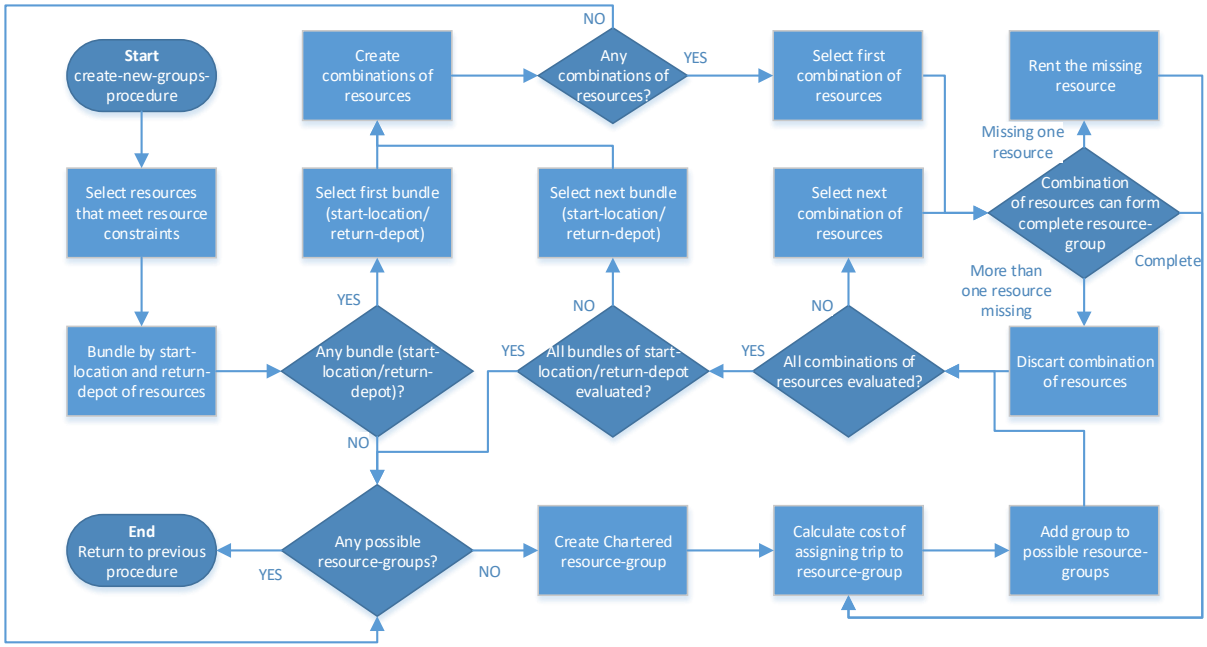



**Figure 5.5:** Create-new-groups-procedure

constraints resulting from the *trip* that is scheduled and its *requested-resources*. *Resource-groups* must be constructed out of *resources* that have the same *start-location* and *return-depot*. Therefore, we bundle the *resources* by *start-location* and *return-depot*. For each bundle of shared *start-location* and *return-depot*, we attempt to create combinations of *resources* that meet the constraints. If a combination is one *resource* short, the missing *resource* is rented. When these attempts of combining and renting *resources* (that share *start-location* and *return-depot*) did not result in any *resource-group*, we add a *Chartered resource-group*. The create-new-groups-procedure is visualized in Figure 5.5.

After this stage has been finished, a list of *resource-groups* remains that can feasibly execute the *trip* under investigation. We rank the list on insertion cost and assign the *trip* to the *resource-group* enabling the insertion at the lowest cost. If we encounter a new *resource-group*, the *resources* are now also linked to the *resource-group*. We now evaluate that *resource-group* and *trips* composing the *resource-group* (using the resource-group-evaluation-procedure and trip-evaluation-procedure as in Section 5.2.2). If the evaluation results in an infeasible *resource-group*, we attempt to assign the *trip* to the subsequent cheapest *resource-group* and once more evaluate that *resource-group*. We continue this procedure until the assignment of all *trips* is realistic and feasible.

### 5.2.5 Improvement-heuristic

As argued in Section 5.1, we choose Simulated Annealing as the *metaheuristic*, guarding our *improvement-heuristic*. We use the algorithm from Aarts and Korst (1989). We reveal the related flowchart in Figure 5.6. The main settings of the heuristic are the 'start temperature' ($T_{start}$), the 'temperature decrease factor' ($T_{decrease}$), the 'stop criterion' ($T_{stop}$), and the 'number of operator attempts to apply per cycle' ($markov_{length}$). The algorithm uses variables for the 'temperature' ($T$) as well as the 'number of operator attempts applied so far' ($N$). The algorithm starts by setting these variables ($T = T_{start}$; $N = 1$) and storing the best solution so far. Then, an improvement-operator is applied, a process which we describe further on. When the improvement-operator is applied, the algorithm evaluates if the new

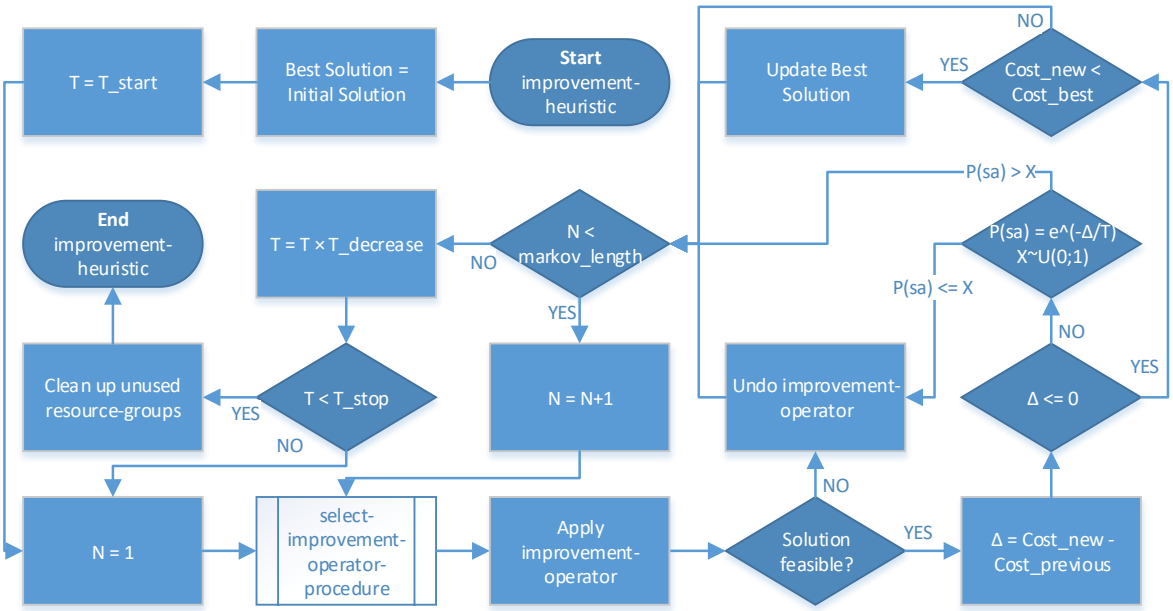



**Figure 5.6:** Improvement-heuristic

solution is feasible (using the resource-group-evaluation-procedure). If not, the new solution is rejected by inversing the improvement-operator. If the new solution appears to be feasible, the 'difference between the costs of the new solution and the previous solution' $\Delta$ is calculated. If the $\Delta$ value is at most zero, the new solution is better or equal good on the previous solution. Subsequently, we check if it is also better than the best solution so far. If so, the best solution so far is updated. If the new solution does not outperform the previous solution, we accept the solution with a certain probability, depending on $\Delta$ and $T$ using a uniform distribution on $X \sim U(0;1)$. In case the solution is not accepted, we discard the solution by inversing the improvement-operator. If $N$ turns out to be smaller than $markov_{length}$, $N$ is incremented by one ($N = N + 1$) and another attempt to apply an improvement-operator is made. When $markov_{length}$ operators are attempted, we decrease $T$ by multiplying it with $T_{decrease}$. We then compare $T$ to the stop criterion ($T_{stop}$). If the algorithm stops, we clean up the solution (i.a., removing *resource-groups* without any *trips*). If not, we reinitialize the process by resetting $N$ and continuing with the new $T$. Theoretically, our algorithm can end without having accepted any feasible improvements (there might be none), because counter $N$ counts all attempts (accepted feasible, rejected feasible, and infeasible). Therefore, the $markov_{length}$ parameter requires a relatively high value to ensure that enough attempts have been made.

### Improvement-operators

The improvement-operators are an essential part of our *improvement-heuristic*. As discussed before, our solution comprises the assignment of two objects: *resources* to *resource-groups*, and *trips* to *resource-groups*. This reasoning results in the following operator-types: (*1*) swap *trips* between *resource-groups* (TRIP-SWAP; Figure 5.7a); (*2*) move a *trip* from one *resource-group* to another (TRIP-MOVE; Figure 5.7b); or (*3*) swap *resources* between *resource-groups* (RESOURCE-SWAP; Figure 5.7c). By definition, moving a *resource* from one *resource-group* to another while having *trips* assigned (RESOURCE-MOVE; Figure 5.7d), without replacing or removing other *resources*, results in an infeasible solution (the *resource-group* has too few or to many *resources*) and yielding a logically impossible


**(a)** TRIP-SWAP (Trips 1&2 of Resource-group A swapped with Trips 4&5 of Resource-group B)

**(b)** TRIP-MOVE (Trip 3 moved from Resource-group A to Resource-group B)

**(c)** RESOURCE-SWAP (Resource 1 of Resource-group A swapped with Resource 2 of Resource-group B)

**(d)** RESOURCE-MOVE (Resource 1 moved from Resource-group A to Resource-group B; both groups become infeasible)

**Figure 5.7:** Improvement-operators

operator. We now continue by examining the three operators-types.

TRIP-MOVE operators are based on insertion-deletion (see Section 2.2.3): a *trip* is removed from a *resource-group* and than inserted again at another place in the attempted solution. We apply this operator only inter-*resource-group* (i.e., the operator is not applied at another place in the same *resource-group*), since *trips* have relatively small *time-windows* (maximally one hour) and *trips* are relatively long (95% over two hours). This operator enables us to vary the number of *resource-groups*. Therefore, we also distinguish between a move to an existing *resource-group* (TRIP-MOVE-EXISTING) and a new *resource-group* (TRIP-MOVE-NEW). New *resource-groups* can be of any kind (*Own*, *Rented*, or *Chartered*).

The TRIP-SWAP operator type is based on Or-Opt (see Section 2.2.3): one, two, or three subsequent *trips* that are assigned to a certain *resource-group* are swapped with the same number of *trips* from another *resource-group* (swapping different numbers of *trips* adds computational complexity). The benefit of this procedure is that the arrangement of *trips* (and thus the *time-windows*) is respected. We distinguish these operators by naming them TRIP-SWAP-N1, TRIP-SWAP-N2, and TRIP-SWAP-N3. Furthermore, sophistication of the algorithm is enhanced by the possibility to swap only *trips* that are similar in nature (both the start- and end-locations of the *trips* are located at most 50 kilometers of each other). This distinction is only applied to the TRIP-SWAP-N1 operator as a consequence of the added computational complexity. We distinguish both operators by naming them TRIP-SWAP-N1-SIMILAR and TRIP-SWAP-N1-RANDOM.

The RESOURCE-SWAP operator enable us to vary the composition of *resource-groups*. *Resources* can be exchanged with *resources* that are represented in another *resource-group*, as well as with *resources* that are not assigned to any *resource-group* at all. Both *Rented* and *Own resources* can be swapped. We distinguish swaps of *drivers* (RESOURCE-SWAP-DRIVER) and swaps of *vehicles* (RESOURCE-SWAP-VEHICLE). Logically, the *resource* constraints should be met.

The operator that is attempted, is selected based on a set of randomly taken decisions, of which the decision-tree is given in Figure 5.8. First, a decision is made between TRIP-MOVE, TRIP-SWAP, and RESOURCE-SWAP. Within the TRIP-MOVE operator type, the decision between moving to a new (TRIP-MOVE-NEW) or an existing *resource-group* (TRIP-MOVE-EXISTING) is made. Within the TRIP-SWAP operator-type, it is decided if the chain should have a length of one (TRIP-SWAP-N1), two (TRIP-SWAP-N2), or three (TRIP-SWAP-N3). Furthermore, within TRIP-SWAP-N1, it is decided if *trips* are randomly chosen (TRIP-SWAP-N1-RANDOM), or that similar *trips* are swapped (TRIP-SWAP-N1-SIMILAR). Finally, within the RESOURCE-SWAP operator type, it is decided if a *vehicle* (RESOURCE-SWAP-VEHICLE) or a *driver* (RESOURCE-SWAP-DRIVER) is swapped.

For each decision, a ratio between outcomes needs to be determined. Within the algorithm, this ratio is used to determine the probability for each outcome (summing up to one for each decision). Using the probabilities and a random number for each decision guides the selection of operators such that the preferred ratio of operators is matched. We illustrate this principle in Example 5.4. In Chapter 6 we discuss these ratios and use experiments to determine the mix of operators that works best.

## 5.3 Trip Assignment Solver

We develop a software tool based on the algorithms and data model described before: the 'Trip Assignment Solver' (screenshot in Figure 5.9). We depend on the programming language and tools used by RGB$^+$ Automatisering. Therefore, the tool is written in C# (C Sharp) as a Windows Forms application on Microsofts .NET framework. The tool (consisting of almost 7000 lines of code) features (*a*) our algorithms; (*b*) functionality to retrieve and store *distance-* and *duration-matrices* from a PTV xServer; (*c*) import and export functionalities; (*d*) visualization of our solutions by drawing a gantt chart; (*e*) calculation of solution statistics; and (*f*) options for doing experiments. A major disadvantage of using C# in our

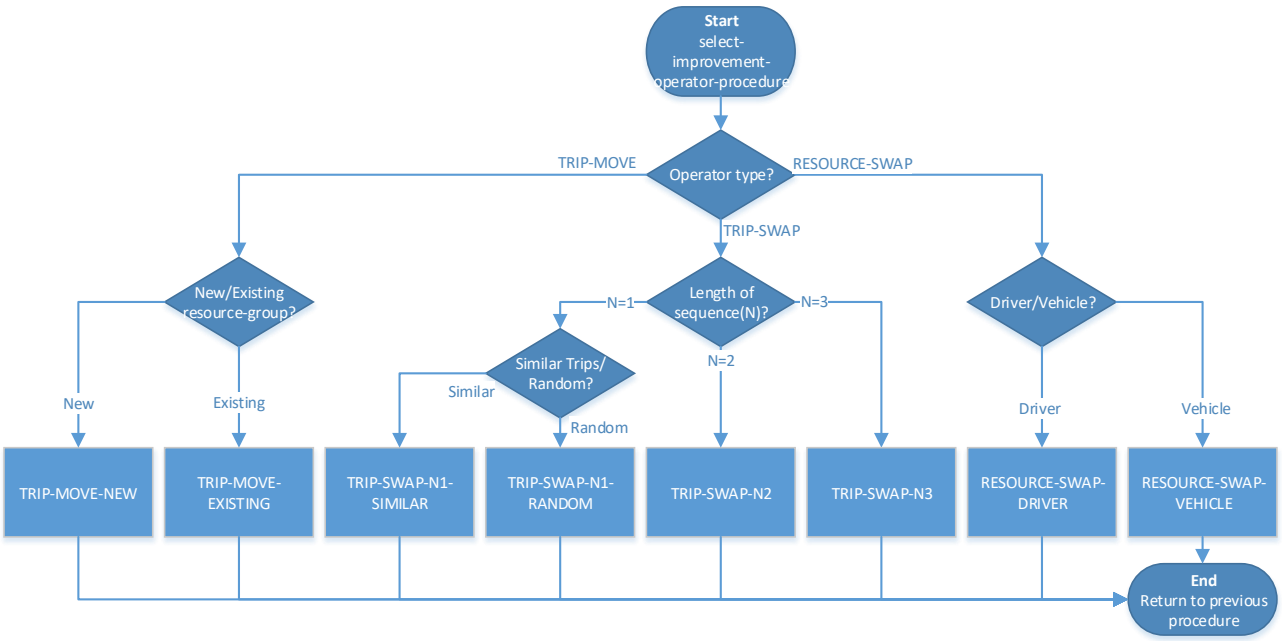



**Figure 5.8:** Decision-tree of improvement-operators (select-improvement-operator-procedure)

**Example 5.4:** Operator ratio

The preferred ratio between TRIP-MOVE (2), TRIP-SWAP (1), and RESOURCE-SWAP (1) is 2:1:1, summing up to 4. Then, the probability of TRIP-MOVE is $\frac{2}{4}$ or $0.50$, the probability of TRIP-SWAP is $\frac{1}{4}$ or $0.25$, and the probability of RESOURCE-SWAP is also $\frac{1}{4}$ or $0.25$.

When this decision is taken, a random number $X$ is taken, $X \sim U(0;1)$. If $0 < X <= 0.50$ then the TRIP-MOVE operator is applied, if $0.50 < X <= 0.75$ then the TRIP-SWAP operator is applied, and in all other cases ($0.75 < X <= 1$) the RESOURCE-SWAP operator is applied.

