Intermodal Transport

Routing Vehicles and Scheduling Containers

K. Kalicharan

Graduation thesis
MSc Applied Mathematics

Supervisors
Dr. D.C. Gijswijt
Dr. F. Phillipson
Ir. A. Sangers

TU Delft

Intermodal Transport

Routing Vehicles and Scheduling Containers

by

K. Kalicharan

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Tuesday May 14, 2018 at 14:00.

Student number:	4159233	
Project duration:	August 14, 2017 – May 14, 2018	
Thesis committee:	Prof. Dr. Ir. K. I. Aardal,	TU Delft
	Dr. D. C. Gijswijt,	TU Delft, supervisor
	Prof. Dr. F. H. J. Redig,	TU Delft
	Dr. F. Phillipson,	TNO, supervisor
	Ir. A. Sangers,	TNO, supervisor

Please print in color, if printing this report.

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Preface

I have had the opportunity to enhance my knowledge and skills and get to know amazing people in the course of my life. So I will take this moment to thank my parents and sister for their care, love and support and my other relatives and friends for their kindness and support.
I consider myself lucky to have met skilled and friendly people during my education and thesis project. Specifically, I would like to thank my supervisors Alex, Dion and Frank for their support and humor, Dick for the interesting conversations, Karen for the feedback and the interns of the TNO CSR department: Dylan, Max, Casper, Suzanne, Bor, Lianne, Gerbrich, Irina, Kim, Chaitali, Laurens, Jurriaan, Tim, Teresa, Finn and Ines for both the substantive and informal conversations. Furthermore, thank you to my primary school teacher Ida, my secondary school math teacher Afke, Frank from my former job at SSL, TNO and the TU Delft for making all of this possible.

K. Kalicharan
Delft, May 2018

Abstract

In intermodal transport multiple types of vehicles are used to transport containers. If the routes of the vehicles are known, then the container allocation can be optimized. This problem can be modelled as an integral multi-commodity min cost flow problem on a time-space graph. This model has an arc-based and path-based form. In this thesis, the path-based form is derived from the arc-based form. Some of the methods that can be used to solve this model are column generation, Lagrangian relaxation and the repeated cheapest path heuristic.

If the routes of the vehicles are not known, then we also need to create routes for the vehicles. The problem of routing vehicles and scheduling containers can be modelled as a multi-commodity network design problem on a time-space graph. Variable reductions, cutting planes and other additional constraints are looked into to make the problem easier to solve. Additions to the model are researched that can make the model more suitable for use in practice. Additionally, ILP based solution methods are developed. Finally, some of these reductions, extensions and solution methods are implemented and reviewed.

Contents

1

Integer Linear Programming

This chapter is meant to give the reader some intuition about integer linear programming. For formal proofs, theorems and definitions, we refer the reader to the many books such as [19] about (integer) linear programming that have been written.
Let us start by looking at the 2-dimensional plane ($\mathbb{R}^2$). Every point in the plane we represent with Carthesian coordinates; (x_1, x_2). Let $a_{1,1}, a_{2,1}, b_1$ be fixed real numbers and x_1, x_2 be real variables. Then the linear equation:

$$a_{1,1}x_1 + a_{2,1}x_2 = b_1$$

represents a line that splits the plane. Every point that is at one side of the line or on the line can be represented with the linear inequality

$$a_{1,1}x_1 + a_{2,1}x_2 \leq b_1.$$

Only a subset of the Carthesian coordinates satisfy the inequality. Therefore, we call such an inequality a constraint. More constraints of this form for other $a_{1,j}, a_{2,j}, b_j \in \mathbb{R}$ for $j \in Z_m := [1, m] \cap \mathbb{Z} = \{1, 2, ..., m\}$ where m is the number of constraints, can be added. We use the constraints to define a linear program (LP):

$$\max f(x_1, x_2) \tag{1.1}$$

subject to

$$a_{1,j}x_1 + a_{2,j}x_2 \leq b_j \qquad \forall j \in Z_m \tag{1.2}$$
$$x_1 \geq 0 \tag{1.3}$$
$$x_2 \geq 0, \tag{1.4}$$

where $f(x_1, x_2) = c_1x_1 + c_2x_2$ is called the objective function of the LP. Below the constraints we already introduced we see two more inequalities (1.3),(1.4). These are called nonnegativity constraints. These constraints prevent the variables x_1, x_2 from becoming negative. Every point that satisfies all the constraints that are added, is called *feasible*. Together all these feasible points make up the *feasible region*:

