Optimization Truck Loading and Scheduling

a thesis for PostNL

Daphne Stok

MSc Econometrics and Management Science
Operations Research & Quantitative Logistics

Optimization Truck Loading and Scheduling

a thesis for PostNL

by

Daphne Stok

to obtain the degree of Master of Science
at the Erasmus University,

Student number:	543591	
Date final version:	July 26, 2022	
Thesis supervision:	Dr. W. van den Heuvel,	Erasmus University, supervisor
	Prof. dr. D. Huisman	Erasmus University, second assessor
	Dr. J.C. de Munck,	PostNL, supervisor
	Ir. J. Mol,	PostNL, supervisor

Acknowledgements

In recent years it has been a challenge for me to find my way in the world of applied mathematics. While educational programs mostly focus on theoretical issues, my interests lie more in applying mathematics to society. I was therefore delighted that I got the opportunity to do my graduation thesis at PostNL. When I told people about my research subject, I noticed that everyone sympathised with the importance of the issue. This gave me excitement, as I was investigating a subject that was so relatable to many people, and hopefully would contribute to the reduction of the struggles that they sometimes experienced. First and foremost, I would like to thank my supervisor from PostNL, Jan de Munck. You gave me the confidence and energy to work hard to achieve this result. You always made time for me if I wanted to discuss something, which I am highly grateful for. Furthermore, I want to thank Joost Mol for your guidance throughout the project. You both always motivated me, wanted to help me tackle issues, and ensured me that everything was going well, which made me feel at ease. Especially during the months when things didn't go as planned, you were always there on behalf of PostNL to advise and help me if necessary. In my opinion, it is an enormous credit to the company PostNL for this reason.

Additionally, I would like to express gratitude to my supervisor from Erasmus University, dr. ir. Wilco van den Heuvel. Thank you for your detailed feedback and the enthusiastic Zoom meetings we have had. Your immense knowledge and plentiful experience have encouraged me all the time in my academic research. Given the circumstances, you were always understanding and even made extra time to help me through this.

Finally, I appreciate all the support I received from my family and friends. Without your tremendous understanding and encouragement over the past few years, it would have been impossible for me to complete my study and to be where I am now.

Daphne Stok

Rotterdam, July 2022

Abstract

The goal of this thesis is to check whether PostNL's current truck loading rules are as efficient as possible. And if not, whether these rules could be optimized by a mathematical-based algorithm. The optimization problem can be seen as a multi-commodity network flow (MCNF) with transport schedules and production times of the trolleys to be transported at each sorting centre. In this paper, two possible methods for solving this optimization problem are discussed. The problem is constructed as an Integer Linear Problem (ILP) and is solved by a commercial solver as Gurobi. To deal with many variables, a Column Generation method has been developed for which a Dantzig-Wolfe decomposition is used. For this last decomposition method, the mathematical model is rewritten into a block-angular structure of the ILP that often comes with MCNF problems. Finally, the mathematical algorithm is tested by comparing the results with the current truck loading rules. Results show that when the number of variables is too large, the original ILP could not be solved by a complex solver like Gurobi. However, the Column Generation algorithm with Dantzig-Wolfe as a decomposition method, lends itself perfectly to solving an MCNF. To speed up the computational time of solving the several optimization problems that comes along with the column generation algorithm, a suitable solver like Gurobi is recommended. Due to the column generation algorithm, the running time is decreased.

Keywords: Multi-commodity, Network Flow Optimization, Integer Linear Programming, Gurobi, Column Generation, Dantzig-Wolfe decomposition

Contents

1 Introduction

This research is motivated by the need to solve the truck loading and transportation problems arising in the delivery process of PostNL. Due to the imbalance in the arrival and departure times of trucks and the production times of the to be transported parcel, the problem of assigning parcels to suitable transport has to be solved. When loading the parcel into trucks optimally and satisfying transportation schedules and corresponding truck capacity, as many packages could be delivered in time at the correct address.

In this thesis, the Truck Loading and Scheduling (TLS) problem is addressed, which is presented by PostNL. PostNL is the largest universal delivery company in The Netherlands and is also publicly listed at Euronext.
Due to the increase in online orders nowadays, on an average day, around 1 million parcels are sent through PostNL's *parcel delivery chain*. These parcels have to be sorted and delivered within 24 hours. That means that around 1 million parcels go into and out of the delivery chain of PostNL.
Due to this rise it is more and more important that PostNL keeps one's head above water when it comes about parcel delivery. The overload of packages must be accommodated by a precisely optimized process for transporting parcels.
Simplified, this parcel delivery chain exists of a collection of parcels to be sent, a first sorting process at a local depot followed by the first transport to a central stationed depot called a 'crossdock', and a second transport to the destination depot. Here the second sorting is started followed by a local postman picking up all of the parcels to deliver. With a total of 25 depots, 5 crossdocks and an average of 1 million packages to deliver every day, one can imagine that this process must be carried out conscientiously.

The PostNL process for the transportation of parcels will be explained further in section 2. The focus of this paper will be to optimize a part of the total delivery chain. By perfectly loading the trucks with trolleys of pre-sorted parcels, the first and second transportation schedules could be used as efficiently as possible. To make sure that for both the first transport and the second transport each trolley is assigned to a truck in the best way possible, an optimization programming tool based on a mathematical model has to be constructed. To construct this, PostNL provides data of one day of the to be transported trolleys and the transport schedules of the trucks driving that same day. With these two datasets a network will be made, where nodes correspond to arriving trolleys that need to be transported and arcs correspond to the truck movements belonging to the given transport schedules. Finally, by optimizing this network, an obtained solution will show which trolley should be transported by which truck while satisfying the transport schedules in order to deliver as many trolleys as possible to the right destination depot in time.

This problem is considered as a Multi-Commodity Network Flow (MCNF) problem. To solve this problem, a well-known linear solver will be used and an algorithm based on a Column Generation method including a Dantzig-Wolfe decomposition method will be developed.

From the suggested research approaches, I hope some valuable insights can be learned. For example, which trolleys can be delivered successfully and which not? Also, how far away are the current load-and-unload rules from the optimum? Could PostNL decrease the number of used trucks? Probably the most valuable question and therefore the main question that will be answered is: 'Is a mathematical-based algorithm the best way to determine the loading rules?' The goal is to improve the current loading rules by 10%.
I expect all of the methods to provide feasible solutions within an acceptable computation time. Therefore, all methods could be used for the planning purposes of PostNL. However, it may be possible that one specific method is the most ideal for PostNL's purposes. In this MCNF problem consisting of many constraints, the Column Generation methodology is expected to be the best working algorithm.

The remainder of this report is organized as follows: A detailed problem description can be found in Chapter 2. Chapter 3 gives an overview of used literature which laid the basis for how the algorithm has been built. Chapter 4 contains a description and short analysis of the provided data. The Integer Linear Program (ILP) is explained in Chapter 5. To improve the results obtained in 5, using a small example, Column Generation as a solution technique for solving the MCNF problem is given in Chapter 6. For this solution technique, a Dantzig-Wolfe decomposition is created later on in this section. Chapter 7 shows the results of the research and the used algorithms and conclusions can be found in Chapter 8.

2 Problem Description

The PostNL process for transporting parcels from webshops to consumers, the other way around, or any parcel that has to be transported, is subdivided into separate processes that can be studied and optimized more or less independently. These processes are: collection, first sorting at a local depot (to level of rough destination area), transport through a so-called crossdocking depot, second sorting at a depot (to consumer addresses) and distribution from second sorting center to consumer. As mentioned before in the Introduction, we will look at a specific part of the total delivery chain. This part is called the *inter transport*.

Each day a different amount of parcel arrives at a depot. On an average day, all depots combined have 1 million parcels to be sorted at the first sorting. By smart machines and people these parcels are sorted and dropped into a trolley with a specific final destination combined with a priority dependent letter and a shift number. The time a trolley is ready for the first transport is known. These trolleys need to be assigned smartly to trucks with the right destination depot within the right shift to satisfy the final destination in the end of the transport. These trucks drive pre-known routes from a depot to a other depot, usually a so-called crossdock, with a pre-known departure and arrival time. The frequency a given route is used depends on the transport planning for that day. The number of trucks that drives the pre-known routes is given in a transport schedule, as are the departure times per depot. A transportation by a truck will be called a *truck movement*. This truck movement has to be done according a strict time-scheme because of the time-dependence deadline of each trolley. Arriving at the destination of the truck after the first movement, the truck has to be unloaded after which these trolleys will be given to the next truck to complete the second truck movement. It is not important to know whether these are the same trucks or not. The two truck movements of above we will call *inter transport 1 and 2* respectively. The routes that will be driven by these trucks and the departure times are again most important and known. It is not only the case that a trolley uses just two truck movements. Sometimes it is better for a trolley to be routed from one crossdocks to another crossdock in order to be delivered in time at the destination. In this case a trolley did make use of three different truck movements. After assigning the trolleys to trucks in the same way as for the first truck movement, the second truck movement will be started followed by the third truck movement in some cases. After completing the last movement, which leads the trolley to the correct destination depot, the trucks will be unloaded and the transportation of the trolley is finished. The transportation process as above mentioned, is illustrated in Figure 2 below. In this figure, the two grey colored blocks represent the collection process before the inter transport and the distribution process after the inter transport. The arrows stand for the truck movements and the different colored trolleys represent trolleys with different destinations.

To be sure the inter transport processes are connected and the concerned trolleys arrive in time at the final destination, the assigning procedure has to be considered precisely. To complete this exercise completely the question 'Which trolley has to be assigned to which truck?' has to be answered at the beginning of both transports. A loading schedule must be developed for perfectly assigning the trolleys to the trucks, in such a way that all time constraints, capacity constraints and priority rules are taken into account for both trucks and trolleys. In the end, all of trolleys have to be assigned to trucks in order to arrive in time at the final destination depot. Also, we may find out that the amount of used trucks is not ideal for the company's current delivery process. In that case we might have to add transports or better in terms sustainability, we may cancel some truck movements. To succeed in solving the above problem a mathematical model will be created and optimized.