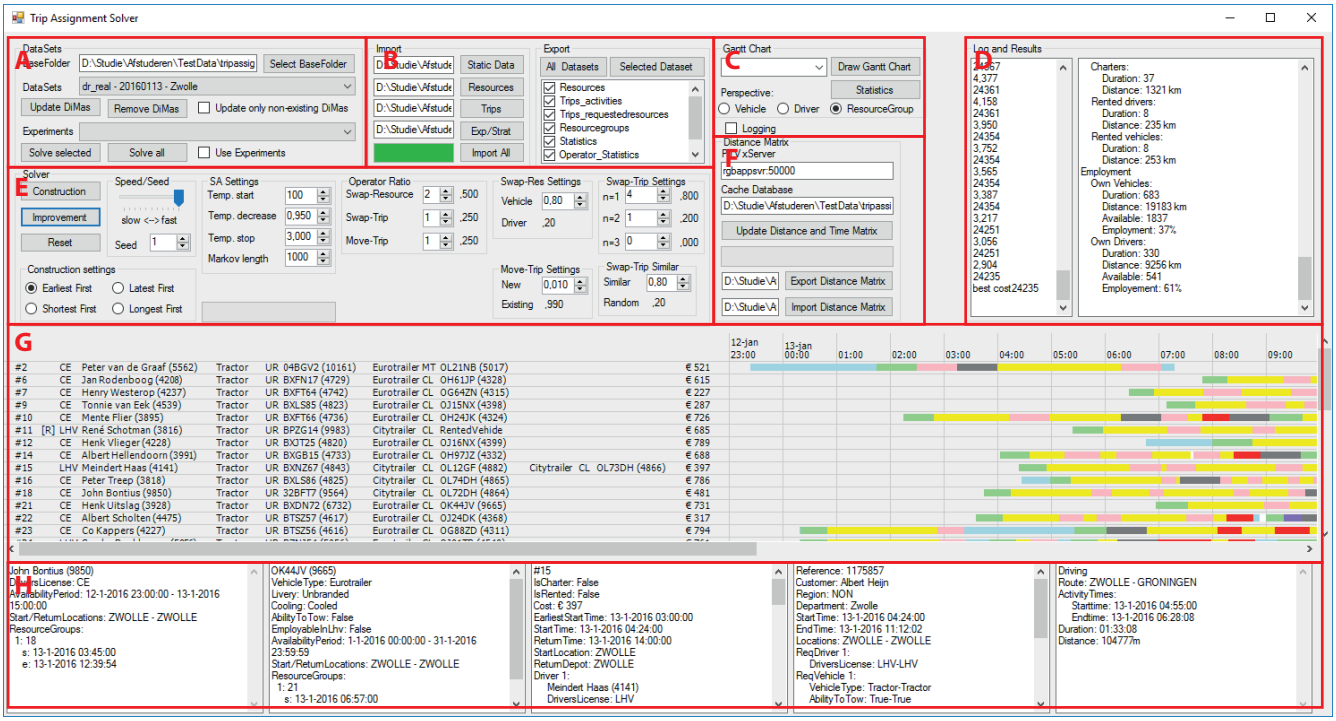



**Figure 5.9:** Trip Assignment Solver, containing of eight parts: (A) dataset selection and automated experiments; (B) import and export functionalities; (C) logging and screen settings; (D) on screen logging and output; (E) distance and duration matrix functionality (xServer API connections); (F) clickable gantt chart and simple information; and (G) object information.

case is its inability to make a deep copy of an object , (i.e., if we copy object A to object B, and subsequently change object B, then object A also changes accordingly). Therefore, we are forced to write deep copy functions ourselves, which step through all objects, which is computationally intensive for copying large objects (such as the entire best solution).

Definition

A **deep copy** is a copy where objects are dereferenced.

## 5.4 Conclusions

In this chapter we described the aspects we had to consider during the development of a solution method that is able to solve our model. We decided to solve the model heuristically and choose a parallel scheduling method for our *construction-heuristic* and Simulated Annealing for our *improvement-heuristic*. We continued with our approach: we discussed the data model, the evaluation of *resource-groups* and *trips*, and the checking of constraints. Furthermore, we use this input to develop our *construction-* and *improvement-heuristics*, which we also implemented in the Trip Assignment Solver tool.

# Chapter 6

# Results

This chapter presents the results of our solution model, using real data. First, we discuss the setup of our experiments (Section 6.1). Before we could run our experiments, we had to clean up the input data, to make them feasible for sound analyses (Section 6.2). Furthermore we discuss the verification and validation of our model in Section 6.3. Then, we describe and analyze our experiments (Section 6.4), followed by the conclusion in Section 6.5.

## 6.1 Setup

In this section we present the setup of our experiments. In Section 6.1.1 we define which datasets we use. We continue with the experimental factors in Section 6.1.2, followed by the number of replications (Section 6.1.3). Finally, we discuss our key performance indicators (KPIs) in Section 6.1.4.

### 6.1.1 Datasets

PAT kindly makes available a huge set of data, covering the entire set of (manually planned) *trips* and *resources* over the course of January 2016, approximating a total of 31,000 *trips* (see Chapter 3). For experiment purposes, we omit the first week of January because of irregularities around New Year.

Random sampling from this large data set is considered undesirable or even impossible: randomly sampled (sets of) *trips* and *resources* could potentially have entirely different proportions that determine the consistency of a day (such as *departments*, *trip-type*, or *resource-availability*). The different sampled *trips* would add up to a sampled dataset, consisting of 'apple and pears', without any logical or empirical rationale to treat this sample as 'fruit'.

Therefore, we decide to sample randomly on the level of entire naturally occurring sequences of *trips*, defined by uncut (full-length) days. These days served as the logical unit for the analyses. To limit calculation times to realistic efforts and durations, three days were randomly selected, according to the following procedure: first all similar days of the week (all Mondays, Tuesdays, Wednesdays and Thursdays of the remaining weeks of January 2016) were lumped together, and one working day was selected randomly (Wednesday, January, 13; 'Day 1'). The procedure was repeated for all Fridays of the month, with Friday, February, 22 drawn from this subset of data ('Day 2'). Finally, the procedure was repeated for all Saturdays and all Sundays of January, 2016, with Saturday, January, 30 as the selected date ('Day 3'). Thus, out of the month defining the whole data set, three entire days of manually assigned *trips* remained for testing the model (Table 6.1).

| # | Date | Day of week | # trips | # departments |
|---|---|---|---|---|
| Day 1 | 13-01-2016 | Wednesday | 1167 | 42 |
| Day 2 | 22-01-2016 | Friday | 1216 | 42 |
| Day 3 | 30-01-2016 | Saturday | 923 | 33 |

**Table 6.1:** Datasets

### 6.1.2 Experimental factors

One of the aims of this study is to test the potential benefits of scheduling at a higher aggregation level. We define three aggregation levels: (*1*) *Local* (L): schedules are set up per *department*; (*2*) *Regional* (R): schedules are created per *region*, containing both *trips* and *resources* of the *departments* that are linked to the respective *region*; and (*3*) *National* (N): all *trips* and *resources* are scheduled together, irrespective the *departments* and *regions*.

| Parameter | Set A | Set B |
|---|---|---|
| Time cost (hour) | 37 | 37 |
| Distance cost (km) | 0.8 | 0.8 |
| Rent factor | 1.5 | 2 |
| Charter factor | 2 | 4 |

**Table 6.2:** Cost Settings

Section 4.6 discussed the structure applied for calculating the costs of our model. A set of cost parameters emerged, that we call *'Set A'*. Because we would like to investigate the sensitivity of our model regarding these cost, we introduce a second set of cost settings (*'Set B'*), enlarging the differences between *Own*, *Rented* and *Chartered resource-groups*. Both sets are included in Table 6.2.

Furthermore, we are searching for the best configuration of our *improvement-heuristic* (see Section 5.2.5). Therefore, we require the ratio between improvement-operators as well as proper settings for the Simulated Annealing algorithm. Determining the ratio between operators is part of the experiments, see Section 6.4.1. However, we define initial settings that are used throughout this chapter (see Table 6.3).

| Description | Operator type | | | New/Existing | | Sequence length | | | Similar/Random | | Vehicle/Driver | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | TRIP-MOVE- | | TRIP-SWAP- | | | TRIP-SWAP-N1- | | RESOURCE-SWAP- | |
| | RESOURCE-SWAP | TRIP-SWAP | TRIP-MOVE | NEW | EXISTING | N1 | N2 | N3 | SIMILAR | RANDOM | VEHICLE | DRIVER |
| Initial settings | 1/3 | 1/3 | 1/3 | 0.1% | 99.9% | 1/3 | 1/3 | 1/3 | 1/2 | 1/2 | 1/2 | 1/2 |

**Table 6.3:** Initial operator settings

| Parameter | Value |
|---|---|
| Operators applied | 100,000 |
| Operators feasible | 11,758 |
| $\Delta < 0$ | 248 |
| $\Delta = 0$ | 10,512 |
| $\Delta > 0$ | 998 |
| $min(\Delta)$ | -395 |
| $max(\Delta)$ | 864 |
| $avg(\Delta)$ | 15 |
| $avg(\Delta > 0)$ | 202 |

**Table 6.4:** Acceptance ratio

With respect to the Simulated Annealing algorithm, we determine $T_{start}$, $T_{stop}$, $T_{decrease}$, and $markov_{length}$. We conduct an experiment using the operator ratios given in Section 6.4.1, and take the results of the first 100,000 attempts, summarized in Table 6.4. It is common to choose $T_{start}$ such that the acceptance ratio of feasible neighbors is approximately equal to 1 (i.e., the number of accepted neighbors is approximately equal to the number of attempted neighbors). Using the method of (Crama & Schyns, 2003), we calculate the temperature for the average increase in the objective value (deterioration; $\Delta > 0$), at which $P_{acceptance} \approx 90\%$ (see Equation 6.1). Because the maximum $\Delta > 0$ is much higher than the average $\Delta > 0$, we increase the calculated value, resulting in $T_{start} = 2500$.

$$e^{\frac{-202}{T_{start}}} = 0.9 \tag{6.1a}$$

$$T_{start} = \frac{-202}{ln(0.9)} \approx 1917 \tag{6.1b}$$

For the stop temperature we take $\Delta = 1$ (taking in mind that the objective value represents costs, an increase of € 1 is acceptable) and calculate the temperature at which $P_{acceptance} \approx 1\%$ (see Equation 6.2), resulting in $T_{stop} = 0.2$.

$$e^{\frac{-1}{T_{stop}}} = 0.01 \tag{6.2a}$$

$$T_{stop} = \frac{-1}{ln(0.01)} \approx 0.22 \tag{6.2b}$$

We analyze the acceptance of operators and the decrease of the solution value. We find that the objective value barely deteriorates after approximately 10 million operators were attempted (feasible and infeasible). Therefore, we set $T_{decrease} = 0.99$ and $markov_{length} = 12,000$, resulting in a total of 11,256,000 attempts.

| Parameter | Full (F) | Constrained (C) |
|---|---|---|
| $T_{start}$ | 2500 | 100 |
| $T_{stop}$ | 0.2 | 0.01 |
| $T_{decrease}$ | 0.99 | 0.98 |
| $markov_{length}$ | 12,000 | 5000 |
| # Temperature changes | 938 | 455 |
| # Operators in algorithm | 11,256,000 | 2,275,000 |

**Table 6.5:** Simulated Annealing settings

We call this set of parameters the *'Full'* (F) set of Simulated Annealing parameters. Because the *Full* set requires very long run times, we also introduce a *'Constrained'* (C) set of parameters. This set has limited deterioration possibilities ($P_{acceptance} = e^{\frac{-213}{100}} = 0.12$). Because we want to focus on the final stage of the Simulated Annealing algorithm (similar to a local search), we also select a lower stop criterion. Both sets of Simulated Annealing settings are summarized in Table 6.5.

### 6.1.3 Number of replications

Our algorithms depend on random factors for both the choice of operators as well as the acceptance of new solutions. Using different random numbers (in several runs) leads to differences in end-results, which is undesirable. Because our experiments are terminating (the algorithm stops when $T < T_{stop}$), using a single long run and taking parts of that run as replications is not an option. Therefore we choose to use multiple replications (repeating the experiment with different predefined random number seeds).

We use the sequential confidence-interval method (Law, 2007). We calculate the 99% confidence-interval of the *total-costs*, (see Equation 6.3a) using the confidence-interval half-length $\delta(n, \alpha)$ (see Equation 6.3b):

$$[\bar{X}(n) - \delta(n,\alpha), \bar{X}(n) + \delta(n,\alpha)] \tag{6.3a}$$

with

$$\delta(n,\alpha) = t_{n-1,1-\alpha/2}\sqrt{\frac{S^2(n)}{n}} \tag{6.3b}$$

$$S^2(n) \quad \text{(the variance of the first } n \text{ replications)} \tag{6.3c}$$

$$\bar{X}(n) \quad \text{(mean of the first } n \text{ replications)} \tag{6.3d}$$

$$t_{n-1,1-\alpha/2} \quad (t\text{-distribution; } \alpha \text{ confidence level; } n-1 \text{ degrees of freedom}) \tag{6.3e}$$

Furthermore, we accept a relative error $\gamma'$ of our 99% confidence-interval of at most 1% (see Equation 6.4a). Based on our experiments (see Table 6.6), we conclude that 4 runs are required for each experiment. When considering the duration of the experiments, we conclude that these four experiments have an average duration of 42:56 minutes.

$$\delta(n,\alpha)/|\bar{X}(n)| \leq \gamma' \tag{6.4a}$$

$$\gamma' = \gamma/(1+\gamma) \quad \text{(corrected relative error)} \tag{6.4b}$$

### 6.1.4 Key Performance Indicators

Our software produces several statistics (see Appendix H). In collaboration with PAT, we define 12 important Key Performance Indicators (KPIs): (*1*) *runtime* (in hour, the lower the better); (*2*) *improvement-percentage* (in % of cost of *improvement-heuristic* relative to *construction-heuristic*, the higher the better); (*3*) *total-cost* (in €, the lower the better); (*4*) *total-duration* (in hours of working, the lower the better); (*5*) *total-distance* (in kilometer, the lower the better); (*6*) *cost-per-hour* (in €, the lower the better); (*7*) *pre/returntrip-duration* (in % of total trip duration, the lower the better); (*8*) *pre/returntrip-distance* (in % of

| Run ($n$) | Cost ($X$) | Cumulative mean ($\bar{X}(n)$) | Standard deviation ($S(n)$) | Lower interval ($\bar{X}(n) - \delta(n,\alpha)$) | Upper interval ($\bar{X}(n) + \delta(n,\alpha)$) | Deviation ($\delta(n,\alpha)/\lvert\bar{X}(n)\rvert$) | Duration of run (minutes) |
|---|---|---|---|---|---|---|---|
| 1 | 141,886 | 141,886.00 | n/a | n/a | n/a | n/a | 43:41 |
| 2 | 142,151 | 142,018.50 | 187.383 | 133,583.98 | 150,453.02 | 5.94% | 42:40 |
| 3 | 142,530 | 142,189.00 | 323.677 | 140,334.29 | 144,043.71 | 1.30% | 41:42 |
| 4 | 142,597 | 142,291.00 | 333.857 | 141,315.99 | 143,266.01 | 0.69% | 43:41 |

**Table 6.6:** Required replications (Dataset: Day 1; Aggregation level: Regional (South); SA: Constrained; Cost: Set A; Initial operator settings)

total trip distance, the lower the better); (*9*) *outsourcing-cost* (in €, cost of *Chartered resource-groups* plus extra cost of *Rented resource-groups* relative to *Own resource-groups*), the lower the better; (*10*) *resource-groups* (counted; number of); (*11*) *driver-employment* (in % of availability, *Own drivers* only, higher is better); and (*12*) *vehicle-employment* (in % of availability, *Own vehicles* only, higher is better). We use these KPIs to compare the results of our experiments.

## 6.2 Input assumptions

This section describes the assumptions and requirements concerning the input data for our model to be used for verification, validation and experimenting. The assumptions involve *working-hour-regulations*, *requested-resources* and *time-windows*.

- We assume that a *driver* has had a full *Daily-rest* at the start of his/her *availability-period*. In practice, this means that we do not take any *drivers' activities* into account on the days before the current schedule: *drivers* are allowed to work a full day (taking into account the constraints described in Chapter 4).
- As already described in Chapter 3, *orders* do not incorporate any information about the required volume or *requested-resources*. However, we assume that the planners at PAT schedule *trips* in the right way (as desired by the customers of PAT), based on their expertise. We derive the *requested-resources* of an *order* from the historical execution of those *orders*, assuming that this corresponds to the actual *requested-resources*. We are aware however that the following errors occur and therefore we correct for them:
  - The Transplan TMS has a maximum of two *vehicles* and one *driver* per *trip*. Therefore, the number of *drivers* is limited to one per *trip*, therefore, the Transplan TMS is unable to cope with LHVs properly (which have more that two *vehicles*). Nevertheless, we know that there are LHVs required, so we make an educated guess. We parse the textual description (comment) of the order (contains 'LHV'?) and combine that information with historical scheduling information to determine if a LHV is required for that *order*.
  - Because PAT uses their *vehicles* for multiple customers, some *trips* that do not obviously need *Cooled* transport are executed by *Cooled* (or *Multi-temperature*) *vehicles*. We correct *trips* for bakeries, postal services, drugstores and non-cooled warehouses to allow at least *Unregulated vehicles* instead of the cooling value of their respective historically scheduled *vehicles*.
- Some *trips* have extremely wide *time-windows* (approximately the entire day) or no *time-windows* at all, which are practically impossible and actually not intended by the customers of PAT. For those *trips*, we set the *time-window* of the first *activity* of the *trip* to be exactly the historically scheduled departure time of the first *activity* of that *trip*. Although this assumption limits the scheduling possibilities, it is more in line with reality. We find out however, that in some of these cases, *trips* include *pre-trips* and *return-trips*: meaning that the departure time is set as the time at which a

*resource-group* must depart from the *depot* to be in time for the first *activity* of the *trip* it is traveling towards. This procedure actually sometimes doubles the *trip-duration* and implicitly includes the *depot* from which the *resources* must depart, something that is actually part of the choices we make in this research. The way the scheduled departure-time is used is contrary to the design of Transplan TMS. Therefore, we are unable to determine in which cases this happens, because this information is simply not captured anywhere. We therefore take this for granted.

- We adapt the *trips* that include a *Transporting activity* with a *duration* of more than 4.5 hours, enabling us to schedule breaks more easily (without splitting *activities* within the algorithm). The *duration* of some *Transporting activities* in our dataset even exceeds 15 hours, which is legally impossible without taking a *Daily-rest*. In practice, those *Transporting activities* have a *duration* of at most 9 hours. Therefore we limit the required *duration* of those *activities* to 9 hours. Furthermore, for break scheduling reasons, we split *Transporting activities* with a *duration* of more than 4.5 hours into two *Transporting activities* (e.g., a 6 hour *Transporting activity* becomes two 3 hour *activities*; a 13 hour *activity* becomes two 4.5 hour *activities*).

- A large part of the *trips* have *time-windows* being infeasible by definition. Infeasible in this case means that, when a *resource-group* starts executing the first *activity* of a *trip* as early as possible, then the *resource-group* is unable to be in time for any or some of the *activities* later on in that *trip*. This can not be explained by the use of another *distance-matrix* (identical as in Transplan TMS) or break scheduling (excluded), but simply by dirty data: *time-windows* in our data do not always match the *time-windows* given by the customers of PAT. Because we are unable to validate which *time-windows* are correct and which are incorrect, we remove all *time-windows*, except the *time-window* for the first *stop* of a given *trip*, assuming that this *time-window* is correct.

