Delivery routing challenge for a European retailer
Master thesis

Vrije Universiteit Amsterdam
Faculty of Sciences
Study programme Business Analytics
De Boelelaan 1105
1081 HV Amsterdam

Centrum Wiskunde & Informatica Amsterdam
Research group Stochastics
Science Park 123
1098 XG Amsterdam

Author: Dimitry Erkin
CWI supervisor: Dr. Elenna R. Dugundji
VU internship supervisor: Prof. dr. Ger Koole
VU second reader: Prof. dr. Joaquim Gromicho

August 13, 2017

Preface

This thesis is the output of my internship at Centrum Wiskunde & Informatica (CWI) Amsterdam. I conducted this internship for my Master Thesis as a student in Business Analytics Programme at the Vrije Universiteit Amsterdam (VU). During these 6 months I studied a practical case of the Vehicle Routing Problem. Together with my colleagues we went from a business case definition through data discovery and analysis, and eventually ended up having a prototype of a prescriptive mathematical model which is capable of solving a formulated business challenge.

I would like to thank my CWI supervisor Dr. Elenna Dugundji for all the effort she made providing me with the necessary data acquired from relevant third parties and supporting me with activities related to gathering business requirements. I would also like to show my gratitude to my VU supervisors Prof. dr. Ger Koole and Prof. dr. Joaquim Gromicho for their valuable recommendations. Next, I would like to express many thanks to our colleagues at the European retailer, the freight transport delivery company and the ICT solution company for giving us meaningful feedback. Last but not least, I would like to thank Rob van der Mei as host at CWI, Walther Ploos van Amstel as Principal Investigator of the ITSLOG project, and Thijs Elsing and Robert de Roos for support from the municipality of Amsterdam.

Amsterdam, August 2017 Dimitry Erkin

Managerial summary

The purpose of this research is to provide decision support for a European retailer with its daily delivery operations of goods. A critical challenge which the retailer faces is the limited capacity of the transportation network in the Amsterdam area. This leads to very strict requirements for the tactical delivery routing plans. Indeed, it is usually the case that there is only one parking place available at each store per time window and any delay in arrival may result in a situation where the next truck due to arrive is forced to drive around in circles or wait somewhere blocking traffic. The uncertainty of travelling times due to traffic congestion is also an important factor which can significantly affect an actual delivery schedule, causing arrival delay and consequent waiting time. This research aims to minimize total travel and waiting time costs first by carefully planning the deliveries in the presence of uncertainty, and second by considering strategic buffer locations for a delivery vehicle to wait temporarily until a loading zone for a particular store becomes available instead of blocking traffic or adding to traffic and vehicle kilometres travelled by circling around.

In order to respond to this challenge a mathematical model is proposed and prototyped which is capable of generating daily delivery routing plans for retail goods given necessary constraints and the cost structure. This model takes into account incurred costs such as the mileage, demand satisfaction and waiting time. These factors are weighted according to their relative importance. The assessment of this model is based on the comparison of the generated plans with the actual plans currently used by the European retailer. In particular, an existing historical routing plan is used as an input for the model, the total solution costs are computed, then a plan is generated by this model and the results are compared.

Analysis demonstrates that buffer locations help to reduce waiting costs without any drop in total solution costs. Furthermore, commonly used buffers which are shared by multiple stores have greater cost reduction than linked buffers which are dedicated to a particular store. These shared buffers allow 7.5% savings of the waiting time costs and 6.9% of the total solution costs respectively, compared to the case without buffers. Moreover, smaller relative importance of demand satisfaction costs less than or equal to 1.0€/item leads to the possibility of considering partial delivery. In addition, an instrument is obtained to verify the usefulness of different buffer locations by using these locations in the model. The more often a buffer is shown to be used, the more reason there is to propose to the municipality to dedicate a buffer at that location.

Contents

1 Introduction

The purpose of this research is to provide decision support for a European retailer with its daily delivery operations of goods. A critical challenge which the retailer faces is the limited capacity of the transportation network in the Amsterdam area. This leads to very strict requirements for the tactical delivery routing plans. Responding to this challenge, both a mathematical analytical model and a numerical method of solving this problem are proposed. This method is capable of generating delivery routing plans given all necessary constraints and the cost structure. The model takes into account incurred costs such as the mileage, demand satisfaction level and waiting time, all being weighted according to their relative importance. Furthermore, in order to reduce waiting time, buffer locations within the Amsterdam area are considered where a truck can park and wait until the loading zone at a store becomes available. The numerical method is approximate in nature and is based on the Column Generation technique. This technique allows iterative exploration of search space by adding new promising one-truck routes (columns). The Regret construction heuristic is applied to generate an initial solution. New promising columns are generated by means of solving the Pricing Subproblem which takes into account duals of the Master problem relaxation. Analysis demonstrates that the buffer locations help to reduce waiting time incurred by early arrivals without any drop in total solution costs. Furthermore, an instrument is obtained to verify the usefulness of different buffer locations by using these locations in the model.

The thesis is organised as follows: This chapter provides the reader with a high level overview of the research. In Chapter 2, the statement of the research is formulated and assumptions are defined. The description of all necessary data sources, data cleaning and transformation procedures as well as the analytical methods for extracting knowledge from the data is provided in Chapter 3. Chapter 4 contains the mathematical model formulation. The numerical method of solving this model is described in Chapter 5. Chapter 6 reports on several experiments which have been conducted in order to verify the hypothesis. In Chapter 7, conclusions are made, and recommendations regarding possible usage of the implemented model as well as promising directions for further research are proposed. While Chapters 1, 2 and 7 are intended for a broad audience, the remaining chapters require general technical background and familiarity with the theory of mathematical optimization in particular.

2 Research statement

The research hypothesis is formulated as follows. Utilization of buffer locations in the Amsterdam city area results in the reduction of the waiting time costs of the European retailer's daily delivery operations of goods while the demand satisfaction level does not decrease. When considering waiting time, two types of delay are taken into account, incurred by either early arrivals at a store when store personnel is not prepared for unloading/loading the truck, or overlapping arrival and departure of multiple vehicle at the same store when multiple vehicles cannot be serviced simultaneously.

There are a number of assumptions which are made. First, it is assumed that different types of costs have different importance. This assumption significantly affects the resulting shipment plan generated by the model. Second, it is assumed that it is possible to deliver less goods than the total demand by a store. This allows consideration of cases when partial demand satisfaction can result in lower travelling time and/or waiting time costs. Third, it is assumed that different types of goods have their own relative priority for delivery. In other words, if a store has demand for all considered types of goods then an attempt is made to deliver fresh goods first. Fourth, for the deterministic case it is assumed that a truck might travel at a maximal allowed speed for a particular road segment. Fifth, it is assumed that a particular buffer would be used only once for a particular one truck trip. Sixth, it is assumed that the most reasonable time to arrive at a store from a buffer is in the middle of corresponding time window. This allows more possibility for a truck which arrived during a previous time window and is still parked to leave this store, so that overlapping arrival and departure are eliminated. Finally, the length of a driver shift is taken into account and it is assumed that a trip duration should not exceed 8 hours.

3 Data Discovery

3.1 Data Selection

First, the set of necessary data sources is defined:

Data Source	Description
Google place id map	– A map with Google Road API places id assigned to Amsterdam roads. Acquired from: Municipality of Amsterdam
Google city flow data	– A dataset with road traffic measurements in Amsterdam. Acquired from: Municipality of Amsterdam
Hoofd en plusnetten map	– The National Road Database map with structural parameters of Amsterdam roads. Acquired from: Municipality of Amsterdam
Wegvakken map	– The National Road Database map with structural parameters of The Netherlands roads. Acquired from: `https://www.rijkswaterstaat.nl`
Parameters related to the European retailer's store locations in Amsterdam	– The spatial locations of the European retailer's delivery destinations in Amsterdam. – The demand of each store location per product type. – The number of vehicles of each vehicle type at each store location which can be unloaded at the same time. – The time windows per day for each day of the week for each store for delivery by each vehicle type. Acquired from: Freight delivery company
Parameters of the freight delivery company fleet which services the retailer's distribution centre	– Types of available delivery vehicles. – Number of delivery vehicles are per vehicle type. – Capacity of each vehicle type per product type. Acquired from: Freight delivery company
Travel plan of each delivery vehicle	– Expected time of departure from the retailer's distribution centre. – Expected time of arrival at each store location. – Expected time of departure from each store location. – Expected time to travel back to the retailer's distribution centre. – Planned travel route in terms of street line segments. Acquired from: ICT solution company

Data Source	Description
Actual travel route of each delivery vehicle	– Actual time of departure from the retailer's distribution centre. – Actual time of arrival and departure at each store location. – Actual time to travel back to the retailer's distribution centre Acquired from: ICT solution company
Buffer locations in the Amsterdam area	– Buffer locations linked to particular stores agreed with the Municipality of Amsterdam. – Multi-purpose parking locations in Amsterdam provided by "Truck Parking Europe" service `https://app.truckparkingeurope.com` – Map of urban development non-residential functions `http://maps.amsterdam.nl/open_geodata/`

Table 1: Data Sources

Next, the training set time period from 1 April until 29 June 2016, and the test set time period on 30 June 2016 are selected for further analysis.

After that, identification of the attributes of each data source is performed, using the defined time period where it is relevant.

The Google place id map contains 77,258 rows of data. Each row represents a directional road segment (PlaceId) for which Google has collected historical aggregate statistics (Table 7). The Google city flow dataset contains 142,436,186 rows. Each row represents a measurement made for a particular road segment (PlaceId) at a particular point in time (Table 8). The "Hoofd en plusnetten" map has 24,044 rows. Each row represents a link (wvk_id) of the Amsterdam transportation network (Table 9). The "Wegvakken" map contains 1,008,045 rows of data. Each row represents a link (wvk_id) of the Netherlands transportation network (Table 10). The spatial location dataset of the European retailer's delivery destinations in Amsterdam contains 82 rows. Each row represents either the distribution center (naam=DC) or a store (Table 11). For the vehicles parameters of the retailer's delivery destinations in Amsterdam there are 84 rows. Each row represents a store (Table 12). For the freight transport delivery fleet parameters there are 6 rows. Each row represents a capacity of a particular vehicle type carrying a particular type of goods (Table 13). For the Travel plan (30 June 2016, Amsterdam) there are 125 rows. Each row represents a delivery to a store (Table 14). For the buffer locations linked to corresponding stores there are 4 selected rows. Each row represents a buffer (Table 15). For the commonly used buffer locations there are 10 selected rows. Each row represents a buffer (Table 16).

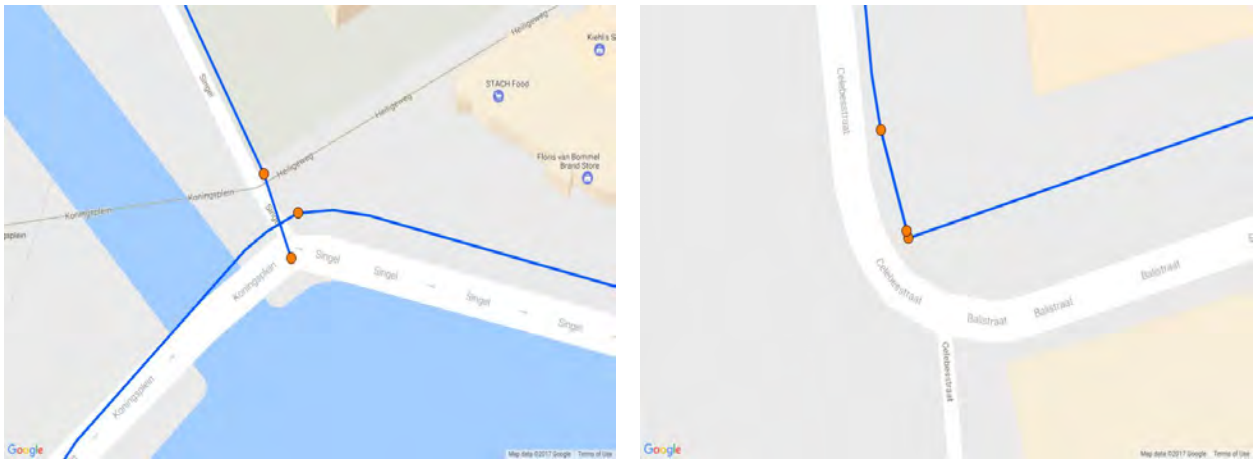
3.2 Data Cleaning and Transformation

The PostGIS extension of PostgreSQL database is utilized (geospatial functions) combined with QGIS and R (rgeos and maptools libraries) to manage the map data. In addition, the igraph library is used to compute the shortest paths.

3.2.1 Correction of Google place id map

The Google place id map has some road segments with incorrect spatial location of starting and/or ending points which prohibit proper identification of intersections. One example of this is at the intersection near

Heiligeweg 46 (Figure 1.a). Another example is where Celebesstraat crosses Balistraat (Figure 1.b). A number of these issues are fixed manually.

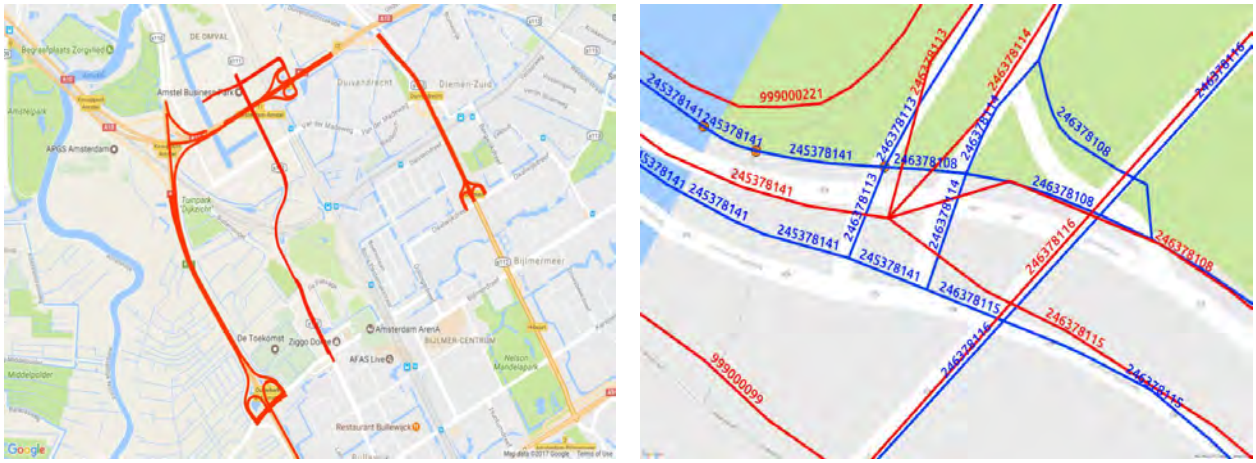


(a) (b)

Figure 1: Google place id map issues

3.2.2 Extention of "Hoofd en plusnetten" map

There are a number of roads missing in the "Hoofd en plusnetten" map which connect the central part of Amsterdam with Amsterdam Zuidoost through Diemen, and near the intersection between Ringweg Zuid and A2 highway. The missing links make it impossible to perform the identification of shortest paths (Figure 2.a). Those missing roads are copied from the "Wegvakken" map and the maximal speed values are estimated from the neighbourhood.