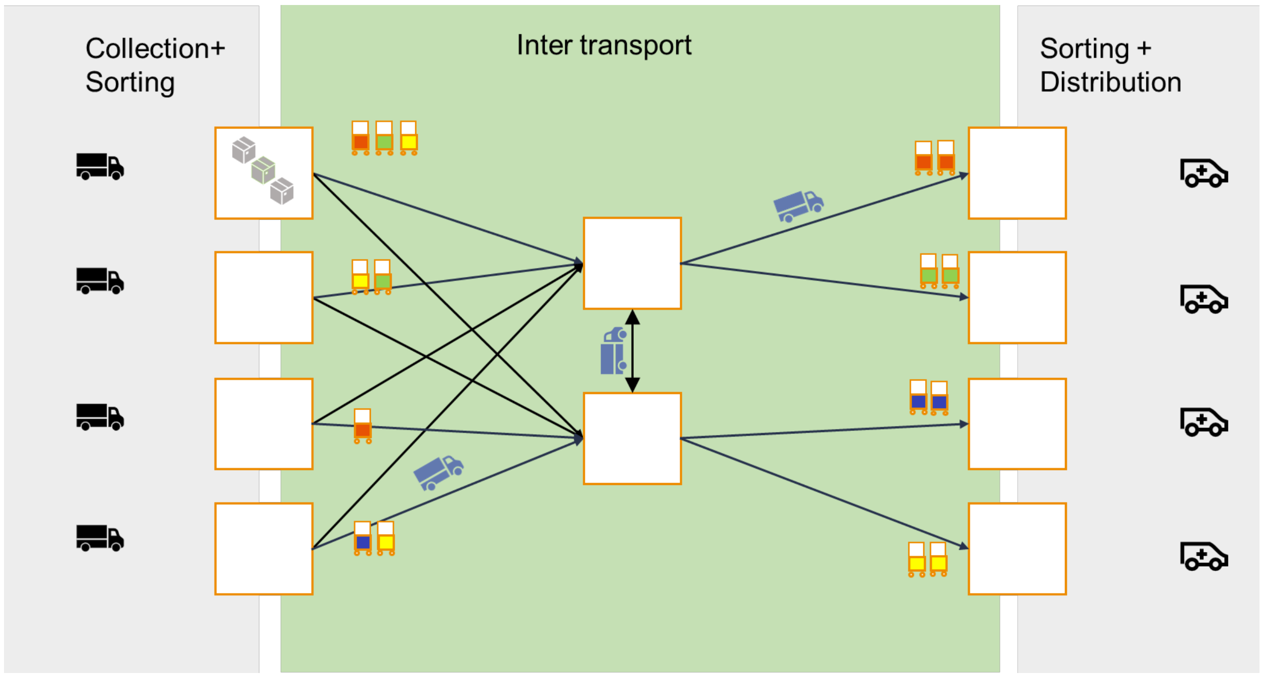

Figure 2: PostNL's inter transport process

3 Literature Study

Over the last few decades, an increasing amount of studies have been conducted on logistic transportation network systems. This is largely due to the huge impact transport has on the greenhouse emissions and consequently on climate change (Bauer et al. (2010)). Also because of the huge raise of online orders our local postal delivery companies need to work more efficiently every day.

A problem that looks at some points similar to the Truck Loading and Scheduling problem is the Freight Car Flow problem. This problem deals with determining an optimal flow of empty and loaded cars in order to maximize profit. Fukasawa et al. (2002) solve this problem given a fixed train schedule with corresponding train capacities while meeting the required demands. They succeed in solving this problem to optimality using an integer multi-commodity flow model and its LP relaxation results in very good upper bounds. This gives rise to applying such a model representation of Fukasawa et al. (2002) to our alike problem. One of the first models that appeared for the Multi-Commodity Network Flow problem, in which multiple commodities need to be transported from source nodes to sink nodes by using arcs while satisfying capacity constraints, is the linear programming model of **Ford1958AFLOW**. In the remainder of this section we review the literature considering the network flow problem that contributes to our research.

3.1 Multi-Commodity Network Flow problem

Ford and Fulkerson introduced the Multi-Commodity Network Flow problem in the late fifties, **Ford1958AFLOW**. Ford and Fulkerson have seen the importance of looking at the structure of maximal multi-commodity flows in networks. They recognize the loss of combinatorial features when looking at multi commodities compared to the single commodity problem.

Although the MCNF is known as one of the most difficult problems yet in network optimization, it is considered a very efficient model in network design. Because the MCNF problem is defined over a network where more than one commodity needs to be transported from source nodes to sink nodes by using arcs while satisfying capacity constraints of these so-called transportation arcs, this MCNF model appears often in logistic applications. For example, Rungwanichsu Kanon et al. (2015) combines two transport schedules to obtain the

optimal path a passenger needs to go through while satisfying capacity constraints.
Not only in the logistics, but also in the telecommunication industry MCNF is a common model to help solve the problem. When you consider a telecommunication problem as a problem in which you need to route messages, you may recognize the similarities between transportation and telecommunication problems. Look for instance at paper Barnhart, Hane, Johnson, et al. (1995) where a message routing problem in the telecommunication industry is modeled as a minimum-cost multi-commodity network flow problem. In particular, the MCNF problem arises when more than one commodity needs to be shipped between specific node pairs without violating any constraints (such as capacity) associated with the arcs.

3.2 Node-arc formulation

Many economic systems can be visualized as networks where nodes stand for commodities, and paths stand for simple or complex production processes. The transportation network is one of the systems that can be described as such a network in the most natural way. Here nodes stand for cities and arcs for connecting two cities. A certain demand is associated with every pair of connected nodes of the network. This demand will be distributed among paths which join the pair of nodes. This gives rise to a traffic pattern. This determination is known as the Traffic Assignment problem. Considering the Traffic Assignment problem of Babonneau (2006) one can obtain an MCNF problem for which the special structure of the constraints is explained in detail. Mathematically, the findings of the optimization problem in this paper have been clarified resulting in a clear overview of the so-called *node-arc formulation*. This node-arc formulation is used as an inspiration for the formulation of this paper where nodes will stand for the 'production' of trolleys at a given depot and arcs will stand for the truck movements between two depots.

3.3 Algorithms for the Multi-Commodity Network Flow problem

Because of the problem's nature of several constraints and a large number of variables, it is even for an easy continuous flow network difficult to find the optimal solution (Salimifard et al. (2020)). It is commonly known that the MCNF problem is hard to solve exactly. Therefore it may be necessary to use a heuristic to approach a good solution for the problem.

The problem in Bevrani et al. (2020) looks very similar to PostNL's case. In this article, the assessment of transportation system flow is reconsidered and an improved multi-commodity network flow model is proposed. The objective of the model is to determine the maximum flows of commodities that a network can sustain. The model makes use of so-called origin-destination pairs (ODP) in which the commodities are transported between specific pairs of locations. Different than in their model we are able to define multiple destination nodes, because we distinguish in time-dependent events at the depots dependent on the shifts. Trolleys could arrive in several shifts at the same depot and still arrive in time for their own shift. A detailed description of the nodes which stand for a shift and depot at the same time can be found in chapter 5.

Also Barnhart, Hane, and Vance (2000) considered a constrained version of linear multi-commodity flow problem, named the origin-destination integer multi-commodity flow problem. The problem was solved successfully using a column generation technique.

The problem of having to deal with an intractable large model also arises with the assignment problem of Cordeau et al. (2001). They solve the problem of simultaneously assigning locomotives and cars to passenger trains, using a multi-commodity network flow model. They take advantage of the fact that the problem can quite easily be decomposed into multiple components. This results in a network representation per order type. This separation technique enables them to use a Benders decomposition approach. Their approach results in an optimal solution to the problem within reasonable computational time for instances of realistically large size. The idea of decomposing the multi-commodity flow model into smaller parts and solve them sequentially can be used in our problem as well.

From paper Dai et al. (2017) there can be obtained column generation turns out to have good properties for solving MCNF problems. Because of the fact that many constraints do not contribute to the optimal solution, only the columns that contribute are taken into account in this solving method. For a new formulation we were inspired by the block angular structure as in paper Wang (2018), as also seen in paper Dai et al. (2017). By this way of formulating the optimization problem suits perfectly when using the column generation method.

The column generation approach, which will be also used for our research, is mostly based on the paper Dai et al. (2017). The idea of only looking at the constraints that contribute to an optimal solution is developed further. To do so the former MILP formulation has been updated into a block angular structure. Thanks to this clear formulation this structure serves perfectly for a Dantzig-Wolfe decomposition. This last decomposition method turns out to be crucial when using the column generation approach. This approach can be seen as an exact solution method which can result in an optimal solution instead of an approximation of the optimal solution.

The way the optimization problem is tackled in this thesis comes down to an exact solution method instead of a heuristic. namely the column generation method. For the company, PostNL, if there exists an optimal solution to the problem, the outcome when using this method is much more meaningful. Due to a limited amount of time reserved for the thesis project only an exact solving method is developed.

4 Data Description

This chapter will give an overview of the data used for the problem as described in Chapter 2. It will describe the data needed to solve the problem. In addition, a description of what both data sets look like will be given.

4.1 Data Description

The data contains information regarding real parcels on an actual, average day that need to be delivered. Parcels have already been assigned to trolleys. The part of the delivery process we will look at is the transportation of these trolleys by trucks. Therefore we need to assign trolleys to the truck movements using up-to-date arrival times of the trolleys. Truck movements connect a given depot to another depot. There are three types of depots. At the *classic depots*, parcels are being sorted and loaded on trolleys followed by the assignment to truck transports. We will call these classic depots just *depots*, even though a better name would be 'a sorting-center'. A central-stationed depot at which parcels will be passed from one transport to another transport is called a *crossdock*. A crossdock therefore serves as an intermediate station. The third type of a depot is called a *depot-plus* which is a depot combined with a crossdock. So a depot-plus has both the functions of a depot and crossdock. There are 30 unique depots, including 3 crossdocks and 2 depot-plus', in the Netherlands. One day of the available trolleys and transport schedules are provided in two separate data sets. These two data sets give useful information about the size of the considered problem.

The first data set includes a number of trolleys, which is equal to 33778. For each trolley the origin depot, destination depot, arrival time at origin depot and the due time in which at least the trolley has to arrive at the destination are available. This due time stands for the moment the trolley has to arrive before. On average there are 10 due times in which a given trolley has to be delivered. To characterize the different trolleys, the origin depot, destination depot and due time are combined to a type k. We call these different trolleys *trolley requests*. Trolley requests thus consist of origin, destination, quantity and due time. Because there are 33778 trolleys, many different types can be obtained. These so-called *trolley types* are not unique which means that they could appear several times in the data set.

The second data set includes a total of 1492 truck movements. This means, when we distinguish in departure and arrival times, there are almost 3000 different *events* of the transports. An event is defined as a time-dependent moment at a given depot and can be seen as a relevant point of time in our network. Therefore an event can represent an arrival or departure time of a truck movement, but can also stand for the due time. Section 5 will look deeper into the definition *event*. For each truck movement, the depots of both begin and end points are known, as well as the time and thus the due time to which the truck movements belong. Furthermore it can be seen that there are also truck movements between the crossdocks. This makes it possible for a trolley request to make use of three truck movements instead of two which might be efficient for optimal delivery.

5 Integer Linear Problem

In this section, the Integer Linear Program (ILP) formulation will be elaborated on in a few steps. First, the sets used for this formulation and the input data will be given in Section 5.1. Next, some assumptions will be introduced in section 5.2 in order to obtain a tractable model. In section 5.3, the multi-commodity network flow formulation to create our mathematical model will be explained.