- We find out that the planners at PAT do not use the *driver-availability* schedule. In general, according to the given *driver-availability* schedule, almost no *drivers* are available at night, although there are *orders* that were executed at night. The *drivers* are thus scheduled based on planners' experience and phone calls. This yields lots of cases where a *driver* is (according to the schedule) available during the day (e.g., 3.00-17.00 hour), and being employed at night (e.g., 16.00-2.00 hour), just based on the planner's experience. Using the given schedule, results in a shortage of *drivers* during the night and therefore in the excessive scheduling of *Rented drivers* and *Chartered resource-groups*. We look for a more realistic *driver-availability* schedule: we take the period of time that the *driver* was actually working historically and add two hours before and after that time span as an adapted *availability-period*.

The number of assumptions we make concerning the data has implications for this research. Historical (manually created) schedules are essential for the adjustments of our input data (i.e., our solution cannot be supplied with data, without manually scheduling the *trips* at first). With this, we created a relationship between manually and automatically created schedules. Therefore, we are now unable to statistically compare the outcomes of our model with the manually created schedules. This conclusion influences our verification and validation as well as our experiments.

## 6.3 Verification and validation

One of the main challenges of this research is to determine if we can replace the manual scheduling method with an automated scheduling method, and to determine what the differences are between manual and automated scheduling. Therefore, we validate our model and the schedules resulting from our solution. During development, we regularly look critically

at the schedules we create in order to verify that those schedules are realistic. Furthermore we aim to validate our solution by comparing the schedules we create on a department level with the schedules created manually at PAT on a department level. However, due to the number of assumptions we have to make regarding our data (see Section 6.2), we are unable to statistically validate our model using the data that is currently available. Unfortunately, the quality of the input data must improve drastically to enable automated optimization of schedules at PAT. All the same, for the present research, we run small experiments (Dataset: *Day 1*; Aggregation level: *Local* (Sint-Annaparochie, Zwolle, Breda, Zaandam, and Geldermalsen); SA: *Constrained*; Cost: *Set A*; Initial operator settings), resulting in five schedules for five local *departments*. We present the results of those experiments to the planners at PAT, who we ask to assess the solutions (experts opinion) by filling in a small questionnaire (see Appendix I), containing 7 open (scaled -2 (Disagree) to 2 (Agree)) and 4 closed questions. We give a summary of the results in Table 6.7.

Based on these results, we conclude that the planners reasonably satisfied with our present planning as a base planning. However they mention some concerns and challenges for improvement, which we point out below. Each of the points mentioned by the planners is followed by our comments.

- Different *shift* lengths and long *shifts*: We agree that there are differences in *shift* lengths between individual *drivers*. The main objective of our model is however to schedule at low cost. In relation to the long *shifts*, our model and solution take legal *working-hour-regulations* into account, which are assessed to be properly accounted for. Our model does not contain other *shift* length related constraints. Therefore, we conclude that the comments related to *shift* length are due to the choices we made in our model. Incorporating secondary objectives such as equal *shift* lengths in the model, involves making considerations between two objectives that might be contradictory. The complexity of the model will increase.

- Preloading and execution are falsely independently scheduled: We agree that preloading and execution of a *trip* should be done by the same *resources*. In our input, these actually consists of two *trips* which are unrelated in the Transplan TMS. Therefore, we also see these *activities* as separate *trips* and our model allows them to be scheduled on different *resources*. The simplest solution for this problem is to include preloading *activities* within the *trip* it belongs to, without having any model consequences. Another options is to add constraints to the model that require related *trips* to be scheduled on the same *resources*.

- Improve *vehicle* usage: Again, the main objective of our model is to schedule at low cost. There might be cases where the assignment of *vehicles* seems illogical, however, this might result in lower cost. Smoothing out the usage of *vehicles* is not an objective or constraint of our model. Again, incorporating secondary objectives increases the complexity of the model.

- Location and *trip* knowledge: We agree that there might be some *drivers* scheduled on *trips* that they have no knowledge of. These are however not part of our model and are therefore not taken into account. The main implication of these constraints is that the data should be properly captured. Adding the corresponding constraints to the model is relatively easy.

- Exchange of *Rented resources* and *Chartered resource-groups*: We agree that the amount of *Rented resources* is relatively high and that sometimes charters can be used for those *trips*. This would however result in higher costs, which is against our main objective. Solutions for this problem are increasing the costs of *Rented resources*, or limiting the number of available *Rented resources* and/or charters, which has limited impact on the model.

| Question | Score | Interpretation | Summary of comments |
|---|---|---|---|
| 1. The planning is of a high quality level | 0.8 | Neutral/Slightly agree | Can't find large mistakes / Shift length differs / Given conditions are met / Preloading and execution should be accomplished with same resources / Vehicle and drivers are scheduled obeying legal rules and regulations. |
| 2. Break scheduling is sufficient | 1.0 | Slightly agree | Breaks seem to be scheduled in line with the collective labor agreement / Even the night's rest are scheduled / Legal driving time rules are respected. |
| 3. The planning is realistic | 1.4 | Slightly agree/Agree | This planning can be executed / Resources are employed multiple times per day, that is okay / Some details might be improved or done differently / The planning seems realistic as far that we can see / We don't see extremely long shifts or things that are not possible. |
| 4. The available vehicles are efficiently deployed | 0.6 | Neutral/Slightly agree | Less vehicles might be used / Exchanges between departments might be possible / Some shifts might be combined on the vehicle level / There are some large waiting gaps between trips, which we want to avoid. |
| 5. The available drivers are efficiently deployed | 0.8 | Neutral/Slightly agree | Shifts are sometimes very long, shorter shifts might be desirable for some drivers / Location and trip knowledge is not incorporated. |
| 6. Charters and rented resources are efficiently deployed | 1.0 | Slightly Agree | Short trips are assigned to charters, that principle is all right / No irregularities / Some rented vehicles and charters could be exchanged. |
| 7. The plan contains major errors | -0.8 | Neutral/Slightly Disagree | We don't see major errors, however we see possible refinements / We are concerned about the amount of employment hours of the drivers / As a basis, a very neat planning is presented. |
| A. What points can be found weak in this planning? | | | There are no major weaknesses / Some improvements regarding the employment of drivers and vehicles could be preferable. |
| B. What points you will recommend for the improvement of this planning? | | | Vehicle employment percentage / Own vehicles should have the longest trips |
| C. What points would you improve this schedule? | | | Improving the length of the driver shifts, and increase the employment of vehicles / Multiple employment of charters (lower cost and operational load) / Incorporate driver knowledge / Combining with other departments. |
| D. Do you have any further remarks? | | | Cool that this planning is generated by a computer / We see potential for detailing and refinement / Cool that automated planning is possible, are we going to use this? |

**Table 6.7:** Validation results

## 6.4 Experiments

We execute five stages of experiments. In the first stage, we select the best values for the operator ratios of our *improvement-heuristic* (Section 6.4.1). In the second stage, we experiment using our '*construction-heuristic* only' (Section 6.4.2). We continue with both the *construction-* and *improvement-heuristics* in the 'production runs' (Section 6.4.3). We perform a 'sensitivity analysis' of the cost parameters in Section 6.4.4. Finally, we run our *improvement-heuristic* with the constrained set of Simulated Annealing parameters in Section 6.4.5.

### 6.4.1 Stage 1: 'improvement-operator settings'

In the first stage of our experiments, we establish the best combination of operator ratios. The theoretical number of experiments (based on the dimensions) can become huge, due to the number of possible values for each of the five operator ratios (Operator type, New/Existing, Sequence length, Similar/Random, and Vehicle/Driver). For example, if we take four different combinations for each possible ratio that we have (e.g., for the Operator ratio 'RESOURCE-SWAP:TRIP-SWAP:TRIP-MOVE', we use the following four ratios: (1) '1:1:1', (2) '1:1:2', (3) '1:1:3', and (4) '1:1:4'), we can run a total of $4^5 = 1024$ experiments, resulting in a total duration of $1024 \times 4$ (replications) $\times$ 3 (regions) $\times$ 43 (minutes) $\approx$ 8800 hour (see Section 6.1.3 for replications and duration). Therefore, we choose to measure the effects of each of the parameters individually (one factor at the time). We start with the execution of a base experiment and then change the dimensions one by one. For ratios that have three values ('Operator type' and 'Sequence length') we also perform experiments to investigate the interaction between these values. We use the dataset of *Day 1*, aggregate on a *Regional* level (medium aggregation), set the costs to *Set A*, and use the *Constrained* Simulated Annealing settings (for duration reasons). Each experiment is replicated four times using different seeds. Table 6.8 presents both the settings (Table 6.8a) and results (Table 6.8b) of the improvement-operator experiments.

Note

We run our experiments on several cores (≈3.4Ghz per core) simultaneously.

**Results**

The experiments have a *runtime* of 773 hours. We now analyze the results of our experiments based on the five operator ratios. Furthermore, we decide which ratios we use within the next stages:

- Operator type (RESOURCE-SWAP/TRIP-SWAP/TRIP-MOVE, experiments 2-4 & 5-7): Experiments 4, 6, and 7 reveal that especially the TRIP-MOVE operator gives better results than the other two operators (being preferred in experiment 2, 3, and 5). Therefore, we select a higher value for the TRIP-MOVE operator, resulting in a ratio of 1/2 for the TRIP-MOVE operator and 1/4 for both the RESOURCE-SWAP and TRIP-SWAP operators in further experiments.
- New/Existing (TRIP-MOVE-NEW/TRIP-MOVE-EXISTING, experiments 8-9): Experiment 1, 8 and 9 show that the highest value (1%) for adding new groups gives better results than lower values for new groups (0.1% and 0%). Therefore, we select the highest value tested with (1% new groups) in further experiments.
- Sequence length (TRIP-SWAP-N1/TRIP-SWAP-N2/TRIP-SWAP-N3, experiments 10-12 & 17-19): Experiments 10, 17, and 19 reveal that a swap-sequence of 1 trip gives far better results than longer sequences, (being preferred in experiment 11, 12, and 18). Therefore, we prefer the TRIP-SWAP-N1 operator for further experiments as well. Due to the large differences in costs, we increase the ratio value to 4/6 for the TRIP-SWAP-N1 operator. Both the TRIP-SWAP-N2 and TRIP-SWAP-N3 operators get a ratio of 1/6.
- Similar/Random (TRIP-SWAP-N1-SIMILAR/TRIP-SWAP-N1-RANDOM, experiments 13-14): From experiment 13 and 14, we conclude that there is a small difference in favor of the TRIP-SWAP-N1-RANDOM operator. We set a ratio value of 3/5 in favor of the TRIP-SWAP-N1-RANDOM operator and 2/5 for the TRIP-SWAP-N1-SIMILAR operator.
- Vehicle/Driver (RESOURCE-SWAP-VEHICLE/RESOURCE-SWAP-DRIVER, experiments 15-16): The differences between the outcomes experiments 15 and 16 appear to be very small. However, the RESOURCE-SWAP-DRIVER operator performs slightly better than the other operator. Therefore, we assign the RESOURCE-SWAP-DRIVER operator a ratio value of 3/5, while the RESOURCE-SWAP-VEHICLE operator gets a ratio of 2/5.

### 6.4.2 Stage 2: 'construction only'

We run our *construction-heuristic* for the three selected days (*Day 1*, *Day 2*, and *Day 3*), using the three aggregation levels (*Local*, *Regional*, *National*). Due to the lack of randomness in the *construction-heuristic*, we do not require multiple replications. Furthermore, we use cost *Set A*.

**Results**

The total *runtime* is 2 hours. The results of the experiments are summarized in Table 6.9. We now analyze the results using the following comments:

- As expected, the best results often occur at the highest aggregation levels (*Regional* and *National*). The *Local* aggregation level gives the best results only for *Day 2* and *Day 3* of the *driver-employment* percentage.
- *Runtimes* are shorter than expected, it stands out that the best runtimes are reached at the *Regional* aggregation level. The *runtime* is approximately 30 times longer for the *Local* level compared to the *Regional* level, and 400 times longer for the *National* level compared to the *Regional* level.

| # | Description | Datasets | Agg. Lvl. | Sim. An. | Cost Set | Operator type | | | New/Existing | | Sequence length | | | Similar/Random | | Vehicle/Driver | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | TRIP-MOVE- | | TRIP-SWAP- | | | TRIP-SWAP-N1- | | RESOURCE-SWAP- | |
| | | | | | | RESOURCE-SWAP | TRIP-SWAP | TRIP-MOVE | NEW | EXISTING | N1 | N2 | N3 | SIMILAR | RANDOM | VEHICLE | DRIVER |
| 1 | Base experiment | Day 1 | R | C | A | 1/3 | 1/3 | 1/3 | 0.1% | 99.9% | 1/3 | 1/3 | 1/3 | 1/2 | 1/2 | 1/2 | 1/2 |
| 2 | 100% RESOURCE-SWAP | Day 1 | R | C | A | 1 | 0 | 0 | 0.1% | 99.9% | 1/3 | 1/3 | 1/3 | 1/2 | 1/2 | 1/2 | 1/2 |
| 3 | 100% TRIP-SWAP | Day 1 | R | C | A | 0 | 1 | 0 | 0.1% | 99.9% | 1/3 | 1/3 | 1/3 | 1/2 | 1/2 | 1/2 | 1/2 |
| 4 | 100% TRIP-MOVE | Day 1 | R | C | A | 0 | 0 | 1 | 0.1% | 99.9% | 1/3 | 1/3 | 1/3 | 1/2 | 1/2 | 1/2 | 1/2 |
| 5 | RESOURCE-SWAP/TRIP-SWAP | Day 1 | R | C | A | 1/2 | 1/2 | 0 | 0.1% | 99.9% | 1/3 | 1/3 | 1/3 | 1/2 | 1/2 | 1/2 | 1/2 |
| 6 | TRIP-SWAP/TRIP-MOVE | Day 1 | R | C | A | 0 | 1/2 | 1/2 | 0.1% | 99.9% | 1/3 | 1/3 | 1/3 | 1/2 | 1/2 | 1/2 | 1/2 |
| 7 | RESOURCE-SWAP/TRIP-MOVE | Day 1 | R | C | A | 1/2 | 0 | 1/2 | 0.1% | 99.9% | 1/3 | 1/3 | 1/3 | 1/2 | 1/2 | 1/2 | 1/2 |
| 8 | More new groups | Day 1 | R | C | A | 1/3 | 1/3 | 1/3 | 1% | 99% | 1/3 | 1/3 | 1/3 | 1/2 | 1/2 | 1/2 | 1/2 |
| 9 | No new groups | Day 1 | R | C | A | 1/3 | 1/3 | 1/3 | 0% | 100% | 1/3 | 1/3 | 1/3 | 1/2 | 1/2 | 1/2 | 1/2 |
| 10 | 100% N1 | Day 1 | R | C | A | 1/3 | 1/3 | 1/3 | 0.1% | 99.9% | 1 | 0 | 0 | 1/2 | 1/2 | 1/2 | 1/2 |
| 11 | 100% N2 | Day 1 | R | C | A | 1/3 | 1/3 | 1/3 | 0.1% | 99.9% | 0 | 1 | 0 | 1/2 | 1/2 | 1/2 | 1/2 |
| 12 | 100% N3 | Day 1 | R | C | A | 1/3 | 1/3 | 1/3 | 0.1% | 99.9% | 0 | 0 | 1 | 1/2 | 1/2 | 1/2 | 1/2 |
| 13 | More similar swaps | Day 1 | R | C | A | 1/3 | 1/3 | 1/3 | 0.1% | 99.9% | 1/3 | 1/3 | 1/3 | 4/5 | 1/5 | 1/2 | 1/2 |
| 14 | More random swaps | Day 1 | R | C | A | 1/3 | 1/3 | 1/3 | 0.1% | 99.9% | 1/3 | 1/3 | 1/3 | 1/5 | 4/5 | 1/2 | 1/2 |
| 15 | More vehicle swaps | Day 1 | R | C | A | 1/3 | 1/3 | 1/3 | 0.1% | 99.9% | 1/3 | 1/3 | 1/3 | 1/2 | 1/2 | 4/5 | 1/5 |
| 16 | More driver swaps | Day 1 | R | C | A | 1/3 | 1/3 | 1/3 | 0.1% | 99.9% | 1/3 | 1/3 | 1/3 | 1/2 | 1/2 | 1/5 | 4/5 |
| 17 | N1/N2 | Day 1 | R | C | A | 1/3 | 1/3 | 1/3 | 0.1% | 99.9% | 1/2 | 1/2 | 0 | 1/2 | 1/2 | 1/2 | 1/2 |
| 18 | N2/N3 | Day 1 | R | C | A | 1/3 | 1/3 | 1/3 | 0.1% | 99.9% | 0 | 1/2 | 1/2 | 1/2 | 1/2 | 1/2 | 1/2 |
| 19 | N1/N3 | Day 1 | R | C | A | 1/3 | 1/3 | 1/3 | 0.1% | 99.9% | 1/2 | 0 | 1/2 | 1/2 | 1/2 | 1/2 | 1/2 |