(a) New roads added (in red colour) (b) Roads matched by wvk_id

Figure 2: Hoofd en plusnetten map extension and matching with Google

3.2.3 Filtering of roads in "Hoofd en plusnetten" map

The "Hoofd en plusnetten" map contains 1,975 rows with positive values of the attribute "auto" representing major roads intended primarily for vehicle traffic. This criterion is alone however insufficient for selecting

all roads where delivery vehicles may travel. Furthermore, the attribute "snelheid" (travel speed) has zero values for some vehicle roads. Thus, vehicle roads not suited for delivery trucks are **excluded** based on the following criteria:

- If the attribute "bst code" in the corresponding "Wegvakken" map record (which has the same "wvk_id") has value of either FP or VP. This criterion filters out dedicated bicycle paths (FP) and dedicated pedestrian paths (VP).
- If either of the attributes "fiets" (bicycle) or "voet" (pedestrian) is non-zero while all of the attributes "ov", "auto" and travel speed are zero in the "Hoofd en plusnetten" map. Here roads are excluded that have indication of minimal vehicle traffic.
- If the attribute "bus_tram" is 2 and all of the attributes "voet", "fiets" and "auto" are zero in the "Hoofd en plusnetten" map. This filters out road segments that are primarily used for public transit.

3.2.4 Google place id and "Hoofd en plusnetten" map matching

In order to perform a vehicle fleet routing algorithm there is a need to know the travelling time along the links of Amsterdam transportation network. Indeed, the possibility to get to an arbitrary location on time depends not only on the travelling distance to this location from a starting point but also on the amount of time a vehicle will spend travelling due to travel speed limitations for each link.

There are two maps which contain necessary information to estimate travelling times on the vehicle transportation network in Amsterdam. The Google place id map has data about directionality of road segments. The "Hoofd en plusnetten" map has information about the maximal travel speed on the transportation links. Given these two data sources, they are matched together in order to combine the directionality of the Google place id map with travel speed data from the "Hoofd en plusnetten" map.

The idea of the map matching algorithm is to look for similarity between segments of both maps using three parameters computed for each pair of those segments such as distance, number of overlapping points and angle. Given these parameters, the most similar segment from the Google map is selected for each segment from the "Hoofd en plusnetten" map.

In order to obtain the distance between two segments, the first segment geometry from the Google map is split into points, and the average 2D Cartesian distance is calculated between the second segment geometry from the "Hoofd en plusnetten" map and each of those points. For the sake of computation of the number of overlapping points, the number of the first segment points is counted which are within 0.0002 distance from the second segment. The distances are defined in units of the spatial reference system ($0.0001 \approx 11.1$ m, $0.00001 \approx 1.11$ m).

The computation of an angle between two segments requires first splitting first segment into points and finding two points which are the closest to the second segment. Then, the angle α_1 is computed from the horizontal to the vector defined by these points, then this angle is adjusted to get rid of directionality as follows:

$$\hat{\alpha_i} = \begin{cases} \alpha_i - 180 & \text{if } \alpha_i \geq 180 \\ \alpha_i & \text{otherwise} \end{cases} \qquad i = 1, 2 \tag{1}$$

Next, the second segment is split into points and two these points are found which are the closest to two first segment points that were identified in the previous step. Given these two second segment points, the angle α_2 is computed from the horizontal to the vector defined by these points and this angle is adjusted to get rid of directionality according to (1). As a result, the slopes of both segments are obtained in an area where these segments are close to each other. This gives an estimate of how similar these two segments are in terms of their spatial orientation.

Given these three parameters computed (distance, number of overlapping points and angle), a heuristic matching procedure is applied as follows:

1. For each first map segment up to two candidates are selected from the ordered list of the second map segments based on the maximal number of overlapping points, minimal angle and minimal distance.
2. The first candidate is picked out if there is only one (obvious choice).
3. A candidate is selected which has the minimal distance if both candidates have the same angle, the difference in number of overlapping points between these candidates is not greater than 0.7, and the difference in distances between these candidates is not greater than 0.7.
4. A candidate is selected which has the minimal distance if both candidates have the same number of overlapping points, and the difference in angles between these candidates is not greater than 15°, and the difference in distances between these candidates is not greater than 0.7.
5. Otherwise, a candidate is selected with the maximal number of overlapping points.

This matching procedure allows matching 53,837 Google map segments out of 77,258. Most of the time this matching is correct but in some difficult cases manual corrections are made. Three examples of such matching results are represented in Figure 2.b and Figure 3.

(a) (b)

Figure 3: Hoofd en plusnetten map matching with Google (continued)

3.2.5 Estimation of missing maximal speed in Google place id map

After the above map matching procedure with the "Hoofd en plusnetten" map there are still 532 segments in the Google place id map which have zero maximum speed. These segments are connected to segments with non-zero maximum speed, so this attribute value is estimated for these segments from the neighbourhood by heuristic.

The idea of this heuristic is based on the visual observation that in most cases the segments with zero maximum speed follow the same trajectory as other segments with known maximum speed values. Indeed, as you can see in the Figure 4.a there is a highway which goes from the lower left corner to the upper right corner. This highway in the lower section has zero maximum speed but after crossing another road it has a maximum speed of 100. Thus, what is done is either the trajectory of a road is followed or there is a movement in the opposite direction. Consequently, known max_speed values are assigned to the missing ones. This results in the following algorithm:

1. Segments are selected with non-zero maximum speed which have outgoing segments with zero maximum speed (starting segments), and segments with non-zero maximum speed which have incoming segments with zero maximum speed (ending segments).

2. Starting and ending segments are sorted in decreasing order according to maximum speed value, so that the higher value has priority. Therefore, for the cases when a segment with zero value has on both its sides segments with non-zero value, the highest of two values will be propagated.

3. For starting segments, the known maximum speed value is propagated forward following the directed route course. For each next segment in this route all outgoing segments are iteratively checked, and for each of them maximum speed value is propagated forward recursively. For each next segment all the incoming segments (except one that the procedure came from) are iteratively checked, and for each of them maximum speed value is propagated back recursively (in the opposite direction to the directed route course).

4. For ending segments the known maximum speed value is propagated backward. For each previous segment all the incoming segments are iteratively checked, and for each of them the maximum speed value is propagated back recursively. For each previous segment all outgoing segments are also iteratively checked (except one the procedure came from), and for each of them the maximum speed value is propagated forward recursively.

As a result of applying this algorithm all the segments in the Amsterdam area are obtained with non-zero maximum speed. The proposed heuristic is admittedly optimistically biased, in that it produces a network where some segments have higher maximum speed values than they might have in reality. Therefore, further analysis to improve this approach is needed. Two examples of such propagation are depicted in Figure 4 (0 in blue colour and corresponding propagated value in green colour).

(a) (b)

Figure 4: Google place id map max_speed estimation

3.2.6 Network construction

After the maximum speed estimation all data necessary to compute the travelling times (and distances) along the links of the Amsterdam transportation network is available. But the network itself is something which needs to be constructed first.

The idea of the network construction algorithm is to form links between two intersections by combining all the segments of the Google place id map which belong to each link. Indeed, the Google place id map has in some cases many consecutive segments corresponding to one link so it is necessary to collect all of them, extract their attributes and associate these segments with a link. The network is constructed using the following algorithm:

1. All the segments are aggregated by longitude and latitude values of their starting and ending points, and the number of unique segments is computed per each combination. This gives all the intersections.

2. All the cases are filtered when the number of unique segments is less than or equal to 2. This gives all the crossroads where more than 2 segments are joined.

3. All the single segments are identified which directly connect two crossroads. These segments are counted as links.

4. All the segments are identified which go from a crossroads A to an intersection B (of exactly 2 segments) and back to this crossroads A. This allows identification of the cycle of 2 links with opposite directions (A $\rightarrow$ B $\rightarrow$ A). These two segments (A $\rightarrow$ B and B $\rightarrow$ A) are counted as links.

5. For the remaining segments which are not counted by the previous steps the procedure starts from a crossroads and adds them iteratively to a link one by one, following the course of a directed network up until it arrives to the next crossroads. In this way all links are constructed which are comprised of more than one segment. The procedure proceeds until all the segments are assigned to links.

This algorithm gives an Amsterdam vehicle travelling links network of 21,077 nodes which are crossroads of more than 2 links, and the 44,395 links that connect those nodes. The algorithm is straightforward and does not apply any sophisticated methods for fast network construction like Hierarchical Routing (Geisberger et al., 2008) because the size of created network is relatively small. This allows computation of this network in a few minutes on a PC with 4x Intel(R) Core(TM) i3-2350M CPU @ 2.30GHz and 16Gb RAM. However, if there was a need to construct such network for the whole country of the Netherlands, advanced methods would be definitely needed to be considered. Two examples of the algorithm results are represented in Figure 5. In purple colour are the links which combine the paths from the original map.

(a) (b)

Figure 5: Google place id map network

3.2.7 Shipment Data extraction

A unique identifier i is assigned to each store from Table 11 (Table 17). A unique identifier i is assigned to each buffer from Tables 16 and 15 respectively (Tables 18 and 19). These two groups are not used at the same time so the identifiers overlap. It is defined that each of these buffers has only one time window $w = 1$ $[0, 1440]$, 0 service time, and 0 demand for all goods types.

The procedure proceeds with extracting time windows data. To start with, it fills missing start and end of store's time window values in Table (14) as corresponding planned arrival and departure respectively. After that, the procedure orders all time windows related to each store according to their start time, and assigns to these time windows numbers from 1 to 4 (w). Next, service duration s is calculated for each combination of i and w as the difference between corresponding planned arrival and departure. Finally, the resulting

start b and end e of time windows are computed as the number of minutes since June 30 00:00 (Table 20).

Identifiers are assigned such as $h = 1$ and $h = 2$ to goods types VS and HB, identifiers $k = 1$ to vehicle type BAK, $k = 2$ to vehicle type CIT and $k = 3$ to vehicle type EUR respectively (Table 21).

Demand of store i within time window w is extracted as a vehicle's capacity divided by the number of stops this vehicle does during this trip. In order to do so first the largest vehicle which can visit all the trip's stores is found, using Table (12). Then, the maximal capacity of a trip is computed as the capacity Q^{k1} of the trip's vehicle type k for the fresh goods type VS($h = 1$) using Table (13). Next, the number of stops N per store id i and type of goods for each trip is calculated using Table (14). After that, for each trip the demand is computed for goods type h of store i visited by this trip within time window w as:

$$\hat{d}_i^{wh} = \left\lfloor \frac{Q^{k1}}{N} \right\rfloor \tag{2}$$

Finally, demand of store i within time window w is adjusted as:

$$d_i^{w2} = \left\lfloor \frac{Q^{k2}}{\max_{o \in H} Q^{ko}} \hat{d}_i^{w2} \right\rfloor, d_i^{w1} = \hat{d}_i^{w1} \tag{3}$$

While extracting demand data it is assumed that each and every store can be visited only by one vehicle within each of its time windows. In fact, there is one exception in the Shipment plan for June 30, that is there are two trips such as DC—2016—26-4-099 and DC—2016—26-4-99 which visit the same stores. The model is deliberately simplified by omitting trip DC—2016—26-4-99. The resulting demand values are represented in Table 22.

3.3 Data Analysis

3.3.1 Nearest crossroads search for European retailer's store locations

Data analysis starts by looking for the nearest network crossroads from each of the given European retailer's store locations.

The idea behind this search is to select a network link such that it has the nearest to this store location Google place id map path, and either starting or ending crossroads of this link are within a certain 2D Cartesian distance ($0.02 \approx 2$ km) from this store location.

3.3.2 Shortest route calculations for European retailer's delivery network

Given the nearest crossroads, the procedure proceeds computing the shortest route in terms of the travelling distance between each pair of the retailer's stores in the Amsterdam area.

The resulting network of shortest routes is depicted in Figure 6 in blue while the retailer's store locations are represented as brown dots. The spatial points for which there are Google data measurements are displayed as green dots.

3.3.3 Nearest buffer location search

Given the network which was constructed in 3.2.6, the nearest node bu is found for each of the buffers from Table 16. Next, for each ordered pair of stores (i, j) node bu is found such that the sum of travelling times from i to bu and from bu to j is minimal for this pair (i, j). This results in set $NB = (i, j, bu)$ which for each pair (i, j) has exactly one element with corresponding to this pair nearest buffer.

3.3.4 Road mean speed forecasting

From Figure 6 it is observed that in many cases the Google speed_mean measurements are made for the spatial points which indeed belong to the shortest routes between the European retailer's stores. This

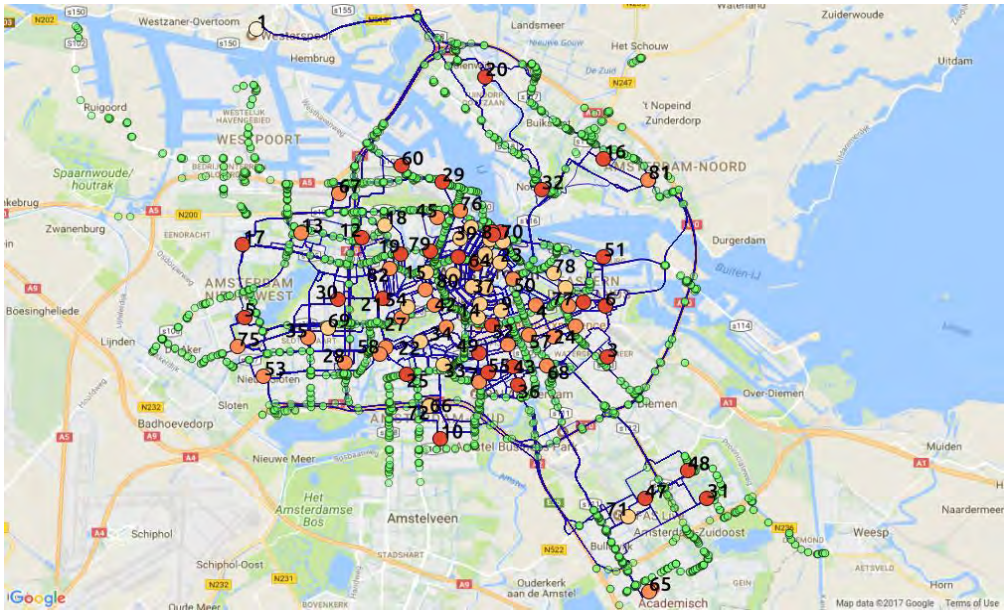


Figure 6: European retailer stores pairs shortest routes and Google data coverage

observation leads to the conclusion that more realistic and adequate fleet route planning can be made for the future if these measurements are taken into account instead of considering fixed constant travelling times. It is only done for the links which are covered by such measurements, while for the remaining (mostly inner) links the static structural travelling times are used based on the maximal speed.