5.1 Sets and Input Data

Sets

Let $\mathcal{K}$ be the set of all commodity types, which means the set of trolley request types that needs to be transported. The type is based on the destination. So the number of unique types is equal to the number of different destination depots. For further research, there is a possibility to extend the meaning of *commodity type* by defining it as the combination of destination depot and due time. Let I be the set of all depots including all three different types. Let P be the set of all truck movements. These truck movements correspond to the trucks scheduled in both transport schedules that are provided. Truck movements take place between two depots of all kinds. Also, each truck movement is unique which means that there is only one transportation by truck $p \in P$ for which this origin-destination and departure and arrival time combination hold. Let P_d and P_a be the set of departures and arrivals of the truck movements respectively, each consisting of the corresponding time and depot. Let S be the set of all due times. The lower the due time the earlier a trolley request with this specific due time has to arrive at the destination depot. It is allowed for a trolley request to arrive at an earlier due time than its assigned due time.

Input data

– k	commodity type index destination
– i	depot index
– p	index number of transport segment, truck movement p
– n	event index (departure or arrival of a truck movement on a given depot)
– s	due time index

– *cap*	capacity of each truck
– N_n^k	production of commodity type k at event n

5.2 Assumptions

To obtain a more tractable model, we make several assumptions that will be explained in this section. Assumption 1 deals with the rounding of relevant time instants.

Assumption 1. *Production times that take place before the first event or between two consecutive events are merged into one inflow on the relevant event node.*

Assumption 2. *The network does not contain parallel arcs (i.e., two or more arcs with the same tail and head nodes).*

Assumption 3. *All truck movements have arrival times that take place before the first existing due time.*

5.3 Multi-Commodity Network Flow

In a Multi-Commodity Network Flow (MCNF) problem, a commodity is a *good* that must be transported from one or more origin nodes to one or more destination nodes in the network. In practice, these commodities might be packages in a distribution network, messages in a telecommunication network or airplanes in an airline flight network. Each commodity has a unique set of characteristics and the commodities are not interchangeable.

In the MCNF problem of this thesis, the set $\mathcal{K} = 1, 2, \ldots, K$ consists of the commodity types k. We define a commodity of type k as a trolley request with a specific destination depot and due time. The commodity of type k has to be transported and arrived at the right destination depot before the start of the corresponding due time. Each trolley request has its source node and sink node which correspond to the origin depot and destination depot of the trolley request respectively. As previously mentioned, the commodity type k only depends on the destination depot. Therefore we assume that when arriving at the destination depot, every trolley is on time. From Assumption 3 in 5.2 we get that all truck movements have arrival times that take place even before the first existing due time. Because of this assumption, it is not necessary to distinguish between the various due times.

5.3.1 Time discretization

At each point in time for each location, we have to decide which trolleys to send or not to send to which destination. The network consists of so-called *event nodes* which represent the time-dependent moments at specific locations that influence the network. This is why our network also can be announced as a spatio-temporal network. There are four different moments: production time-event of a trolley request of type k at depot i (N), due time of the trolley request of type k, the departure of a truck movement and the arrival of a truck movement. When these four types of events are considered as nodes, in a simplified version the network will look as in Figure 3. In this auxiliary graph, the horizontal arcs between two events at the same depot imply the stock at this given depot. These arcs are called *stock arcs*. Diagonal arcs between two depots represent the truck movements and are called the *truck movement arcs*. M denotes the stock of trolley requests with type k between two events.

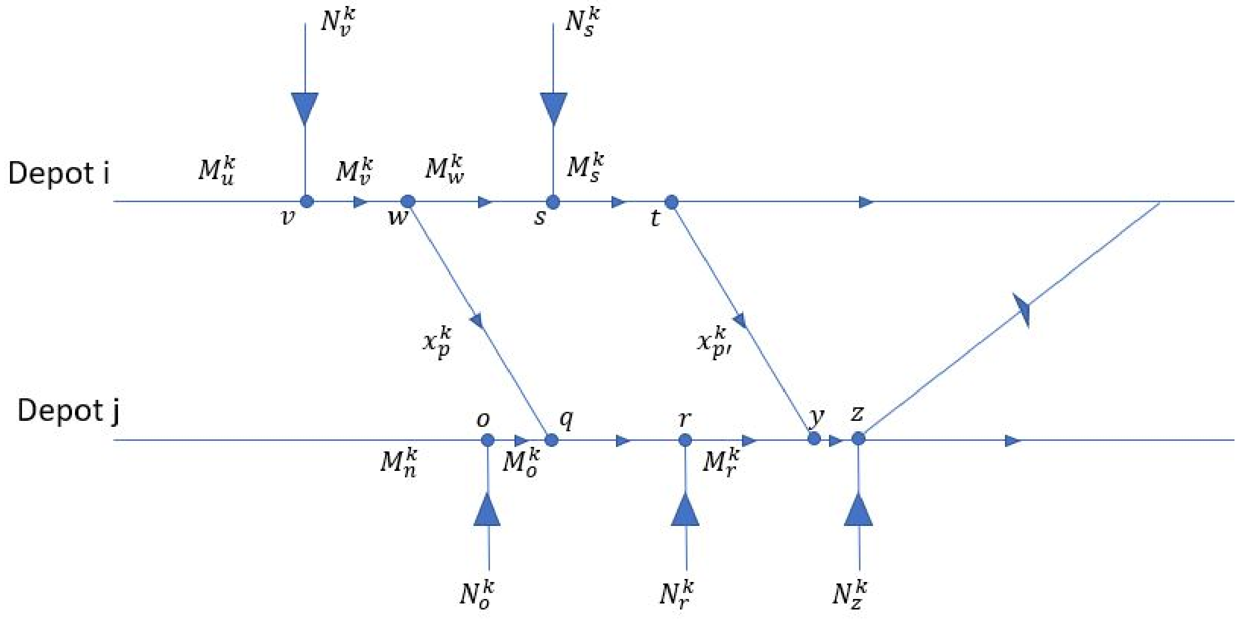

Figure 3: Auxiliary graph of a network flow with four types of events

However, as this is a simplified network, one can imagine that because of the four different types of events the network consists of a lot of nodes. It even consists of nodes which are not as necessary as they seem. There is a way to reduce the size of nodes while still taking the events into account. Instead of adding a node for each production of a trolley, which causes more than 33000 nodes, one can look at the stock of the trolley requests for each type k every time just before the departure or arrival of a truck movement. Clearly it is not important to know if trolleys are produced at the same time or five minutes before the departure of a truck movement. The only thing that is important to know is the stock right before the departure of a truck movement. By not taking into account the production nodes anymore, the number of nodes in our network has decreased to 33000. The same holds for the due time nodes. When the arrival events of the truck movements are known, it can be checked if a given trolley request of type k is on time for the corresponding due time s. By removing these nodes we again reduce the number of nodes in our network. In total, when we only consider the departures and arrivals of truck movements to be nodes, it differs around the 50000 nodes. Look at Figure 4 for an impression of what this network will look like. Note that again this network is used as an example and is not based on real data.

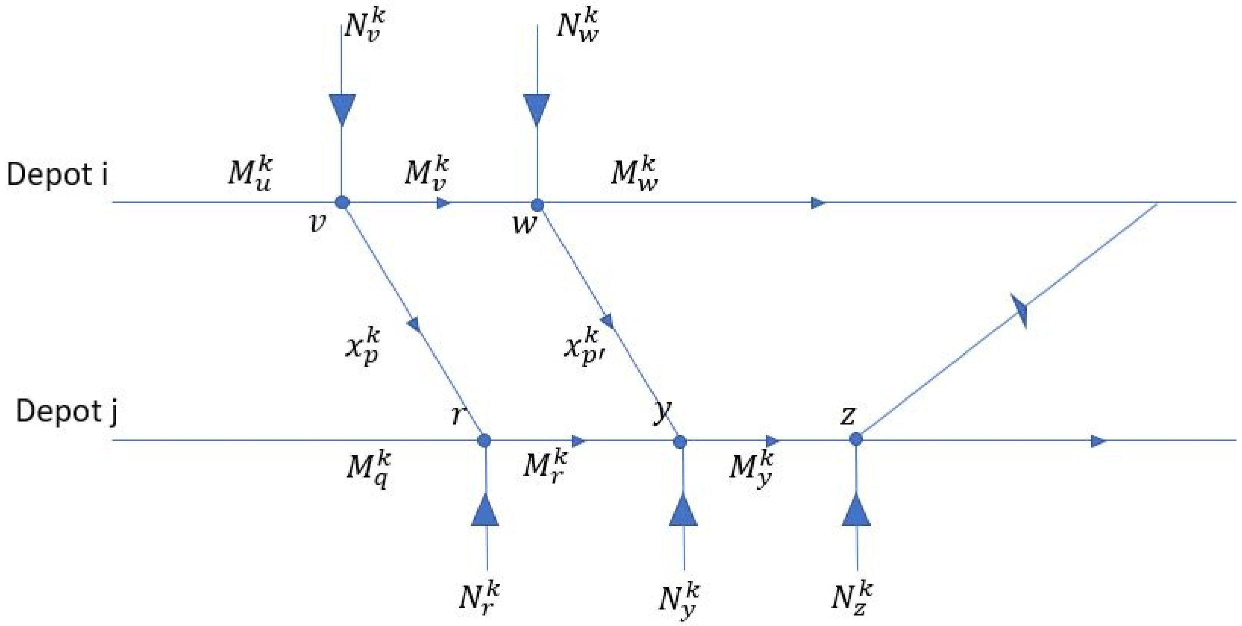

Figure 4: Auxiliary graph of a network flow with two types of events

As in Figure 3, the horizontal arcs between two events at the same depot imply the stock at this given depot. Diagonal arcs between two depots represent the truck movements. The only difference between the two networks above is the number of different events and therefore the number of nodes and arcs in the network. Due to the number of nodes, the network that will be considered in this thesis looks like Figure 3 with nodes that stand for the arrival or departure event of a given truck movement $p \in P$.

A solution is defined as a maximum feasible flow through the network. The transported trolley requests through the network are the most interesting. We want as many trolley requests as possible to arrive in time at the right destination depot right before the corresponding due time. Therefore the problem to tackle is

> *How to decide which trolleys to send from the corresponding origin depot to the destination depot while satisfying all trolley requests and obeying given joint capacities on the arcs?*

Complicating things

Note that the auxiliary graphs from the above figures become quite large due to the discretized time horizon. To make a compact mixed-integer programming model we have to take the following into account:
- for each truck movement arc, how many trucks are used to transport how many trolleys?
- for each stock arc and for each destination, how many trolleys travel along this arc?
- for each location, for each time step and for each purpose, how many trolleys are in stock?