**(a)** Settings of Stage 1.

| # | Description | Runtime | Improvement-percentage | Total-cost | Total-duration | Total-distance | Cost-per-hour | Pre/returntrip-distance | Pre/returntrip-duration | Outsourcing-cost | Resource-groups | Driver-employment | Vehicle-employment |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | ◷hh:mm:ss | % | € | ◷hour | km | € | % | % | € | # | % | % |
| 1 | Base experiment | 7:13:14 | 12% | € 517,861 | 7941 | 154,672 | € 65.21 | 18.89% | 18.24% | € 89,726 | 887 | 60.8% | 19.8% |
| 2 | 100% Resource swap | **5:42:32** | 1% | € 583,375 | 8525 | 188,672 | € 68.43 | 33.50% | 33.84% | € 92,011 | 922 | **58.5%** | 22.1% |
| 3 | 100% Trip swap | 12:20:28 | 2% | € 574,787 | 8368 | 180,819 | € 68.69 | 30.62% | 30.50% | **€ 89,244** | 922 | 57.4% | **22.2%** |
| 4 | 100% Trip move | 6:24:53 | **9%** | **€ 534,738** | **7982** | **155,766** | **€ 67.00** | **19.46%** | **19.06%** | € 100,966 | 925 | 57.5% | 20.0% |
| 5 | Resource swap/Trip swap | 10:07:33 | 4% | € 568,164 | 8399 | 183,843 | € 67.65 | 31.76% | 31.91% | **€ 85,514** | 922 | 59.3% | **22.0%** |
| 6 | Trip swap/Trip move | 10:42:53 | **11%** | € 525,540 | **7931** | **153,248** | € 66.26 | **18.13%** | **17.63%** | € 95,782 | 910 | 58.1% | 19.6% |
| 7 | Resource swap/Trip move | **6:31:12** | **11%** | **€ 525,013** | 7967 | 154,972 | **€ 65.90** | 19.04% | 18.55% | € 95,950 | 893 | **60.3%** | 18.9% |
| 8 | More new groups | **6:26:46** | **13%** | **€ 512,585** | **7855** | **149,965** | **€ 65.25** | **16.34%** | **15.71%** | € 94,409 | 919 | **61.0%** | 19.7% |
| 9 | Less new groups | 12:50:04 | 7% | € 546,271 | 8264 | 173,549 | € 66.10 | 27.71% | 27.50% | **€ 80,131** | 849 | 60.2% | **21.3%** |
| 10 | 100% N=1 | **9:38:30** | **13%** | **€ 515,627** | **7935** | **154,284** | **€ 64.98** | **18.68%** | **18.08%** | **€ 87,286** | 879 | **61.4%** | 19.8% |
| 11 | 100% N=2 | 12:20:48 | 10% | € 528,490 | 8011 | 157,365 | € 65.97 | 20.28% | 19.78% | € 94,668 | 891 | 60.0% | **20.5%** |
| 12 | 100% N=3 | 13:13:56 | 10% | € 528,942 | 8013 | 157,147 | € 66.01 | 20.17% | 19.62% | € 94,965 | 896 | 60.0% | 19.7% |
| 13 | More similar swaps | **11:41:44** | **12%** | € 519,807 | 7943 | 154,660 | € 65.45 | 18.88% | 18.21% | € 91,868 | 890 | 60.6% | **20.1%** |
| 14 | More random swaps | 12:18:28 | **12%** | **€ 516,600** | **7936** | **154,067** | **€ 65.09** | **18.57%** | **17.93%** | **€ 88,685** | 891 | **60.9%** | 19.7% |
| 15 | More vehicle swaps | 12:16:52 | **12%** | € 517,813 | 7945 | 154,497 | **€ 65.18** | 18.80% | 18.23% | **€ 89,072** | 885 | **60.9%** | 19.7% |
| 16 | More driver swaps | **12:10:33** | **12%** | **€ 517,119** | **7930** | **154,040** | € 65.21 | **18.55%** | **17.89%** | € 89,772 | 889 | 60.6% | **19.9%** |
| 17 | N=1/N=2 | **8:42:33** | **12%** | **€ 515,941** | **7932** | **153,738** | **€ 65.05** | **18.40%** | **17.75%** | **€ 89,149** | 888 | **61.0%** | 19.0% |
| 18 | N=2/N=3 | 10:07:06 | 10% | € 527,647 | 7996 | 156,815 | € 65.99 | 20.00% | 19.50% | € 94,908 | 893 | 60.0% | **19.6%** |
| 19 | N=1/N=3 | 10:07:34 | **12%** | € 518,611 | 7940 | 154,468 | € 65.31 | 18.78% | 18.33% | € 90,285 | 885 | 60.9% | **19.6%** |

**(b)** Results of Stage 1 (best values bold)

**Table 6.8:** Settings and results for Stage 1: 'improvement-operator settings'

| # | Description | Day | Runtime | Total-cost | Total-duration | Total-distance | Cost-per-hour | Pre/returntrip-distance | Pre/returntrip-duration | Outsourcing-cost | Resource-groups | Driver-employment | Vehicle-employment |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | ◷hh:mm:ss | € | ◷hour | km | € | % | % | € | # | % | % |
| I | Agg. Local (Cost A) | Day 1 | 00:02:57 | € 589,404 | 8525 | 188,672 | € 69.14 | 33.50% | 13.54% | € 96,026 | 922 | 57.1% | 22.0% |
| II | Agg. Regional (Cost A) | Day 1 | **00:00:06** | € 585,530 | **8196** | **182,387** | € 71.44 | **31.21%** | **12.42%** | € 112,347 | 959 | 57.8% | **35.8%** |
| III | Agg. National (Cost A) | Day 1 | 00:41:04 | **€ 576,147** | 8489 | 185,474 | **€ 67.87** | 32.36% | 13.18% | **€ 88,835** | 900 | **59.2%** | 20.6% |
| I | Agg. Local (Cost A) | Day 2 | 00:03:35 | € 645,296 | 9198 | 204,740 | € 70.15 | 33.04% | 13.20% | € 113,114 | 950 | **58.2%** | 27.3% |
| II | Agg. Regional (Cost A) | Day 2 | **00:00:07** | € 655,005 | **8977** | **192,959** | € 72.97 | **28.95%** | **10.81%** | € 142,327 | 1006 | 53.6% | **36.2%** |
| III | Agg. National (Cost A) | Day 2 | 00:43:09 | **€ 635,684** | 9128 | 203,022 | **€ 69.64** | 32.47% | 13.32% | **€ 107,854** | 925 | 57.6% | 24.9% |
| I | Agg. Local (Cost A) | Day 3 | 00:01:06 | € 512,721 | 7084 | 164,360 | € 72.37 | 30.88% | 12.99% | € 95,659 | 730 | **63.1%** | 22.4% |
| II | Agg. Regional (Cost A) | Day 3 | **00:00:02** | € 521,344 | **6910** | **153,103** | € 75.45 | **25.79%** | **9.69%** | € 121,426 | 763 | 56.7% | **30.4%** |
| III | Agg. National (Cost A) | Day 3 | 00:17:08 | **€ 505,633** | 7050 | 162,305 | **€ 71.72** | 30.00% | 12.66% | **€ 91,780** | 725 | 62.0% | 20.0% |

**Table 6.9:** Results of Stage 2: 'construction only' (best values bold).

- The results are comparable for all three days: the best values are for most KPIs (all except *driver-employment* percentage) reached at equal aggregation levels (e.g., all shortest *runtimes* are retrieved at the *Regional* level). This indicates that the three days share the same conclusions.
- The lowest *total-costs*, *cost-per-hour*, and *outsourcing-cost* are reached at the *National* aggregation level. The *Regional* aggregation level has the highest costs. The *total-costs* at the *National* aggregation level is on average 1.7% lower than the *total-costs* at the *Local* aggregation level and even 2.5% lower than at the *Regional* level.
- We hypothesized that the *pre/returntrip-distance* and *-duration* would decrease at higher aggregation levels. However, we obtain somewhat counter-intuitive results: the lowest *pre/returntrip-values* are reached at the *Regional* level. Also the *total-distance* and *total-duration* are the lowest at the *Regional* level. These low *Pre/returntrip-values* at the *Regional* level are (logically) accompanied by the highest *outsourcing-costs* (*Regional* level 23% higher than the *Local* level and 30% higher than the *National* level).
- We compare the differences between the *driver-employment* and *vehicle-employment* percentages at different aggregation levels. We argued before that higher *resource-employment* percentages reflect a superior outcome. However, we now have to refine that argument. Higher values are preferable only if this can be obtained by lower *outsourcing-costs*. On the other hand, high values might also be the result of inefficient use of *resources* (e.g., by longer *pre-/return-trips*): in that case, lower values are better when they are the result of a more efficient use of *resources*. We clearly see the difference between the *driver-employment* and *vehicle-employment* percentages. The *driver-employment* percentage has the best (*Day 1*) or second best (*Day 2* and *Day 3*) results in relation to the lowest *outsourcing-costs* (on average 60% and € 96,156 respectively), compared to highest *outsourcing-cost* and lowest *driver-employment* (*Regional*; 56% and € 125,367 respectively), from which we conclude that scheduling at the *National* aggregation level results in less outsourcing of *drivers*, and therefore a higher *driver-employment* percentage. On the other hand, we see that the *vehicle-employment* percentage and *outsourcing-costs* are high at the *Regional* aggregation level (on average 49% higher *vehicle-employment* and 26% higher *outsourcing-cost* than at other levels). At the *Regional* level, the *pre/returntrip-values* as well as the *total-duration* and *total-distance* are relatively small compared to the other levels. Therefore, we conclude that *vehicles* are employed very inefficient at the *Regional* level.

### 6.4.3 Stage 3: 'production runs'

Using the results from Section 6.4.1, we now conduct our production runs. Each experiment is again replicated 4 times, the aggregation level is varried, we use the datasets from *Day 1*, *Day 2*, and *Day 3* as our input. In this stage, analyze the results of our *improvement-heuristic* and we compare the results of the 'construction' only experiments (Section 6.4.2)

with the combination of *construction-* and *improvement-heuristics*. We use the *Full* Simulated Annealing settings and cost *Set A* (experiment 20-22; settings in Table 6.10a).

**Results**

We discuss the results from experiment 20-22 as presented in Table 6.10b. The total *runtime* of all experiments is 1714 hour (distributed over multiple processing units).

- The first thing we point out are the very long *runtimes*. In most cases, the *runtimes* are longer than the 24 hour *planning-period* as defined in Section 4.8. Because new *orders* arrive every day and scheduling is currently done according to the 'today we schedule tomorrow' principle, *runtimes* longer than a day are unrealistic. Furthermore, the *runtimes* become longer for higher aggregation levels.

- The results are comparable for all three days: the best values are in all cases reached at equal aggregation levels (e.g., all shortest *total-distances* are retrieved at the *Local* level). This indicates that the three days can share conclusions.

- Taking a look at the results per aggregation level, something that seems counter-intuitive is detected. The *total-costs*, *costs-per-hour*, and *outsourcing-cost* when scheduling are the best at the *Regional* level, compared to the *Local* and *National* aggregation levels (in both cases on average 2.3% lower), although we expected increasingly better results at both the *Regional* and *National* aggregation levels. At *Day 1*, the *total-costs* at the *National* level are (unexpectedly) 2.4% higher than at the *Local* aggregation level. We presume that this might be due to the fact that our *improvement-heuristic* parameters are chosen based on the *Regional* level: the algorithm might not be completely finished (although few improvements are found at the end for all aggregation levels) and thus might require longer *runtimes*, which can be acquired by increasing the values of $markov_{length}$, $T_{decrease}$, or both.

- We hypothesized that the *pre/return-distance* and *-duration* would decrease at higher aggregation levels. However, we obtain counter-intuitive results: when the algorithm is able to combine *trips* of several *departments* or even *regions*, the *pre/returntrip-values* increases (an average increase of 1.6% percent point per aggregation level). This outcome might be explained by a simple logical argument: the *total-distance* and *total-duration* increase as well. The decrease in *total-costs* is expected to be the result from decreased *outsourcing costs*.

- The *Regional* aggregation level provides the best results, based on *total-cost*, *cost-per-hour*, *outsourcing-cost*, *improvement-percentage*, and *driver-employment*. The relative average improvement percentage is 2.3% for the *Regional* aggregation level, compared tot the *Local* aggregation level.

- We compare the results for the *driver-employment* and *vehicle-employment* percentages as well. We clearly see the difference between the *driver-employment* and *vehicle-employment* percentages. The *driver-employment* percentage shares its best results with the *outsourcing-costs* (on average 64% and € 99,376 respectively), compared to other levels (on average 61% and € 113,296 respectively), from which we conclude that the *Regional* aggregation level result in less outsourcing, and therefore a higher *driver-employment* percentage. On the other hand, we see that the *vehicle-employment* percentage drops on higher aggregation levels (31% decrease from *Local* to *Regional* and another 14% decrease from *Regional* to *National*), from which we conclude that *vehicles* are probably more efficient deployed on higher aggregation levels. Clearly, there is a trade-off between outsourcing and the use of *Own resources*. We conclude that this trade-off leads to different results at different aggregation levels.

| # | Description | Datasets | Agg. Lvl. | Sim. An. | Cost Set | Operator type | | | New/Existing | | Sequence length | | | Similar/Random | | Vehicle/Driver | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | TRIP-MOVE- | | TRIP-SWAP- | | | TRIP-SWAP-N1- | | RESOURCE-SWAP- | |
| | | | | | | RESOURCE-SWAP | TRIP-SWAP | TRIP-MOVE | NEW | EXISTING | N1 | N2 | N3 | SIMILAR | RANDOM | VEHICLE | DRIVER |
| 20 | Agg. Local (Full/Cost A) | Day 1,2,3 | L | F | A | 1/4 | 1/4 | 1/2 | 1% | 99% | 4/6 | 1/6 | 1/6 | 2/5 | 3/5 | 2/5 | 3/5 |
| 21 | Agg. Regional (Full/Cost A) | Day 1,2,3 | R | F | A | 1/4 | 1/4 | 1/2 | 1% | 99% | 4/6 | 1/6 | 1/6 | 2/5 | 3/5 | 2/5 | 3/5 |
| 22 | Agg. National (Full/Cost A) | Day 1,2,3 | N | F | A | 1/4 | 1/4 | 1/2 | 1% | 99% | 4/6 | 1/6 | 1/6 | 2/5 | 3/5 | 2/5 | 3/5 |

**(a)** Settings of Stage 3

| # | Description | Day | Runtime | Improvement-percentage | Total-cost | Total-duration | Total-distance | Cost-per-hour | Pre/returntrip-distance | Pre/returntrip-duration | Outsourcing-cost | Resource-groups | Driver-employment | Vehicle-employment |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | ⏲hh:mm:ss | % | € | ⏲hour | km | € | % | % | € | # | % | % |
| 20 | Agg. Local (Full/Cost A) | Day 1 | **24:49:09** | 13% | € 507,337 | **7532** | **143,369** | € 67.36 | **12.49%** | **4.88%** | € 106,431 | 957 | 62.0% | **29.9%** |
| 21 | Agg. Regional (Full/Cost A) | Day 1 | 63:02:48 | **14%** | **€ 505,439** | 7807 | 147,740 | **€ 64.74** | 15.03% | 5.87% | **€ 91,854** | 923 | **62.7%** | 19.0% |
| 22 | Agg. National (Full/Cost A) | Day 1 | 58:28:39 | 10% | € 519,642 | 7870 | 152,060 | € 66.03 | 17.49% | 6.84% | € 101,028 | 936 | 60.4% | 16.4% |
| 20 | Agg. Local (Full/Cost A) | Day 2 | **29:56:15** | 12% | € 577,666 | **8363** | **155,778** | € 69.07 | **11.99%** | **4.16%** | € 134,459 | 1003 | 58.4% | **31.1%** |
| 21 | Agg. Regional (Full/Cost A) | Day 2 | 48:58:47 | **13%** | **€ 558,248** | 8444 | 160,960 | **€ 66.11** | 14.82% | 5.66% | **€ 109,651** | 960 | **61.5%** | 22.9% |
| 22 | Agg. National (Full/Cost A) | Day 2 | 84:42:24 | 10% | € 573,288 | 8497 | 164,035 | € 67.47 | 16.42% | 6.35% | € 121,300 | 977 | 59.1% | 19.7% |
| 20 | Agg. Local (Full/Cost A) | Day 3 | **23:46:14** | 11% | € 463,440 | **6463** | **125,711** | € 71.71 | **9.62%** | **3.54%** | € 114,837 | 749 | 62.3% | **26.7%** |
| 21 | Agg. Regional (Full/Cost A) | Day 3 | 37:16:43 | **12%** | **€ 44,012** | 6527 | 129,623 | **€ 68.79** | 12.35% | 4.80% | **€ 96,625** | 724 | **67.0%** | 18.5% |
| 22 | Agg. National (Full/Cost A) | Day 3 | 57:31:06 | 10% | € 455,987 | 6562 | 131,461 | € 69.49 | 13.58% | 5.36% | € 101,721 | 733 | 65.6% | 15.8% |

**(b)** Results of Stage 3 (best values bold)

**Table 6.10:** Settings and results for Stage 3: 'production runs'

Comparing the results from the current experiments with the results from the 'construction only' experiments in Section 6.4.2, we see a few differences:

- The *runtimes* exploded (867 times longer) from several minutes/seconds to several days/hours.
- The *improvement-heuristic* works best for the *Regional* aggregation level: the improvement is so good that the best aggregation level changes: before improvement, the *National* aggregation level yields the lowest *total-costs*, after improvement this role is for the *Regional* aggregation level.
- Great improvements are made in the *pre/returntrip-values*: we measure a decrease of 55% and 58% on *distance* and *duration* respectively between construction and improvement.
- Comparing the results of the *construction-heuristic* and *improvement-heuristic*, the *outsourcing-costs* increase by 0.9% on average only, while the *total-costs* decrease by an average of 12% when applying the *construction-heuristic* and *improvement-heuristic*.
- The *improvement-heuristic* increases the *driver-employment* by an average of 6%, while the *vehicle-employment* is decreased by 17%.

From this analysis we conclude that the *improvement-heuristic* does its job, however at the price of a very long *runtimes*. Furthermore, we conclude that the *improvement-heuristic* improves both the routing and assignment aspects: there is are large decreases in overhead (*pre/returntrip-values*) and we can see that the composition of *resource-groups* changes due to the changes in *resource-employment* and *outsourcing-cost*.

### 6.4.4 Stage 4: 'sensitivity analysis'

We continue by executing experiments for the determination of the sensitivity for the rent and charter factors respectively, while varying the aggregation level at the same time, using the *Full* Simulated Annealing settings and cost *Set B* (experiment 23-25; settings in Table 6.11a).

