Optimizing Parcel Transportation of PostNL

Ruben Hoeksma[1], Christopher Hojny[2], Jan-Kees van Ommeren[3], Matthias Walter[4]

Abstract

We consider a multi-commodity transportation problem that arises in parcel delivery in the Netherlands. We focus on its deterministic variant and propose a mixed-integer programming model to solve it. We provide an implementation that is based on a multi-stage solution approach in order to overcome computational difficulties. This allows us to solve practical instances to reasonable accuracy.

Keywords: transportation, network design, multi-commodity flow

2.1 Introduction

PostNL is a Netherlands-based mail and parcel delivery company. On a daily basis, the company collects, sorts and delivers mail and parcels throughout, mostly, the Netherlands. This report addresses the optimization of the daily transportation of parcels by trucks between sorting centers in the Netherlands.

[1]University of Twente, The Netherlands
[2]Eindhoven University of Technology, The Netherlands
[3]University of Twente, The Netherlands
[4]University of Twente, The Netherlands

These parcels need to be carried between different *sorting centers*, which are the hub locations where collected parcels are redistributed and from which actual delivery to the recipients starts. To avoid having to send several trucks from every sorting center to every other sorting center, PostNL runs several so-called *cross docks* which are simply hubs in the distribution network. We consider only the transportation of *trolleys*, which are the units of identical size that are loaded with parcels at their origin sorting center and are sent through the network to their destination sorting center.

Deterministic problem. For the deterministic version of the problem we are given a set L of sorting locations, of which some have cross-dock capabilities, which we denote by $L^{\times} \subseteq L$. For every pair $i, j \in L$ $(i \neq j)$ of locations we know the driving times $d(i,j) \in \mathbb{R}_{\geq 0}$ from i to j. There is a set K of commodities to be sent through the network, which correspond to the labels that are attached to each trolley sent through the network. Each such commodity $(i,t) \in K$ consists of a location $i \in L$ and a deadline $t \in \mathbb{R}$ until which all trolleys of this commodity have to be delivered to their destination i. In the idealized deterministic setting, a set G of generated trolleys is given. For each trolley, we are given a tuple $(i,t,i',t') \in G$, where $i \in L$ denotes the location where it appears, a *release time* t at which it appears and a commodity $(i',t') \in K$ that indicates to which destination i' and by which time t' it must be delivered. Note that although identical tuples can occur multiple times, we consider them as unique for ease of notation. Our implementation deals with multiplicities properly. Transportation is done by identical trucks that each have a capacity of $U \in \mathbb{Z}_{\geq 0}$ trolleys. Moreover, loading and unloading a truck at some location takes an amount of time, $t_{\text{load}}(i) \in \mathbb{R}_{\geq 0}$ and $t_{\text{unload}}(i) \in \mathbb{R}_{\geq 0}$ for each location $i \in L$, respectively. At each location $i \in L$, at most $c_{\text{park}}(i) \in \mathbb{Z}_{\geq 0}$ trucks can load or unload at the same time. Finally, at most $c_{\text{out}}(i) \in \mathbb{Z}_{\geq 0}$ generated trolleys can wait for being transported, at most $c_{\text{in}}(i) \in \mathbb{Z}_{\geq 0}$ trolleys can wait at their destination for their deadline and $c_{\times}(i) \in \mathbb{Z}_{\geq 0}$ trolleys can wait for further transportation after they have arrived at some cross dock $i \in L^{\times}$.

Demand forecasts. We were provided with a deterministic instance of the optimization problem we just described. However, knowledge of the exact time at which a trolley is ready to be shipped is quite unrealistic. In practice, PostNL has a demand forecast whose accuracy is unknown to the authors. Moreover, there are further sources of uncertainty, e.g., the actual traveling times or breakdowns in any part of the logistics chain. However, as it will turn out already the deterministic optimization problem is not easy to solve at all. This justifies our focus on the perfect-knowledge version of the real-world problem.

Outline. The paper is structured as follows. We first describe our modeling approach by means of time discretization in Section 2.2. In that section we also derive a mixed-integer programming model and refine it as an attempt to deal with robustness problems. Our solution approach is described in Section 2.3 and the corresponding implementation prototype is explained in Section 2.4. Our results are presented in Section 2.5. We conclude our paper with future recommendations for PostNL in Section 2.6.

2.2 Time discretization and MIP models

In this section we first describe how we discretize the times that are relevant to our problem. Then we explain an auxiliary graph that is useful to derive a mixed-integer programming model for the problem. Finally, a basic and a refined MIP model are described.

2.2.1 Time-expanded network

We first choose a parameter $\Delta > 0$ and then transform all times t to t/Δ, which we call *ticks*. In fact, we only consider integer tick values. To this end, transformed release, loading and unloading times are rounded up to the next integer, while deadlines are rounded down. We denote these rounded values by a Δ superscript, e.g., $t^{\Delta}_{\text{unload}}(i) := \lceil t_{\text{unload}}(i)/\Delta \rceil$. This rounding procedure is conservative in the sense that the available time to route a trolley is never increased, while the time necessary to transport it is never decreased. In particular, if Δ is

larger than $t_{\text{load}}(i)$ or $t_{\text{unload}}(i)$, then rounding up several parameters yields values that sometimes overestimate the actual transport time by too much. To resolve this problem, we do not transform the driving times independently, but instead define the *travel ticks* as

$$d^{\Delta}(i,j) \coloneqq \lceil (d(i,j) + t_{\text{load}}(i) + t_{\text{unload}}(j))/\Delta \rceil \,.$$

We denote by $t_{\min} \in \mathbb{Z}$ and $t_{\max} \in \mathbb{Z}$ the largest (resp. smallest) tick such that $T \coloneqq [t_{\min}, t_{\max}] \cap \mathbb{Z}$ contains all transformed release times and deadlines.