The directed graph $G = (V, A)$ is defined more formally as follows:

Vertices

– $V_d = \{v_p^d\}$: event index nodes standing for departure event of truck movement $p \in P$

– $V_a = \{v_p^a\}$: event index nodes standing for arrival event of truck movement $p \in P$

– $V = V_d \bigcup V_d$

Arcs

- A^{TM} : (v_p^d, v_p^a) : truck movement arcs from departure event node to arrival event node corresponding to the combination of departure of truck movement p and arrival of truck movement p
- A^{ST} : (v, w) : stock arcs with $v, w \in V$ and v, w are consecutive event nodes at the same location
- $A = A^{TM} \bigcup A^{ST}$

Other symbols

- i_k : index of destination depot corresponding to trolley request with type k
- v_i : last event node at depot i
- N_v^k : production of commodity type k at event v

5.3.2 Mathematical Model

In this section, the mathematical model corresponding to the multi-commodity network flow formulation is explained. The first decision variable x_p^k denotes the number of commodities of type $k \in \mathcal{K}$ that makes use of truck movement $p \in P$ and so flows on arc (v_p^d, v_p^a). The second decision variable depends on the stock of commodity type $k \in \mathcal{K}$ right after event node $v \in V$ and therefore denotes the number of commodities of type $k \in \mathcal{K}$ that flows on stock arc $(v, w) \in V$, which will be denoted as M_v^k.

Time discretization represented by events n

At each depot, we distinguish arrival and departure times for all truck movements p, which leads us to the vertices. So each node represents the arrival or departure of a truck p at a given depot i, which we denote with event n. As given above, the set N_v^k stands for the number of trolley requests of a specific type $k \in \mathcal{K}$ that is produced at a depot right before event v, such that it is ready to be transported away by the next truck. N_v^k can be seen as the *inflow* at the network's event node v of specific trolley type k. Later on in this research, the inflow is also indicated by the so-called *production vector* b^k. The decision variable M_v^k denotes the stock of trolley requests of type k right after event v.

What is important, is the number of trolley requests that is available between two events, and which type. Therefore constraints need to be build, which denote the stock at a specific event. These constraints are called the *flow conservation constraints*. Basically this implies that the inflow of a node equals the outflow.
These conservation constraints are given in an iterative way:

$$M_u^k + N_v^k = x_p^k + M_v^k$$
$$M_r^k + N_r^k = -x_p^k + M_q^k$$

Note that the used indices are based on the auxiliary graph from Figure 4

Based on Figure 4, the first constraint stands for the flow conservation of the *begin node* v of an arc with $v \in V_d$. The second constraint stands for the flow conservation of the *end node* r of the same arc with $r \in V_a$. Both constraints together imply what enters the flow of the arc and also leaves the flow of the arc.

The objective is to maximize the total number of *correctly* positioned trolley requests at the end of the process, which is only the case when a trolley request of type k is in time, so before the corresponding due time, and at the right destination depot corresponding to the type k. This has to be done in a way such that each commodity k will be transported through an arc of decision variable x_p^k satisfying capacity constraints and corresponding transport schedules of transports p. Therefore we define two sets to make this objective easier to formulate.

- $E_i = \{k | i_k = i\}$: set of trolley requests with destination depot i

By using these sets, we could check which trolley request of type k has arrived at the right destination depot corresponding to type k. The following summation implies the above:

- $\sum_i \sum_{k \in E_i} M_{v_i}^k$: number of trolley requests arrived at the right destination depot in time for their due time

By looking at the intersection of the two defined sets, the number of trolley requests arrived at the right destination depot corresponding to each type k, can be obtained.

Maximum flow of the MCNF problem

From the above it can be seen that when we are looking for the greatest number of correctly delivered trolley requests $\sum_i \sum_{k \in E_i} M_{v_i}^k$ needs to be maximized. Therefore this paper will focus on this objective.

The Integer Linear Program (ILP) can be formulated as follows:

$$\begin{aligned}
\max \quad & \sum_i \sum_{k \in E_i} M_{v_i}^k & & (1)\\
\text{s.t.} \quad & M_u^k + N_v^k = x_p^k + M_v^k & \forall u, v \in V_d, p \in A, k \in \mathcal{K} \qquad & (2)\\
& M_r^k + N_r^k = -x_p^k + M_q^k & \forall r, q \in V_a, p \in A, k \in \mathcal{K} \qquad & (3)\\
& \sum_k x_p^k \leq \text{cap}_p & \forall p \in A \qquad & (4)\\
& \sum_k M_v^k \leq \text{cap}M & \forall v \in V \qquad & (5)\\
& x_p^k \in \mathbb{N} & & (6)\\
& M_v^k \in \mathbb{N} & & (7)
\end{aligned}$$

Objective (1) maximizes the number of trolley requests for each type k that has been transported to the destination depot within or before the due time corresponding to each type. The first set of constraints (2) and (3) reflect the flow conservation constraints from which the number of trolley requests M could be obtained. Constraint (4) states that the sum of the path flows passing through the arc is at most the capacity of the arc (Ahuja et al. (1993)). Constraint (5) states that the sum of the path flows passing through this arc can be no more than the allowed the stock capacity at each depot. Constraint (6) enforces non-negative and integer values for x_p^k and constraint (7) enforces non-negative and integer values for M_v^k.

5.4 Solving the ILP

To solve the ILP from equations (1)-(7), a mathematical model is created in *PyCharm* which is a programming environment for the use of the programming language *Python*. To fulfill the maximization of the objective, some test nodes and test arcs are created. These test nodes and test arcs stand for the *last moment* for each commodity type, which is in our case a possible destination depot. From these test nodes and arcs useful information about the success of satisfying the trolley requests can be obtained.
The optimization problem will be solved using the solver Gurobi. Because of the original data

set size is very large, a test data set has been made. This set consists of less truck movements and fewer parcels. The results of this test data set are not as expected. The objective is very low and therefore stands for a small amount of succeeded trolley requests. The most plausible explanation for this is the choice of the nodes and arcs of the network. The nodes are made of four depots and two sorting centers. The arcs are all known truck movements between the possible nodes. After the network has been made, all matching trolley requests are collected to one data set. These are used as the inflow of the model. In reality, certain trolleys may be transported via other depots and therefore make use of different truck movements. As a result, it is possible that while looking at this test network, there is currently no suitable truck movement, also taking into account the capacity of each arc. This explains that the size of the objective is too small and therefore the poor outcome. The functioning of the model must therefore be checked with another data set.
By making another data set containing 24 trolley requests and two movements, it can be checked whether the objective generated by the Gurobi model is the right objective or not. In section 6.1, the instance and solution of this example are illustrated. By solving this problem by hand first and comparing this result with the Gurobi result, the reliability of the model can be checked. It follows that the objective is indeed the total amount of the 24 trolley requests. Therefore we can conclude that the mathematical model mentioned before behaves in a good way. In the next chapter, this small example is discussed to illustrate using a specific solving method.

The original data set is still too large to run. One option is to run the ILP in a cloud based service such as Amazon Web Services. Another possibility is to divide the solving of the ILP into smaller parts to solve. This can be done with a heuristic which will bring us to the next chapter. This approach will take some optimization but seems promising to yield positive results.

6 Methodology

As seen in section 5.3.2, the ILP contains two conservation constraints and two capacity constraints, but at the same time deals with a large number of variables. One can imagine that taking into account a large computational time, solving subproblems of the original problem may save some time. Pfetsch et al. (2006) explains that a problem with a network of 1000 nodes and 10000 edges through which 1000 commodities move, would require at least 80 GB of storage to solve the linear programming model. Our network consists of 3016 nodes, 4470 edges and 33778 commodity types which therefore requires at least 450 GB. Following the recommendation of Pfetsch et al. (2006), which states that often column generation can solve a network problem like this with much less storage and time.
Another reason why using a more efficient solving method than just solving the ILP with Gurobi will be explained by the following *small example.*

6.1 Small example

A small example that will be discussed as a case study, is shown in Figure 5 and table 1. There are 5 nodes spread over two different depots, 5 arcs between these nodes and a total of 24 trolley requests that need to be fulfilled. As can be seen in Figure 5, at nodes 1 and 2 there are inflows of 17 and 7 trolley requests respectively. Because these trolley requests all have the same destination depot, there has to be dealt with just one commodity type k. Nodes 3 and 4 correspond with the event nodes, which are arrivals of truck movements from nodes 1 and 2 respectively. Node 5 is created by the model to make it more obvious to know whether the trolley requests are fulfilled. The arc connected to this self-created last event node represents all of the trolleys that are transported to the depot belonging to this last event node. This arc is therefore not limited to a predefined capacity.

Remark. *A commodity type is based on the destination depot of the corresponding trolley request.*

For each trolley request with a specific commodity type k, the production vector b^k must be satisfied. This means that paths between the origin depot of the trolley request with type k ($O(k)$) and the destination depot belonging to this type k ($D(k)$) need to be created. With the vertical arcs in the direction of nodes 1 and 2, the inflow is indicated. Each number next

Table 1: Sample commodities

Trolley request with type k	$O(k)$	$D(k)$	transportation demand d^k
Trolley request 1	1	5	17
Trolley request 2	2	5	7

to the other arcs indicates the number of trolley requests that are being moved along these arcs. Their total flows are equal to the transportation demand. Because all trolley requests have the same destination depot -thus the same commodity type k- we want them to be transported to either event nodes 3, 4 or 5.

The optimal solution of this small problem is obtained using the solver Gurobi and is shown in table 2. From this table, we could also obtain that the total transportation demand for each trolley request is fulfilled which implies that our solution is optimal.
The path flows that are used to obtain the solution of Table 2 are shown in Table 3.