In order to set up the experimental environment two sets are considered such as a training data set of 3 months from 1 April till 29 June 2016, and 30 June 2016 as a test set. These sets for each path have about 25,500 and 250 observations respectively.

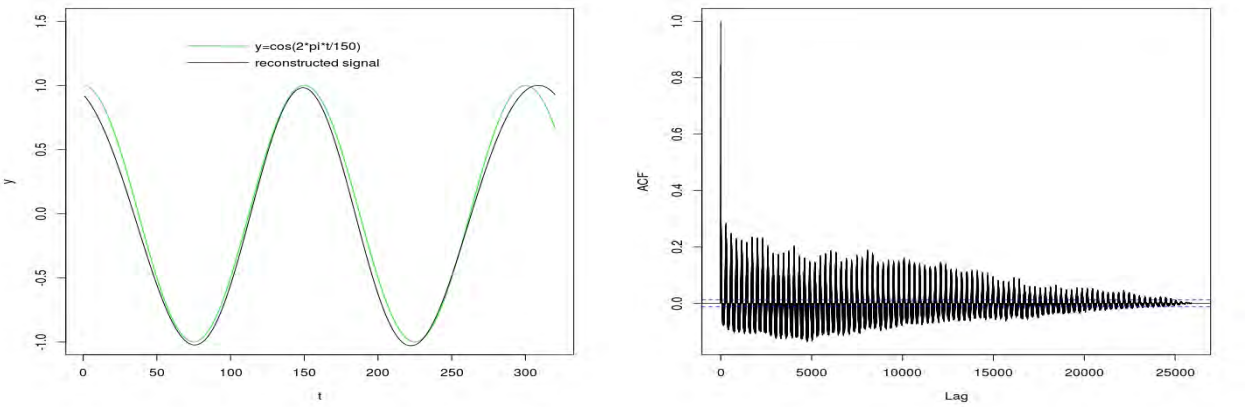
The idea of the forecasting algorithm is based on the Fourier analysis of time series (Bloomfield, 2000). The predicted mean speed is reconstructed as the sum of certain harmonics extracted from the training set time series. These harmonics correspond to the peaks of Autocorrelation function which indicate the periodicity patterns in the time series. In other words, if there is peak p_1 at lag l_1 then harmonic $h_c = \left\lfloor \frac{N_{training}}{p_1} \right\rfloor + 1$ needs to be added to the resulting reconstructed signal along with a number of harmonics on both sides of h_c. For instance, if there is a time series constructed from cosine function with period 150 and the number of observations is 320 (two complete periods and a bit more) then there are 2 peaks of Autocorrelation function at lags 147 and 287. $\left\lfloor \frac{320}{147} \right\rfloor + 1 = 3$ thus the resulting signal is reconstructed from harmonics $2, 3, 4$. This signal is depicted in black colour in Figure 7.a. The described above idea results in the following algorithm:

1. A sub-set of the Google city flow data is considered corresponding to a certain path (fixed placeId).
2. Outliers are eliminated by removing all the measurements which being rounded to tenths (0,10,20,30,etc.) occur in the data set less than a certain number of times (10% of the data set).
3. Missing measurements are estimated by averaging the values of two nearest points on both sides.
4. A linear regression model is fitted with dependent variable *speed_mean* and independent variable *start_interval* using the training data set. This model gives an estimate for the trend T in the data and residuals e which are needed to be studied further.
5. Autocorrelation in residuals e is studied by applying the Autocorrelation function $\rho_e(l)$ (Chatfield,

2003) to the residuals time series:

$$\rho_e(l) = \frac{\gamma_e(l)}{\gamma_e(0)}, \gamma_e(l) = Cov(e_t, e_{t+l}) = E(e_t e_{t+l}) - E(e_t)E(e_{t+l}), \tag{4}$$
$$l = 1..N_{training}, N_{training}\text{- the number of rows in the training set}$$

6. It is observed that in most cases the residuals e demonstrate clear periodic patterns like those which are depicted in Figure 7.b.



(a) (b)

Figure 7: Signal reconstruction example and residuals autocorrelation

7. The lags are identified which correspond to the positive peaks P^+ (local maximums) of the autocorrelation function. These peaks are what the periodic pattern is comprised of. For the sake of finding these maximums the peak is defined as a point which has on both sides not less than a certain number of other points with lower values. This peak neighbourhood size PN is the parameter of the model.

8. Fast Discrete Fourier Transform is applied (Bloomfield, 2000) to the residuals time series in order to get the magnitudes m_h, phases ph_h and frequencies f_h of $\frac{N_{training}}{2}$ harmonics (fft function from R stats package). In particular, these parameters are needed for the peaks which were identified in the previous step.

9. Filtered residuals are reconstructed as follows. First, only peaks are considered which correspond to lags up to three days $p \leq \frac{3}{90} N_{training}$ given that the training set has 90 days of measurements. Next, for each selected peak p corresponding central harmonic number h_c is computed. Finally, the symmetric neighbourhood of h_c of size $2PN + 1$ is considered. All considered waves are summed and the result is scaled by the number of harmonics $\frac{N_{training}}{2}$

$$\hat{e} = \frac{2}{N_{training}} \sum_{p \in P^+, p \leq \frac{3}{90} N_{training}, h_c = \left\lfloor \frac{N_{training}}{p} \right\rfloor + 1, h \in [h_c - PN, h_c + PN]} m_h \cos(2\pi t f_h + ph_h) \tag{5}$$
$$t = 1..(N_{training} + N_{test}), N_{test}\text{- the number of rows in the test set}$$

10. The response variable is reconstructed as $\widehat{speed\_mean} = T + \hat{e}$

11. The goodness of fit is measured as follows:

$$R = \sum_{t=1..(N_{training}+N_{test})} (speed\_mean_t - \widehat{speed\_mean}_t)^2 \tag{6}$$

12. The resulting model is cross-validated using peak neighbourhood size $PN = 5..25$ with $t = 1..(N_{training} + N_{test})$.

This algorithm gives a forecast for the future *mean_speed* value for a path given that there is historical data for this path. This predicted *mean_speed* value is time dependent and it is computed for each time interval (5 munutes). Some examples of such forecasting are depicted in Figure 8. For the network links which are comprised of more than one path such forecast is made separately for each path (when there are measurements) and later combine these forecasts for the link.

4 Model

The problem of optimal vehicle fleet route search can be formulated using a complete directed graph $G(V, A)$ where V is the set of nodes and A is the set or arcs. V includes the origin depot v_1, the destination depot v_{n+2}, and all the stores and buffers $v_2..v_{n+1}$. The buffers form subset $B \subset V$. A is comprised of directed arcs between graph nodes.

In addition to the graph, there are a number of static parameters such as the set of vehicle types K, the set of vehicle numbers L (the same cardinality is assumed for all vehicle types), the set of time window numbers per day W (the same cardinality is assumed for all locations but not all of these time windows might have demand for a particular store), and the set of goods types H. Each vehicle type $k \in K$ has the maximal capacity Q^k in terms of the number of containers which can be loaded.

Each node $v_i, i \in V \setminus \{1, n+2\}$ has its service time s_i and demand per product type per time window d_i^{wh}. Furthermore, for each node a binary variable z_i^w is introduced which indicates the presence of demand. Moreover, each node $v_i, i \in V \setminus \{1, n+2\}$ has a set of time windows in which a vehicle must arrive, described by the beginning and ending time moments b_i^w and e_i^w respectively.

Each arc from node i to node j $(i, j \in V)$ is characterised by its travelling distance c_{1ij} and travelling time t_{ij}. The second parameter can be either deterministic or stochastic. In the latter case t_{ij} is represented by its mean $E(t_{ij})$ and variance $Var(t_{ij})$. Furthermore, the stochastic travelling times are not stationary.

The set of all feasible routes Ω is defined which start at the origin depot, visit a sub-set of stores within their time windows and return to the destination depot. Given that at most one vehicle can visit a store within each time window, a binary constant α_{ri}^{klw} is introduced which specifies that store i is visited by vehicle l of type k within time window w following route r. Finally, a binary decision variable x_r is defined which indicates that route r is included in a solution.

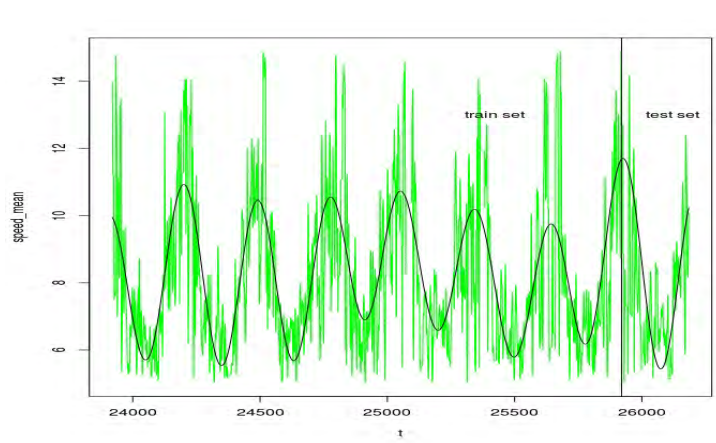
For each route $r \in \Omega$ three types of costs are introduced: the travelling costs c_{1r}, the costs of not satisfying demand c_{2r}, and the costs of waiting at a store location due to the fact that the previous vehicle has not left yet c_{3r}.

The costs of travelling along a route (7) are computed as the sum of all travelling distances c_{1ij} for the arcs which this route is comprised of.

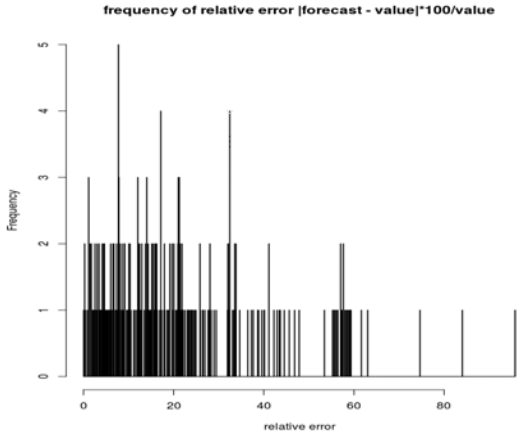
$$c_{1r} = \sum_{(i,j) \in r} c_{1ij} \qquad \forall r \in \Omega \tag{7}$$

The costs incurred for not delivering the full amount of goods of type h to store i during time window w (8) are computed as the difference between demand d_i^{wh} and the actual amount of delivered goods u_i^{wh}. The costs of not delivering the entire demand of goods for store i within time window w (9) are computed as the sum of c_{2i}^{wh} for all types of goods h. The costs of not satisfying the demand for route r (10) are computed as the sum of c_{2i}^{w} for all locations visited by route r.

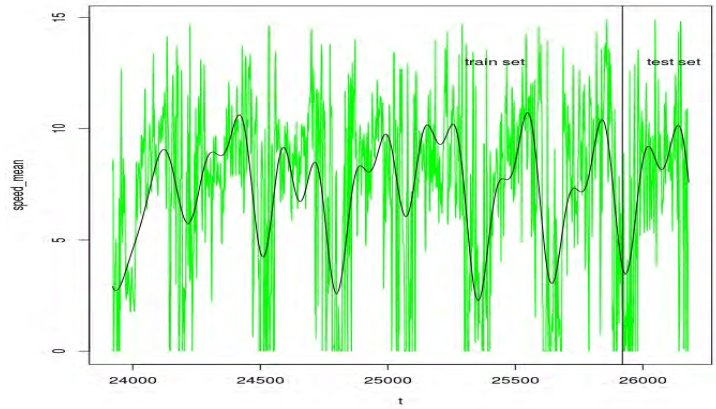
$$c_{2i}^{wh} = d_i^{wh} - u_i^{wh} \qquad \forall w \in W, \forall h \in H, \forall i \in V \setminus B \setminus \{1, n+2\} \tag{8}$$



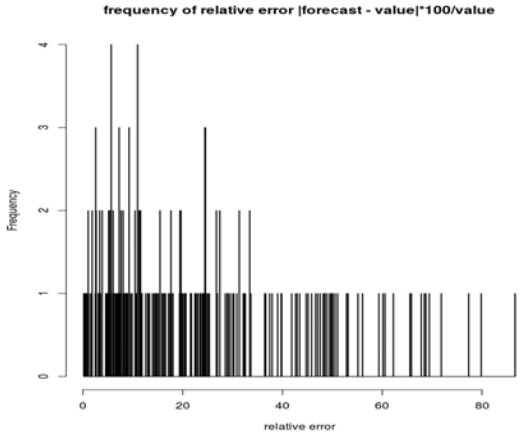
(a)



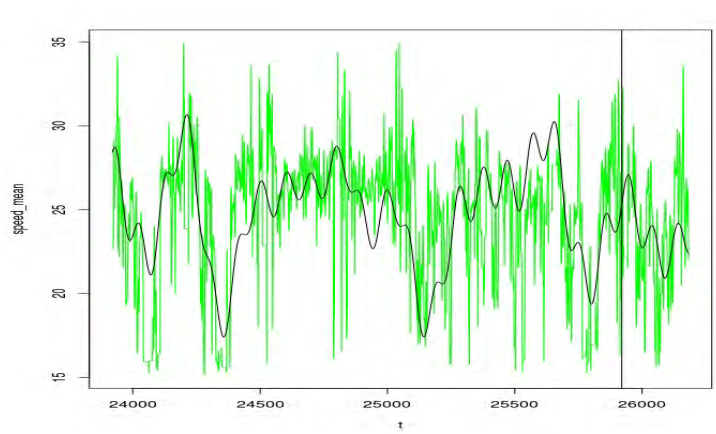
(b)