Auxiliary network. We now define the auxiliary directed graph $D = (V, A)$ with $V = L \times T$. In particular, we consider multi-commodity flows (with further restrictions) in D, where a flow unit that traverses through node (i,t) represents a trolley that is at i in tick t. Note that the distinction of incoming and outgoing trolleys is easy, since the commodity $k = (j,t)$ contains the destination information and we have $i = j$ if and only if the trolley has i as its destination. The arc set of D consists of arcs $A = A^{\text{move}} \cup A^{\text{stay}}$ with

$$A^{\text{move}} \coloneqq \{((i,t),(j,t') : i \in L,\ j \in L,\ t,t' \in T,\ t' = t + d^{\Delta}(i,j)\} \,,$$
$$A^{\text{stay}} \coloneqq \{((i,t),(i,t+1) : i \in L,\ t,t+1 \in T\} \,.$$

We also introduce the following notation:

$$\delta_t^{\text{out}}(i) \coloneqq \{a \in A : a = (i,t,j,t') \text{ for some } j \in L, t' \in T\} \,,$$
$$\delta_t^{\text{in}}(i) \coloneqq \{a \in A : a = (j,t',i,t) \text{ for some } j \in L, t' \in T\} \,.$$

2.2.2 Basic model

We first describe a base model for the problem. The main decision of the multi-commodity flow interpretation of the trolley routing problem is to decide how many commodities of a specific type traverse an arc $a \in A$ at a certain point of time. For arcs $a \in A^{\text{stay}}$, we need to guarantee that the capacities of the corresponding location are not exceeded. For arcs $a \in A^{\text{move}}$, we must ensure that also sufficiently many trucks are provided for transportation along a. For this reason, we introduce the

following variables, modeling the previously mentioned decisions. The *truck variables*

$$x_a \in \mathbb{Z}_{\geq 0} \qquad \forall a \in A^{\text{move}} \tag{2.1}$$

model how many trucks are available for transportation along arc $a = ((i,t),(j,t'))$. We assume that the loading procedure of these trucks starts at tick t, i.e., the trucks are not necessarily leaving immediately. To make sure that capacities of locations are not exceeded, we introduce *inventory variables*

$$s^t_{i,k} \in \mathbb{R}_{\geq 0} \qquad \forall i \in L,\ \forall k \in K,\ \forall t \in T, \tag{2.2}$$

which keep track of the number of trolleys of commodity k that are at location i at time t. Finally, *flow variables*

$$y_{a,k} \in \mathbb{R}_{\geq 0} \qquad \forall a \in A,\ \forall k \in K \tag{2.3}$$

model the number of trolleys of commodity k that traverse arc a. Note that we model the flow of commodities using continuous variables. Preliminary computational experiments showed that switching between integer and continuous variables does not make much of a difference for the optima. One potential reason is that actually sending a truck along an arc often yields enough capacity that the full demand of one commodity (at the source node) can be sent without the need to split it across different arcs. However, already for smaller examples the running time increased significantly when requiring integrality of the y-variables. Hence, we decided to make them continuous.

Constraints. To make sure that the previously introduced variables model a solution of the trolley routing problem, we introduce the following constraints. The *truck capacity constraints* guarantee that the total number of trolleys sent via an arc $a \in A^{\text{move}}$ does not exceed the capacity f the available trucks:

$$\sum_{k \in K} y_{a,k} \leq U \cdot x_a, \qquad \forall a \in A^{\text{move}}. \tag{2.4}$$

Similarly, we need to make sure that the docking capacities at each location are not exceeded. That is, the total number of trucks arriving

and departing from a certain location i must not exceed the number of docks D_i:

$$\sum_{\tau=0}^{t^{\Delta}_{\text{unload}}(i)-1} \sum_{a \in \delta^{\text{in}}_{t+\tau}(i)} x_a + \sum_{\tau=0}^{t^{\Delta}_{\text{load}}(i)-1} \sum_{a \in \delta^{\text{in}}_{t-\tau}(i)} x_a \leq D_i \quad \forall i \in L,\ \forall t \in T. \tag{2.5}$$

Note that we need to take the summation over τ into account to also consider the trucks whose unloading (resp. loading) process at location i has not finished yet until tick t.

When routing the trolleys through the directed graph, we need to make sure that the routing adheres to a flow structure, i.e., the following *flow balance constraints* need to be satisfied for all $i \in L$, $k = (j, t') \in K$, and $t \in T \setminus \{0\}$:

$$s^{t-1}_{i,k} + \sum_{a \in \delta^{\text{in}}_t(i)} y_{a,k} - \sum_{a \in \delta^{\text{out}}_t(i)} y_{a,k} + \beta^{\text{in}}_{i,k,t} - \beta^{\text{out}}_{i,k,t} = s^{t}_{i,k}, \tag{2.6}$$

where $\beta^{\text{in}}_{i,k,t}$ is the number of trolleys of commodity k that are due at time t at location i, and $\beta^{\text{in}}_{i,k,t}$ is the number of trolleys created. Note that these values can be easily computed from the instance data.