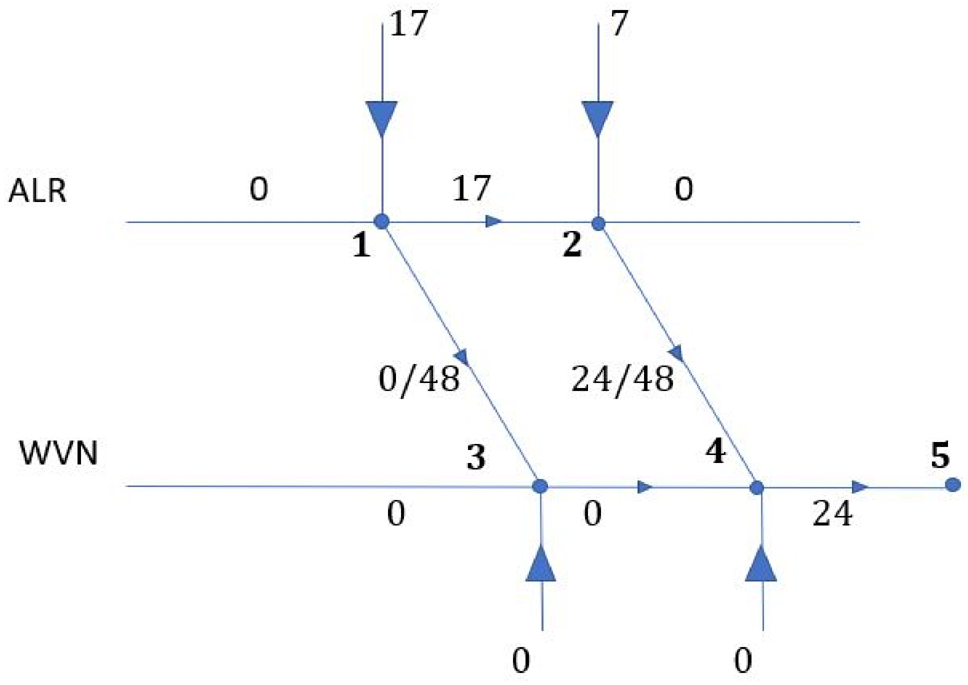

Figure 5: Small example

Note that two truck movement arcs are not being used while satisfying all of the trolley

Table 2: Solution with Gurobi for the small example

Arc	Trolley request 1	Trolley request 2
(1,2)	17	0
(1,3)	0	0
(2,4)	17	7
(3,4)	0	0
(4,5)	17	7

Table 3: Solution for the small example with the path flows

Commodity	Path
Trolley request 1	$1 \to 2 \to 4 \to 5$
Trolley request 2	$2 \to 4 \to 5$

requests. This means that two arcs do not contribute to an optimal solution. One can imagine that when using a much larger data set, having unused arcs is, even more, the case. This property shows the potential for solving MCNF problems in a more efficient way. This will lead us to the Column Generation method.

6.2 Column Generation method

Column generation is a method to efficiently solve linear programs with a large number of variables and is therefore commonly used to solve an MCFP problem. The basic idea is that new columns are added into the solution step by step as necessary to improve the objective function, which prevents the use of a large number of variables that was looked at before. In this subsection, the exact solving method will be explained and discussed.

We want to solve an integer linear programming model, called master ILP, and consider a restricted master problem (RMP), which contains all constraints of the master ILP, but only a subset of the variables. From this primal problem dual information can be obtained. This information, in the form of dual variables, is used to generate pricing problems for all the commodities to try to identify columns with a *profitable reduced cost*. The subproblem (SP) is to solve these pricing problems. Because these subproblems have fewer variables included it makes it easier for the algorithm to solve. While an optimal solution exists to one of these subproblems which is equal to a negative number, the column corresponding to this optimal value needs to be added to the current RMP. Imagine that a column represents a total flow through the network of a commodity-type k. This while-loop will continue until all of the pricing problems return positive values and an optimal solution to the original MCNF can be obtained.

6.2.1 Block-angular structure of the ILP

In this section, the block-angular structure of the MCNF constraints from the paper Wang (2018) will be elaborated, which suits perfectly when using a decomposition technique such as Dantzig-Wolfe decomposition as used in paper Dai et al. (2017). Namely, due to the structure the set of possible outcomes, which are the flows of the commodity types, can be modeled as the integer points (extreme points) of a polyhedron which is defined using this specific structure. This will be further explained in section 6.3.1.

The directed graph $G = (V, A)$ as defined in section 5.3 again describes the network. To simplify some notations, an arc (v, w) can also be announced as arc l. Also, we assume V to have size n and A to have size m. To reformulate the Integer Linear Program as seen in constraints (1) - (7), the two types of decision variables are merged into one type of decision variable $\mathbf{x}$. This new type of decision variable $\mathbf{x}$ stands for the flow vector of each edge for commodity type k with $k \in \mathcal{K}$ and will be denoted as $\mathbf{x}^k = [x_1^k, x_2^k, \ldots, x_m^k]^T$. To define the maximum flow objective in the same way as in paper Dai et al. (2017) we need to find a cost vector $\mathbf{c}$. Because we only want to take into account the trolleys that have been transported to the correct destination depot, values 0 and 1 are given to $\mathbf{c}$ to check whether the commodity is placed right. Therefore for each type k the vector $\mathbf{c}$ gets value 1 when the index matches the last moment at the depot corresponding to k. From the test nodes and test arcs that have been created, we can obtain useful information about the success of satisfying the trolley requests. So for each commodity type k, which corresponds to a destination depot, the vector value is 1 for the last event at this depot and 0 otherwise.
The capacity vector is also made by merging the two capacity constraints (4) and (5) and will be denoted as $\boldsymbol{cap}$.
The incidence matrix between nodes and arcs is described as $n \times m$ matrix $\mathbf{B} = [B_{il}]_{n \times m}$. It has size $n \times m$ with n the number of nodes, m the number of arcs and for the lth arc (i, j), $B_{il} = 1$ and $B_{jl} = -1$. In our case, the vector usually known as the demand vector equals

the *production vector* $\mathbf{b}^k = [b_1^k, b_2^k, \ldots, b_n^k]^T$.

Two important constraints need to be considered. The first constraint is the arc capacity constraint. This means that the flow on each edge can not exceed the flow capacity. The second constraint can be seen as the transportation demand which means that all commodity types need to be transported from their origin to their destination. The capacity constraint considers all commodity types together. However, the transportation constraint is essentially the sum of a set of single-commodity flow problems.

The new formulation of the LP using the above details is given as follows:

$$\max \quad \sum_{k \in \mathcal{K}} (\mathbf{c}^k)^T \mathbf{x}^k \tag{8}$$

$$\text{s.t.} \quad \sum_{k=1}^{K} \mathbf{x}^k \leq \mathbf{cap} \tag{9}$$

$$\mathbf{B}\mathbf{x}^k = \mathbf{b}^k \qquad \forall k \in \mathcal{K} \tag{10}$$

$$\mathbf{x}^k \geq 0 \qquad \forall k \in \mathcal{K} \tag{11}$$

Equation (8) is the objective function of the total cost. Equations (9) and (10) are the edge capacity constraint and node flow equilibrium equation, respectively. Equation (11) is a non-negative constraint. As seen in the small example (section 6.1), there is a lot of potential for solving the MCNF problem more efficiently.

Consider equations (8) to (11). The flow x^k indicates the size of the flow for each arc per commodity type. When one looks at the zeros in this vector, it becomes clear which arcs are not used for this x^k. The diagonal arcs, which represent the truck movements, for which x^k is equal to 0 for all $k \in \mathcal{K}$, actually indicate unnecessary arcs in the network.

6.3 Column Generation as a solution technique

Section 6.2.1 shows that the formulation of the MCNF problem has a structure that could be exploited during the computation of a solution. Equation (10) can be divided into K independent sets of equations for each commodity k. In the case that commodities do not influence each other in this transportation constraint, the MCNF problem can be solved easily by looking at the independent single-commodity flow problems. We have seen in the

small example from section 6.1 that most variables do not contribute to the solution, see table 2. Namely, several arcs are not being used by the commodity-type k. As might be expected, when considering a larger network, still many arcs will not be used. Based on this observation a column generation algorithm is obvious to use as a solution technique. In the equations (8)-(11) there are $K \times m$ variables in the model. However, as mentioned previously, many variables do not contribute to the optimal solution. Therefore, using the column generation algorithm, new columns are added into the solution step by step when necessary to improve the objective function. This process is explained below.
First a set of initial columns, which is a set of complete flows of corresponding commodity types k, is created to construct RMP. The solutions to this primal problem will return dual variables. This dual information is used to generate pricing problems for all the commodities to try to identify columns with a *profitable reduced cost.* If $u_\star^k$ is the optimal objective function of the kth subproblem following from the pricing problem, then $u_\star^k \geq 0$ implies the MCNF to reach the optimal solution and $u_\star^k < 0$ implies the solution to the pricing problem has to be added as a new column into the set of columns of the RMP. Remember, this column represents a complete flow through the network of corresponding commodity type k. The algorithm will continue until the optimal solution is reached.

6.3.1 Dantzig-Wolfe decomposition and master problem

In this section, the Dantzig-Wolfe decomposition is described in detail and split into a master problem and several sub-problems. The master problem consists of *active* columns.
The constraints (9) and (10) can be written as (13) and (14), respectively:

$$\max \quad \sum_{k\in\mathcal{K}} (\mathbf{c}^k)^T \mathbf{x}^k \tag{12}$$

$$\sum_{k=1}^{K} \mathbf{x}^k + \mathbf{s} = \mathbf{cap} \tag{13}$$

$$\mathbf{B}\mathbf{x}^k = \mathbf{b}^k \qquad \forall k \in \mathcal{K} \tag{14}$$

$$\mathbf{x}^k \geq 0 \qquad \forall k \in \mathcal{K} \tag{15}$$

Consider the auxiliary set

$$G^k = \{\mathbf{x}^k \in \mathbb{R}^m | \mathbf{B}\mathbf{x}^k = \mathbf{b}^k, \mathbf{x}^k \geq 0\}$$

which is also a polyhedron. If $\mathbf{x}^k$ is feasible for the ILP problem then $\mathbf{x}^k \in G^k$.
For each commodity k, the decision variable vector $\mathbf{x}^k$ can then be decomposed as

$$\mathbf{x}^k = \sum_{j=1}^{n_k} \lambda_j^k \mathbf{y}_j^k \tag{16}$$

where $\mathbf{y}_j^k$ for $k = 1, 2, \ldots, K$ and $j = 1, 2, \ldots, n_k$ are the extreme points of G^k and n_k is the number of extreme point solutions (Tomlin (1966)). The extreme directions of G^k can be omitted, as discussed in the next section. Now $\mathbf{x}^k$ can be seen as a convex combination of the variables $\mathbf{y}_j^k$ with λ_j^k the coefficients of these convex combinations for $j = 1, 2, \ldots, n_k$ and $k \in \mathcal{K}$. It holds that λ_j^k are the coefficients of the convex combinations of $\mathbf{y}_j^k$ satisfying

$$\sum_{j=1}^{n_k} \lambda_j^k = 1$$
$$\lambda_j^k \geq 0$$

In the LP (8) - (11), we can substitute $\mathbf{x}^k$ by $\sum_{j=1}^{n_k} \lambda_j^k \mathbf{y}_j^k$ and add the constraints $\sum_{j=1}^{n_k} \lambda_j^k = 1$ and $\lambda_j^k \geq 0$. Furthermore equations (13) and (14) are used instead of (9) and (10) respectively.