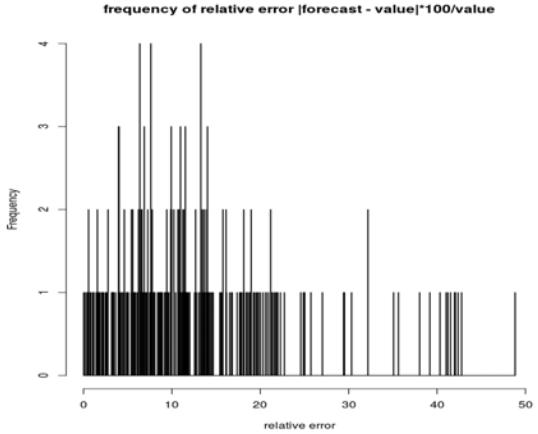
(c)



(d)



(e)



(f)

Figure 8: Path mean speed forecasting

$$c_{2i}^{w} = \sum_{h \in H} c_{2i}^{wh} \qquad \forall w \in W, \forall i \in V \setminus B \setminus \{1, n+2\} \tag{9}$$

$$c_{2r} = \sum_{(w,i) \in r} c_{2i}^{w} \qquad \forall r \in \Omega \tag{10}$$

The costs of early arrival at store j within time window y (11) are defined as being greater than the difference between the beginning of time window b_j^y and the time needed to come to this location (j, y) from a previous location (i, w) visited during this route. The time of reaching (j, y) is computed as the sum of the arrival time a_i^w at (i, w), the service time s_i at i and the travelling time t_{ij} from i to j. Furthermore, these costs should also be greater than 0 because there is no intention to incur negative costs (profit).

$$c_{3j}^{y} \geq b_j^y - (a_i^w + s_i + t_{ij}), c_{3j}^{y} \geq 0 \tag{11}$$

$$c_{3r} = \sum_{(y,j) \in r} c_{3j}^{y} \qquad \forall r \in \Omega \tag{12}$$

The costs of waiting at store i within time window w for any pair of routes r, q (13) are defined as being greater than the difference between departure de_{ri}^{w-1} from this store i within the previous time window $w-1$ following route r and arrival a_{qi}^{w} at this store i within time window w following route q. Furthermore, these costs should also be greater than 0 because there is no intention to incur negative costs (profit). The costs of waiting for pair of routes r, q (14) are computed as the sum of waiting costs for all locations visited by these routes.

$$c_{3rqi}^{w} \geq de_{ri}^{w-1} - a_{qi}^{w}, c_{3rqi}^{w} \geq 0 \qquad \forall w \in W \setminus \{1\}, \forall i \in V \setminus B \setminus \{1, n+2\}, \forall r, q \in \Omega \tag{13}$$

$$c_{3rq} = \sum_{\forall w \in W \setminus \{1\}} \sum_{\forall i \in V \setminus B \setminus \{1, n+2\}} c_{3rqi}^{w} \qquad \forall r, q \in \Omega \tag{14}$$

The last type of costs turns the problem into a non-linear (quadratic) formulation unless the following is defined:

$$x_{rq} = \begin{cases} x_r x_q & \text{if } r \neq q \\ x_r & \text{otherwise} \end{cases} \qquad c_{1rq} = \begin{cases} c_{1r} & \text{if } r = q \\ 0 & \text{otherwise} \end{cases}$$

$$c_{2rq} = \begin{cases} c_{2r} & \text{if } r = q \\ 0 & \text{otherwise} \end{cases} \qquad c_{3rq} = \begin{cases} c_{3r} & \text{if } r = q \\ 0 & \text{otherwise} \end{cases} \qquad \forall r, q \in \Omega \tag{15}$$

Given these definitions, the objective function is formulated as follows:

$$\min \sum_{r \in \Omega} (\gamma_1 c_{1rr} + \gamma_2 c_{2rr} + \gamma_3 c_{3rr}) x_{rr} + \gamma_3 \sum_{r \in \Omega} \sum_{q \in \Omega} c_{3rq} x_{rq} \tag{16}$$

subject to:

$$\sum_{k \in K} \sum_{l \in L} \sum_{r \in \Omega} \alpha_{ri}^{klw} x_{rr} = z_i^w \qquad \forall w \in W, \forall i \in V \setminus B \setminus \{1, n+2\} \tag{17}$$

$$0 \leq \sum_{r \in \Omega} x_{rr} \leq K_{\max} \tag{18}$$

$$x_{rq} \in \{0, 1\} \qquad \forall r, q \in \Omega \tag{19}$$

$$x_{rq} = x_{qr} \qquad \forall r, q \in \Omega \tag{20}$$

$$x_{rq} >= x_{rr} + x_{qq} - 1 \qquad \forall r, q \in \Omega \tag{21}$$

The objective function (16) aims to minimize the sum of costs for all routes that are included in a solution. These costs are weighted by coefficients γ which determine the relative importance of each type of costs. Constraint (17) guarantees that all stores that have demand for delivery of goods are visited. The total number of routes in a solution should not exceed a certain threshold (18). The above formulation is named the Integer Master Problem (IMP).

5 Algorithm

5.1 Column Generation

It is technically difficult to explicitly solve the IMP because the number of feasible routes becomes very large even for instances of a moderate size. For a given day in June 2016 there are 71 stores and 125 "locations" (combinations of a store and a time window) so that the upper bound of the number of feasible solutions is about 125! which is a huge number. Of course, in reality this number is much smaller because it is unlikely that a vehicle will visit more than 2 stores during one route, because its capacity is relatively small compared to demand. Nevertheless, this number is still large so a special technique is applied to solve this problem. This technique is called Column Generation (Hu and Kahng, 2016).

The idea of Column Generation is to build Ω iteratively by adding new promising routes (columns) which could potentially improve a current solution. In order to generate new columns a linear relaxation of the IMP is formulated by replacing (19) with:

$$x_r \geq 0 \qquad \forall r \in \Omega \tag{22}$$

This relaxation is named the Linear Master Problem (LMP) and it allows application of the Simplex method to search for new promising columns. These new columns being added to Ω should have potential to improve the objective function (16). In terms of the Simplex method it means that they should have negative reduced cost. The reduced cost of a route is computed as follows:

$$\begin{aligned} \hat{c_r} &= c_r - \beta^T a_r - \beta_0 - \beta_{K_{\max}} \\ c_r &= c_{1r} + c_{2r} + c_{3r} \\ a_r &= \sum_{k \in K} \sum_{l \in L} a_{ri}^{klw} \qquad \forall w \in W, \forall i \in V \setminus B \setminus \{1, n+2\} \end{aligned} \tag{23}$$

In this formulation, c_r is the total cost of route r, β is the vector of $|W||V \setminus B \setminus \{1, n+2\}|$ dual variables corresponding to (17), a_r is the vector of the same cardinality as β such that each element indicates whether store i is visited within time window w following route r, β_0 and $\beta_{K_{\max}}$ are the dual variables related to both parts of inequality (18) respectively.

Therefore, the search for new promising columns is an optimization problem which is formulated as follows:

$$\begin{aligned} &\min_{r \in \Omega} \hat{c_r} \\ &\text{subject to all constraints related to a route} \end{aligned} \tag{24}$$

This problem is named the Pricing Sub-Problem.

5.2 Dynamic Programming Formulation for Pricing Sub-Problem

The Pricing Sub-Problem is solved by the Dynamic Programming Algorithm as it is proposed in Ropke and Cordeau(2009). First, the state space is defined at any location $(i, w), i \in V \setminus \{1, n+2\}, w \in W$ as the set of the following parameters:

- k - the type of vehicle k
- a_i^w - the arrival time
- $\hat{c}_i^w$ - the partial reduced cost of arrival
- $\{q_i^{wh} | \forall h \in H\}$ - the amount of goods of type h for which the cargo space is reserved
- S_i^w - the set of locations that have been visited before arriving at (i, w)

Given these definitions, the value function is formulated as follows:

$$\begin{aligned} &V_{iw}(k, a_i^w, \hat{c}_i^w, \{q_i^{wh} | \forall h \in H\}, S_i^w) = \\ &\min_{(j,y) \in (V,W)}((j = (n+2)) \wedge \hat{c}_j^y \vee (j \neq (n+2)) \wedge \end{aligned}$$

$$\min(V_{jy}(k, a_j^y, \hat{c}_j^y, \{q_j^{yh}|\forall h \in H\}, S_j^y), V_{jy}(k, ab_j^y, \hat{cb}_j^y, \{q_j^{yh}|\forall h \in H\}, SB_j^y)) \tag{25}$$

Subject to:

$$a_j^y = \max(b_j^y, a_i^w + s_i + t_{ij}), ab_j^y = \max(\frac{b_j^y + e_j^y}{2}, a_i^w + s_i + t_{i,bu} + t_{bu,j}) \qquad (i,j,bu) \in NB \tag{26}$$

$$u_j^{yh} = \min(d_j^{yh}, \max_{o \in H} Q^{ko} - \sum_{o \in H} q_i^{wo} - \mathbb{1}_{h>1} \sum_{o=1..(h-1)} \hat{u}_j^{yo}),$$

$$\hat{u}_j^{yh} = \left\lceil \frac{\max_{o \in H} Q^{ko}}{Q^{kh}} u_j^{yh} \right\rceil \qquad h = 1..|H| \tag{27}$$

$$q_j^{yh} = q_i^{wh} + \hat{u}_j^{yh} \qquad \forall h \in H \tag{28}$$

$$\hat{c}_j^y = \hat{c}_i^w + c_{1ij} + c_{2j}^y + c_{3j}^y - \beta_j^y, \hat{cb}_j^y = \hat{c}_i^w + c_{1,i,bu} + c_{1,bu,j} + c_{2j}^y - \beta_j^y \tag{29}$$

$$S_j^y = S_i^w \cup \{(j,y)\}, SB_j^y = S_i^w \cup \{(bu,1),(j,y)\} \tag{30}$$

Stopping criteria:

$$a_j^y > e_j^y, ab_j^y > e_j^y \tag{31}$$

$$j \neq (n+2) \wedge \sum_{h \in H} u_j^{yh} = 0 \tag{32}$$

$$(j,y) \in S_i^w \tag{33}$$

In this formulation, $V_{wi}(k, a_i^w, \hat{c}_i^w, \{q_i^{wh}|\forall h \in H\}, S_i^w)$ is the minimal reduced cost of reaching the destination depot $n+2$ from the current position at store i within time window w. $V_{w1}(k, 0, -\beta_0 - \beta_{K_{\max}}, \{0|\forall h \in H\})$ gives the minimal reduced cost of travelling from the origin depot 1 to the destination depot $n+2$.

The equality (26) along with inequality (31) assure that the current vehicle arrives to store j within time window y but not earlier than it finishes serving the previous store i within time window w taking into account travelling time from store i to store j. The amount of goods u_j^{yh} of each type $h \in H$, that is delivered to store j within time window y, and the corresponding amount of cargo space that needs to be reserved $\hat{u}_j^{yh}$, are computed by (27). It is assumed that the types of goods are ordered according to their priority $h = 1..|H|$. The reserved capacity of the current vehicle after visiting store j is calculated by (28). The equality (29) gives the partial reduced cost of this route up until reaching store j within time window y. A recursion step is not made if any goods can not be delivered to the next location, either because the current vehicle capacity is exceeded or there is no demand for delivery of goods (32).

The value function formulation (25) is extended to deal with buffering locations. In order to do so, a second V_{jy} term and minimum operator are added. This second V_{jy} represents an option when instead of travelling directly from i to j a truck goes to buffer bu first. The minimum among two V_{jy} indicates that both options are considered, that is to say, direct travel from i to j as well as travel through the nearest buffer bu, and the option with minimal total reduced costs is chosen. Next, It is (heuristically) assumed that the waiting costs due to overlapping arrivals/departures for the entire solution will get lower if a truck arrives to a store from its nearest buffer not earlier than in the middle of the store's time window. Given this assumption, the arrival time equality (26) is adjusted accordingly. The partial reduced cost equality (29) is also extended for this case to take into account the travelling costs to and from buffer bu, and to exclude the early arrival costs c_{3j}^y which can not occur in this case.

The value function (25) needs to be solved $|K|$ times but not necessarily to optimality because in practice a limited number of new promising columns during are needed during each Column Generation iteration.

5.3 Route Construction Heuristic

The Column Generation technique described in subsection 5.1 requires a good initial set of promising columns Ω to start with. A regret construction heuristic is applied in order to populate this set. The idea of this

heuristic (Floudas and Pardalos, 2009) is to extend a route by visiting a location which would cause the biggest regret if this location was not selected as the next destination.

To start with, the last visited store i is defined within time window w as location l, and the set of remaining unvisited locations as R. Then, all the possible route extensions $l \rightarrow u \rightarrow v, \forall v \in R, u \in R \setminus \{v\}$ are evaluated. For each of these extensions the measure of regret is computed, and after that next route location v is chosen which has the highest value of this measure.

Different regret measures are used. First, regret is defined as the amount of demand which exceeds the remaining vehicle capacity:

$$\max \left(\sum_{o \in H} d_i^{oh} - (\min_{o \in H} Q^{ko} - \sum_{o \in H} q_i^{wo}), 0 \right) \tag{34}$$

Next, another option is considered by defining regret as the amount of time which a vehicle is late travelling from location (i, w) to location (j, y):

$$e_i^w + s_i + t_{ij} - e_j^y + t_{\max} \in [0, 2t_{\max}] \tag{35}$$

5.4 Local Search

In case if the algorithm gets stuck not being able to improve current best solution, a local search procedure is applied in order to explore the search space and hopefully escape from the current local optimum. This procedure is based on the shaking of current solution, which applied as follows:

1. Up to 100 pairs of routes of more than one location each are uniformly selected which belong to the current best solution, then the Regret Construction Heuristic (subsection 5.3) is applied to the set of locations corresponding to each pair.
2. Up to 100 triples of routes of more than one location each are uniformly selected which belong to the current best solution, then the Regret Construction Heuristic (subsection 5.3) is applied to the combined set of each pair of locations.