**Results**

We discussed the insights obtained from experimenting with various aggregation levels. The next step is to repeat the experiments with another cost set (*Set B*), in order to test the sensitivity for the cost factors (experiment 23-25). The total *runtime* of these experiments adds up to 1912 hour. We show the results in Table 6.11b, and give a short analysis below:

| # | Description | Datasets | Agg. Lvl. | Sim. An. | Cost Set | Operator type | | | New/Existing | | Sequence length | | | Similar/Random | | Vehicle/Driver | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | RESOURCE-SWAP | TRIP-SWAP | TRIP-MOVE | TRIP-MOVE-NEW | TRIP-MOVE-EXISTING | TRIP-SWAP-N1 | TRIP-SWAP-N2 | TRIP-SWAP-N3 | TRIP-SWAP-N1-SIMILAR | TRIP-SWAP-N1-RANDOM | RESOURCE-SWAP-VEHICLE | RESOURCE-SWAP-DRIVER |
| 23 | Cost B Local (Full) | Day 1,2,3 | L | F | B | 1/4 | 1/4 | 1/2 | 1% | 99% | 4/6 | 1/6 | 1/6 | 2/5 | 3/5 | 2/5 | 3/5 |
| 24 | Cost B Regional (Full) | Day 1,2,3 | R | F | B | 1/4 | 1/4 | 1/2 | 1% | 99% | 4/6 | 1/6 | 1/6 | 2/5 | 3/5 | 2/5 | 3/5 |
| 25 | Cost B National (Full) | Day 1,2,3 | N | F | B | 1/4 | 1/4 | 1/2 | 1% | 99% | 4/6 | 1/6 | 1/6 | 2/5 | 3/5 | 2/5 | 3/5 |

**(a)** Settings of Stage 4

| # | Description | Day | Runtime | Improvement-percentage | Total-cost | Total-duration | Total-distance | Cost-per-hour | Pre/returntrip-distance | Pre/returntrip-duration | Outsourcing-cost | Resource-groups | Driver-employment | Vehicle-employment |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | ◷hh:mm:ss | % | € | ◷hour | km | € | % | % | € | # | % | % |
| 23 | Cost B Local (Full) | Day 1 | **28:36:47** | 14% | € 678,890 | **7727** | 154,576 | € 87.86 | 18.84% | **7.05%** | € 102,356 | 925 | **64.1%** | **32.4%** |
| 24 | Cost B Regional (Full) | Day 1 | 49:16:30 | **15%** | **€ 576,214** | 7922 | **154,356** | **€ 72.73** | **18.72%** | 7.34% | **€ 91,343** | 886 | 63.8% | 20.6% |
| 25 | Cost B National (Full) | Day 1 | 109:48:36 | 7% | € 684,770 | 8227 | 171,804 | € 83.24 | 26.98% | 10.92% | € 91,775 | 887 | 63.4% | 15.4% |
| 23 | Cost B Local (Full) | Day 2 | **15:42:05** | 12% | € 805,530 | **8567** | **168,132** | € 94.03 | **18.46%** | **6.53%** | € 132,185 | 968 | 60.6% | **33.6%** |
| 24 | Cost B Regional (Full) | Day 2 | 56:20:17 | **15%** | **€ 724,229** | 8740 | 178,558 | **€ 82.86** | 23.22% | 9.13% | **€ 99,600** | 907 | **64.4%** | 26.2% |
| 25 | Cost B National (Full) | Day 2 | 85:43:28 | 7% | € 771,104 | 8859 | 185,921 | € 87.04 | 26.26% | 10.39% | € 111,114 | 920 | 61.9% | 22.8% |
| 23 | Cost B Local (Full) | Day 3 | **13:18:52** | 11% | € 660,430 | **6609** | **135,056** | € 99.93 | **15.88%** | **5.84%** | € 116,485 | 734 | 64.1% | **29.2%** |
| 24 | Cost B Regional (Full) | Day 3 | 40:07:44 | **13%** | **€ 592,813** | 6772 | 144,936 | **€ 87.53** | 21.61% | 8.62% | **€ 88,278** | 682 | **69.3%** | 21.2% |
| 25 | Cost B National (Full) | Day 3 | 79:14:25 | 9% | € 609,122 | 6840 | 148,648 | € 89.05 | 23.57% | 9.58% | € 91,641 | 686 | 68.5% | 18.6% |

**(b)** Results of Stage 4(best values bold)

**Table 6.11:** Settings and results for Stage 4: 'sensitivity analysis'

- These experiments require even longer *runtimes* than the previous experiments. Apparently, a more diverse set of costs makes it harder for our algorithms to make a decision between *Rented*, *Own*, and *Chartered resource-groups*.
- Furthermore, we see similar, yet more extreme (higher values and larger differences) results as in experiment 20-22 (as expected). Although it is meaningless to compare individual values (there are different cost factors after all), we perform a similar analysis for experiment 23-25 as for experiment 20-22, from which we conclude that the same conclusions apply.

Experiment 23-25 produce approximately similar results as experiment 20-22, besides the increase in *runtimes*. Because the results and conclusions are similar, we state that results appear to be quit insensitive for the difference in cost factors.

### 6.4.5 Stage 5: 'constrained improvement'

Finally, we examine the differences between using the *Full* and *Constrained* Simulated Annealing settings, while varying the aggregation Level and using cost *Set A* (experiment 26-28; settings in Table 6.12a).

**Results**

Finally, we repeat our experiments with constrained settings for the Simulated Annealing algorithm. The main reason to do this is to investigate the results for shorter runtimes. The total *runtime* of these experiments is 380 hour. We give an overview of our results in Table 6.12b, and draw the following three conclusions:

- The *runtimes* for experiments 26-28 are much shorter than for our previous experiments (78% shorter than in 'production runs'). Most improvements are made on higher aggregation levels. Looking at the *total costs* (+0.3%), *costs-per-hour* (+0.04%), and *outsourcing-costs* (-1.2%), we see that the results are almost equal compared to the 'production runs'.
- When comparing the differences between aggregation levels and days between the current experiments and the 'production runs', we see almost no differences. Therefore, we draw the same conclusions as for experiments 20-25.
- Although we used different settings for the Simulated Annealing algorithm, the *National* aggregation level still yields the worst results. Our earlier statement that 'the *National* aggregation level yields the worst results because of improper Simulated Annealing settings' should now be doubted, because different settings give similar results.

| # | Description | Datasets | Agg. Lvl. | Sim. An. | Cost Set | Operator type | | | New/Existing | | Sequence length | | | Similar/Random | | Vehicle/Driver | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | RESOURCE-SWAP | TRIP-SWAP | TRIP-MOVE | TRIP-MOVE-NEW | TRIP-MOVE-EXISTING | TRIP-SWAP-N1 | TRIP-SWAP-N2 | TRIP-SWAP-N3 | TRIP-SWAP-N1-SIMILAR | TRIP-SWAP-N1-RANDOM | RESOURCE-SWAP-VEHICLE | RESOURCE-SWAP-DRIVER |
| 26 | Constrained Local (Cost A) | Day 1,2,3 | L | C | A | 1/4 | 1/4 | 1/2 | 1% | 99% | 4/6 | 1/6 | 1/6 | 2/5 | 3/5 | 2/5 | 3/5 |
| 27 | Constrained Regional (Cost A) | Day 1,2,3 | R | C | A | 1/4 | 1/4 | 1/2 | 1% | 99% | 4/6 | 1/6 | 1/6 | 2/5 | 3/5 | 2/5 | 3/5 |
| 28 | Constrained National (Cost A) | Day 1,2,3 | N | C | A | 1/4 | 1/4 | 1/2 | 1% | 99% | 4/6 | 1/6 | 1/6 | 2/5 | 3/5 | 2/5 | 3/5 |

**(a)** Settings of Stage 5

| # | Description | Day | Runtime | Improvement-percentage | Total-cost | Total-duration | Total-distance | Cost-per-hour | Pre/returntrip-distance | Pre/returntrip-duration | Outsourcing-cost | Resource-groups | Driver-employment | Vehicle-employment |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | hh:mm:ss | % | € | hour | km | € | % | % | € | # | % | % |
| 26 | Constrained Local (Cost A) | Day 1 | 10:53:53 | **14%** | **€ 505,020** | **7524** | **142,738** | € 67.12 | **12.11%** | **4.70%** | € 104,337 | 964 | **61.9%** | **30.5%** |
| 27 | Constrained Regional (Cost A) | Day 1 | **9:50:35** | **14%** | € 508,951 | 7846 | 148,846 | **€ 64.86** | 15.71% | 6.05% | **€ 92,352** | 927 | 61.6% | 19.8% |
| 28 | Constrained National (Cost A) | Day 1 | 14:12:03 | 10% | € 521,309 | 7910 | 153,109 | € 65.90 | 18.06% | 7.12% | € 97,882 | 941 | 59.8% | 17.0% |
| 26 | Constrained Local (Cost A) | Day 2 | **9:07:00** | 12% | € 576,625 | **8365** | **156,095** | € 68.93 | **12.17%** | **4.24%** | € 133,049 | 1001 | 58.6% | **31.4%** |
| 27 | Constrained Regional (Cost A) | Day 2 | 9:28:07 | **13%** | **€ 563,546** | 8479 | 162,851 | **€ 66.46** | 15.81% | 6.10% | **€ 110,723** | 961 | **60.8%** | 23.0% |
| 28 | Constrained National (Cost A) | Day 2 | 16:09:12 | 9% | € 577,944 | 8531 | 166,170 | € 67.75 | 17.49% | 6.79% | € 119,558 | 975 | 58.0% | 20.0% |
| 26 | Constrained Local (Cost A) | Day 3 | 7:33:32 | 11% | € 462,306 | **6456** | **125,641** | € 71.61 | **9.57%** | **3.56%** | € 113,974 | 754 | 62.3% | **26.9%** |
| 27 | Constrained Regional (Cost A) | Day 3 | **7:28:14** | **12%** | **€ 449,380** | 6538 | 130,297 | **€ 68.73** | 12.81% | 5.06% | **€ 95,035** | 721 | **66.1%** | 18.8% |
| 28 | Constrained National (Cost A) | Day 3 | 10:22:51 | 9% | € 459,263 | 6593 | 133,182 | € 69.66 | 14.69% | 5.83% | € 99,749 | 730 | 63.7% | 16.4% |

**(b)** Results of Stage 5 (best values bold)

**Table 6.12:** Settings and results for Stage 5: 'constrained improvement'

We conclude that using a constrained set of parameters for our Simulated Annealing algorithm (for these experiments) does not necessarily produce worse results, despite the 78% decrease in *runtime*. We expect that the shorter (more realistic) *runtimes* might even counterbalance small deterioration of results.

## 6.5 Conclusions

This chapter focused on the empirical part of this research. We used the input data provided by PAT, consisting of historical data as discussed in Chapter 3, in a (semi) random sample of *trip* assignments for three days in January 2016. These input data unfortunately contains a considerable amount of errors, limiting the validity of conclusions drawn from our experiments in advance. Nevertheless, the experiments conducted with our imperfect input data allow for some preliminary and tentative conclusions.

We continued the chapter with our experimental phase, which is divided into five stages: the 'operator experiments', 'construction only', 'production runs', 'sensitivity analysis', and 'constrained improvement'. We conducted our last three experiments based on the results of the 'operator experiments'. From the 'construction only' experiments we concluded that the *total-cost* decreases at higher aggregation levels: a change from *Local* to *Regional* scheduling yields a decrease of 1.7% in *total-cost*, a shift from *Regional* to *National* yields another 2.5% decrease.

The 'production runs' showed us different results: now the *Regional* aggregation level yields the best results (-2.3% compared to *Local*). Surprisingly, the worst results were retrieved at the *National* level (+2.4% compared to *Regional*). However, it should be noted that this might be an artifact of improper Simulated Annealing parameters for the *National* aggregation level (although not supported by the repetition of results within the 'constrained improvement' experiments using different settings). Looking at the *total-cost*, the *improvement-heuristic* decreases the *total-cost* by an average of 12%, while improving both the routing and scheduling aspects of the *trip-assignment-problem*. Despite these improvements, the combined *runtimes* of the heuristics became extremely long (867 times longer for the combination of heuristics, compared to the *construction-heuristic* only).

Furthermore, we concluded that the increase in cost factors for renting and chartering increases the *runtime* in particular, however results in similar conclusion as our initial cost set, meaning that the solution is insensitive for changes in the cost parameters. Finally, we concluded that a constrained set of Simulated Annealing parameters does not benefit the outcomes in terms of lower *total-costs* (+0.3%), although the corresponding shorter and more realistic *runtimes* (4.5 times shorter) outweigh small deteriorations of results.

# Chapter 7

# Conclusions & discussion

In this final chapter we put the different facets of our project together and summarize our conclusions. We reflect on our goal and research questions by presenting conclusions in Section 7.1, followed by the discussion in Section 7.2.

## 7.1 Conclusions

We will now reflect on the research questions that were derived from our research goal in Section 1.3. We answer the research questions one by one, followed by a reflection and conclusions on the research goal:

*Develop an algorithm for solving the trip-assignment-problem at PAT.*

We answer the following research questions to achieve the research goal:

**Which methods can be used to solve the trip-assignment-problem according to literature? (RQ 1)**

We attempted to model our *trip-assignment-problem* after comparable problems in the literature (see Chapter 2). The Vehicle Routing Problem, the Pick-up and Delivery Problem and the multi-Traveling Salesman Problem contain aspects similar to aspects of the *trip-assignment-problem*. We found that our algorithms need to focus on the assignment of *trips* to *resources* before considering the routing aspect, because a large part of the routing aspect (within *trips*) is given up front. We compared the size of the *trip-assignment-problem* with the size of several benchmark problems and concluded that our PAT case is relatively large. Looking at the methods that we might use to solve the *trip-assignment-problem*, we concluded that *metaheuristics* and especially Simulated Annealing were the most likely to solve the problem.

**Which data is available for solving the trip-assignment-problem, and what is the current scheduling performance of PAT? (RQ 2)**

Our data for the PAT case are obtained from the RGB$^+$ Automatisering Transplan TMS and other software systems at PAT. We distinguished three major components: (*1*) *vehicles* (fleet); (*2*) *drivers* (employees); and (*3*) *orders* (*trips*). In Chapter 3 we described the characteristics and structure of the data. We found several *vehicle-(combination)-types*, *vehicle-properties*, *drivers-licenses* and *trip-types* being relevant for our research. We are aware that one might take other properties of *resources* or *orders* into consideration. However, documentation and data availability on other then the selected characteristics turned out to be scarce. Furthermore, we described the *working-hour-regulations* applicable to road transport. We also examined the performance of PAT as far as possible and establish some interesting results: (*1*) *drivers* take breaks in chunks of almost exactly 15 minutes; (*2*) charter companies are actually employed when needed the most; and (*3*) there are plenty of *resources* available, compared to the amount of employment.

**How can the trip-assignment-problem be solved using a limited time optimization algorithm? (RQ 3)**

In Chapter 4 we develop a model to describe the *trip-assignment-problem*. We simplify and structure the Peter Appel Transport case by limiting the number of constraints involved and using simple mechanisms for the calculation of costs and *working-hour-regulations*. We introduce the principle of *resource-groups* and define the input data required, as well as the output desired.

In Chapter 5 we developed an algorithm that solves our model using both a *construction-* and an *improvement-heuristic*. We used a parallel scheduling method for the *construction-heuristic*, and Simulated Annealing for the *improvement-heuristic*. We developed the Trip Assignment Solver tool using C# as the programming language, implementing our algorithm as described before. We used the software to validate our algorithms (see Section 6.3) and experimented with different algorithm settings (see Section 6.4.1).

**What are the consequences of implementing the automated scheduling? (RQ 4)**

We used data of PAT, stored in the RGB$^+$ Automatisering Transplan TMS as the input of our experiments. We concluded that the data is pretty dirty and we had to make a lot of assumptions and adaptations to circumvent the dirty parts, resulting in input data that depends on historically manually created schedules. In other words: our input data cannot exist before human planners created the schedules manually, resulting in a correlation between manually created and software generated schedules. Due to this relationship, we can impossibly compare the performance of our algorithms with the performance of the human planners. Before implementation is possible, many data related improvements are required (see Section 7.2.2).

We were, however, able to compare automatically created schedules on several aggregation levels (*Local*, *Regional*, and *National*), using both the *construction-* and *improvement-heuristics*. Based on the *construction-heuristic* only, we concluded that the best results were retrieved at the *National* aggregation level, showing average decreases of 1.7% (*Local* to *Regional*) and 2.5% (*Regional* to *National*) in *total-cost*. Our *improvement-heuristic* yielded an average decrease in *total-cost* of 12% compared to the *construction-heuristic*, although the *runtime* was multiplied 867 times. We concluded that our *improvement-heuristic* gives the best results at the *Regional* level, and the worst results at the *National* level (noting that the chosen settings might not be fully applicable to the *National* aggregation level). We furthermore concluded that the settings for the cost parameters were not decisive for the conclusions on the best aggregation level. Moreover, we conclude that the *improvement-heuristic* results in very long and unrealistic *runtimes* on higher aggregation levels (over 24 hours or multiple days of *runtime* for a single schedule). We experimented with constrained parameters leading to only the final part of the Simulated Annealing algorithm (similar to a local search), resulting in much shorter *runtimes* (4.5 times shorter) and only negligibly worse results (+0.3% in *total-cost*).

The main benefit of our research is that we were able to develop the *trip-assignment-model*, as well as a first step in the development of the required algorithms for solving the model. We thus show that the problem can be solved automatically. Again, we are unfortunately unable to compare the results of the automated scheduling to the manually created schedules. However, despite the potential (untested) gain of efficiency, being able to schedule some, or a large part, of the *trips* automatically, instead of manually, can be an improvement in itself. Our solution might be used as a decision support system, using algorithms for the construction of a base planning and repetitive tasks, enabling the planners to focus on the challenging parts of planning.

## 7.2 Discussion

In Section 7.2.1, we discuss the limitations of our model and solution. Furthermore, we give recommendations for overcoming of these limitations in Section 7.2.2. Finally, we give suggestions for future research in Section 7.2.3.

### 7.2.1 Limitations

In this section we discuss the limitations of the present research. First, we discuss the limitations regarding the model given in Chapter 4:

- In practice, there are many more *resource*-related constraints that are not accounted for in our model, related to properties such as: (*1*) *driver* certificates (related to food safety or hazardous goods); (*2*) knowledge of specific customers of certain locations or customers; (*3*) preference of *drivers* to be scheduled on the same *vehicle* as often as possible; (*4*) the full set of *working-hour-regulations* (see Section 3.5) (*5*) the existence of BE-type-vehicles (small vans); (*6*) environmental comparability of *vehicles* (related to low emission zones); (*7*) height of *vehicles*; (*8*) specific compartmentalization of *vehicles*; (*9*) *vehicle* tailgates (see Appendix D); (*10*) the capacity of *vehicles*; and different other properties.

- There are many more *trip*-related properties in practice, such as: (*1*) the required volume to be transported; (*2*) a proper registration of *time-windows*; and (*3*) the required *cooling* level.

- In practice, there are dependencies among *trips*, which are currently not properly documented in the Transplan TMS, but are crucial for the scheduling of these *trips*. Examples are: (*1*) a split *trip*: preloading on the present day, while driving on the next day, logically using the same *vehicle*; (*2*) combined LHV: an LHV is driving a long distance from a distribution center to the outskirts of a city, then split into two regular (non-LHV) *vehicle-combinations* to distribute the goods within the inner city using two *trips*; and many more.

- As stated before, the structure we use to calculate the cost of our solutions is overly simplistic, compared to reality. In practice, there are differences between *drivers*, *vehicles*, *Rented resources*, and charters. Even the choice of a specific charter company might influence the cost of the solution.

- We do not take the availability of charters and *Rented resources* into account.

- We ignore the fact that *Own resources* also raise costs when they are not employed (*vehicles* are depreciated and *drivers* might have a permanent labor agreement).

- Secondary objectives are not taken into account. Examples are: (*1*) striving for an even distribution of shift length between *drivers*; (*2*) *vehicle*-preferences of *drivers* (employee satisfaction); and many more.