The master problem (master LP) can then be formulated as

$$\max z(\lambda) = \sum_{k=1}^{K} (\mathbf{c}^k)^T \sum_{j=1}^{n_k} \lambda_j^k \mathbf{y}_j^k \tag{17}$$

$$\text{s.t.} \quad \sum_{k=1}^{K} \sum_{j=1}^{n_k} \lambda_j^k \mathbf{y}_j^k \leq \mathbf{cap} \tag{18}$$

$$\sum_{j=1}^{n_k} \lambda_j^k = 1, \qquad \forall k \in \mathcal{K} \tag{19}$$

$$\lambda_j^k \geq 0, \qquad j = 1, 2, \ldots, n_k \tag{20}$$

and in standard form with slack variable vector $\mathbf{s}$ as:

$$\max\ z(\lambda) = \sum_{k=1}^{K} (\mathbf{c}^k)^T \sum_{j=1}^{n_k} \lambda_j^k \mathbf{y}_j^k \tag{21}$$

$$\text{s.t.}\ \sum_{k=1}^{K} \sum_{j=1}^{n_k} \mathbf{I}\lambda_j^k \mathbf{y}_j^k + \mathbf{I}\mathbf{s} = \mathbf{cap} \tag{22}$$

$$\sum_{j=1}^{n_k} \lambda_j^k = 1, \qquad \forall k \in \mathcal{K} \tag{23}$$

$$\lambda_j^k \geq 0, \qquad j = 1, 2, \ldots, n_k \tag{24}$$

$$\mathbf{s} \geq 0 \tag{25}$$

In the LP problem of constraints (21) - (25) λ is the variable now. This master problem is called the Dantzig-Wolfe reformulation (DWR).
When the above master problem is solved, a solution to the primal problem can be obtained. We have to find a variable λ with a positive reduced cost. We reuse the reformulated problem to a RMP and assign dual variables β to the linking constraints (22) and dual variable μ_k is assigned to constraint (23), respectively. Assume that after initializing the Dantzig-Wolfe algorithm and the corresponding RMP, some columns have been found.

To check the optimality of the LP solution, the reduced costs are considered.
The reduced cost corresponding to a non-basic λ_j^k is:

$$((\mathbf{c}^k)^T - \boldsymbol{\beta}^T)\mathbf{y}_j^k - \mu_k,$$

which is equivalent to

$$-\mu_k - (\boldsymbol{\beta}^T - (\mathbf{c}^k)^T)\mathbf{y}_j^k \tag{26}$$

From these reduced cost the following subproblem follows for $k \in \mathcal{K}$:

$$\rho_k = \max_{j=1,2,\ldots,n_k} \{-\mu_k - (\boldsymbol{\beta}^T - (\mathbf{c}^k)^T)\mathbf{y}_j^k\} \tag{27}$$

The pricing problem is to solve the above subproblems to find

$$\rho = \max\{\rho_k\} \tag{28}$$

Note that the following observation can be made:

$$\begin{aligned}\rho_k &= \max_{j=1,2,\dots,n_k}\{-\mu_k - (\boldsymbol{\beta}^T - (\mathbf{c}^k)^T)\mathbf{y}_j^k\} \\ &= -\mu_k + \max_{j=1,2,\dots,n_k}\{-(\boldsymbol{\beta}^T - (\mathbf{c}^k)^T)\mathbf{y}_j^k\} \\ &= -\mu_k - \min_{j=1,2,\dots,n_k}\{(\boldsymbol{\beta}^T - (\mathbf{c}^k)^T)\mathbf{y}_j^k\}\end{aligned} \tag{29}$$

The pricing problem is solved to try to identify columns with a profitable reduced cost. If such columns are found, the DWR and thus the RMP are reoptimized. The dual variables that came out the primal problem, will be used as an input for the pricing problems.

6.3.2 Pricing problems

As mentioned previously, the optimal solution to the primal problem from above returns the optimal value of the dual variables $\boldsymbol{\beta}$ and μ_k.
These variables are used as an input for the pricing problems for each commodity type k. For each commodity k, the pricing problem as described in equation (29) can be decomposed into subproblems SP_k for $k = 1, 2, \dots, K$, for which each subproblem can be seen as the following LP.

$$\min u(\mathbf{x}^k) = (\boldsymbol{\beta}^T - (\mathbf{c}^k)^T)\mathbf{x}^k \tag{30}$$

$$\text{s.t. } \mathbf{B}\mathbf{x}^k = \mathbf{b}^k \tag{31}$$

$$\mathbf{x}^k \geq \mathbf{0} \tag{32}$$

Where $\boldsymbol{\beta}$ is the vector with dual variable from the constraints of the RMP. The original problem is bounded due to the fact that flows can not be negative and that costs and capacities are finite. Since the original problem is bounded, the subproblems are as well bounded. This means that the parts in DWR containing extreme directions can essentially be removed from the problem formulation.

Solving the subproblems shows whether the current master LP problem has to be updated with new columns or not. We assume that $u_\star^k$ is the optimal value of the above subproblem. Using the pricing problems from equation (29), it is proven by Ahuja et al. (1993) that when $\rho_k = -\mu_k - u_\star^k \leq 0$, which is equal to $\rho_k = u_\star^k + \mu_k \geq 0$, an optimal solution for the original MCNF problem can be obtained. If $\rho_k = -\mu_k - u_\star^k > 0$, which is equal to $\rho_k = u_\star^k + \mu_k < 0$, for $k = 1, 2, \dots, K$, then $x_\star^k$ with $k = 1, 2, \dots, K$ needs to be added as a new column into the current column set. A new version of the RMP will be solved. These obtained new dual variables will be used again as an input for the pricing problems. Again the value of ρ_k with the new $u^k_\star$ will be checked. A loop will continue this process until an optimal solution is reached, which is the case when $u^k_\star \geq 0$ is found.

The procedure as above described is schematically represented in the following pseudocode.

Table 4: Pseudocode

```
Procedure: Column generation with Dantzig-Wolfe decomposition (CGDW)
begin
Initialize the master problem and subproblems
and initialize the RMP.
Assume that the optimal solution of the pricing problem of 29 is equal to $u^k_\star + \mu_k < 0$.
while ($u^k_\star + \mu_k < 0$):
    do
        Solve the RMP and compute dual variables $\beta$ and $\mu$
        Update the Pricing Problem with dual information and solve SP$_k$ ((30)-(32))
          if there exist a ($u^k_\star + \mu_k < 0$) for $k = 1, 2, \ldots, K$:
              add $x^k_\star$ as a column to the RMP
          else:
              an optimal solution for the original MCNF is found
end
```

6.3.3 Initialization of the RMP

As described before, to find a variable λ the Dantzig-Wolfe algorithm has to be initialized. Therefore, the reformulated problem is reused to an RMP where no variables λ^k_j are included. In the situation where all λ^k_j's are zero in constraints (21) - (25), no choice has to be made for a certain λ^k_j that deviates from zero in order to satisfy constraint (23). Slack variable vector $s = (s_1, ..., s_n)^T$ is included and artificial variables $t_1, t_2, \ldots t_K$ are introduced to ensure that a feasible solution exists. The auxiliary variables t_k can be added to constraint (23) such that λ^k_j equals zero and still satisfies this constraint (23).
We choose M so high that the artificial variables are never part of an optimal solution to the original problem. After trying out different values for M, setting M equal to 10000 turns out to be large enough.

After the procedure as described above and setting all λ_j^k's equal to zero, the initialization of the RMP can be written as the following set of constraints:

$$\max \; -M\sum_{j=1}^{K} t_j \tag{33}$$

$$\text{s.t. } \mathbf{s} = \mathbf{cap} \tag{34}$$

$$t_j = 1, \qquad \forall j \in 1, 2, \ldots, K \tag{35}$$

$$\mathbf{s}, t_j \geq 0 \qquad \forall j \in 1, 2, \ldots, K \tag{36}$$

From this, it is clear to see what the optimal solution to this RMP is. Dual variables β have been assigned to the linking constraints (22) and dual variable μ_k has been assigned to constraint (23) respectively. When solving the primal problem from equations (17) - (20) we get the vectors of the dual variables. Thus, from the primal problem the dual information can be obtained. To get this information the optimal basis to the RMP from above is used and the dual variables are acquired. The optimal basis to this RMP is, which follows immediately from the fixed notation in (33) - (36), $(s_1, \ldots, s_K, t_1, t_2, \ldots, t_K)$.
We find the first set of dual variables using standard notation, with I the identity matrix and $c_b = [0 \ldots 0 \; -M \cdots -M]$:

$$\overrightarrow{\beta} = c_b^T I^{-1} = c_b^T = [0 \ldots 0 \; -M \cdots -M] \tag{37}$$

6.4 Benchmark

PostNL's current charging rules are used as a benchmark. The current trolley loading rules at PostNL are based on the *hinterland* of the crossdock destination of incoming trucks. Each crossdock has its own hinterland. This hinterland is the grouping of a certain number of predetermined depots. Every trolley is sent by trucks to the crossdock, whose hinterland contains the destination depots of the trolleys. Arriving at the crossdock, after inter transport 1, the trolley enters the truck for inter transport 2, which has the final destination of the trolley as its destination.

One can imagine that it can be done more efficiently by looking for each trolley and corresponding destination depot, to which truck it should be assigned. In this way the loading process will be optimized instead of hinterland-based loading. The algorithm developed in this thesis that behaves best is expected to be close to the optimal solution in a fraction of a time. This would improve the current loading planning of trolleys into trucks.

6.5 Extension to the current LP

As announced in Chapter 1, it is also important to look at the number of truck movements used. In the LP of constraints (8) - (11), the number of correctly transported trolleys is maximized.
When it is assumed that the trolleys arrive at the correct destination depot, this can be made into a constraint. Instead of looking at this as an objective, i.e. maximizing the corrected transported trolley flow, it can now be seen as an objective to minimize the use of the truck movement arcs. For example, there are certain costs associated with using an arc. These costs represent the duration of the arc. The longer a truck movement lasts, the more expensive the arc is. A new cost vector **c** is created, which consists of the number of time units necessary for each diagonal arc. Remember that each truck movement is represented by a diagonal arc in the network. Note that for the horizontal stock arcs the cost vector **c** contains a zero.
The objective can be seen as the following constraint with $\mathbf{x}^k$ the flow of commodity type k and $\mathbf{c}^k$ the cost vector as described above:

$$\min \quad \sum_{k\in\mathcal{K}} (\mathbf{c}^k)^T \mathbf{x}^k \tag{38}$$

The number of arcs is fixed in PostNL's current network. With this objective, we create an overview of the number of necessary time units. From this, we obtain the number of time units that could be saved.
Applying this extension results in the following new formulation for the LP.

$$\begin{aligned}
\min \quad & \sum_{k\in\mathcal{K}} (\mathbf{c}^k)^T \mathbf{x}^k && && (39)\\
\text{s.t.} \quad & \sum_{k=1}^{K} \mathbf{x}^k \leq \mathbf{cap} && && (40)\\
& \mathbf{B}\mathbf{x}^k = \mathbf{b}^k && \forall k \in \mathcal{K} && (41)\\
& \mathbf{x}^k \geq 0 && \forall k \in \mathcal{K} && (42)
\end{aligned}$$

The results obtained when minimizing the costs flows with CGDW can be seen in Chapter 7.