5.5 Summary

Based on the ideas of the previous subsections, the following algorithm is proposed to solve the model formulated in section 4:

1. An initial solution is generated by applying the Regret Construction Heuristic (subsection 5.3), then this initial solution is shaken by applying the Local Search procedure (subsection 5.4).
2. The LMP is iteratively solved which is defined in subsection 5.1 considering the optimal solution of this relaxation as the lower bound. The improvement is defined as the fact of getting better (smaller) value of the lower bound.
    - 2.1. If there are no improvements of the LMP lower bound in 3 consecutive iterations, the IMP is solved which is defined in subsection 5.1, non basic routes are dropped and the current best solution is shaken by applying the Local Search procedure (subsection 5.4). If this shaking gives better solution then a new iteration is started, otherwise the procedure stops and reports the best obtained solution.
    - 2.2. If there are some improvements in the LMP lower bound, up to 100 new promising routes are obtained for each vehicle type by solving the defined in subsection 5.2 Pricing Sub-Problem using current LMP duals values. If there are no new routes then the procedure loops back to step (2.1.).
    - 2.3. The Pricing Sub-Problem is defined as being solved if there is no vehicle type for which exactly 100 new promising routes were obtained. If this is the case the procedure proceeds to the last iteration solving the LMP and the Pricing Sub-Problem only one more time, and then applying step (2.1.).

This algorithm gives a heuristic sub-optimal solution.

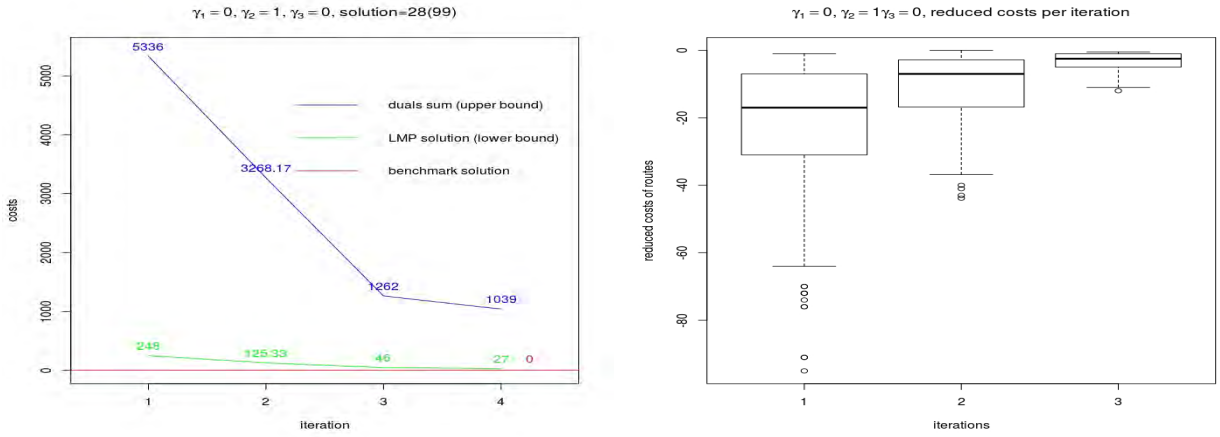
6 Results

The proposed algorithm is prototyped in Python and Gurobi is used as a linear solver. All the tests are performed on a PC with 4x Intel(R) Core(TM) i3-2350M CPU @ 2.30GHz and 16Gb RAM.

6.1 Experiments

6.1.1 $\gamma_1 = 0, \gamma_2 = 1.0, \gamma_3 = 0$

First, the simplest option is considered when the only costs which are taken into account are the demand satisfaction costs c_2. For this particular case the optimal solution is known (0) because according to the experts' requirement the stores' demand is estimated as the proportion of trucks' capacity. Therefore, it is known that the shipment plan exists which is capable of completely satisfying demand for delivery of goods. This knowledge gives an opportunity to test the ability of the model to generate (near) optimal delivery plan because the model uses only demand as input but not the known shipment plan. From Figure 9 it is observed that, despite the algorithm demonstrates a good convergence, only a sub optimal solution of 28 is obtained, which means that $(4421 - 28)/4421 = 0.9937$ of demand is satisfied.



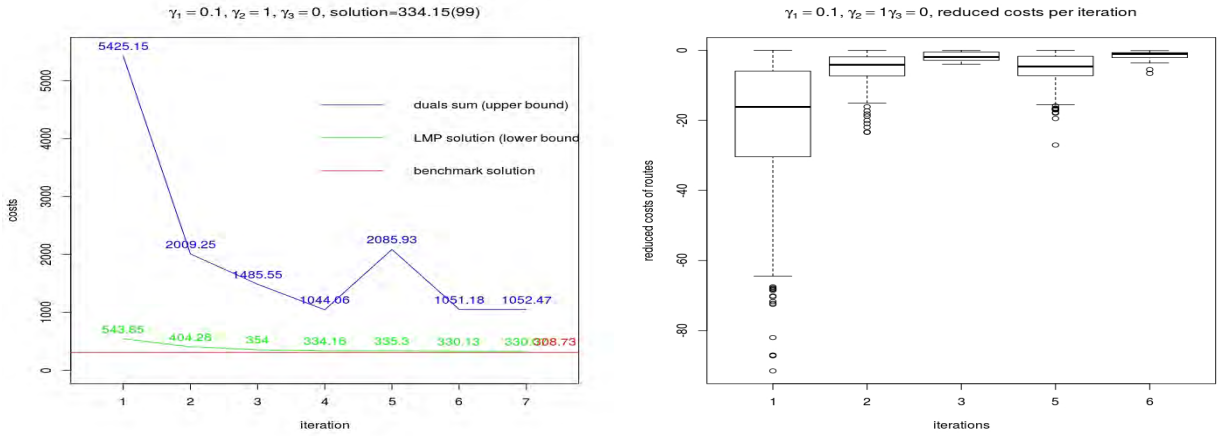
(a) The solution value convergence

(b) The reduced costs convergence

Figure 9: $\gamma_1 = 0, \gamma_2 = 1.0, \gamma_3 = 0$

6.1.2 $\gamma_1 = 0.1, \gamma_2 = 1.0, \gamma_3 = 0$

The travelling costs c_1 are added into consideration. Furthermore, it is assumed that one container that is not delivered costs as much as 10 travelled kilometres ($\gamma_1 = 0.1$). Figure (10) indicates that a slightly worse solution is obtained which is $334.1512/308.7328 = 1.0823$ multiplied by benchmark. Moreover, the c_2 component of the solution is 30 so that (less) $(4421 - 30)/4421 = 0.9932$ of demand is satisfied. On the other hand, this solution has c_1 of 304.1512 which is slightly less compared to that of the benchmark solution (308.7328).



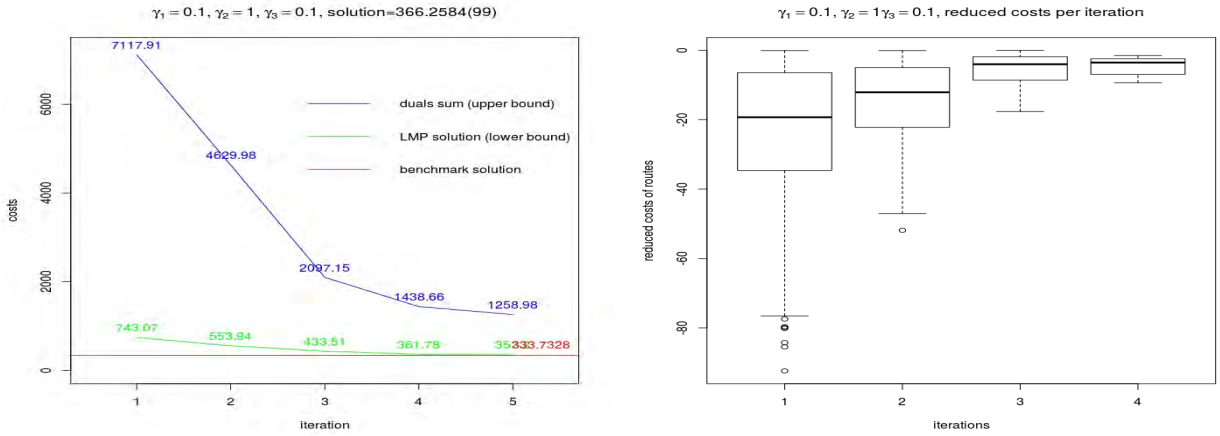
(a) The solution value convergence

(b) The reduced costs convergence

Figure 10: $\gamma_1 = 0.1, \gamma_2 = 1.0, \gamma_3 = 0$

6.1.3 $\gamma_1 = 0.1, \gamma_2 = 1.0, \gamma_3 = 0.1$ without buffers

The waiting time costs c_3 are added into consideration regarding those as being 10 times less important as the demand satisfaction costs c_2 (or just as important as the travelling costs c_1). In other words, it is assumed that one container that is not delivered costs as much as 10 minutes of waiting ($\gamma_3 = 0.1$). From Figure 11 it is seen that the method smoothly converges in 5 iterations.



(a) The solution value convergence

(b) The reduced costs convergence

Figure 11: $\gamma_1 = 0.1, \gamma_2 = 1.0, \gamma_3 = 0.1$

The costs of the benchmark solution in this case get increased by 25 (333.7328) because this solution has 3 overlapping arrivals/departures (6.2 units in total) which are highlighted in red in Figure 12. Furthermore, there are early arrivals which sum up to 18.8 units (Table 2), no demand satisfaction costs and travelling costs of 308.7328 ($308.7328 + 0 + 18.8 + 6.2 = 333.7328$).

path, (i,w)	sum of early arrivals, in units
(1,1) (18,3) (24,2) (1,1)	1.10
(1,1) (18,2) (24,1) (1,1)	0.60
(1,1) (43,1) (3,2) (1,1)	0.60

path, (i,w)	sum of early arrivals, in units
(1,1) (4,1) (77,2) (1,1)	0.40
(1,1) (71,1) (14,3) (1,1)	0.90
(1,1) (3,3) (75,1) (1,1)	3.20
(1,1) (15,1) (14,2) (1,1)	0.60
(1,1) (3,1) (36,1) (1,1)	0.70
(1,1) (80,1) (22,1) (1,1)	0.70
(1,1) (59,1) (6,1) (1,1)	0.20
(1,1) (77,1) (42,1) (1,1)	0.60
(1,1) (64,1) (39,1) (1,1)	0.60
(1,1) (27,1) (46,1) (1,1)	0.40
(1,1) (41,1) (79,1) (1,1)	0.50
(1,1) (29,2) (21,2) (1,1)	0.80
(1,1) (22,3) (33,1) (1,1)	0.40
(1,1) (80,2) (44,2) (1,1)	0.50
(1,1) (56,1) (15,4) (1,1)	1.00
(1,1) (32,1) (4,2) (1,1)	1.00
(1,1) (45,1) (35,2) (1,1)	0.60
(1,1) (63,1) (8,1) (1,1)	0.20
(1,1) (50,2) (68,1) (1,1)	2.10
(1,1) (79,2) (70,1) (1,1)	0.20
(1,1) (44,1) (38,1) (15,3) (1,1)	0.90

Table 2: Benchmark solution early arrivals $\gamma_1 = 0.1, \gamma_2 = 1.0, \gamma_3 = 0.1$

On the other hand, the model gives solution 366.2584 (366.2584/333.7328 = 1.0975 times the benchmark solution) without overlapping arrivals/departures (Figure 13),with 18.3 units of early arrivals (Table 3), but 39 ((4421 − 39)/4421 = 0.9912) containers which are not delivered and 308.9584 units of travelling costs (308.9584 + 39 + 18.3 + 0.0 = 366.2584)

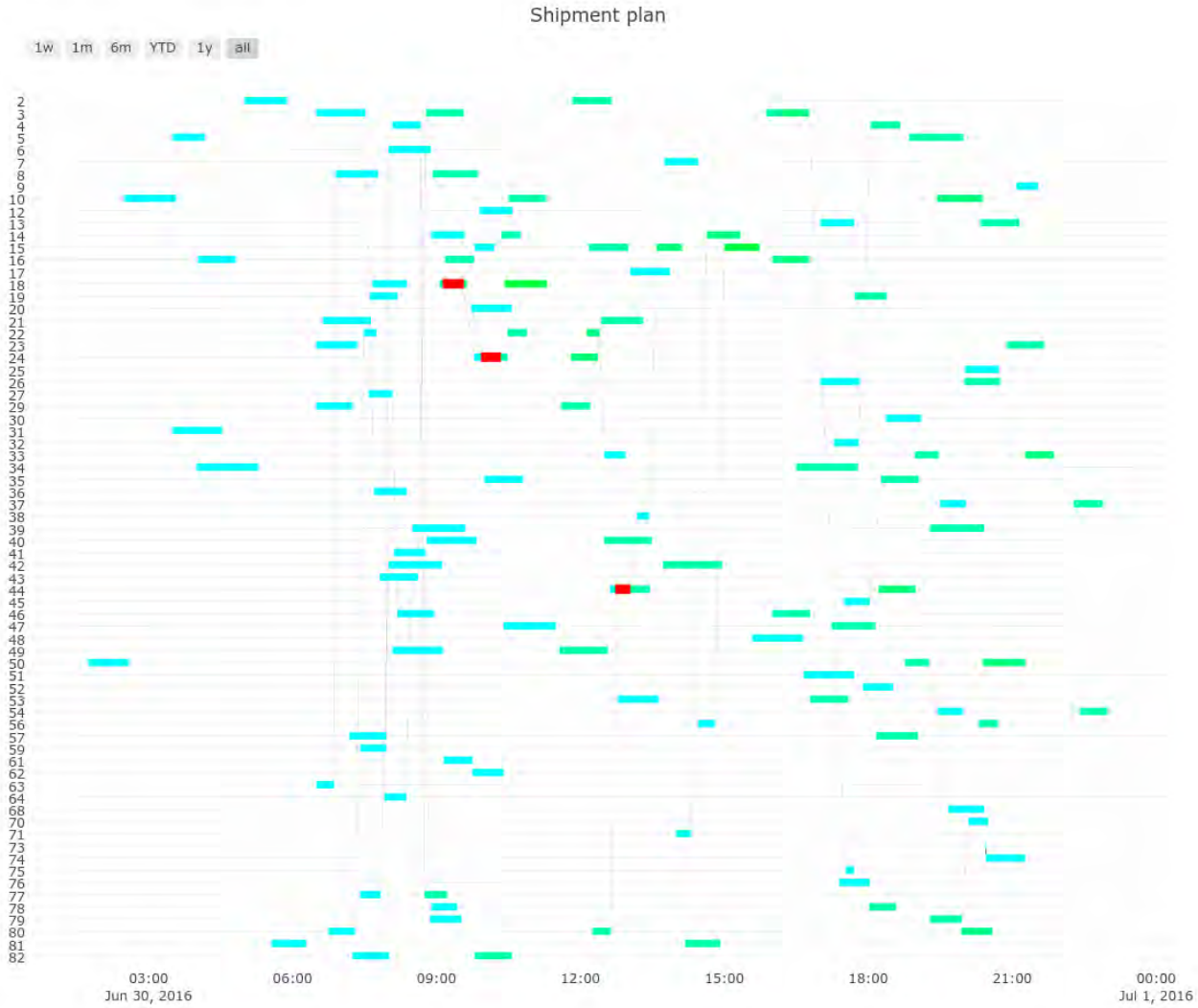


Figure 12: Benchmark Shipment plan $\gamma_1 = 0.1, \gamma_2 = 1.0, \gamma_3 = 0.1$