Finally, we need to guarantee that the inventory capacities at all locations are not exceeded. Since locations have different capacities for outgoing and incoming trolleys, we introduce

$$\sum_{\substack{(j,t') \in K: \\ j \neq i}} s^t_{i,(j,t')} \leq c_{\text{out}} \qquad \forall i \in L \setminus L^{\times},\ \forall t \in T, \tag{2.7}$$

$$\sum_{\substack{(j,t') \in K: \\ j \neq i}} s^t_{i,(j,t')} \leq c_{\times} \qquad \forall i \in L^{\times},\ \forall t \in T, \tag{2.8}$$

$$\sum_{t' \in T} s^t_{i,(i,t')} \leq c_{\text{in}} \qquad \forall i \in L,\ \forall t \in T. \tag{2.9}$$

Objective function. Since we are interested in small total driving times, we

$$\text{minimize} \sum_{a=((i,t),(j,t')) \in A} d(i,j) x_a. \tag{2.10}$$

Note that while we use the rounded driving times (by means of travel ticks $d^{\Delta}(i,j)$) in order to define the auxiliary graph, we use the actual driving times in the objective function. Hence, the objective values of computed solutions can be related to the real world and do not require a conversion from ticks to actual times.

2.2.3 Refined model

Later, we solve the MIP (2.1)–(2.10) for a certain problem instance with a certain trolley production G. We then evaluate the solution for different other sets $G_1, \ldots, G_\ell$ of generated trolleys. The purpose is to investigate the robustness of the computed solution $(x^\star, s^\star, y^\star)$ with respect to modified demands. For this we fix the truck decision variables $x = x^\star$ and try to find vectors y and s that constitute a transportation plan for a particular set G_i of generated trolleys. However, it may happen that the instance for a G_i is infeasible: for instance, after the last truck from a location leaves, another trolley appears, which has no chance of reaching its destination.

A similar problem can appear already for the first optimization with trolley production G if the discretization parameter Δ is too large: in this case, there might not be enough time to go to a destination via a cross dock, and hence much of the transportation would have to arrive exactly at the deadline which may in turn overload the available docks.

To this end, we extended the base model represented in the previous section as follows.

Not delivering trolleys. We introduce additional variables

$$p_k^{\text{in}} \in \mathbb{R}_{\geq 0} \qquad \forall k \in K \tag{2.11}$$

that count the number of trolleys of commodity k that are not delivered. To match this to the total flow balance in the network, we also introduce variables

$$p_{i,k,t}^{\text{out}} \in [0, \beta_{i,k,t}^{\text{in}}] \qquad \forall i \in L,\ \forall k \in K,\ \forall t \in T, \tag{2.12}$$

that indicate the number of trolleys if commodity k that would be produced in i at time t, but that are not released. These variables

are incorporated into the basic model by adding, to the left-hand side of (2.6), the term $p^{\text{out}}_{i,k,t}$ as well as subtracting p^{in}_k in case $k = (i,t)$ holds. The modified constraint shall be denoted by (2.6'). In order to encourage delivery, we penalize these variables with a certain factor in the objective. In our case, we used a coefficient of 10, which is larger than the extra costs of sending a single truck (say, with the considered trolley) along the longest connection.

Extending the number of depots. For strategic planning, one may want to analyze the effect of certain restrictions. In particular, it could be interesting to judge the benefit of increasing certain capacities such as the sorting capacity of a location or the number of depots. To illustrate this flexibility of our proposed MIP approach we added a corresponding extension regarding the depot numbers. To this end, we introduced variables

$$e_i \in \mathbb{R}_{\geq 0} \qquad \forall i \in L, \tag{2.13}$$

to indicate extended docking capacity at location i. The modification of the docking constraint (2.5) is straight-forward: we replace its right-hand side with $D_i + e_i$. The modified constraint shall be denoted by (2.5') Again, we add $\sum_{i \in L} 10 e_i$ to the objective function in order to penalize this dock extension. This is solely for demonstration purposes, and for answering an actual strategic question a suitable value would have to be found. The objective function (2.10) augmented with all discussed penalty terms is denoted by (2.10').

Final model. For further reference we denote the final model as

$$\begin{aligned} &\text{minimize (2.10') over } (x, s, y, p^{\text{in}}, p^{\text{out}}, e) \\ &\qquad \text{subject to (2.1)–(2.4), (2.5'), (2.6'), (2.7)–(2.13).} \end{aligned} \tag{2.14}$$

2.3 Solution approach

Our main challenge is to produce solutions with total travel time as small as possible. For this reason, we have focused in our approach on

generating solutions with a small objective value rather than deriving strong lower bounds on the optimal travel time. To find solutions that are as realistic as possible, one is interested solving Model (2.14) for a time discretization of about 15 minutes. Refining the discretization parameter Δ to $\frac{\Delta}{2}$, however, roughly doubles the number of variables and constraints in Model (2.14). This makes it a challenge to solve Model (2.14) or finding good solutions for a very fine time discretization. Our solution approach therefore consists of multiple phases in which solutions for coarse time discretizations are used to initialize the search for good solutions with a finer time discretization.

Phase 1. In Phase 1, we start with a very coarse time discretization $\Delta = 120\,\text{min}$, and our aim is to find a solution in a very limited amount of time. To this end, we define a placeholder solution $(x^\star, s^\star, y^\star)$, and initialize an upper bound on the optimal objective value of $u = \infty$. Then, we iteratively attempt to solve Model (2.14). In each iteration, we specify a time limit of 300 s and provide the model the best known solution $(x^\star, s^\star, y^\star)$ from a previous iteration as start solution. In the first iteration, this solution is a placeholder solution which results in not providing a solution at all. After the time limit is hit, we extract the best solution found during this iteration. If the objective value of the latter is smaller than $0.99u$, we replace $(x^\star, s^\star, y^\star)$ by the newly found solution and update u to its objective value. Otherwise, Phase 1 terminates and returns the best solution found so far.