7 Results

In this section, the results obtained from the two methods are given. For solving the optimization problems, linear solver Gurobi version 9.1.2 is used together with Python version 3.8. Also, a comparison is made between the methods and the benchmark instance, which consists of the current truck loading rules of PostNL. Firstly, the different datasets that are used will be explained. After that, the results of the ILP problem solved with Gurobi are given. Then, the results generated by the Column Generation method with Dantzig-Wolfe decomposition are given and lastly, a comparison is made between these two methods.

Due to the limited time for the sorting process, it is important that the algorithm gives good results in a very short time. Therefore it is assumed that a reasonable computation time is less than two minutes. To understand the results section well, it is recommended to consider one more time the data on which the network used in this thesis is based. This is explained in section 6.1.

7.1 Data sets

Because the original data set consists of more than 30000 trolleys and almost 2000 truck movements, we deal with a network consisting of many nodes and arcs, not to mention the number of commodity types moving through it known as the flow. Running this data set requires a memory of 153 MiB, which therefore causes an error during computation. To successfully obtain results for this data set, the code should be executed in a cloud computing program. For example, PostNL uses Amazon Web Services. However, the behavior of the algorithm could also be checked by looking at smaller versions of this data set. To create these, depots have been deleted randomly from the data set, just like the truck movements that took place at one of these depots and the trolleys that have one of these depots as their origin or destination depot. This process is repeated four times, creating four new different-sized datasets which will be considered. Data set 1 contains three depots of which one crossdock. Data set 2 contains six depots of which two crossdocks. Data set 3 consists of 16 depots of which three crossdocks and data set 4 consists of 22 depots of which four crossdocks. Also, the small example described in section 6.1 will be considered.

To be able to say more about the difficulty of the original data set of an average day, the data from an extra day is simulated. The original data set for this extra day has twice as many trolleys and 1.5 times as many truck movements. The two groups of data sets that are considered, are called group 1 and group 2, respectively. This extra day could represent a day in November in which the hustle and bustle of the coming December month are already visible. It is expected that the methods for group 2 require a longer runtime and therefore perform less well than for group 1.
In group 2, the small example from Figure 5, section 6.1 is not considered again. In Tables 5 and 6, the information of the two groups is presented in order of size. The headers of the columns are the number of trolleys, the number of truck movements, the number of commodity types k, the number of nodes in the network, and the number of arcs in the network respectively.

Data set group 1	nr. of trolleys	nr. of transports	nr. of comm. types k	nr. of nodes	nr. of arcs
Small ex. Figure 5	24	2	1	5	5
Data set 1	410	48	3	96	144
Data set 2	670	129	6	262	384
Data set 3	12202	437	16	890	1311
Data set 4	22111	939	22	1900	2817
Original	33778	1492	32	3016	4474

Table 5: Information of the used data sets of group 1

Data set group 2	nr. of trolleys	nr. of transports	nr. of comm. types k	nr. of nodes	nr. of arcs
Data set 1	820	79	3	161	237
Data set 2	1348	199	6	404	597
Data set 3	24403	437	16	1352	2004
Data set 4-	44221	939	22	2860	4257
Original -	67556	2232	32	4524	6736

Table 6: Information of the used data sets of group 2

7.2 ILP model

In this section, the results that follow after solving the ILP program with integer linear solver Gurobi will be discussed. Table 7 shows the information obtained after running the five datasets. The headers of the columns are the number of trolleys, the objective obtained after solving the ILP problem, the runtime of the model in seconds and the transportation score, which means the percentage of right transported trolleys, respectively. A 100% transportation score implies that there could not be a better solution and the obtained objective gives the optimal solution.

ILP group 1	nr. of trolleys	objective	runtime (seconds)	transportation score (%)
Small ex. Figure 5	24	24	0.0156	100
Data set 1	410	410	0.178	100
Data set 2	670	670	27.1	100
Data set 3	12202	4570	120	37.5
Data set 4	22111	7611	120	34.4
Original	33778	16222	120	48.0

Table 7: Results after solving the ILP problem with Gurobi

ILP group 2	nr. of trolleys	objective	runtime (seconds)	transportation score (%)
Data set 1	820	820	0.25	100
Data set 2	1348	740	32.9	55.0
Data set 3	24403	8541	120	35.0
Data set 4	44221	12824	120	29.0
Original	67556	18667	120	27.6

Table 8: Results after solving the ILP problem with Gurobi

For a maximisation model, the lower bound is the objective of the best known feasible solution, while the upper bound gives a bound on the best possible objective.
From Table 7, one can easily see that only for the small example, data set 1 and 2 of group 1 optimal solutions have been obtained. The same holds for the data sets of group 2, see Table 8. This can be assumed because the objective is based on the number of successfully transported trolleys. When the number of total trolleys in the network is equal to the objective, an optimal solution is found. For data sets 3, 4 and the original instance, the objective of the best known feasible solution represents the lower bound to the problem. When looking at the last column, it becomes clear that the transportation score decreases as the data set increases. Concluding there could be assumed that solving the problem as an ILP problem with a linear solver such as Gurobi only works well for relatively small multi-commodity networks.

The reason why the CGDW algorithm has been developed, is to obtain an optimal solution in a more reasonable time for all of the data sets. The motivation behind this methodology is described in section 6.3. The next section will describe the results of using CGDW.

7.3 Column Generation with Dantzig-Wolfe

In this section, the results that follow after using the Column Generation method with a Dantzig-Wolfe reformulation will be discussed.
As said in 6.2, a column generation method works when dealing with an optimization problem consisting of many variables. The results that follow this methodology show that this statement is correct. The results obtained after running the CGDW can be found in Table 9

and Table 10 for group 1 and group 2 respectively.

The headers of the columns are the number of trolleys, the objective obtained after solving the optimization problem with CGDW, the runtime of the model in seconds, the number of iterations - which means the number of times one or more columns have been added to the RMP - and the transportation score respectively.

Column Generation with Dantzig-Wolfe group 1	nr. of trolleys	obj	runtime (s)	nr. of iterations CG	transportation score (%)
Small example Figure 5	24	24	0.141	1	100
Data set 1	410	410	0.178	100	
Data set 1	410	410	0.297	5	100
Data set 2	670	670	0.864	8	100
Data set 3	12202	12202	13.0	82	100
Data set 4	22111	22111	57.7	102	100
Original	33778	26544	120	stops after 104 iterations	78.6

Table 9: Results after solving the ILP with Column Generation and Dantzig-Wolfe

Column Generation with Dantzig-Wolfe group 2	nr. of trolleys	obj	runtime (s)	nr. of iterations CG	transportation score (%)
Data set 1	820	820	0.826	9	100
Data set 2	1348	1348	2.68	15	100
Data set 3 -	24403	24403	35.06	82	100
Data set 4 -	44221	21774	120	stops after 105 iterations	49.2
Original -	67556	29433	120	stops after 105 iterations	43.6

Table 10: Results after solving the ILP with Column Generation and Dantzig-Wolfe Christmas

Note that from these tables we see that the CGDW returns optimal solutions after reasonable computation times for datasets 1, 2, 3 and 4 of group 1 and data sets 1, 2 and 3 of group 2. For the other datasets lower bounds has been found.
For the original data set in group 1, a lower bound has been achieved with the CGDW, which gives a transportation score of 78%. It means that at least 78% is transported correctly. Looking at group 2 - due to the limit of two minutes on the runtime - a lower bound has been found for data set 4. Furthermore, the original data set of group 2 has a much lower transportation score than that of group 1 due to the doubled amount of trolleys and the one and a half times as large network. As expected before, transportation scores are worse, and runtimes are longer.

Another remarkable result that follows from the Tables 9 and 10 above is the fact that the objectives are integers. Because an LP-relaxation has been created from the ILP problem in order to be able to solve the ILP problem, integer solutions are not necessarily expected. The fact that integers have been found can potentially be seen as a coincidence. When the problem is solved as an ILP problem, the amount of zeros in the solution is ignored. This is caused by a commodity type not moving over every arc. For this reason, the CGDW is still a useful method. In this case, the CGDW is a better alternative to the common simplex method.

7.4 Comparison ILP and Column Generation

To check whether the ILP method with linear solver Gurobi or the CGDW method performs better when solving the MCNF problem, a few valuable results are put together for the data sets of group 1 and group 2 respectively. The original data set and the created data sets 1, 2, 3 and 4 are used.
When comparing the objectives and the corresponding runtimes found for the ILP with Gurobi and CGDW a few conclusions could be made.
Firstly, a comparison of the methods of group 1 will be handled. After that, the results of the methods of group 2 will be compared.

7.4.1 Comparison results group 1

Valuable results are put together for the data sets of group 1 in Table 11. The objectives that are colored orange imply optimal solutions. The underlined objectives represent lower bounds to the solution.

Comparison of the methods for data group 1	obj. **ILP**	runtime **ILP** (s)	obj. **CGDW**	runtime **CGDW** (s)	method with best performance
Data set 1	410	0.178	410	0.297	CGDW
Data set 2	670	27.1	670	0.864	CGDW
Data set 3	<u>4570</u>	120	12202	13.0	CGDW
Data set 4	<u>7611</u>	120	22111	57.7	CGDW
Original	<u>16222</u>	120	<u>26544</u>	120	CGDW

Table 11: Comparison of the two methods

It can be observed that for every data set the column generation method gives a better performance compared to the ILP method. Also, it can be stated that CGDW always performs well as the objectives found all give a 100% transportation score except for the original data set. When looking at the original data set, it can be seen that the lower bound has improved compared to the ILP method. The transportation score here has increased from 48% to 78.6%.

To get a better sense of how good the solutions found are, we will look at the so-called *optimality gap*. Therefore the optimal objectives, which equal the upper bounds, and lower bounds as found with ILP and CGDW are compared. When no optimal objective is found, we assume that the upper bound equals the number of trolleys in the data set. It follows that the optimality gap only takes values between 0 and 1. The optimality gap is determined as follows: $\frac{\text{lower bound}}{\text{upper bound}}$. The optimality gaps for group 1 are given in Tables 12 and **??**.

Group 1	ILP	CGDW
Data set 1	1	1
Data set 2	1	1
Data set 3	0.37	1
Data set 4	0.34	1
Original	0.48	0.79

Table 12: Optimality gaps for group 1

Looking at the results of Table 7, there can be concluded that CGDW performs better than the ILP. However, it is a quite complex mathematical method to use and therefore for small instances the ILP solving method works fine and results in good objectives. When using large data sets, high computation times and no optimal solutions should be expected. With these, the objectives can be seen as lower bounds to the solution. Lastly, by looking at the runtimes of the program, it can be seen that CGDW outperforms the linear solving method. This becomes clearer when a larger data set is used.

7.4.2 Comparison results group 2

In Table 13 objectives of both methods are set out. The orange colored objectives imply optimal solutions. In the case a lower bound to the solution is found this objective is underlined.