In most cases, the limitations of the model are the result of missing, faulty, dirty, or inconsequently used data (we keep out constraints because of missing information). Unfortunately, we are confronted with the (resolvable) limitations of the input data, which we describe below:

- Breaks and rests are not property documented, and rests taken before the start of the *planning-period* can therefore not be taken into account. We simply assume that *drivers* finished a *Daily-rest* at the beginning of their *availability-period*.

- Maximum of one *driver* per *trip*, which is true for most cases, however not all.

- Maximum of two *vehicles* per *trip*, therefore, the Transplan TMS is unable to cope with LHVs, resulting in a complex workaround for the determination of requested LHVs.

- *Orders* (as currently captured in the Transplan TMS) do not contain any information regarding *requested-vehicle-type* (or volume to be transported), and the required *cooling*. Currently, we deduce this data based on historically manually created schedules. Consequently, we (currently) require the manual scheduling to be completed before automated scheduling can be applied.

- *Time-window* registration is improperly and inconsequently. We found cases where the *time-windows* gathered from the Transplan TMS do not comply with the *time-windows* given by the customers of PAT.

- *Resource-availability* information is missing for most *resources*.

There are also some limitations regarding our solution (heuristics and Trip Assignment Solver), which are the following:

- Break scheduling looks only one *activity* ahead, resulting in cases where a *Daily-rest* is scheduled, before only the final *return-trip*. When the *return-trip* is very short, this might result in unhappy *drivers*.

- Our algorithms do not allow to pickup *resources* from other *depots*.

- From our data we know that there is a fair amount of *trips* that is roughly long enough to keep one *resource-group* busy for the entire day. It might be beneficial to keep these *trips* out of the *improvement-heuristic*, because those *trips* cannot logically be combined with other *trips*.

- We have substantial issues with the programming language we used for our solution, mainly related to the inability of C# to deep copy objects (relationship between objects maintains). We expect this to be the main reason for the extremely long *runtimes*.

Finally, our experiments also have some limitations:

- Due to the issues we have with our data, we are unable to make statements about the improvement related to the current (manually scheduling) situation.

- The *runtimes* of our algorithms are generally fairly long.

- The setup of our experiments is based on the *Regional* scenario. The selected Simulated Annealing parameters might therefore not be fully applicable to the *National* scenario.

We conclude that there are many limitations regarding this research. These limitations result in recommendations, which we give in the following section.

### 7.2.2 Recommendations

We now discuss the recommendations resulting from the limitations listed in Section 7.2.1.

- Regarding the model we have the following recommendations: (*1*) investigate the use a more realistic cost structure; (*2*) include more constraints, both *resource* and *trip* related; (*3*) select *vehicles* for *trips* based on properties of the *trip*, such as volume instead of manually made assumptions; and (*4*) include the selection of specific *Rented resources* and specific charters.

- The quality of data must drastically be improved to implement automatic scheduling at PAT. We recommend to start with improvements on the following elements: (*1*) start monitoring the *DWH-status* of *drivers* at any point in time; (*2*) implement the opportunity to schedule multiple *drivers* per *trip*; as well as (*3*) the opportunity to schedule more than two *vehicles* per *trip*; (*4*) ensure that *orders* (coming from customers) involve the relevant properties on which the scheduling can be based, such as the requested volume; (*5*) implement proper and consequent registration of *time-windows*; and (*6*) register *resource-property* information neatly and consequently.

- Our recommendations regarding the solution are the following: (*1*) try another programming language that is able to cope with deep copying of objects; (*2*) improvements of input data and model should also be implemented in the solution; (*3*) implementing distributed (parallel) optimizations, smarter than just *Local* or *Region*, might improve *runtimes* of the algorithm. Furthermore (*4*) the *Regional* division of departments might be improved; and (*5*) dropping entire day *trips* (mainly *Hire-orders*) out of the optimization automatically (decreasing problem size), because these *trips* can usually not be combined with other *trips*.

- Regarding the experiments, we recommend the following improvements: (*1*) reapply the experiments using a smart grouping algorithm of *departments*; and (*2*) attune parameters for each aggregation level individually.

### 7.2.3 Future research and implementation

Both RGB$^+$ Automatisering and PAT would like to implement some form of automated scheduling as soon as possible. Although our model and solution are not perfect, there are possibilities for partial implementation at first. Examples are: (*1*) as a decision support system (the solution advises planners in their work); (*2*) automatically creating a base schedule, followed by manual refinements; or (*3*) scheduling only a subset (e.g., *trips* without missing information) of the *trips* (i.e., decreasing the problem size). Even a partial implementation requires measures to improve data quality. We recommend more research and improvements upon several points before the implementation and use of (partially) automated scheduling in the Transplan TMS. We suggest the following points of improvement (in order of execution):

1. Improve the structure, implementation, and administration of data. Using wrong or missing data for automated scheduling can do more harm than good. Because our solution is currently depending on manually created schedules (for the adaption of input data), the algorithms can currently not be implemented without the requirement for manual work beforehand.

2. We advise to implement our recommendations regarding the model and the solution method in future research projects. Beforehand, however, improvement of data quality should be accomplished.

3. In the mean time we recommend to find out if *National* scheduling is actually worse than *Regional* scheduling, by attuning the parameters for the *National* aggregation level, as well as improving the regional division using a clustering method. Because using *Regional* or *Local* scheduling is comparable to scheduling *departments* or *regions* in parallel, implementing parallelization of the algorithm and smart exchanges between *departments* and/or *regions* might improve automated scheduling.

4. Try to decrease the problem size by excluding *trips* that hardly can be improved upon (partial implementation). As mentioned before, entire day *trips* (*Hire-orders*) might be excluded. It should be examined if there are other types of *trips* or *resources*

which decrease the problem size in advance and make optimization easier (hopefully decreasing *runtimes*).

5. Finally, research might be done using a hybrid algorithm that divides optimization in multiple parts: (*a*) entire day *trips* (easy); (*b*) round *trips* with (almost) the same start- and end-location (moderate); and (*c*) other *trips* (hard). The main improvement to be expected here is that the algorithms can be attuned to fit the parts (each having its own scheduling challenges). On the other hand, it might be beneficial to integrate the schedules in certain cases (e.g., a small round *trip* after a long distance *trip*).

# Bibliography


Aarts, E. & Korst, J. (1989). *Simulated annealing and boltzmann machines: a stochastic approach to combinatorial optimization and neural computing*. New York, NY, USA: John Wiley & Sons, Inc.

Afshar-Nadjafi, B. & Afshar-Nadjafi, A. (2014). A constructive heuristic for time-dependent multi-depot vehicle routing problem with time-windows and heterogeneous fleet. *Journal of King Saud University - Engineering Sciences*. doi:http://dx.doi.org/10.1016/j.jksues.2014.04.007

Aho, A. & Ullman, J. (1994). *Foundations of computer science*. Principles of computer science series. W. H. Freeman. Retrieved from http://infolab.stanford.edu/~ullman/focs.html

Archetti, C. & Savelsbergh, M. (2009). The trip scheduling problem. *Transportation Science*, *43*(4), 417–431. doi:10.1287/trsc.1090.0278

Baker, B. & Ayechew, M. (2003). A genetic algorithm for the vehicle routing problem. *Computers and Operations Research*, *30*(5), 787–800. cited By 303. doi:10.1016/S0305-0548(02)00051-5

Bettinelli, A., Ceselli, A. & Righini, G. (2014). A branch-and-price algorithm for the multi-depot heterogeneous-fleet pickup and delivery problem with soft time windows. *Mathematical Programming Computation*, *6*(2), 171–197. doi:10.1007/s12532-014-0064-0

Błocho, M. & Czech, Z. (2012). A parallel algorithm for minimizing the number of routes in the vehicle routing problem with time windows. In R. Wyrzykowski, J. Dongarra, K. Karczewski & J. Waśniewski (Eds.), *Parallel processing and applied mathematics* (Vol. 7203, pp. 255–265). Lecture Notes in Computer Science. Springer Berlin Heidelberg. doi:10.1007/978-3-642-31464-3_26

Bräysy, O. & Gendreau, M. (2005a). Vehicle routing problem with time windows, part I: route construction and local search algorithms. *Transportation Science*, *39*(1), 104–118. doi:10.1287/trsc.1030.0056

Bräysy, O. & Gendreau, M. (2005b). Vehicle routing problem with time windows, part II: metaheuristics. *Transportation Science*, *39*(1), 119–139. doi:10.1287/trsc.1030.0057

Bullnheimer, B., Hartl, R. & Strauss, C. (1999). An improved ant system algorithm for the vehicle routing problem. *Annals of Operations Research*, *89*, 319–328. cited By 348. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-0033446619&partnerID=40&md5=d6883074a362430f97a49922ba8e7e4d

Caris, A. & Janssens, G. (2009). A local search heuristic for the pre- and end-haulage of intermodal container terminals. *Computers and Operations Research*, *36*(10), 2763–2772. cited By 33. doi:10.1016/j.cor.2008.12.007

Caseau, Y. & Laburthe, F. (1999). Heuristics for large constrained vehicle routing problems. *Journal of Heuristics*, *5*(3), 281–303. cited By 41. doi:10.1023/A:1009661600931

Christofides, N., Mingozzi, R. & Toth, P. (1979). *The vehicle routing problem* (N. Christofides, R. Mingozzi, P. Toth & C. Sandi, Eds.).

Coffey, J. (2009, February 18). Surface area of the earth. Retrieved November 18, 2015, from http://www.universetoday.com/25756/surface-area-of-the-earth/

Cordeau, J., Gendreau, M., Hertz, A., Laporte, G. & Sormany, J. (2004). *New heuristics for the vehicle routing problem.* (Technical Report No. G-2004-33). GERAD. Montreal, Canada.

Cordeau, J., Gendreau, M., Laporte, G., Potvin, J.-Y. & Semet, F. (2002). A guide to vehicle routing heuristics. *Journal of the Operational Research Society*, *53*(5), 512–522. cited By 238. doi:10.1057/palgrave/jors/2601319

Crama, Y. & Schyns, M. (2003). Simulated annealing for complex portfolio selection problems. *European Journal of Operational Research*, *150*(3), 546–571. Financial Modelling. doi:http://dx.doi.org/10.1016/S0377-2217(02)00784-1

Dayarian, I., Crainic, T., Gendreau, M. & Rei, W. (2015). A column generation approach for a multi-attribute vehicle routing problem. *European Journal of Operational Research*, *241*(3), 888–906. doi:10.1016/j.ejor.2014.09.015

Deineko, V. & Woeginger, G. (2000). A study of exponential neighborhoods for the travelling salesman problem and for the quadratic assignment problem. *Mathematical Programming, Series B*, *87*(3), 519–542. cited By 39. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-8344239073&partnerID=40&md5=641a4b8be368bc8a7086657b74d2f25c

Dominguez, O., Juan, A., Barrios, B., Faulin, J. & Agustin, A. (2014). Using biased randomization for solving the two-dimensional loading vehicle routing problem with heterogeneous fleet. *Annals of Operations Research*. cited By 0; Article in Press. doi:10.1007/s10479-014-1551-4

Dongarra, J. (2013, June 3). Visit to the national university for defense technology changsha, china. Retrieved November 18, 2015, from www.netlib.org/utk/people/JackDongarra/PAPERS/tianhe-2-dongarra-report.pdf

Eksioglu, B., Vural, A. & Reisman, A. (2009). The vehicle routing problem: a taxonomic review. *Computers and Industrial Engineering*, *57*(4), 1472–1483. cited By 139. doi:10.1016/j.cie.2009.05.009

Franceschelli, M., Rosa, D., Seatzu, C. & Bullo, F. (2013). Gossip algorithms for heterogeneous multi-vehicle routing problems. *Nonlinear Analysis: Hybrid Systems*, *10*(1), 156–174.

Fritsma, K. (2015, June 6). *Peter Appel Transport: Truck*. Transportfotos.nl. Retrieved October 27, 2015, from http://www2.picturepush.com/photo/a/14713270/img/Bedrijven-7/Peter-Appel-Middenmeer.jpg

Funke, B., Grünert, T. & Irnich, S. (2005). Local search for vehicle routing and scheduling problems: review and conceptual integration. *Journal of Heuristics*, *11*(4), 267–306. cited By 37. doi:10.1007/s10732-005-1997-2

Gehring, H. & Homberger, J. (1999). A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. In *Proceedings of eurogen99–short course on evolutionary algorithms in engineering and computer science* (pp. 57–64).

Gehring, H. & Homberger, J. (2002). Parallelization of a two-phase metaheuristic for routing problems with time windows. *Journal of Heuristics*, *8*(3), 251–256. cited By 34. doi:10.1023/A:1015053600842

Goel, A. (2009). Vehicle scheduling and routing with drivers' working hours. *Transportation Science*, *43*(1), 17–26. cited By 33. doi:10.1287/trsc.1070.0226

Goel, A. & Gruhn, V. (2006). Drivers' working hours in vehicle routing and scheduling. In *Ieee conference on intelligent transportation systems, proceedings, itsc* (pp. 1280–1285).

Goel, A. & Kok, A. (2012). Truck driver scheduling in the united states. *Transportation Science*, *46*(3), 317–326. doi:10.1287/trsc.1110.0382

Goel, A. & Vidal, T. (2014). Hours of service regulations in road freight transport: an optimization - based international assessment. *Transportation Science*, *48*(3), 391–412. doi:10.1287/trsc.2013.0477

Golden, B. L., Wasil, E. A., Kelly, J. P. & Chao, I.-M. (1998). The impact of metaheuristics on solving the vehicle routing problem: algorithms, problem sets, and computational results. In T. G. Crainic & G. Laporte (Eds.), *Fleet management and logistics* (pp. 33–56). Centre for Research on Transportation. Springer US. doi:10.1007/978-1-4615-5755-5_2

Hollis, B., Forbes, M. & Douglas, B. (2006). Vehicle routing and crew scheduling for metropolitan mail distribution at australia post. *European Journal of Operational Research*, *173*(1), 133–150. cited By 19. doi:10.1016/j.ejor.2005.01.005

Hu, X., Ding, Q. & Wang, Y. (2010). A hybrid ant colony optimization and its application to vehicle routing problem with time windows. *Communications in Computer and Information Science*, *97 CCIS*(PART 1), 70–76. doi:10.1007/978-3-642-15853-7_10

Huang, S.-H., Yang, T.-H. & Wang, R.-T. (2011). Ant colony optimization for railway driver crew scheduling: from modeling to implementation. *Journal of the Chinese Institute of Industrial Engineers*, *28*(6), 437–449. doi:10.1080/10170669.2011.599433

Inspectie Leefomgeving en Transport. (2009, June 30). Rekenregels vo 561/2006. Retrieved April 8, 2016, from https://www.ilent.nl/Images/Rekenregels%20Vo%20561%202009%20Internet%20IVW%2030%20juni%202009_tcm334-325287.pdf

Inspectie Leefomgeving en Transport. (2010a, June 1). Rij-en rusttijden vrachtauto en touringcar (Vo. (EG) nr. 561/2006). Retrieved April 8, 2016, from https://www.ilent.nl/Images/Infoblad%20Rij-%20en%20rusttijden%20vrachtauto%20en%20touringcar%20obv%20Vo%20561_tcm334-318004.pdf

Inspectie Leefomgeving en Transport. (2010b, September 2). Specifieke regels: Wegvervoer. Retrieved April 8, 2016, from https://www.rijksoverheid.nl/binaries/rijksoverheid/documenten/brochures/2010/09/02/specifieke-regels-arbeidstijden-wegvervoer/specifieke-regels-wegvervoer.pdf

Iori, M. & Riera-Ledesma, J. (2015). Exact algorithms for the double vehicle routing problem with multiple stacks. *Computers and Operations Research*, *63*, 83–101. doi:10.1016/j.cor.2015.04.016

Jula, H., Dessouky, M., Ioannou, P. & Chassiakos, A. (2005). Container movement by trucks in metropolitan networks: modeling and optimization. *Transportation Research Part E: Logistics and Transportation Review*, *41*(3), 235–259. cited By 59. doi:10.1016/j.tre.2004.03.003

Koç, Ç., Bektaş, T., Jabali, O. & Laporte, G. (2015). Thirty years of heterogeneous vehicle routing. *European Journal of Operational Research*. cited By 0; Article in Press. doi:10.1016/j.ejor.2015.07.020

Kok, A., Meyer, C., Kopfer, H. & Schutten, J. (2010). A dynamic programming heuristic for the vehicle routing problem with time windows and european community social legislation. *Transportation Science*, *44*(4), 442–454. cited By 30. doi:10.1287/trsc.1100.0331

Kolen, A., Rinnooy Kan, A. & Trienekens, H. (1987). Vehicle routing with time windows. *Operations Research*, *35*(2), 266–273.

Krulwich, R. (2012, September 17). Which is greater, the number of sand grains on earth or stars in the sky? Retrieved November 18, 2015, from http://www.npr.org/sections/krulwich/2012/09/17/161096233/which-is-greater-the-number-of-sand-grains-on-earth-or-stars-in-the-sky

Kumar, S. N. & Panneerselvam, R. (2012). A survey on the vehicle routing problem and its variants. *Intelligent Information Management*, *4*(3), 66–74.

Lai, M., Crainic, T. G., Di Francesco, M. & Zuddas, P. (2013). An heuristic search for the routing of heterogeneous trucks with single and double container loads. *Transportation Research Part E: Logistics and Transportation Review*, *56*, 108–118. Retrieved from www.scopus.com

Laporte, G. (2009). Fifty years of vehicle routing. *Transportation Science*, *43*(4), 408–416. cited By 182. doi:10.1287/trsc.1090.0301

Law, A. (2007). *Simulation modeling and analysis*. McGraw-Hill series in industrial engineering and management science. McGraw-Hill.

Li, F., Golden, B. L. & Wasil, E. (2005). Very large-scale vehicle routing: new test problems, algorithms, and results. *Computers and Operations Research*, *32*(5), 1165–1179. doi:10.1016/j.cor.2003.10.002

Li, X., Leung, S. & Tian, P. (2012). A multistart adaptive memory-based tabu search algorithm for the heterogeneous fixed fleet open vehicle routing problem. *Expert Systems with Applications*, *39*(1), 365–374.