Our motivation for this strategy is based on the observation that in many cases the solver was not able to improve on a found solution in a reasonable amount of time. Restarting the entire solution process and providing the best incumbent, however, the solver was able to very quickly find an improving solution.

Phase 2. In Phase 2, we refine the time discretization to $\Delta = 60\,\text{min}$. The remaining steps are essentially the same as in Phase 1. The only difference is that we provide also the first iteration a solution, namely the final solution of Phase 1, and that the time limit for each iteration is set to 1800 s.

Phase 3. Phase 3 iteratively attempts to solve Model (2.14) for $\Delta = 30\,\text{min}$. We provide the entire phase a total time limit of 86 400 s, i.e., one day. The remaining structure of Phase 3 is essentially the same as before except for the following differences. Instead of providing each iteration a fixed time limit, we work with a solution time limit. That is, we do not specify an initial time limit for each iteration, but we wait until the solver has found a solution improving on the initially provided one. Afterwards, we allow the solver to continue with the search for better solutions within the solution time limit. Our motivation for this strategy is that we observed that the solver could rather often quickly improve on a found solution, i.e., interrupting the solver right after the first improving solution has been found might have blocked it from providing even better solutions in a reasonable amount of time. For the first iteration, the solution time limit has been set to 300 s. In every succeeding iteration, we double the solution time limit if the newly found solution does not improve on the previously best known solution by at least 1 %. Otherwise, the same solution time limit is used.

2.4 Instance format and software

We implemented the MIP model (2.14) and the solution approach described in the previous section in Python. Our code is available on github[5]. Since we cannot publish the actual instances as they contain some confidential information, we describe the instance format that is used by our code.

The instances are described in 2 files, one *network file* and one *trolley file*. The former describes all properties of the network except for the actual trolleys that are sent through it. This separation allowed us to test a solution for a network with different trolley sets. The network file has the following format:

```
U <NUMBER INDICATING THE CAPACITY OF EACH TRUCK>
i <UNLOADING TIME IN HOURS>
o <LOADING TIME IN HOURS>

# List of locations, one per line.
# <NAME> is a string
```

[5] https://github.com/discopt/postnl

```
# <X> is the longitude
# <Y> is the lattitude
# <OUT-CAPAC.> is the outgoing capacity
# <IN-CAPAC.> is the incoming capacity
# <CROSS-CAPAC.> is the crossdocking capacity.
# <NR. OF DOCKS> is the number of docks.
l <NAME> <X> <Y> <OUT-CAPAC.> <IN-CAPAC.> <CROSS-CAPAC.> <NR. OF DOCKS>
...

# List of distances, one per line.
# <i> and <j> are numbers from 0 up to |L|-1.
d <i> <j> <DISTANCE FROM i to j>
...

# List of commodities, one per line.
# c <TARGET LOCATION> <SHIFT NUMBER> <DEADLINE TIME IN HOURS>
...
```

The *trolleys file* is a CSV file with ; as a separator character. It contains one header line and otherwise lines of the form

```
<NAME1>;<NAME2>;<SHIFT NUMBER>;<TIME>
```

where first two columns refer to names of some locations $i, j \in L$ from the network file, j together with the shift number constitute a commodity, and the last column specifies the time at which the trolley is spawned.

All our experiments were run on a cluster with 32 Intel Xeon Gold 5217 CPU (3.00 GHz) processors, a total of 64 GB of RAM running an Ubuntu Linux with Gurobi 9.5.1rc2 on 4 threads.

2.5 Results

PostNL provided us one instance on that we could test our solution approach. This instance features 31 locations of which six are classified as cross docks; the remaining 25 locations are regular sorting centers. A regular sorting center has a capacity of 400 outgoing and 1200 ingoing trolleys, for cross docks no limits on the trolley capacities are imposed. Trolleys are assumed to start arriving at sorting centers late afternoon and need to be shipped to their destination until an individually specified time the next morning. Finally, a truck is assumed

to have a capacity of 48 trolleys and loading (resp. unloading) a truck takes 10 minutes (resp. 15 minutes).

2.5.1 Solution quality and robustness

One downside of our solution approach is that the produced solution arguably is not robust against changes in the data. In practice, this means that arrival times of trolleys are not deterministic and one needs to find a schedule of trucks that is able to transport as many trolleys on time while still minimizing total mileage. Therefore, we have tried increase robustness of our solution by reducing capacities of sorting centers and/or trucks with which we compute it. If we reduce, for example, the outgoing capacities of sorting centers, we are able to deal with uncertainties of arrival times of trolleys; reducing the truck capacities sends more trucks than strictly needed in the considered scenario. I.e., if more trolleys arrive than expected, we increase the chance that all trolleys can be delivered on time.

In our experiments, we used the method described in Section 2.3 to produce solutions for the original instance (referred to as "original" in the following) provided by PostNL as well as three variations to add aspects of robustness to the solution approach. To this end, we either reduced the outgoing capacity (O) by 25 %, the truck capacity (T) by 8.3 %, or both (OT). Table 2.1 summarizes our results. For each of the three phases, it shows the value of the best known solution (columns 2–4) as well as the number of iterations within this phase (columns 8–10). Moreover, it provides the best known final dual bound (column 5) and the corresponding gap (column 6) in the 30 minutes discretization as well as the final penalty value (column 7) caused by trolleys not delivered on time.