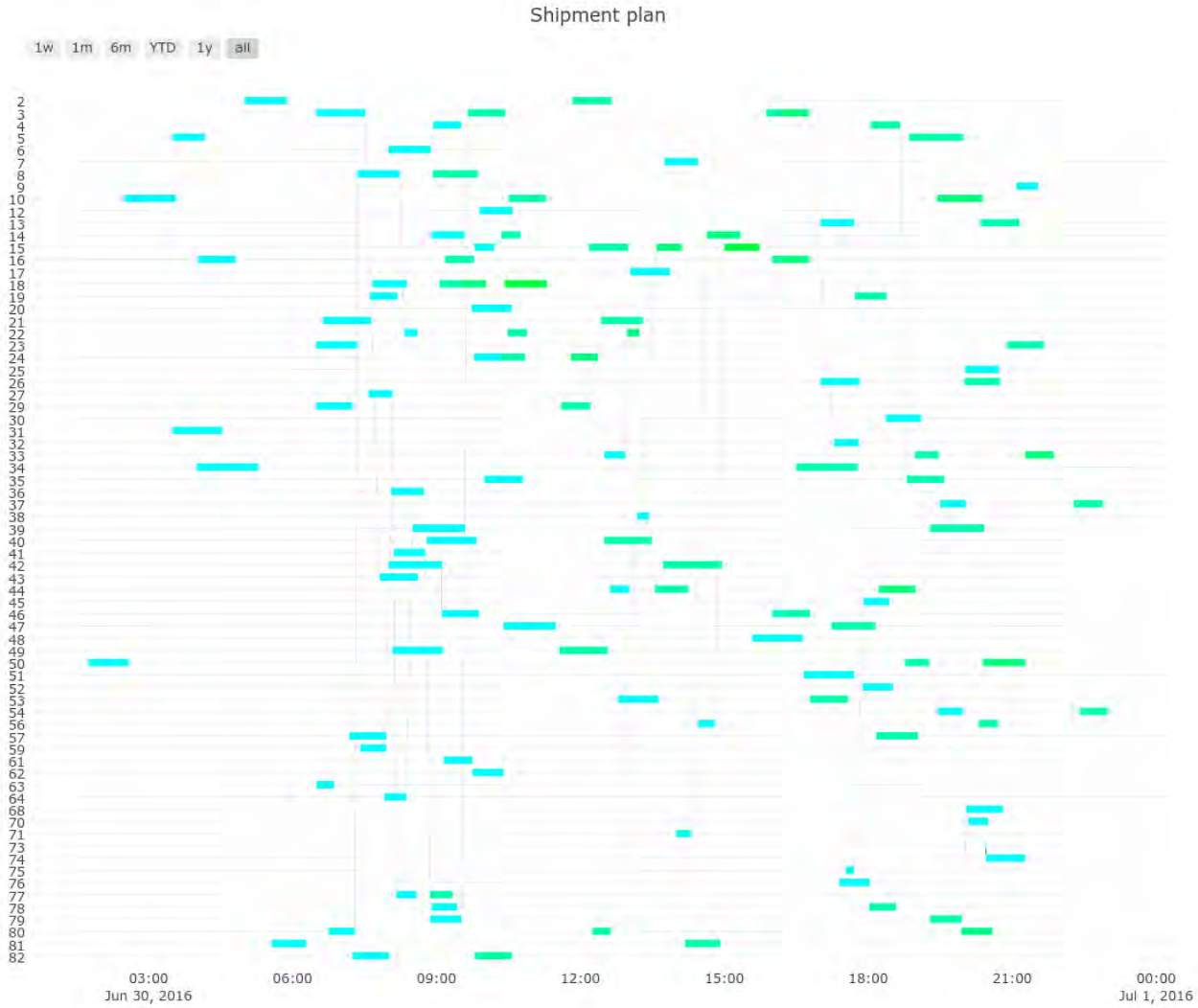


Figure 13: Generated Shipment plan $\gamma_1 = 0.1, \gamma_2 = 1.0, \gamma_3 = 0.1$

path, (i,w)	sum of early arrivals, in units
(1,1) (3,1) (43,1) (1,1)	1.50
(1,1) (80,2) (15,3) (1,1)	5.60
(1,1) (64,1) (39,1) (1,1)	0.60
(1,1) (15,1) (14,2) (1,1)	0.60
(1,1) (44,1) (38,1) (44,2) (1,1)	0.40
(1,1) (56,1) (15,4) (1,1)	1.00
(1,1) (71,1) (14,3) (1,1)	0.90
(1,1) (29,2) (21,2) (1,1)	0.80
(1,1) (3,3) (32,1) (1,1)	2.30
(1,1) (50,2) (70,1) (1,1)	4.60

Table 3: Generated solution early arrivals $\gamma_1 = 0.1, \gamma_2 = 1.0, \gamma_3 = 0.1$

6.1.4 $\gamma_1 = 0.1, \gamma_2 = 1.0, \gamma_3 = 0.1$ with linked buffers

Next, four buffering locations linked to particular stores are considered across the Amsterdam city area depicted in green in Figure 14. Those buffers have no associated early arrivals and overlapping arrivals/departures costs.

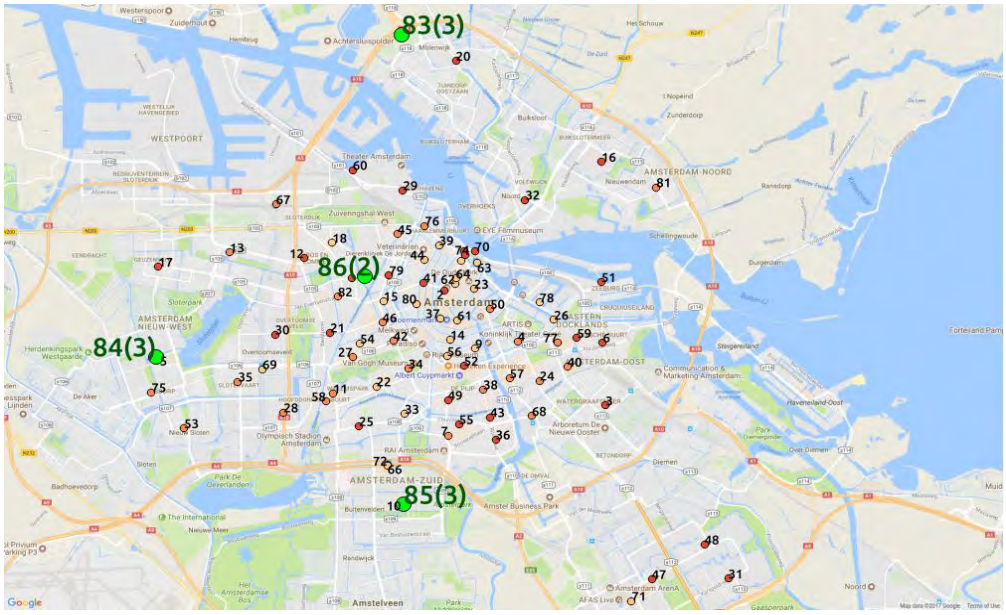
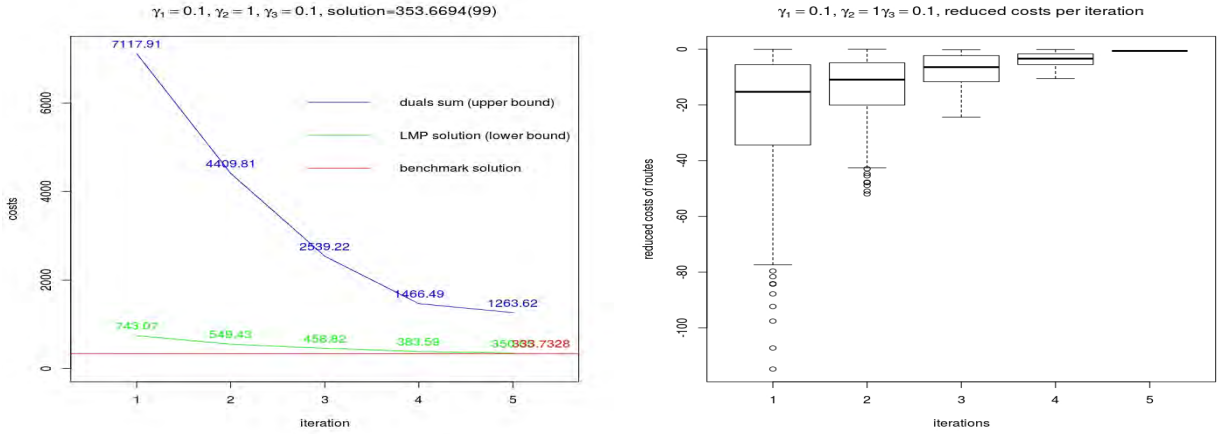


Figure 14: European retailer stores and linked buffers in Amsterdam, in i(k) format

From Figure 15 it can be observed that the method converges in 5 iterations, the same as the experiment without buffers.



(a) The solution value convergence

(b) The reduced costs convergence

Figure 15: $\gamma_1 = 0, \gamma_2 = 1.0, \gamma_3 = 0$ with linked buffers

The model gives solution 353.6694 (353.6694/333.7328 = 1.0597 times the benchmark solution) with only 0.1 units of overlapping arrivals/departures (Figure 16), 11.0 units of early arrivals (Table 4), 33 ((4421 − 33)/4421 = 0.9925) containers which are not delivered and 309.5694 units of travelling costs (309.5694+33+ 11.0+0.1 = 353.6694). It is seen that only one buffer 86 is utilized. This buffer is linked to stores 15, 34, 80.

Furthermore, it is noticed that there is no drop in demand satisfaction (33 compared to 39 for 6.1.3) while lower waiting costs c_3 are obtained ($11.0 + 0.1 = 11.1$ compared to $18.3 + 0.0 = 18.3$ for 6.1.3).

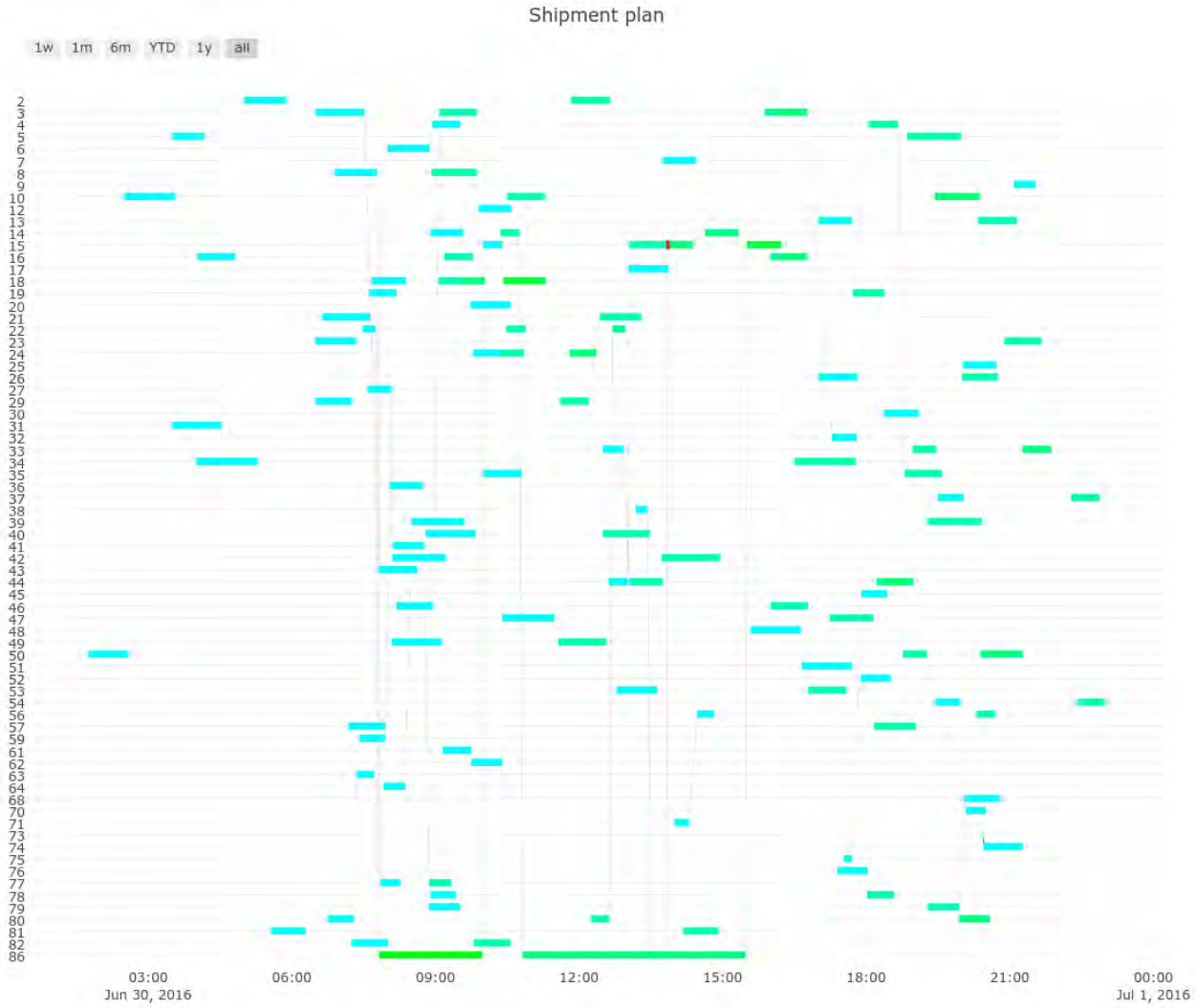


Figure 16: Generated Shipment plan $\gamma_1 = 0.1, \gamma_2 = 1.0, \gamma_3 = 0.1$ with linked buffers

path, (i,w)	sum of early arrivals, in units
(1,1) (3,1) (43,1) (1,1)	1.50
(1,1) (64,1) (39,1) (1,1)	0.60
(1,1) (38,1) (86,1) (15,3) (14,3) (1,1)	1.20
(1,1) (29,2) (21,2) (1,1)	0.80
(1,1) (50,2) (70,1) (1,1)	4.60
(1,1) (3,3) (32,1) (1,1)	2.30

Table 4: Generated solution early arrivals $\gamma_1 = 0.1, \gamma_2 = 1.0, \gamma_3 = 0.1$ with linked buffers

6.1.5 $\gamma_1 = 0.1, \gamma_2 = 1.0, \gamma_3 = 0.1$ with commonly used buffers

Next, ten commonly used buffering locations are considered across the Amsterdam city area depicted in green in Figure 17. The commonly used buffers, just as the linked buffers, have no associated early arrivals and overlapping arrivals/departures costs.