Lin, S. (1965). Computer solutions of the traveling salesman problem. *Bell System Technical Journal*, *44*(10), 2245–2269. cited By 800. doi:10.1002/j.1538-7305.1965.tb04146.x

Logistiek.nl. (2015, March 13). Top 100 logistiek dienstverleners 2015. Retrieved September 22, 2015, from http://www.logistiek.nl/logistieke-dienstverlening/nieuws/2015/3/download-de-online-tabel-top-100-logistiek-dienstverleners-101120170

Mester, D. & Bräysy, O. (2005). Active guided evolution strategies for large-scale vehicle routing problems with time windows. *Computers and Operations Research*, *32*(6), 1593–1614. cited By 113. doi:10.1016/j.cor.2003.11.017

Metters, R. (1996). Interdependent transportation and production activity at the united states postal service. *Journal of the Operational Research Society*, *47*(1), 27–37. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-0029777609&partnerID=40&md5=a59cd71df698d66bb70a6412855b8eae

Miller, C., Zemlin, R. & Tucker, A. (1960). Integer programming formulation of traveling salesman problems. *Journal of the ACM (JACM)*, *7*(4), 326–329. doi:10.1145/321043.321046

Ministerie van Sociale Zaken en Werkgelegenheid. (2011, April 29). Brochure: De Arbeidstijdenwet. Retrieved April 8, 2016, from https://www.rijksoverheid.nl/binaries/rijksoverheid/documenten/brochures/2011/04/29/de-arbeidstijdenwet-nederlands/brochure-de-arbeidstijdenwet.pdf

NovoFerm. (2014, January 30). *Dockleveller NovoFerm NovoDock*. NovoFerm. Retrieved April 18, 2016, from http://www.novoferm.nl/media/image/4000/4000/1/dockleveller_novoferm_novodock-l730i-6_1200x930.jpg

Osman, I. (1993). Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research*, *41*(4), 421–451. cited By 483. doi:10.1007/BF02023004

Pisinger, D. & Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers and Operations Research*, *34*(8), 2403–2435. doi:10.1016/j.cor.2005.09.012

Potvin, J.-Y. (2009). State-of-the art review:evolutionary algorithms for vehicle routing. *INFORMS Journal on Computing*, *21*(4), 518–548. doi:10.12S7/ijoc.1080.0312

Potvin, J.-Y. & Naud, M.-A. (2011). Tabu search with ejection chains for the vehicle routing problem with private fleet and common carrier. *Journal of the Operational Research Society*, *62*(2), 326–336. cited By 17. doi:10.1057/jors.2010.102

Prescott-Gagnon, E., Desaulniers, G., Drexl, M. & Rousseau, L.-M. (2010). European driver rules in vehicle routing with time windows. *Transportation Science*, *44*(4), 455–473. cited By 24. doi:10.1287/trsc.1100.0328

Prescott-Gagnon, E., Desaulniers, G. & Rousseau, L.-M. (2009). A branch-and-price-based large neighborhood search algorithm for the vehicle routing problem with time windows. *Networks*, *54*(4), 190–204. cited By 33. doi:10.1002/net.20332

Qu, Y. & Bard, J. (2015). A branch-and-price-and-cut algorithm for heterogeneous pickup and delivery problems with configurable vehicle capacity. *Transportation Science*, *49*(2), 254–270. doi:10.1287/trsc.2014.0524

Raff, S. (1983). Routing and scheduling of vehicles and crews. the state of the art. *Computers and Operations Research*, *10*(2), 63–67, 69–115, 117–147, 149–193, 195–211. doi:10.1016/0305-0548(83)90030-8

Rijksoverheid. (2016a, March 2). Arbeidstijdenbesluit Vervoer. Retrieved April 8, 2016, from http://wetten.overheid.nl/BWBR0009386/2016-03-02

Rijksoverheid. (2016b, January 1). Arbeidstijdenwet. Retrieved April 8, 2016, from http://wetten.overheid.nl/BWBR0007671/2016-01-01

Ropke, S. & Cordeau, J. (2009). Branch and cut and price for the pickup and delivery problem with time windows. *Transportation Science*, *43*(3), 267–286. cited By 54. doi:10.1287/trsc.1090.0272

Ropke, S., Cordeau, J. & Laporte, G. (2007). Models and branch-and-cut algorithms for pickup and delivery problems with time windows. *Networks*, *49*(4), 258–272. cited By 80. doi:10.1002/net.20177

Ropke, S. & Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, *40*(4), 455–472. cited By 232. doi:10.1287/trsc.1050.0135

Salhi, S., Wassan, N. & Hajarat, M. (2013). The fleet size and mix vehicle routing problem with backhauls: formulation and set partitioning-based heuristics. *Transportation Research Part E: Logistics and Transportation Review*, *56*, 22–35. cited By 6. doi:10.1016/j.tre.2013.05.005

Savelsbergh, M. (1990). An efficient implementation of local search algorithms for constrained routing problems. *European Journal of Operational Research*, *47*(1), 75–85. cited By 46. doi:10.1016/0377-2217(90)90091-O

Seixas, M. & Mendes, A. (2013). Column generation for a multitrip vehicle routing problem with time windows, driver work hours, and heterogeneous fleet. *Mathematical Problems in Engineering*, *2013*. cited By 2. doi:10.1155/2013/824961

Sieben, B. (2011, June 21). *Peter Appel Transport: Tractor, Trailer, Dolly and Trailer*. Transportfotos.nl. Retrieved October 27, 2015, from http://www3.picturepush.com/photo/a/5935416/img/Foto%60s-21-06-2011/Foto%60s-21-06-2011-017-BorderMaker.jpg

Smilowitz, K. (2006). Multi-resource routing with flexible tasks: an application in drayage operations. *IIE Transactions (Institute of Industrial Engineers)*, *38*(7), 577–590. cited By 32. doi:10.1080/07408170500436898

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, *35*(2), 254–265. cited By 1249. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-0023313252&partnerID=40&md5=d413145a436dd31d1d4ac3acdc14f3d3

Stewart Jr., W. & Golden, B. L. (1984). A lagrangean relaxation heuristic for vehicle routing. *European Journal of Operational Research*, *15*(1), 84–88. cited By 19. doi:10.1016/0377-2217(84)90050-X

Taillard, É. (1999). A heuristic column generation method for the heterogeneous fleet vrp. *RAIRO - Operations Research*, *33*(1), 1–14.

top500.org. (2015, November 17). Top500 list - november 2015. Retrieved November 18, 2015, from http://www.top500.org/list/2015/11/

Toth, P. & Vigo, D. (2002). Models, relaxations and exact approaches for the capacitated vehicle routing problem. *Discrete Applied Mathematics*, *123*(1-3), 487–512. cited By 109. doi:10.1016/S0166-218X(01)00351-1

van den Brink, W. (2013, October 23). *Peter Appel Transport: Truck, Dolly and Trailer*. Transportfotos.nl. Retrieved October 27, 2015, from http://s20.postimg.org/4plwt6wsd/a_9.jpg

Verspaansdonk, W. (2013, October 13). *Peter Appel Transport: Tractor and Trailer*. Transportfotos.nl. Retrieved October 27, 2015, from http://www3.picturepush.com/photo/a/13835871/800/Anonymous/IMG-6059c-tf-Appel-Transport,-Peter,-Zwaagdijk-MB-.jpg

Wen, M., Krapper, E., Larsen, J. & Stidsen, T. (2011). A multilevel variable neighborhood search heuristic for a practical vehicle routing and driver scheduling problem. *Networks*, *58*(4), 311–322. cited By 4. doi:10.1002/net.20470

Xu, H., Chen, Z.-L., Rajagopal, S. & Arunapuram, S. (2003). Solving a practical pickup and delivery problem. *Transportation Science*, *37*(3), 347–364. doi:10.1287/trsc.37.3.347.16044

Zäpfel, G. & Bögl, M. (2008). Multi-period vehicle routing and crew scheduling with outsourcing options. *International Journal of Production Economics*, *113*(2), 980–996. cited By 40. doi:10.1016/j.ijpe.2007.11.011

# Appendices

# Appendix A

# Fully automated planning process

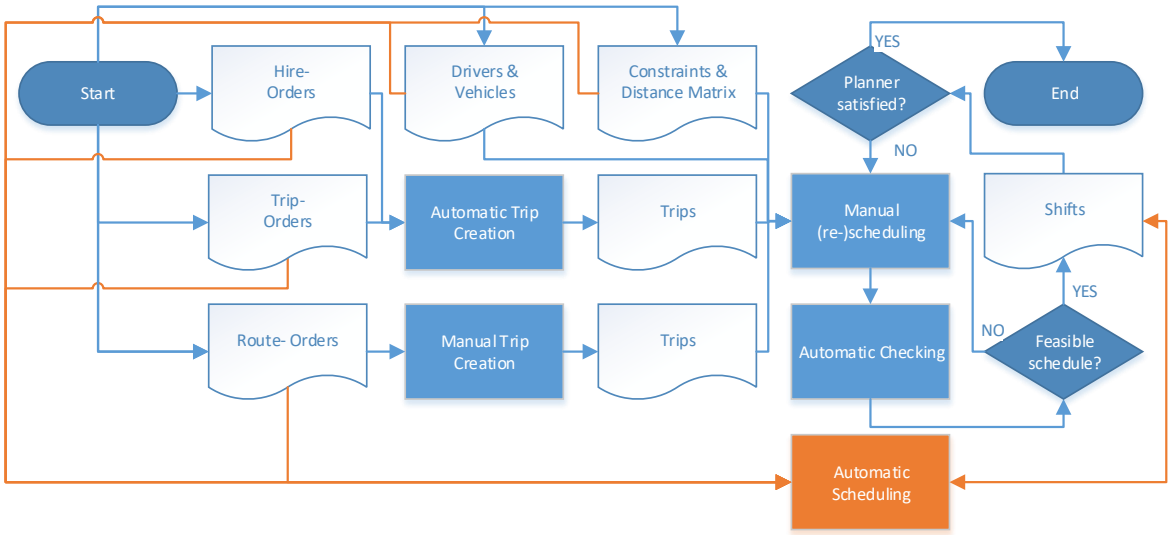



**Figure A.1:** Fully automated planning process: All orders will be automatically grouped where usefull and automatically be assigned.

# Appendix B

# Problem size

## B.1 Assignment of orders to vehicles

Adapted from Aho and Ullman (1994), the number of possible assignments of $n$ orders $X_i$, $i = 1, 2, ..., n$ to $m$ vehicles $V_j$, $j = 1, 2, ..., m$ can be approximated by the formula in Equation B.1, which will be explained below.

$$\frac{(n + v - 1)!}{(v - 1)!} \tag{B.1}$$

An abstract version of a schedule can then be represented as follows. Vehicles are distinguished by a separator ($\star$). Within a vehicle, the orders are stated in order of execution. As an example, the possible assignments and routes of two orders $(X_1, X_2)$ to two vehicles $(V_1, V_2)$ are expressed in Table B.1 (six possible assignments) and a similar example assignment of three orders to three vehicles is given in Table B.2 (sixty possible assignments). There are $n$ orders and $v - 1$ separators (for $v$ vehicles) that must be placed in a certain order, where both the sequence of vehicles and order execution matter, due to time-windows and vehicle types.

There are $(n + v - 1)!$ ways to create a sequence for vehicles and orders. The separators however (not the vehicles) are all equal, so permuting the separators does not change the link between orders and vehicles. Therefore a division by $(v - 1)!$ is performed, resulting in the formula in Equation B.1. The number of possible assignments in the PAT case ($n = 1000, v = 700$) is approximately $5 \times 10^{3065}$ (see Equation B.2).

$$\frac{(1000 + 800 - 1)!}{(800 - 1)!} \approx 4 \times 10^{3102} \tag{B.2}$$

Despite the fact that a large part of these assignments might be infeasible, we would like to give an impression of the size and complexity of the problem: compared to the huge number of solutions to this problem, the amount of stars in the universe or number of grains of sand on the earth is much smaller. These numbers have respectively an estimated 22 and 27 number of digits (Krulwich, 2012). If instances of the currently fastest supercomputer are built on every square meter of land surface on earth, it would still take a number of years with over 3000 digits to enumerate all options (see Section B.2).

## B.2 Computer power

The earth has a land surface of 149 million square kilometre, which is $149 \times 10^{12}$ square meters (Coffey, 2009) and there are $365.25 \times 24 \times 60 \times 60 = 31557600$ seconds in a year. The fastest supercomputer on earth is Tianhe-2, with a surface of 720 square meters and a speed of 33863 Teraflops (top500.org, 2015; Dongarra, 2013), which is $33863 \times 10^{12}$ calculations per second or $31557600 * 33863 \times 10^{12} = 1.069 \times 10^{24}$ calculations per year. There will fit

| $V_1$ | $V_2$ | Representation |
|---|---|---|
| $X_1X_2$ | none | $X_1X_2\star$ |
| $X_2X_1$ | none | $X_2X_1\star$ |
| $X_1$ | $X_2$ | $X_1 \star X_2$ |
| $X_2$ | $X_1$ | $X_2 \star X_1$ |
| none | $X_1X_2$ | $\star X_1X_2$ |
| none | $X_2X_1$ | $\star X_2X_1$ |

**Table B.1:** Combinations for the assignment of two orders to two vehicles.

$149\times10^{12}/720 = 5.519\times10^{12}$ Tianhe-2 computers on earth. So we can do $5.519\times10^{12}\times1.069\times10^{24} = 5.9\times10^{36}$ calculations per year. Therefore it will take $5\times10^{3065}/5.9\times10^{36} = 8.5\times10^{3028}$ years to do these calculations.

| $V_1$ | $V_2$ | $V_3$ | Representation | $V_1$ | $V_2$ | $V_3$ | Representation |
|---|---|---|---|---|---|---|---|
| $X_1X_2X_3$ | none | none | $X_1X_2X_3\star\star$ | $X_3$ | $X_1X_2$ | none | $X_3\star X_1X_2\star$ |
| $X_1X_3X_2$ | none | none | $X_1X_3X_2\star\star$ | $X_3$ | $X_2X_1$ | none | $X_3\star X_2X_1\star$ |
| $X_2X_1X_3$ | none | none | $X_2X_1X_3\star\star$ | $X_2$ | $X_1X_3$ | none | $X_2\star X_1X_3\star$ |
| $X_2X_3X_1$ | none | none | $X_2X_3X_1\star\star$ | $X_2$ | $X_3X_1$ | none | $X_2\star X_3X_1\star$ |
| $X_3X_1X_2$ | none | none | $X_3X_1X_2\star\star$ | $X_1$ | $X_2X_3$ | none | $X_1\star X_2X_3\star$ |
| $X_3X_2X_1$ | none | none | $X_3X_2X_1\star\star$ | $X_1$ | $X_3X_2$ | none | $X_1\star X_3X_2\star$ |
| none | $X_1X_2X_3$ | none | $\star X_1X_2X_3\star$ | none | $X_1X_2$ | $X_3$ | $\star X_1X_2\star X_3$ |
| none | $X_1X_3X_2$ | none | $\star X_1X_3X_2\star$ | none | $X_2X_1$ | $X_3$ | $\star X_2X_1\star X_3$ |
| none | $X_2X_1X_3$ | none | $\star X_2X_1X_3\star$ | none | $X_1X_3$ | $X_2$ | $\star X_1X_3\star X_2$ |
| none | $X_2X_3X_1$ | none | $\star X_2X_3X_1\star$ | none | $X_3X_1$ | $X_2$ | $\star X_3X_1\star X_2$ |
| none | $X_3X_1X_2$ | none | $\star X_3X_1X_2\star$ | none | $X_2X_3$ | $X_1$ | $\star X_2X_3\star X_1$ |
| none | $X_3X_2X_1$ | none | $\star X_3X_2X_1\star$ | none | $X_3X_2$ | $X_1$ | $\star X_3X_2\star X_1$ |
| none | none | $X_1X_2X_3$ | $\star\star X_1X_2X_3$ | $X_3$ | none | $X_1X_2$ | $X_3\star\star X_1X_2$ |
| none | none | $X_1X_3X_2$ | $\star\star X_1X_3X_2$ | $X_3$ | none | $X_2X_1$ | $X_3\star\star X_2X_1$ |
| none | none | $X_2X_1X_3$ | $\star\star X_2X_1X_3$ | $X_2$ | none | $X_1X_3$ | $X_2\star\star X_1X_3$ |
| none | none | $X_2X_3X_1$ | $\star\star X_2X_3X_1$ | $X_2$ | none | $X_3X_1$ | $X_2\star\star X_3X_1$ |
| none | none | $X_3X_1X_2$ | $\star\star X_3X_1X_2$ | $X_1$ | none | $X_2X_3$ | $X_1\star\star X_2X_3$ |
| none | none | $X_3X_2X_1$ | $\star\star X_3X_2X_1$ | $X_1$ | none | $X_3X_2$ | $X_1\star\star X_3X_2$ |
| $X_1X_2$ | $X_3$ | none | $X_1X_2\star X_3\star$ | none | $X_3$ | $X_1X_2$ | $\star X_3\star X_1X_2$ |
| $X_2X_1$ | $X_3$ | none | $X_2X_1\star X_3\star$ | none | $X_3$ | $X_2X_1$ | $\star X_3\star X_2X_1$ |
| $X_1X_3$ | $X_2$ | none | $X_1X_3\star X_2\star$ | none | $X_2$ | $X_1X_3$ | $\star X_2\star X_1X_3$ |
| $X_3X_1$ | $X_2$ | none | $X_3X_1\star X_2\star$ | none | $X_2$ | $X_3X_1$ | $\star X_2\star X_3X_1$ |
| $X_2X_3$ | $X_1$ | none | $X_2X_3\star X_1\star$ | none | $X_1$ | $X_2X_3$ | $\star X_1\star X_2X_3$ |
| $X_3X_2$ | $X_1$ | none | $X_3X_2\star X_1\star$ | none | $X_1$ | $X_3X_2$ | $\star X_1\star X_3X_2$ |
| $X_1X_2$ | none | $X_3$ | $X_1X_2\star\star X_3$ | $X_1$ | $X_2$ | $X_3$ | $X_1\star X_2\star X_3$ |
| $X_2X_1$ | none | $X_3$ | $X_2X_1\star\star X_3$ | $X_1$ | $X_3$ | $X_2$ | $X_1\star X_3\star X_2$ |
| $X_1X_3$ | none | $X_2$ | $X_1X_3\star\star X_2$ | $X_2$ | $X_1$ | $X_3$ | $X_2\star X_1\star X_3$ |
| $X_3X_1$ | none | $X_2$ | $X_3X_1\star\star X_2$ | $X_2$ | $X_3$ | $X_1$ | $X_2\star X_3\star X_1$ |
| $X_2X_3$ | none | $X_1$ | $X_2X_3\star\star X_1$ | $X_3$ | $X_1$ | $X_2$ | $X_3\star X_1\star X_2$ |
| $X_3X_2$ | none | $X_1$ | $X_3X_2\star\star X_1$ | $X_3$ | $X_2$ | $X_1$ | $X_3\star X_2\star X_1$ |