We can see that the formulations with a very coarse discretization of $\Delta = 120\,\text{min}$ are able to provide good solutions in terms of total mileage as the primal bound of Phase 1 is much smaller than the primal bound of Phases 2 and 3. In particular, the primal bounds of Phase 1 almost match the final dual bounds. From a practical point of view, however, these solutions are not useful as the trucks can only leave every two hours. Introducing finer discretizations, Gurobi is only able to find solutions with a relatively large mileage in comparison to the coarse

Table 2.1: Overview of numerical results using the approach of Section 2.3.

	primal bounds/phase			best dual	gap	penalty	#iter./phase		
	1	2	3				1	2	3
original	1333.6	1743.3	1559.6	1330.6	14.7 %	60	4	1	8
O	1371.0	1858.7	1546.1	1326.1	14.2 %	40	6	1	8
T	1439.0	1815.9	1641.9	1417.1	13.7 %	40	5	1	8
OT	1468.6	1933.7	1775.4	1416.8	20.2 %	40	5	1	8

discretization. But note that we cannot conclude that the mileages from Phase 1 are also the right mileages for Phase 3 as also the arrival time of trolleys gets discretized.

Reductions of outgoing- or truck capacities lead, in general, to an increase of value for the best known incumbent solution. The only exception is the reduction of outgoing capacity, where the objective value drops slightly in comparison with the original instance. Reducing the truck capacity (both capacities) leads to increase of the best incumbent's objective value by 5.3 % (13.8 %). Of course, since we could solve neither of the four models to optimality, we cannot conclude that the price of robustness in the sense of the different variations is exactly this value. However, it indicates that the increase in total mileage and penalty values can be rather large. For this reason, it might be interesting to explore different ways to robustify our approach to find robust solutions that have less impact on total mileage.

Table 2.2 depicts the results for different scenarios. A scenario is given by its trolley production G_i and we evaluated our solution (keeping the truck decision that we computed for our instance) for 9 such sets, all of which were provided by PostNL. Interestingly, in terms of robustness the original instance and the one with a reduction of the truck and the outgoing capacities (OT) gives the most robust results. Unfortunately, we could not determine an actual reason for this surprising behavior. In fact, for different solutions that we had produced during the project the robustness of the original solution was much worse. Hence, we conjecture that more computation time (per scenario) would

Table 2.2: Robustness of solution when challenged with different sets of trolley productions. The column base indicates the number of undelivered trolleys and required extra docks for the trolley production that was used as input for our solution approach. The nine further columns indicate these amounts for different productions, and the right-most column shows the average over these 9 instances.

instance	obj. value	undelivered trolleys / extra docks for different scenarios										
		base	1	2	3	4	5	6	7	8	9	average
original	1559.6	3/0	17/1	11/3	11/0	18/0	14/0	15/0	16/0	14/1	17/1	15.1/0.7
O	1546.1	2/0	27/0	80/0	16/0	44/0	18/3	47/1	29/0	20/0	49/4	37.0/0.9
T	1641.9	2/0	65/0	75/0	14/0	39/0	21/0	42/0	20/0	16/0	49/0	38.3/0.0
OT	1775.4	2/0	63/0	19/0	7/0	5/0	7/0	2/0	13/0	12/0	10/0	15.6/0.0

be needed to get a fair assessment of the robustness.

2.5.2 Insights from the solution

We now analyze the properties of the computed solution. Figure 2.1 depicts all connections that are used at all over the day.

It is easy to see that the resulting graph is relatively dense, but also that many of the connections are green or blue, indicating that at most 2 trucks use the connection. However, we believe that this large amount of direct connections is due to the disretization error. If $\Delta \geq 30\,\text{min}$, some indirect connections are infeasible, although in practice they would have been feasible. Since one cannot extract any detailed information from this map, we also made corresponding maps that only depict the direct (resp. indirect connections (see Figure 2.2)).

To untangle the large amount of connections from Figure 2.1 even further, we depict in Figure 2.4 the proposed truck activity on an hourly basis. First, most of the trucks do not depart very early since there are not enough trolleys to be transported. Second, there are only a few late trucks, which is to be expected because the deadlines of the commodities differ significantly. Note that the figures depict truck activity per hour whereas our final time discretization is $\Delta = 30\,\text{min}$, that is, solution values are aggregated. Moreover, our very first time