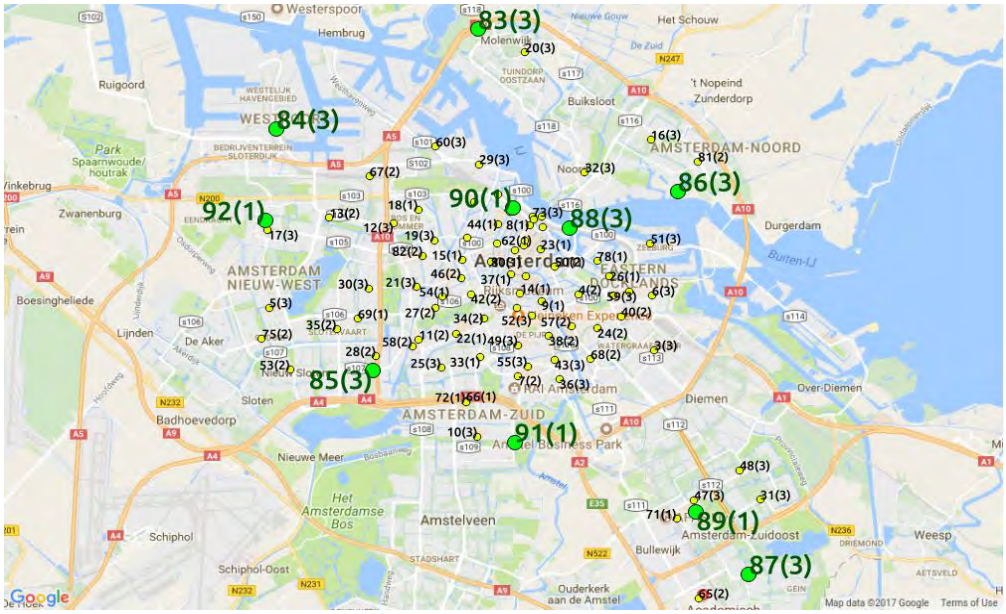
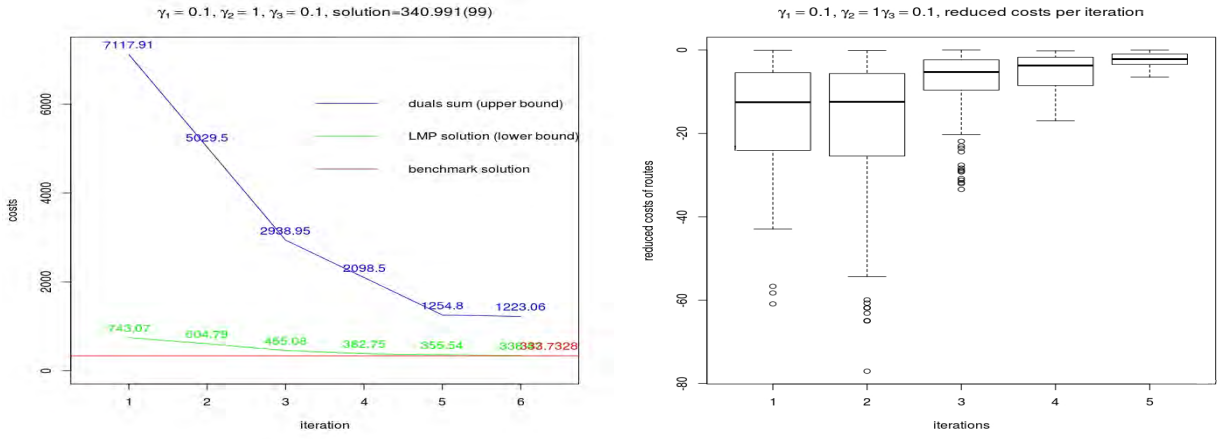


Figure 17: European retailer stores and buffers in Amsterdam, in i(k) format

From Figure 18 it is observed that the method converges in 6 iterations which is 1 more compared to the two cases without buffers and with linked buffers.



(a) The solution value convergence

(b) The reduced costs convergence

Figure 18: $\gamma_1 = 0, \gamma_2 = 1.0, \gamma_3 = 0$ with commonly used buffers

The model gives solution 340.9910 (340.9910/333.7328 = 1.0217 times the benchmark solution) with only 0.1 units of overlapping arrivals/departures (Figure 19), 4.4 units of early arrivals (Table 5), 26 ((4421 − 26)/4421 = 0.9941) containers which are not delivered and 310.491 units of travelling costs (310.491 + 26 + 4.4 + 0.1 = 340.9910). It is seen that 4 out of 10 buffers are utilized such as 88, 89, 90, 91. Furthermore, it is

noticed that there is no drop in demand satisfaction (26 compared to 39 for 6.1.3) while lower waiting costs c_3 are obtained ($4.4 + 0.1 = 4.5$ compared to $18.3 + 0.0 = 18.3$ for 6.1.3 and $11.0 + 0.1 = 11.1$ for 6.1.4).

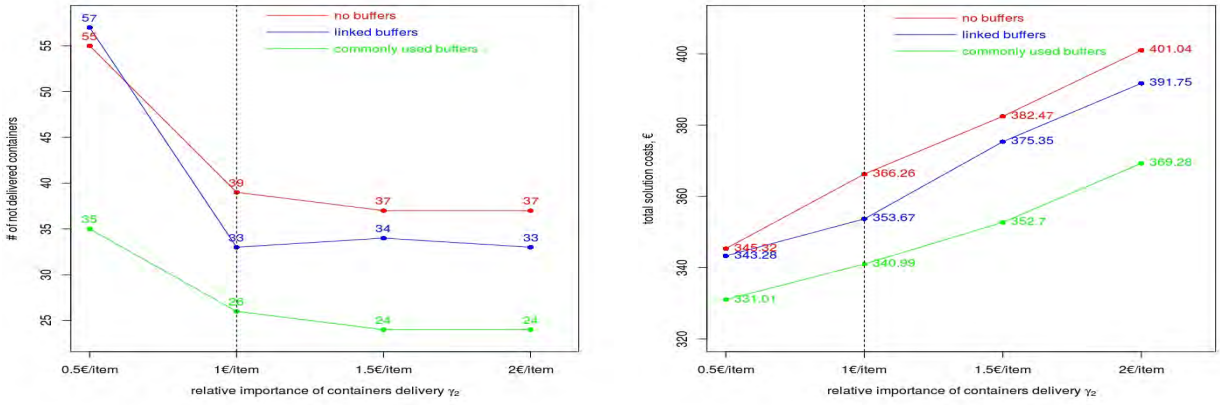
Figure 19: Generated Shipment plan $\gamma_1 = 0.1, \gamma_2 = 1.0, \gamma_3 = 0.1$ with commonly used buffers

path, (i,w)	sum of early arrivals, in units
(1,1) (3,1) (43,1) (1,1)	1.50
(1,1) (38,1) (88,1) (32,1) (50,2) (1,1)	2.40
(1,1) (8,1) (90,1) (33,1) (15,3) (1,1)	0.50

Table 5: Generated solution early arrivals $\gamma_1 = 0.1, \gamma_2 = 1.0, \gamma_3 = 0.1$ with commonly used buffers

6.1.6 Relative importance of containers delivery

The relative importance γ_2 of delivery of containers is varied from 0.5 to 2.0 with step 0.5 and the effect of this change on the demand satisfaction and total solution costs is analysed. The results of these experiments are represented in Figure 20. From the plot 20.a it is seen that the number of containers not delivered first goes down significantly when γ_2 is changed from 0.5 to 1.0, and after that it flattens when γ_2 is increased further from 1.0 to 1.5,2.0. On the other hand, plot 20.b demonstrates that the total solution costs increase linearly while γ_2 is changed from 0.5 to 2.0.



(a) Demand satisfaction

(b) Total solution costs

Figure 20: Effects of containers delivery importance γ_2

6.2 Summary

The results of conducted experiments are represented in Table 6

	benchmark solution					generated solution				
	$\gamma_1 c_1$	$\gamma_2 c_2$	$\gamma_3 c^y_{3j}$	$\gamma_3 c_{3rq}$	$\sum$	$\gamma_1 c_1$	$\gamma_2 c_2$	$\gamma_3 c^y_{3j}$	$\gamma_3 c_{3rq}$	$\sum$
$\gamma_1 = 0.0, \gamma_2 = 1.0, \gamma_3 = 0.0$	0.00	0.00	0.00	0.00	0.00	0.00	28.00	0.00	0.00	28.00
$\gamma_1 = 0.1, \gamma_2 = 1.0, \gamma_3 = 0.0$	308.73	0.00	0.00	0.00	308.73	304.15	30.00	0.00	0.00	334.15
$\gamma_1 = 0.1, \gamma_2 = 1.0, \gamma_3 = 0.1$ without buffers	308.73	0.00	18.80	6.20	333.73	**308.96**	**39.00**	**18.30**	**0.00**	**366.26**
$\gamma_1 = 0.1, \gamma_2 = 1.0, \gamma_3 = 0.1$ with linked buffers	308.73	0.00	18.80	6.20	333.73	**309.57**	**33.00**	**11.00**	**0.10**	**353.67**
$\gamma_1 = 0.1, \gamma_2 = 1.0, \gamma_3 = 0.1$ with commonly used buffers	308.73	0.00	18.80	6.20	333.73	**310.49**	**26.00**	**4.40**	**0.10**	**340.99**

Table 6: Experiments summary

It is seen that use of linked buffering locations allows reduction of the waiting times costs c_3 from $18.3+0.0 = 18.3$ to $11.0 + 0.1 = 11.1$ units while the number of containers not delivered decreases from 39 to 33. Furthermore, utilization of commonly used buffering locations allows reduction of the waiting times costs c_3 even further to $4.4 + 0.1 = 4.5$ while the number of containers not delivered also decreases to 26. The reduction in waiting times is achieved due to decrease in the early arrivals costs c^y_{3j} from 18.3 to 11.0 and 4.4 respectively while the overlapping arrivals/departures costs c_{3rq} slightly increase from 0.0 to 0.1. Therefore, the hypothesis is true. Moreover, commonly used buffers give higher cost reduction than buffering locations linked to stores.

The relative importance is defined for the travelling costs c_1, the demand satisfaction costs c_2 and the waiting times costs c_3 as $\gamma_1 = 0.1$€/km, $\gamma_2 = 1.0$€/item and $\gamma_3 = 0.1$€/minute respectively. This definition allows to conclude that utilization of commonly used buffers leads to total cost savings of $366.26 - 340.99 = 25.27$€ per day, which is $(366.26 - 340.99)/366.26 = 6.9\%$ of the total solution costs without using buffers. Furthermore, the waiting times costs for the case with commonly used buffers are also reduced by $18.3 - 4.5 = 13.8$€ which is $(18.3 - 4.5)/18.3 = 7.5\%$ of the waiting times costs without using buffers. On the other hand, the travelling costs get slightly increased by $310.49 - 308.96 = 1.53$€ (15.3km more kilometres are travelled) which is $(310.49 - 308.96)/308.96 = 0.0049\%$

It is observed that linked buffer number 86 and commonly used buffer numbers from 88 to 91 are utilized but the remaining ones are not so it gives an indicator of which buffering locations are of higher importance. Furthermore, the stores are identified which require utilization of buffers before visiting them $(14, 15, 22, 26, 32, 33, 44, 70, 71)$. Thus, the experiment allows identification of utilized buffers as well as buffer-prone stores (Figure 21).

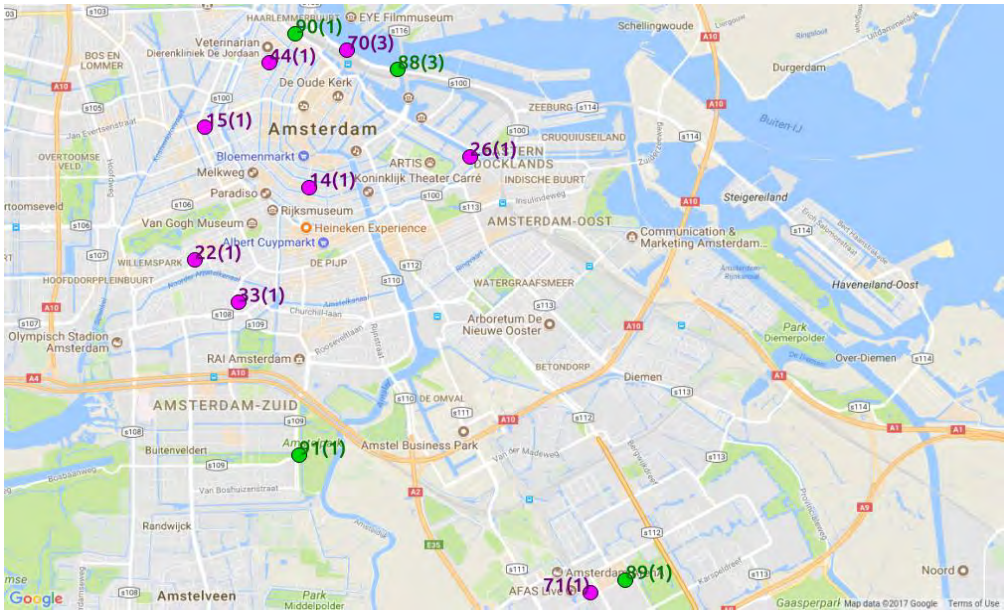


Figure 21: Utilized buffers (in green) and buffer-prone stores (in purple), in i(k) format

7 Conclusion and recommendations

It is seen that buffering locations help to reduce waiting costs without drop in total solution costs. Furthermore, commonly used buffers have higher solution cost reduction than linked buffers. In particular, these shared buffers allow savings of 7.5% of the waiting times costs and 6.9% of the total solution costs respectively, compared to the case without buffers. The solution with cost reduction is obtained at the expense of small increase in travelling costs (0.0049%).