Comparison of the methods for data group 2	obj. **ILP**	runtime **ILP** (s)	obj. **CGDW**	runtime **CGDW** (s)	method with best performance
Data set 1	820	0.25	820	0.826	CGDW
Data set 2	740	31.9	1348	2.68	CGDW
Data set 3	8541	120	24403	35.6	CGDW
Data set 4	12824	120	21774	120	CGDW
Original	18667	120	29733	120	CGDW

Table 13: Comparison of the two methods

The methods for the datasets from group 2 yield less good results than for those from group 1. This is caused by the twice as many trolleys and 1.5 times as many truck movements contained in the original data set of group 2. As expected, objectives are worse, which imply lower transportation scores, and runtimes are larger.
As for group 1, the CGDW method gives a better performance compared to the ILP method. For data sets 1, 2 and 3 an optimal solution is obtained when using CGDW. The optimal solutions to data set 4 and the original set could not be obtained within the permitted runtime and therefore these objectives represent lower bounds. However, the lower bounds that follow from CGDW are much better that the bounds that follow from the ILP.

The optimality gaps for the data sets of group 2 are shown in Table 14. As mentioned before, when no optimal solution is obtained, the upper bound equals the number of trolleys in the corresponding data set.

Group 2	ILP	CGDW
Data set 1	1	1
Data set 2	0.55	1
Data set 3	0.35	1
Data set 4	0.29	0.49
Original	0.27	0.44

Table 14: Optimality gaps for group 2

7.5 Benchmark

In this section the transportation score of the current loading rules will be discussed.

According to the current loading rules, one day a percentage of 85 % trolleys are transported to the correct final destination. This means that an improvement of maximum 15% is possible. As said in chapter 1, the goal is to improve this percentage by at least 10%.

7.6 Minimization of costs flows by arc costs

The extension to the LP of constraints (8) - (11), as described in section 6.5, is solved with the second method. This method solves the problem with CGDW for the data sets of group 1. A comparison is made between the number of time units used in the network as described in equations (8) - (11) and the objective resulting from the modified problem (39) - (42). This objective represents the minimum number of time units required, assuming that all trolleys are transported to the correct destination depot. Comparing these two quantities provides insight into how many units of time in truck movements can be saved. This rate is called the *necessary time unit rate*. The results are shown in Table 15.

Adjusted LP data group 1	Runtime **CGDW** (s)	*Necessary time unit rate*
Small example	0.0211	0.96
Data set 1	0.308	0.91
Data set 2	1.23	0.88
Data set 3	17.5	0.86
Data set 4	61.3	0.83
Original	120	-

Table 15: Necessary time unit rate for adjusted LP using the CGDW solving method

As with the previous problem - the maximization of successfully transported trolleys-, the original data set has a lower bound as an objective after solving with CGDW. Table 9 shows the results of the previous LP problem. Therefore, no necessary time unit rate can be calculated for this instance. However, it can be cautiously assumed that, given the rates for the other data sets, a considerable number of time units can be saved. It can be stated that there exist unnecessary arcs in the network if the rate value is unequal to 1. This is the case for the problem considered in this report. The percentage of time units that can be saved in the original data set is probably between 10% and 20%.

8 Conclusion

The main goal of this thesis was to develop an algorithm that can be used as a scheduling tool for the truck loading problem of PostNL. Because on an average day, more than 1 million packages pass through the PostNL delivery network, it is important to know which package should be transported by which truck. This optimization case can be described as a Multi-Commodity Network Flow problem. In this MCNF problem trolleys can be seen as commodity-types that flow through a network in which arcs denote truck movements or the stock of a specific commodity-type and the nodes represent the departures and arrivals of these truck movements. Nodes are also provided with the inflow of trolleys. To obtain useful insights, the truck loading problem is solved in two ways. The first one is as an Integer Linear Programming problem. The second one is by considering its Linear Programming Relaxation with a Column Generation technique within a Dantzig-Wolfe decomposition. The results that follow from these two methods can be seen as extremely useful for PostNL. Not only for financial purposes but also for sustainable purposes.

When we compare the two methods with the PostNL's current loading rules, the following could be concluded. By modeling the algorithm, the measured success of transporting trolleys could be increased. In addition, the duration of the currently used transports can be reduced by 10% to 20%. Moreover running both of the methods is less time-consuming and is more likely to achieve optimal results than when done by manpower. In general, there can be said that no manpower beats the mathematical power of a good algorithm. Column Generation has lent itself perfectly for optimizing an MCNF because it returns an optimal solution in a reasonable time. Even for large data sets, an optimal solution can be obtained. However, this model requires more mathematical knowledge and could therefore be harder to use in general.

By looking at the use of arcs, a relevant adjustment has been made to the objective. By turning the problem into a minimization problem in which the costs of using an arc are minimized, it is possible to find out how many time units of truck movements can be saved. Using this new objective, it follows that for each data set, between 4% and 17% of time units can be saved. In practice, this means that a reasonable number of transports could be saved. This is explained by the fact that, when solving the MCNF problem, short routes are more attractive for trolleys to make use of and still fulfill the requested number of trolley assignments.

It can be stated that for all of the data sets, as for the original data, the CGDW algorithm lends itself perfectly to the maximization of solving the problem. Using this developed method, it can be concluded that every trolley will be transported successfully. Even though no results are displayed for the original data set, it can be argued that the use of this algorithm makes sense for PostNL. Not only because the algorithm results in accurate, optimal solutions and runs in a reasonable time, but it is also because the optimization of this truck loading case could be done automatically in the near future. The latter is a major advantage in the rapidly growing digitalization of companies. In addition, adjusting the objective has provided insight into the efficiency of the use of trucks for PostNL. It has become clear that costs can also be saved by looking at the inter transports. Of course, this also has a positive impact on the environment.

8.1 Future Research

Further research may concern environmental considerations. For example, the column generation method could also be used to see which transports are not being used. This equates to arcs that are not used in the network and this information could be obtained directly from the current algorithm. Another potential future research could be expanding the original mathematical problem by for instance adding the trolleys' due times.

Sustainability purposes

As already introduced above, the CGDW method used can also be used for sustainability purposes. PostNL is currently deploying a pre-calculated number of trucks to meet the inter transport processes. What already followed from the expansion of looking at the costs of using an arc, is that in any case fewer time units are needed for successfully completing the transportation of the trolleys. It is still interesting to see which truck movements can be omitted exactly. Therefore, more attention should be paid to the results that could be generated by the CGDW algorithm. From the Column Generation method, it is known that several truck movements do not contribute to the network and therefore do not have to drive. This saves money and is above all better for the environment. Take equations (8)-(11). By looking in detail at vector $\mathbf{x}^k$ unused arcs can be observed. The truck movements that correspond to these arcs can therefore be omitted from the transport planning. It should be clear that this is not only cost-effective, but also better for the environment.

Expansion of the mathematical problem

In this thesis for determining the route of a trolley within the inter transport network, only the final destination of the trolley was considered. That is, the commodity type k is defined as the specific destination depot of this trolley. In practice, we see that working with deadlines is important. PostNL's trolleys are divided into so-called 'shifts'. As already introduced in Chapter 2, a possible extension would be to include the shift in the commodity type. Then each commodity type k becomes a destination depot-shift combination. In the case that some trucks arrive later than certain shift times start, it is important that earlier shifts are given higher priority. Integrating this is not as easy as it seems. PostNL currently works with about ten shifts. In this case, ten new commodity types will replace each current commodity type. This makes the problem a lot bigger than it already is. The relevance of this also needs to be considered. It is possible that with the current truck movements, each trolley will arrive on time at the destination depot for the first shift that starts. In this case, adding the shift to the commodity type is not necessary.

References

Ahuja, Vindra Ba K, Thomas L Magnanti, and James B Orlin (1993). *NETWORK FLOWS Theory, Algorithms, and Applications*. Tech. rep.

Babonneau, Frédéric (2006). *Solving the multicommodity flow problem*. Tech. rep. Genève: Université de Genève.

Barnhart, Cynthia, Christopher Hane, Ellis Johnson, and Gabriele Sigismondi (1995). *A column generation and partitioning approach for multi-commodity flow problems*. Tech. rep., pp. 239–258.

Barnhart, Cynthia, Christopher Hane, and Pamela Vance (2000). *Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems*. Tech. rep.

Bauer, J., T. Bektaş, and T. G. Crainic (2010). “Minimizing greenhouse gas emissions in intermodal freight transport: An application to rail service design”. In: *Journal of the Operational Research Society*. Vol. 61. 3. Palgrave Macmillan Ltd., pp. 530–542. DOI: `10.1057/jors.2009.102`.

Bevrani, Bayan, Robert Burdett, Ashish Bhaskar, and Prasad K.D.V. Yarlagadda (Jan. 2020). “A multi-criteria multi-commodity flow model for analysing transportation networks”. In: *Operations Research Perspectives* 7. ISSN: 22147160. DOI: `10.1016/j.orp.2020.100159`.

Cordeau, Jean-Francois ; Francois ; Soumis, and Jacques Desrosiers (2001). *Simultaneous assignment of locomotives and cars to passenger trains*. Tech. rep.

Dai, Weibin, Jun Zhang, and Xiaoqian Sun (Aug. 2017). “On solving multi-commodity flow problems: An experimental evaluation”. In: *Chinese Journal of Aeronautics* 30.4, pp. 1481–1492. ISSN: 10009361. DOI: `10.1016/j.cja.2017.05.012`.

Fukasawa, Ricardo, Marcus Vinicius Poggi de Aragao, Oscar Porto, and Eduardo Uchoa (2002). “Solving the Freight Car Flow Problem to Optimality”. In: *Elsevier* 6.

Pfetsch, Marc, Ralf Borndörfer, and Christian Liebchen (Dec. 2006). *Multicommodity Flows and Column Generation*. Tech. rep. Technische Universität Berlin.

Rungwanichsu Kanon, Thanat and Manoj Lohatepanont (2015). “Multicommodity Flow Model for Multimodal Transportation Planning”. In: *Journal of the Eastern Asia Society for Transportation Studies* 11, pp. 788–798. ISSN: 1881-1124. DOI: `10.11175/easts.11.788`.

Salimifard, Khodakaram and Sara Bigharaz (2020). "The multicommodity network flow problem: state of the art classification, applications, and solution methods". In: *Operational Research*. ISSN: 18661505. DOI: 10.1007/s12351-020-00564-8.

Tomlin, J. A. (Feb. 1966). "Minimum-Cost Multicommodity Network Flows". In: *Operations Research* 14.1, pp. 45–51. ISSN: 0030-364X. DOI: 10.1287/opre.14.1.45.

Wang, I-Lin (2018). "Multicommodity Network Flows: A Survey, Part I: Applications and Formulations". In: *International Journal of Operations Research* 15.4, pp. 145–153. DOI: 10.6886/IJOR.201812{_}15(4).0001. URL: http://doi.org/10.6886/IJOR.201812_15.