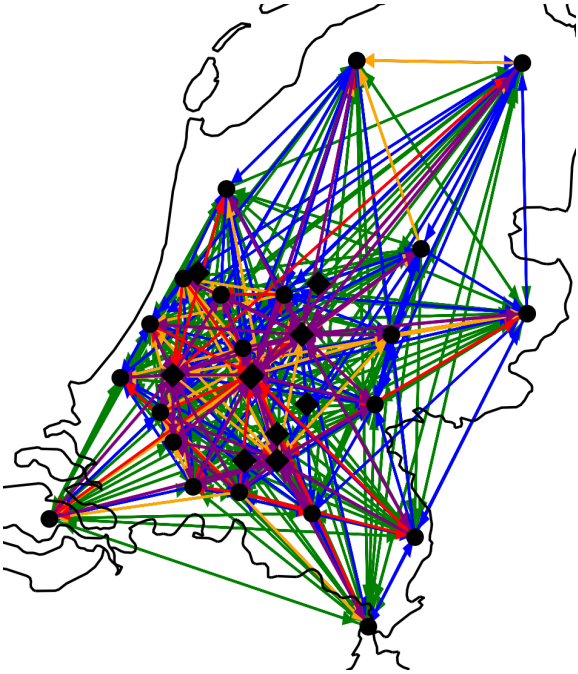
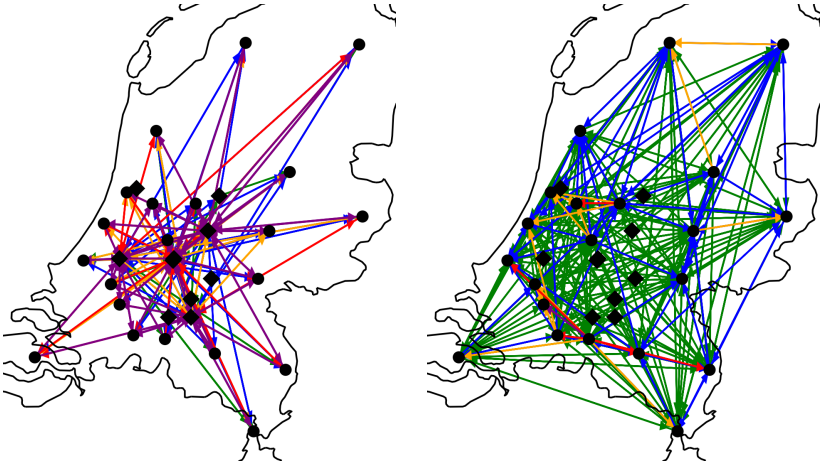
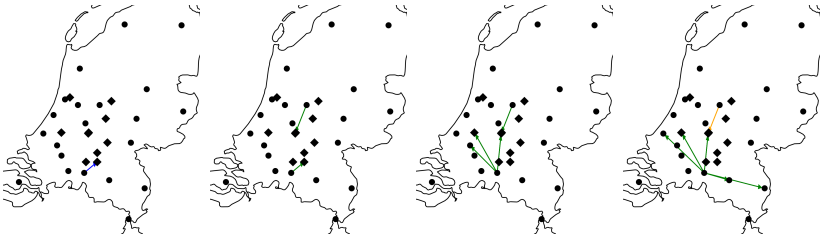


Figure 2.1: Map with all used connections in our computed solution. Colors indicate the number of times a connection is used (green: 1 truck; blue: 2 trucks; orange: 3 trucks; red: 4 trucks; purple: ≥ 5 trucks).

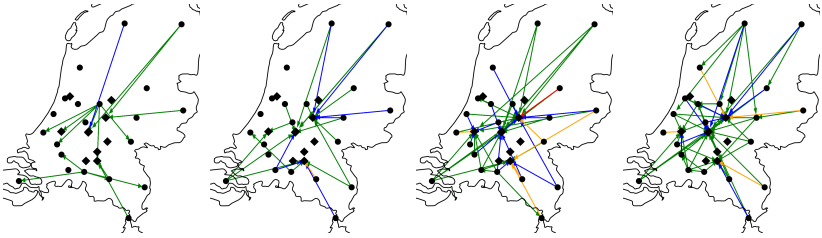


(a) Indirect connections. (b) Direct connections.

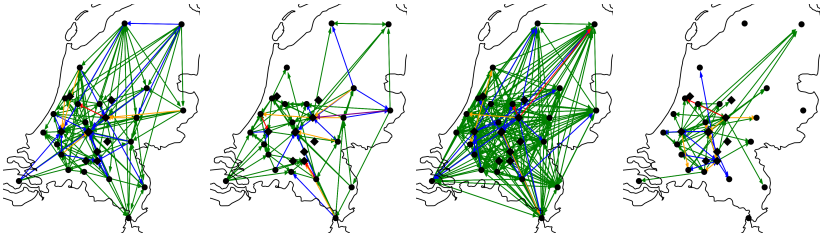
Figure 2.2: Maps with all connections in our computed solution that either involve a cross dock (2.2a) or are direct (2.2b). Colors indicate the number of times a connection is used (green: 1 truck; blue: 2 trucks; orange: 3 trucks; red: 4 trucks; purple: ≥ 5 trucks).



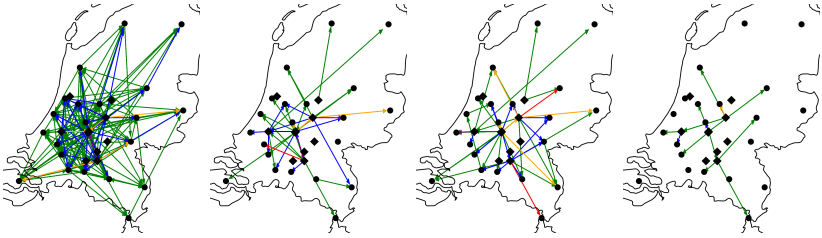
(a) Time slot 1. (b) Time slot 2. (c) Time slot 3. (d) Time slot 4.



(e) Time slot 5. (f) Time slot 6. (g) Time slot 7. (h) Time slot 8.



(i) Time slot 9. (j) Time slot 10. (k) Time slot 11. (l) Time slot 12.



(m) Time slot 13. (n) Time slot 14. (o) Time slot 15. (p) Time slot 16.

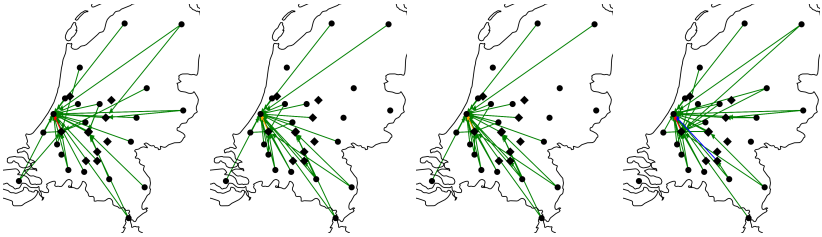
(p) Time slot 17. (q) Time slot 18. (r) Time slot 19. (s) Time slot 20.