It is observed that there are two clearly separable intervals of the relative importance γ_2 of demand satisfaction such as $[0.5, 1.0]$ and $(1.0, 2.0]$. γ_2 greater than 1.0€/item corresponds to the case when delivery costs dominate other types of costs thus there is no way to consider partial delivery. γ_2 less than or equal to 1.0€/item stands for the case when delivery costs are comparable to other types of costs so it becomes possible to trade partial demand satisfaction for shorter travelling distances and/or smaller waiting times. Therefore, smaller relative importance of demand satisfaction costs leads to the possibility of considering partial delivery.

It is recommended to use the implemented model to assess the usefulness of the buffering locations in the Amsterdam area. In order to do so, a number of experiments might be conducted for different shipment dates, and the usage of buffers could be monitored. Based on these statistics more precise inference would be possible to determine which buffers could be established.

It is recommended to consider this model as an alternative option to perform daily routes planning. Firstly, it can provide comparable results to the benchmark solution. Secondly, it is flexible for extension in general and for experimenting in particular with different importance of relative costs.

It is recommended to extend this model to take into account drivers' shift scheduling and stores' efficiency.

It is recommended to extend this model in order to deal with the cases when more than one truck is allowed within the same time window at the same store. This will make the model more realistic.

It is recommended to extend the numerical method described in chapter 5 in order to take uncertain travel-

ling times into account. One of the possible ways to estimate the mean travelling speed is proposed in 3.3.4. This will also make the model more realistic.

8 Appendix

Name	Type	Description	% missing	min	max	mean
id	numerical	Identifier of a row	0	1	77258	38629.50
placeId	categorical	Identifier of a Google place	0			
geom	spatial	Spatial coordinates of a directional road segment (EPSG SRID 4326)	0			

Table 7: The Google place id map attributes

Name	Type	Description	% missing	min	max	mean
placeId	categorical	Identifier of a corresponding Google place id	0			
start_interval	numerical	Start of the measurement time epoch, in seconds since 1970-01-01 00:00:00	0	1459468800	1467323700	1463397714
flow_bucket	numerical	Indicator of the relative road traffic intensity	0	0	9	6.62
speed_mean	numerical	Average road travelling speed	0	0	45.00	11.37

Table 8: The Google city flow data attributes

Name	Type	Description	% missing	min	max	mean
voet	numerical	Number of bicycle paths	0	0.00	2.00	0.23
fiets	numerical	Number of foot paths	0	0.00	5.00	0.44
ov	numerical	Number of bus and tram paths	0	0.00	2.00	0.21
auto	numerical	Number of car paths	0	0.00	2.00	0.13
bus_tram	categorical	1 - bus, 2 - tram				
wvk_id	numerical	Identifier of a road	0	219383002	999000386	253049886
snelheid	numerical	Maximal travelling speed	0	0.00	100.00	34.52
geom	spatial	Spatial coordinates of a directed road segment (EPSG SRID 28992)	0			

Table 9: The "Hoofd en plusnetten" map attributes

Name	Type	Description	% missing	min	max	mean
wvk_id	numerical	Identifier of a road	0	27142004	555557003	308030531
bst_code	categorical	FP -bicycle path, VP -foot path				

Table 10: The "Wegvakken" map attributes

Name	Type	Description	% missing	min	max	mean
naam	categorical	Name of a store	0			
x	numerical	Longitude of a store's spatial location	0	4.75	4.98	4.88
y	numerical	Latitude of a store's spatial location	0	52.29	52.44	52.37

Table 11: European retailer's delivery destinations spatial attributes

Name	Type	Description	% missing	min	max	mean
Winkelnr	categorical	Name of a store	0			
Wagentype	categorical	Type of the largest truck which can be parked: EUR - Euro trailer (largest), CIT - City trailer, BAK - Bakwagen (smallest)	0			

Table 12: European retailer's delivery destinations vehicles parameters

Name	Type	Description	% missing	min	max	mean
vehicle_type	categorical	Type of a truck: EUR - Euro trailer, CIT - City trailer, BAK - Bakwagen	0			
type_of_goods	categorical	Type of goods: VS - Vers, HB - Houdbaar	0			
Q	numerical	Number of containers which can be loaded	0	30	61	45.67

Table 13: Freight transportation company delivery fleet parameters

Name	Type	Description	% missing	min	max	mean
tripid	categorical	Identifier of a trip	0			
stop_number	numerical	Ordered position of a store within this trip	0	1.00	4.00	1.79
stop_name	categorical	Name of a destination	0			
vehicle_type	categorical	Abstract type of a truck: TREKKER - Euro or City trailer, BAKWAGEN(_KOEL) - Bakwagen	0			
type_of_goods	categorical	Type of goods: VS - Vers, HB - Houdbaar	0			
start_time_tw	datetime	Start of a store's time window	13	2016-06-30 01:45:00	2016-06-30 22:25:00	2016-06-30 12:56:08
end_time_tw	datetime	End of a store's time window	15	2016-06-30 02:45:00	2016-06-30 23:25:00	2016-06-30 13:48:35
start_time	datetime	Planned time of arrival at a store	0	2016-06-30 01:07:00	2016-06-30 22:55:00	2016-06-30 12:46:39
end_time	datetime	Planned time of departure from a store	0	2016-06-30 01:34:00	2016-06-30 23:29:00	2016-06-30 13:20:45

Table 14: Travel plan

naam	description	address	vehicle_type	x	y
1	buffer for store 1001	Paleisstraat 25 Amsterdam	EUR	4.88	52.42
2	buffer for store 1027	Van Sonsbeeckstraat Amsterdam	EUR	4.80	52.36
3	buffer for store 1080	Loowaard 8 Amsterdam	EUR	4.88	52.33
4	buffer for stores 1117,1391 and 8684	Galenstraat 35 Amsterdam	CIT	4.87	52.38

Table 15: Linked buffer locations

naam	description	address	vehicle_type	x	y
1	Oostzanerdijk Amsterdam	Oostzanerdijk 1035 Amsterdam	EUR	4.88	52.42
2	La Cantina Westpoort Amsterdam	Hornweg 48 1044 AN Amsterdam	EUR	4.80	52.40
3	Overschiestraat Amsterdam	Overschiestraat 1062 Amsterdam	EUR	4.84	52.35
4	Lay-by Nieuwendammerdijk Amsterdam	Nieuwendammerdijk 542 1023 BX Amsterdam	EUR	4.95	52.39
5	Lay-by Reigersbospad Amsterdam	Reigersbospad 1106 Amsterdam-Zuidoost	EUR	4.97	52.30
6	Cruise Terminal Amsterdam	De Ruijterkade 1013 Amsterdam	EUR	4.91	52.38
7	Public car parking garage	Hoogoorddreef 1102 Amsterdam-Zuidoost	BAK	4.96	52.31
8	Public car parking garage	Haarlemmer Houttuinen 1013 GM Amsterdam	BAK	4.89	52.38
9	Public car parking garage	Europaboulevard 1083 AD Amsterdam	BAK	4.89	52.33
10	Public car parking garage	Sam van Houtenstraat 1067 JG Amsterdam	BAK	4.80	52.38

Table 16: Commonly used buffer locations

Table 17: European retailer's delivery destinations with identifiers (available upon request)

naam	x	y	i
1	4.88	52.42	83
2	4.80	52.40	84
3	4.84	52.35	85
4	4.95	52.39	86
5	4.97	52.30	87
6	4.91	52.38	88
7	4.96	52.31	89
8	4.89	52.38	90
9	4.89	52.33	91
10	4.80	52.38	92

Table 18: Commonly used buffer locations with identifiers

naam	x	y	i
1	4.88	52.42	83
2	4.80	52.36	84
3	4.88	52.33	85
4	4.87	52.38	86

Table 19: Linked buffer locations with identifiers

i	w	b	e	s
2	1	300	360	53
2	2	710	770	49
3	1	390	450	61
3	2	527	587	47
3	3	952	1012	54
4	1	485	545	35
4	2	1083	1143	37
5	1	210	270	40
5	2	1131	1191	68
6	1	480	540	53
7	1	825	885	42
8	1	414	474	53
8	2	535	595	57
9	1	1265	1325	27
10	1	150	210	64
10	2	630	690	47
10	3	1166	1226	57
12	1	594	654	41
13	1	1020	1080	42
13	2	1220	1280	49
14	1	534	594	41
14	2	621	645	24
14	3	878	938	42
15	1	588	612	24
15	2	731	791	49
15	3	815	847	32
15	4	900	960	44
16	1	242	302	47
16	2	551	611	36
16	3	960	1020	46
17	1	782	842	50
18	1	460	520	43
18	2	544	573	29
18	3	549	578	29
18	4	625	685	53
19	1	457	517	34
19	2	1063	1123	40
20	1	584	644	50
21	1	398	458	60
21	2	746	806	52
22	1	449	509	16
22	2	629	689	24
22	3	728	788	16
23	1	390	450	51
23	2	1253	1313	47
24	1	587	619	32
24	2	597	629	32
24	3	708	742	34
25	1	1201	1261	42
26	1	1020	1080	49
26	2	1200	1260	45
27	1	455	515	30
29	1	390	450	45
29	2	696	756	36
30	1	1102	1162	44
31	1	210	270	62
32	1	1037	1097	31
33	1	750	810	26
33	2	1138	1198	30
33	3	1276	1336	36
34	1	240	300	77
34	2	990	1050	77
35	1	600	660	48
35	2	1096	1156	47
36	1	462	522	41
37	1	1170	1230	32
37	2	1337	1397	36
38	1	791	806	15
39	1	510	570	66
39	2	1157	1217	68
40	1	528	588	62
40	2	750	810	59
41	1	487	547	39
42	1	480	540	67
42	2	823	883	74
43	1	469	529	48
44	1	757	781	24
44	2	765	825	42
44	3	1093	1153	46
45	1	1050	1110	32
46	1	491	551	46
46	2	960	1020	47
47	1	624	684	65
47	2	1034	1094	55
48	1	935	995	63
49	1	485	545	63
49	2	694	754	60
50	1	105	165	50
50	2	1126	1186	30
50	3	1223	1283	53
51	1	999	1059	63
52	1	1073	1133	38
53	1	767	827	51
53	2	1007	1067	48
54	1	1167	1227	31
54	2	1345	1405	34
56	1	867	927	21
56	2	1218	1278	24
57	1	431	491	46
57	2	1090	1150	52
59	1	445	505	32
61	1	549	609	36
62	1	585	645	39
63	1	390	450	22
64	1	475	535	27
68	1	1180	1240	45
70	1	1205	1265	25

71	1	840	900	18
73	1	1226	1257	1
74	1	1226	1286	49
75	1	1052	1062	10
76	1	1044	1104	38
77	1	445	505	25
77	2	525	585	28
78	1	534	594	32
78	2	1081	1141	34
79	1	532	592	39
79	2	1157	1217	40
80	1	405	465	33
80	2	735	795	23
80	3	1196	1256	39
81	1	334	394	43
81	2	851	911	44
82	1	435	495	46
82	2	588	648	46

Table 20: Time windows

k	h	Q
1	1	34
1	2	30
2	1	50
2	2	45
3	1	61
3	2	54

Table 21: Goods types with identifiers

i	w	h	d
2	1	1	61
2	2	2	54
3	1	1	30
3	2	2	26
3	3	2	22
4	1	2	22
4	2	1	25
5	1	2	54
5	2	1	61
6	1	1	30
7	1	1	50
8	1	1	11
8	2	2	30
9	1	2	30
10	1	1	61
10	2	2	54
10	3	2	54
12	1	2	54
13	1	1	50
13	2	2	45
14	1	1	34
14	2	2	15
14	3	2	15
15	1	2	15
15	2	2	30
15	3	1	11
15	4	1	17
16	1	1	61
16	2	2	54
16	3	2	54
17	1	2	54
18	1	1	34
18	2	2	15
18	3	2	15
18	4	2	30
19	1	2	54
19	2	1	61
20	1	2	54
21	1	2	54
21	2	1	30
22	1	1	17
22	2	2	30
22	3	1	17
23	1	2	30
23	2	1	34
24	1	2	15
24	2	2	15
24	3	1	50
25	1	2	54
26	1	2	30
26	2	1	34
27	1	1	25
29	1	2	54
29	2	1	30
30	1	2	54
31	1	2	54
32	1	1	25
33	1	1	17
33	2	1	34
33	3	2	30
34	1	1	50
34	2	2	45
35	1	2	45

35	2	1	25
36	1	1	30
37	1	1	34
37	2	2	30
38	1	1	11
39	1	1	17
39	2	2	30
40	1	2	45
40	2	1	50
41	1	1	30
42	1	1	25
42	2	2	45
43	1	2	26
44	1	1	11
44	2	1	17
44	3	2	30
45	1	1	25
46	1	1	25
46	2	2	45
47	1	2	54
47	2	1	61
48	1	2	54
49	1	1	61
49	2	2	54
50	1	1	50
50	2	2	14
50	3	2	45
51	1	2	54
52	1	2	54
53	1	1	50
53	2	2	45
54	1	1	34
54	2	2	30
56	1	1	17
56	2	2	30
57	1	2	45
57	2	1	50
59	1	1	30
61	1	1	17
61	1	2	15
62	1	1	17
62	1	2	15
63	1	1	11
63	1	2	9
64	1	1	17
68	1	1	16
68	1	2	14
70	1	1	20
70	1	2	17
71	1	2	15
73	1	1	15
73	1	2	13
74	1	1	15
74	1	2	13
75	1	2	22
76	1	2	45
77	1	1	25
77	2	2	22
78	1	1	34
78	2	2	30
79	1	1	30
79	2	2	17
80	1	1	17
80	2	1	17
80	3	2	30
81	1	2	45
81	2	1	50
82	1	2	45
82	2	1	50

Table 22: Demand

References

Bloomfield P. (2000). Fourier Analysis of Time Series: An Introduction (2nd ed.). New York, NY : Wiley, 2000. - 261 p.

Chatfield C. (2003). The Analysis of Time Series: An Introduction (6th ed.). Chapman and Hall/CRC, 2003.

Floudas C.A. and Pardalos P.M. (Eds.) (2009). Encyclopedia of Optimization (2nd ed.). New York, NY : Springer, 2009. - 4626 p.

Geisberger R., Sanders P., Schultes D., Delling D. (2008). Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks. In: McGeoch C.C. (eds) Experimental Algorithms. WEA 2008. Lecture Notes in Computer Science, vol 5038. Springer, Berlin, Heidelberg

Hu T.C. and Kahng A.B. (2016). Linear and Integer Programming Made Easy (1st ed.). Switzerland, AG: Springer International Publishing, 2016. - 143 p.

Ropke S. and Cordeau J-F. (2009). Branch-and-Cut-and-Price for the Pickup and Delivery Problem with Time Windows. Transportation Science, 43(3), 267-286.