**Table B.2:** Combinations for the assignment and sequence of three orders to three vehicles.

# Appendix C

# Algorithms

## C.1 Algorithm details

**Algorithm 1** Simulated Annealing

**procedure** SimulatedAnnealing
  Initialization($S$)
  **while** $r < R$ **do** ▷ Stop criterion
    Generate $N_\lambda(S)$
    $a = 0$ ▷ Accepted solution counter
    **while** $N_\lambda(S) \neq \emptyset$ **and** $a = 0$ **do**
      Select $S' \in N_\lambda(S)$
      $k = k + 1$
      $\Delta$ = CalculateChange($S', S$)
      $\theta$ is a uniform random number, $0 < \theta < 1$
      **if** $\Delta \leq 0$ or ($\Delta > 0$ and $e^{(-\Delta/T_k)} > \theta$) **then** ▷ Accept $S'$
        $a = a + 1$
        $S = S'$
        **if** $C(S') < C(S_b)$ **then** ▷ Update Best Solution
          $S_b = S'$
          $T_b = T_k$
        **end if**
      **else**
        Discard $S'$ from $N_\lambda(S)$
      **end if**
      NormalTemperatureDecrease($T_s, T_f, T_k, \alpha, \gamma, k$)
    **end while**
    **if** $a = 0$ **then** ▷ Zero moves accepted, reset
      OccasionalTemperatureIncrease($T_r, T_b$)
      $r = r + 1$
    **end if**
  **end while**
**end procedure**

**Algorithm 1 (Continued)** Simulated Annealing Functions

**function** Initialization($S$) ▷ Initialization of *simulated annealing*
  Generate a neighborhood $N_\lambda(S)$ based on the $\lambda$-opt
  $N_{feas} = 0$
  **for all** $S'_i \in N_\lambda(S)$ **do**
    $\Delta_i =$ CalculateChange($S'_i, S$)
    **if** $S'_i$ is feasible **then**
      $N_{feas} = N_{feas} + 1$
    **end if**
  **end for**
  $\Delta_{max} = \max\{\Delta_i\}$
  $\Delta_{min} = \min\{\Delta_i\}$
  $n =$number of customers ▷ Constant
  $T_s = \Delta_{max}$ ▷ Start temperature
  $T_f = \Delta_{min}$ ▷ Final temperature
  $T_r = T_s$ ▷ Reset temperature
  $\alpha = n \times N_{feas}$ ▷ Constant
  $\gamma = n$ ▷ Constant
  $R =$maximum number of resets ▷ Stopping criterion
  $S_b = S$ ▷ Current best = start solution
  $k = 0$ ▷ Iteration counter
  $r = 0$ ▷ Reset counter
**end function**

**function** CalculateChange($S'$,$S$)
  **return** $C(S') - C(S)$ ▷ C is some cost calculation function
**end function**

**function** OccasionalTemperatureIncrease($T_r, T_b$)

$$T_r = \max\left\{\frac{T_r}{2}, T_b\right\}$$

$$T_k = T_r$$

**end function**

**function** NormalTemperatureDecrease($T_s, T_f, T_k, \alpha, \gamma, k$)

$$\beta_k = \frac{T_s - T_f}{(\alpha + \gamma\sqrt{k})T_s T_f}$$

$$T_k = \frac{T_k}{(1 + \beta_k T_k)}$$

**end function**

**Algorithm 2** Tabu Search

```
1: procedure TabuSearch
2:     initialization
3:     while iterationCount < maxIterations do
4:         S = iterationBest
5:         iterationBest = ∅
6:         while counter < neighborhoodSize do
7:             s' = s
8:             while not isFeasible(s'') do
9:                 Randomly choose neighborhood operator n ∈ {·}
10:                s'' = applyNeighborhood(s', n)
11:            end while
12:            updateTabuList(n)
13:            evaluateQuality(s'')
14:            s = s''
15:            if s better than s_best then
16:                s_best = s
17:            end if
18:            if s better than iterationBest then
19:                iterationBest = s
20:            end if
21:            counter = counter + 1
22:        end while
23:        iterationCount = iterationCount + 1
24:        counter = 0
25:    end while
26: end procedure
```

**Algorithm 2 (Continued)** Tabu Search Functions

```
27: function initialization
28:     Construct initial solution s
29:     evaluateQuality(s)
30:     s_best = s
31:     iterationBest = s
32:     counter = 0
33:     iterationCount = 0
34: end function

35: function
36:     t
37: end function
```

**Algorithm 3** Adaptive Large Neighborhood Search

Initialization($\cdot$)
$x$ =ConstructFeasibleSolution($\cdot$)
$x^* = x$
**repeat**
    $N^-$ =ChooseNeighborhood($\{\pi_j\}$,Destroy)
    $N^+$ =ChooseNeighborhood($\{\pi_j\}$,Repair)
    $x'$ =CreateNeighborhood($x, N^-, N^+$)
    **if** $x'$ can be accepted **then**
        **if** $x < x'$ **then**
            $\sigma = \sigma_2$ ▷ Better than previous
        **else**
            $\sigma = \sigma_3$ ▷ Worse than previous
        **end if**
        $x = x'$
    **end if**
    **if** $f(x) < f(x^*$ **then**
        $x^* = x$
        $\sigma = \sigma_1$ ▷ Better than overall
    **end if**
    UpdateScores($\bar{\pi}_{N^-,j}, \sigma$)
    UpdateScores($\bar{\pi}_{N^+,j}, \sigma$)
    **if** $k = 100$ **then** ▷ End of segment
        UpdateSegment($\cdot$)
    **else**
        $k = k + 1$
    **end if**
**until** Stop criterion
**return** $x^*$

**Algorithm 3 (Continued)** Adaptive Large Neighborhood Search Functions

**function** Initialization(·)
    $\sigma_1 =$ ▷ Score Adjustment parameters
    $\sigma_2 =$
    $\sigma_3 =$
    $\eta =$ ▷ Noise parameter
    $j = k = 1$ ▷ Segment calculators
**end function**

**function** ChooseNeighborhood($\{\pi_{\cdot,j}\}$,type)
    $r = \text{rand}\,[0, 1]$
    $P = 0$
    Select neighborhood of specified type with probability:
    **for** $i = 1 \rightarrow \omega$ **do**

$$p_i = \frac{\pi_{i,j}}{\sum_{k=1}^{\omega} \pi_{k,j}}$$

        $P = P + p_i$
        **if** $r < P$ **then**
            **return** $N_i^{type}$
        **end if**
    **end for**
**end function**

**function** CreateNeighborhood(·)
    Create Neighborhood based on input
    CalculateCost(·)
**end function**

**function** CalculateCost(·)
    $N_{max} = \eta \max_{i,j \in V}\{d_{ij}\}$
    $\delta = \text{rand}\,[N_{max}, N_{max}]$
    $C' = \max\{0, C + \delta\}$
**end function**

**function** UpdateScores($\bar{\pi}_{i,j}, \sigma$)
    $\bar{\pi}_{i,j} = \bar{\pi}_{i,j} + \sigma$
    **return**
**end function**

**function** UpdateSegment(·)

$$\pi_{i,j+1} = \rho \frac{\bar{\pi}_{i,j}}{a_i} + (1 - \rho)\pi_{i,j}$$

    $j = j + 1$
    $k = 1$
    $\bar{\pi}_{i,j} = 0$
**end function**

**Algorithm 4** Genetic Algorithm

1: **procedure** GeneticAlgorithm
2: Create initial population $population$ of $P$ solutions
3: Evaluate each solution in $population$
4: **for** $i = 1 \rightarrow P$ **do**
5: Select two parent solutions $parent_1$ and $parent_2$ (with replacement) from $population$ based on a randomized selection of the solution values.
6: Apply crossover to $parent_1$ and $parent_2$ to generate $child_1$ and $child_2$.
7: Apply mutation (improvement heuristic) to $child_1$ and $child_2$.
8: Insert $child_1$ and $child_2$ in $population_{new}$.
9: **end for**
10: Evaluate each offspring in $population_{new}$.
11: $population = population_{new}$.
12: **return** Best solution found
13: **end procedure**

# Appendix D

# Vehicle-properties

Besides the properties treated in Section 3.2.1, there are some other *vehicle-properties* that exist in practice and are probably relevant, however data about them is missing, and available knowledge is mainly stored in peoples heads. Therefore, we name them and do not further pay attention to them. We examine *Box-vehicles only* (hard sheet walls; loaded from the back). Besides that, there are *Curtain-sider-vehicles* (sides made of canvas; enabling loading from the side) and *Container-vehicles* (able to carry containers). *Vehicles* might have a *tailgate* for *Loading* or *Unloading*. There are several types of *tailgates*: a *Folding-tailgate* or a *Closing-tailgate*, that might have a sideloader (used for unloading on walkways). The *Folding-tailgate* folds under the *Truck* while driving, while the *Closing-tailgate* is in a vertical position while *Driving*, sometimes functioning as the *vehicle's* door. *Vehicles* with a *Closing-tailgate* can only visit distribution centers that are equipped with a tailgate opening ('brievenbus'; in Dutch, see Figure D.1). *Vehicles* might further be equipped with a pallet-transporter, have a limited height, sleeping cabin (for daily rests), or are provided with country specific toll-equipment.


**Figure D.1:** Docks with a tailgate opening, the opened tailgate fits under the dock. (NovoFerm, 2014)

# Appendix E

# Relationship between vehicle-combination-type and vehicle-type


**(a)** Euro-Combi: Tractor and Eurotrailer (Verspaansdonk, 2013)


**(b)** Truck (Fritsma, 2015)


**(c)** Euro-Dolly-LHV: Truck, Dolly and Eurotrailer (van den Brink, 2013)


**(d)** City-Slider-LHV: Tractor, City-slidertrailer and City-trailer (Sieben, 2011)

**Figure E.1:** Vehicle Combination Types at PAT

| | | Tractor | Truck | Euro-Combi | Bilevel-Combi | Euro-Combi + Euro-Combi | City-Combi | Axle-Combi | Truck + Truck | City-Slider-LHV | City-Combi + City-Combi | Euro-Dolly-LHV | Euro-Combi + Truck | Bilevel-Dolly-LHV | Bilevel-Combi + Truck |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| drivers | C | 1 | 1 | | | | | | 2 | | | | 1 | | 1 |
| | CE | | | 1 | 1 | 2 | 1 | 1 | | | 2 | | 1 | | 1 |
| | LHV | | | | | | | | | 1 | | 1 | | 1 | |
| vehicles | Tractor | 1 | | 1 | 1 | 2 | 1 | | | 1 | 2 | | 1 | | 1 |
| | Truck | | 1 | | | | | | 2 | | | | 1 | | 1 |
| | Truck (towing) | | | | | | | 1 | | | | 1 | | 1 | |
| | Dolly | | | | | | | | | | | 1 | | 1 | |
| | Eurotrailer | | | 1 | | 2 | | | | | | 1 | 1 | | |
| | Eurotrailer (towing) | | | | | | | | | | | | | | |
| | Bileveltrailer | | | | 1 | | | | | | | | | 1 | 1 |
| | Citytrailer | | | | | | 1 | | | 1 | 2 | | | | |
| | City-slidertrailer | | | | | | | | | 1 | | | | | |
| | Euro-slidertrailer | | | | | | | | | | | | | | |
| | Axletrailer | | | | | | | 1 | | | | | | | |
| | Axletrailer (towing) | | | | | | | | | | | | | | |

**Table E.1:** Possible vehicle-combination-types and their composition out of vehicle-types. ‘Towing’ means ability-to-tow=‘True’.

# Appendix F

# Stop durations at customers

We relate the differences in *stop* duration to individual customers, addresses, *vehicle-combination-types* and employees. Due to the fact that most customers of PAT provide planned *stop* durations in their *orders*, it can be seen that the addresses with the largest differences are usually frequently visited by the same customers of PAT. A similar reasoning can be made for the employees: *drivers* that have large differences are usually frequent visitors at several addresses. In Figure F.1 we see the average stop durations per customer. Consequently working with deviating planned *stop* durations (i.e., always faster or longer execution times) results in an unreliable planning.

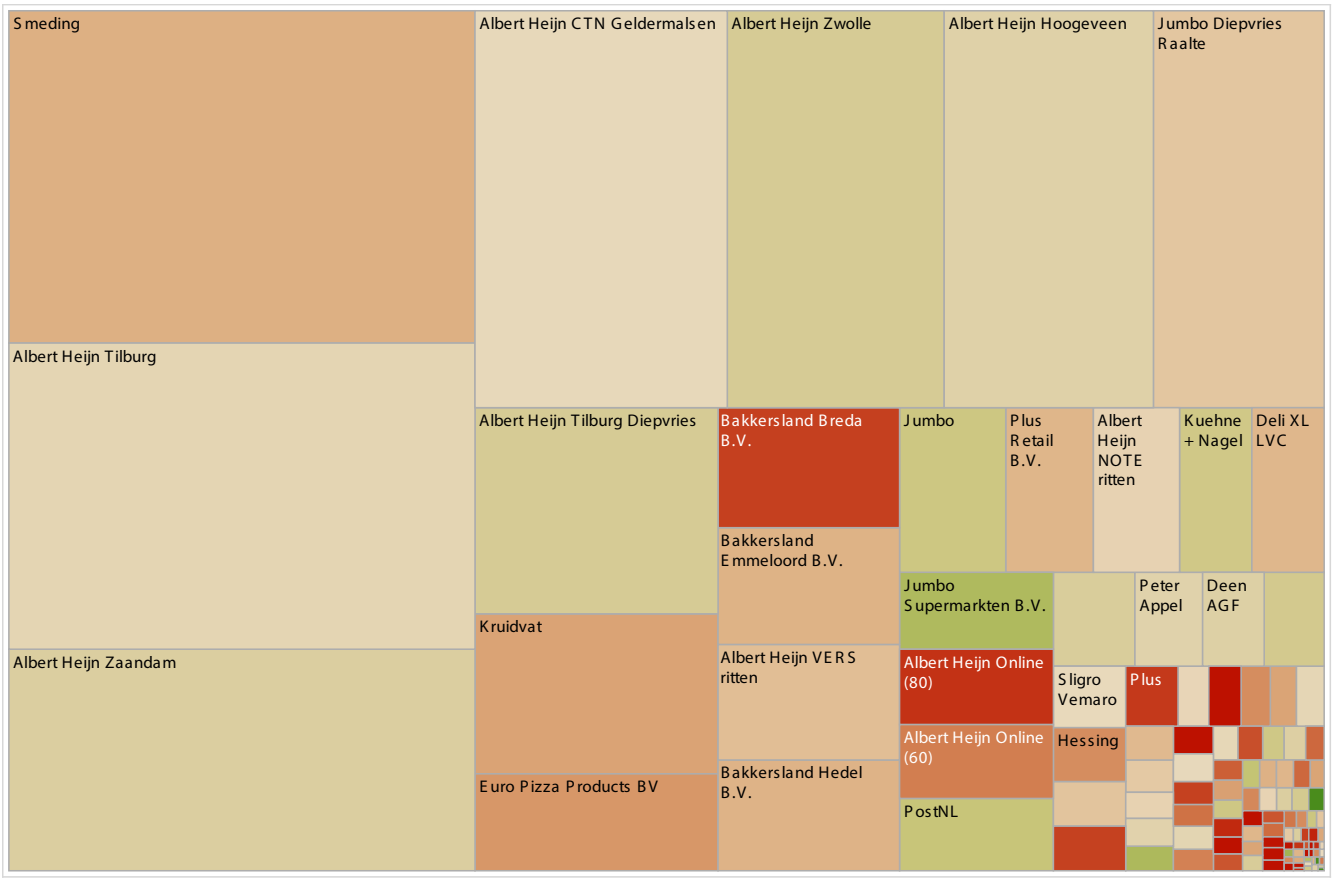



**Figure F.1:** Average difference in stop duration per customer. The size corresponds with the number of stops for that customer. Average stop durations shorter than planned are green, longer average stop durations are red. The larger the difference, the intenser the color.

# Appendix G

# Data model


**Figure G.1:** Data Model. Some objects have been left out (DwhStatus, Address, AvailabilityPeriod, Customer, and ActivityType).

# Appendix H

# Software statistics

- Duration
  - Total Runtime
  - Construction Runtime
  - Improvement Runtime
- Number of
  - Resourcegroups
  - Chartered resourcegroups
  - Rented resourcegroups
  - Own resourcegroups
- Totals
  - Duration
  - Distance
  - Vehicle duration
  - Vehicle distance
  - Driver duration
  - Driver distance
- Cost
  - Total cost
  - Cost charters
  - Cost rented
  - Cost own
  - Cost pre/return trips
- Waste
  - Pre/return-trips
    - Duration
    - Distance
  - Charters

        - Duration
        - Distance
    - Rented Drivers
        - Duration
        - Distance
    - Rented Vehicles
        - Duration
        - Distance
- Employment
    - Own Vehicles
        - Duration
        - Distance
        - Availability
    - Own Drivers
        - Duration
        - Distance
        - Availability

# Appendix I

# Questionnaire Expert Opinion

**Vragenlijst**

Kunt u gegeven de input data en restricties een score geven aan de volgende stellingen, gerelateerd aan deze planning? De vragen kunnen beantwoord worden op een schaal van -2 tot 2, waarbij de volgende scores gegeven kunnen worden.

- -2 mee oneens
- -1 beetje mee oneens
- 0 neutraal
- 1 beetje mee eens
- 2 mee eens

Daarnaast is het mogelijk om bij elke vraag opmerkingen te plaatsen. Let op: er zijn geen goede of foute antwoorden. Het gaat ons om het inzicht in de kwaliteit van de planningen. We stellen zeven meerkeuzevragen en vervolgens nog vier open vragen.

1. De planning is van een kwalitatief goed niveau.
2. Er zijn voldoende pauzes gepland.
3. De planning is realistisch.
4. De beschikbare voertuigen worden doelmatig ingezet.
5. De beschikbare chauffeurs worden doelmatig ingezet.
6. Charters en huurvoertuigen worden doelmatig ingezet.
7. De planning bevat grote fouten.

Daarnaast zouden wij graag zien dat u de volgende open vragen beantwoord:

A. Welke punten vindt u zwak aan deze planning?

B. Welke punten vindt u sterk aan deze planning?

C. Welke punten zou u verbeteren aan deze planning?

D. Heeft u verder nog opmerkingen?