Figure 2.4: Maps with connections in our computed solution distributed over 20 subsequent time intervals of length 1 h each. Time slot 1 is the hour of the first released trolley, and time slot 20 is the hour in which the last trucks arrived at their destinations. Colors indicate the number of times a connection is used (green: 1 truck; blue: 2 trucks; orange: 3 trucks; red: 4 trucks; purple: ≥ 5 trucks).

discretization was $\Delta = 120\,\text{min}$, which seems to have an effect on the computed solution: it is apparent that there is much more activity in time slots 9, 11 and 13 than in 8, 10, 12 or 14. We suspect that indeed this is because our computation for $\Delta = 60\,\text{min}$ was warm-started from a solution with $\Delta = 120\,\text{min}$. This implies that there might be potential for improving the computed solution by making use of these less used slots.

Finally, it is worth to have a closer look at how a specific sorting center is connected. In Figures 2.5 and 2.6, we depict the trucks that have a particular sorting center as a destination, where we vary the shifts (and thus the deadlines). Note that while a connection indicates that there is a truck going from a sorting center to another sorting center with trolleys for the specific deadline, this does not mean that all trolleys on such a truck have the same deadline. In particular, if there is only a direct connection from a location i to destination location j for two shifts then it is likely that these connections represent the same truck.

Our first observation is that the overall picture does not change significantly over subsequent shifts, which may indeed be due to sharing of trucks. Of course, some connections appear or disappear, which often



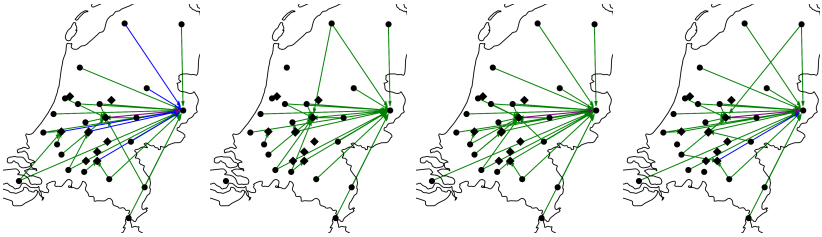
(a) Shift 4. (b) Shift 5. (c) Shift 6. (d) Shift 7.

Figure 2.5: Maps with connections in our computed solution that have a distribution center in the west as its destination, but have different shfits/deadlines. Colors indicate the number of times a connection is used (green: 1 truck; blue: 2 trucks; orange: 3 trucks; red: 4 trucks; purple: ≥ 5 trucks).

happens for single truck connections (colored in green).

In Figure 2.5 one can see that the indirect connections mainly use a cross dock that is very close to the destination, and only make limited use of other cross docks. On the contrary, the destination in the east (Figure 2.6) does not have a cross dock nearby. While there are 3 cross docks of similar distance (the three most eastern ones), only one of them is heavily used. This highlights the optimization potential in our approach as compared to a solution with only direct connections, i.e., that effective aggregation of trolleys can save many truck rides.

We have pointed out a few observations on our computed solution, but we believe that much more insight can be gained when studying it with appropriate background knowledge. In particular, an in-depth comparison to the actual transportation plan is beyond the scope of this paper. Similarly, an evaluation of the computed solution by means of a simulation (that might be based on a more realistic model) would be interesting.



(a) Shift 1. (b) Shift 2. (c) Shift 3. (d) Shift 4.

Figure 2.6: Maps with connections in our computed solution that have a distribution center in the east as its destination, but have different shfits/deadlines. Colors indicate the number of times a connection is used (green: 1 truck; blue: 2 trucks; orange: 3 trucks; red: 4 trucks; purple: ≥ 5 trucks).

2.6 Conclusions and recommendations

We believe that our computed solutions indicate that our solution approach is promising in general. However, we also think that the ability to solve instances with $\Delta \leq 15$ min would be crucial to obtain solutions that are close to being practically relevant. For this reason we would like to point out relevant work in the literature as we think that this is most useful for PostNL to finally obtain a method that can be used in practice.

2.6.1 Column generation

One idea to deal with the large number of variables is to apply column generation in order to only explicitly work with a subset of the variables. The other variables are implicitly set to 0, and a so-called *pricing problem* has to be solved in order to generate promising columns that are turned into explicit variables. Computational results are reported in Gendron and Larose (2014). For a general introduction to column generation, we refer to Desrosiers and Lübbecke (2005), Lübbecke and Desrosiers (2005), and Vanderbeck (2005).

2.6.2 Benders' reformulation

One idea to avoid a large number of continuous variables (in our case all but the x-variables) is that of a Benders' reformulation Benders (1962). Instead of working with an LP relaxation $Q \subseteq \mathbb{R}^{n+d}$ (where n indicates the number of x-variables and d the number of other variables), one works with the projection $P \subseteq \mathbb{R}^n$ of Q on the these n variables. The obvious advantage is that the number of variables is decreased, which in our case would be a reduction by a factor larger than 200. The disadvantage is that one has to be able to describe P by means of linear inequalities, which are usually too many to state explicitly. Hence, one needs to generate the inequalities describing P on demand, i.e., be able to find out if a given $\hat{x}$ lies in P or not, and in the latter case, find a violated inequality. This problem is known as the *separation problem*.

One way to do this is to try to *lift* the vector $\hat{x}$ to one in Q, i.e., to solve the LP of finding $(\hat{x}, z) \in Q$ for given $\hat{x}$. If there exists such a z, then $\hat{x} \in P$ follows. Otherwise, one obtains so-called Farkas multipliersSchrijver (1986, Section 7.3) which can be used to derive a (violated) inequality in the x-space. This approach still requires to solve an LP with a huge amount of y-variables and s-variables, which may not be practical. However, certain classes of inequalities (that are part of P's inequality description) are known for which the separation problem can be solved more effectively. A prime example are cut-based inequalities: if one partitions the node set of a network into two sets V_1 and V_2, then the total transportation demand for trolleys that originate somewhere in V_1 but must be carried to a destination in V_2 is known. This implies that the total number of trucks sent along arcs from V_1 to V_2 must be at least this total demand divided by the truck capacity. Such inequalities can be computed using maximum flow algorithms, which is generally much faster than large LPs. More such inequalities and their computational impact can be found in Bienstock and Günlük (1996), Sridhar and Park (2000), Costa, Cordeau, and Gendron (2009), Raack (2012), and Agarwal and Aneja (2017)

2.6.3 Solving multi-commodity flow subproblems

Despite the use of known inequalities in the Benders' approach sketched above, one may need to solve the underlying multi-commodity flow problem for a fixed number of trucks $\hat{x}$. This effectively yields a multi-commodity flow problem in the (time-expanded network) where the $\hat{x}$-vector imposes capacities on certain arcs. While this problem can be phrased as an LP of which several variables can be removed because of 0 capacity, solution times may still be prohibitively large. An alternative is to cast the problem by means of path variables Tomlin (1966). More precisely, for each commodity and suitable path (having a source node with production and a destination node with demand) there is a path variable. These path variables also have to be dealt with by means of column generation. Here, the pricing problem turns out to be a shortest path problem in the network which can be solved very efficiently. This approach is also promising because a commodity is typically sent via very few paths only, even way fewer than one has arcs in the network. Hence, a final LP solution can be represented very compactly in such a model and the hope is that also for intermediate solution steps, not too many such path variables must be considered at the same time.

There also exist other approaches to solve the problem, e.g. based on *Lagrangean relaxation*, but care must be taken that proper dual information can be extracted in order to be able to obtain inequalities for the Benders' reformulation. Computational insights appear, for instance, in Caprara (2015).

References

Agarwal, Yogesh Kumar and Yash P Aneja (2017). "Fixed charge multicommodity network design using p-partition facets". In: *European Journal of Operational Research* 258.1, pp. 124–135. DOI: 10.1016/j.ejor.2016.09.015.

Benders, Jacques F. (1962). "Partitioning procedures for solving mixed-variables programming problems". In: *Numerische Mathematik* 4.1, pp. 238–252. ISSN: 0945-3245. DOI: 10.1007/BF01386316.

Bienstock, Daniel and Oktay Günlük (1996). "Capacitated Network Design–Polyhedral Structure and Computation". In: *INFORMS Journal on Computing* 8.3, pp. 243–259. DOI: 10.1287/ijoc.8.3.243.

Caprara, Alberto (2015). "Timetabling and assignment problems in railway planning and integer multicommodity flow". In: *Networks* 66.1, pp. 1–10. DOI: 10.1002/net.21611.

Costa, Alysson M., Jean-François Cordeau, and Bernard Gendron (2009). "Benders, metric and cutset inequalities for multicommodity capacitated network design". In: *Computational Optimization and Applications* 42.3, pp. 371–392. DOI: 10.1007/s10589-007-9122-0.

Desrosiers, Jacques and Marco E. Lübbecke (2005). "A Primer in Column Generation". In: *Column Generation*. Ed. by Guy Desaulniers, Jacques Desrosiers, and Marius M. Solomon. Boston, MA: Springer US, pp. 1–32. ISBN: 978-0-387-25486-9. DOI: 10.1007/0-387-25486-2_1.

Gendron, Bernard and Mathieu Larose (2014). "Branch-and-price-and-cut for large-scale multicommodity capacitated fixed-charge network design". In: *EURO Journal on Computational Optimization* 2.1, pp. 55–75. ISSN: 2192-4406. DOI: 10.1007/s13675-014-0020-9.

Lübbecke, Marco E. and Jacques Desrosiers (2005). "Selected Topics in Column Generation". In: *Operations Research* 53.6, pp. 1007–1023. ISSN: 0030-364X. DOI: 10.1287/opre.1050.0234.

Raack, Christian (2012). "Capacitated Network Design - Multi-Commodity Flow Formulations, Cutting Planes, and Demand Uncertainty". Doctoral Thesis. Berlin: Technische Universität Berlin, Fakultät II - Mathematik und Naturwissenschaften. DOI: 10.14279/depositonce-3291.

Schrijver, Alexander (1986). *Theory of Linear and Integer Programming*. New York, NY, USA: John Wiley & Sons, Inc.

Sridhar, Varadharajan and June S. Park (2000). "Benders-and-cut algorithm for fixed-charge capacitated network design problem". In: *European Journal of Operational Research* 125.3, pp. 622–632. ISSN: 0377-2217. DOI: 10.1016/S0377-2217(99)00272-6.

Tomlin, J.A. (1966). "Minimum-cost multicommodity network flows". In: *Operations Research* 14.1, pp. 45–51. DOI: 10.1287/opre.14.1.45.

Vanderbeck, François (2005). "Implementing Mixed Integer Column Generation". In: *Column Generation*. Ed. by Guy Desaulniers, Jacques Desrosiers, and Marius M. Solomon. Boston, MA: Springer US, pp. 331–358. ISBN: 978-0-387-25486-9. DOI: 10.1007/0-387-25486-2_12.