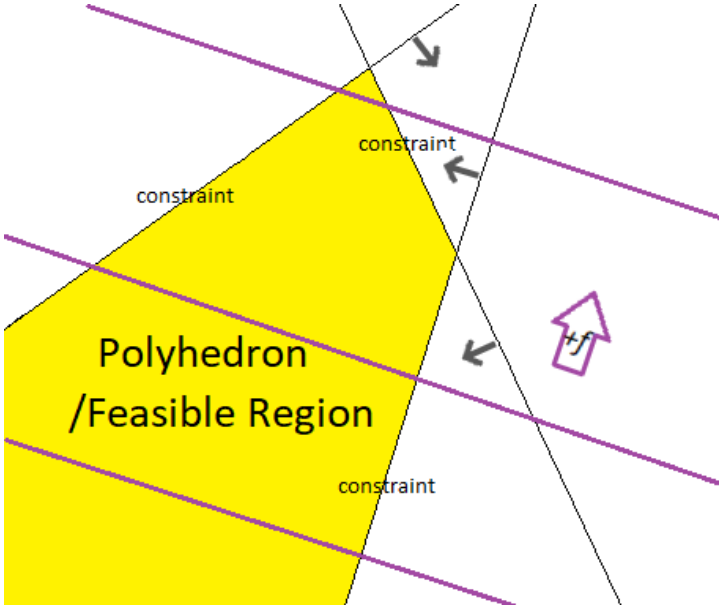
$$\{(x_1, x_2) | a_{1,j}x_1 + a_{2,j}x_2 \leq b_j \forall j \in Z_m, x_1 \geq 0, x_2 \geq 0\}.$$

The feasible region of an LP is a *polyhedron*. An example of such a polyhedron can be seen in Figure 1.1. Note that between each pair of points in the polyhedron a line segment can be drawn that lies completely in the polyhedron. Sets for which this property holds, such as polyhedra are called *convex sets*. An example of a non-convex set can be found in Figure 1.2.

If the feasible region can be contained in a circle (or higher dimensional circle), then the polyhedron is bounded and we call it a *polytope*.

Not all the points/solutions in the feasible region are equally desirable. The linear objective function f maps every coordinate to a value. Points with a higher function value are considered superior, because we are maximizing the objective function value (1.1). A solution to the LP has to be in the feasible region; so satisfy

Figure 1.1: Polyhedron



The black lines denote the (linear) constraints. A gray arrow depicts which side of the line is allowed by the constraint. The yellow area shown is a part of the feasible region. The purple lines are the equations $f(x_1, x_2) = y$ for several values of y. The purple arrow shows in which direction f increases.

(1.2),(1.3),(1.4). Furthermore, we want to maximize the objective function (1.1). An optimal solution to the LP has the highest objective function value of all the solutions in the feasible region. Graphically, we see the $f(x_1, x_2) = y$ for some values of y drawn in Figure 1.1. An optimal solution solution is found by lifting one of the purple lines in the direction of the purple arrow (that way the objective function value increases), such that it still intersects with the feasible region. We see that if we do that we get to the upper most intersection of two constraints. The corresponding coordinates represent the optimal solution of the LP. Furthermore, there are no two points in the feasible region (other than this point), between which we can draw a line section such that this point is on the line section. In other words, the point is no convex combination of two other points in the feasible region. Such points are called *vertices*. Solution methods for LPs such as the simplex method try to find a vertex that corresponds to an optimal solution. If the purple arrow would be in the exact opposite direction, then we could lift one of the purple lines indefinitely. In that case the LP would be unbounded. Then for every feasible solution, it is possible to find a feasible solution with a higher objective function value. One other thing that may occur is that the feasible region is empty, such LPs we call infeasible.

Figure 1.2: Non-Convex Set

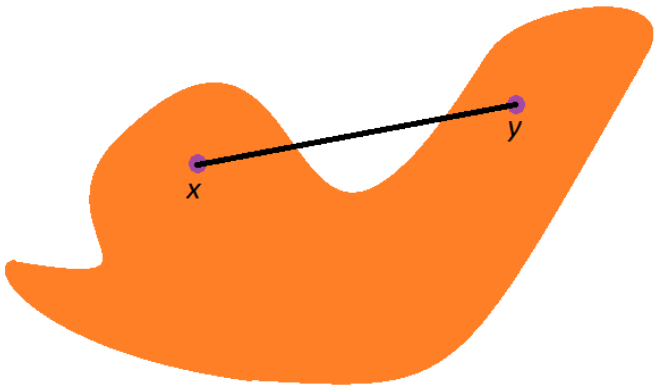
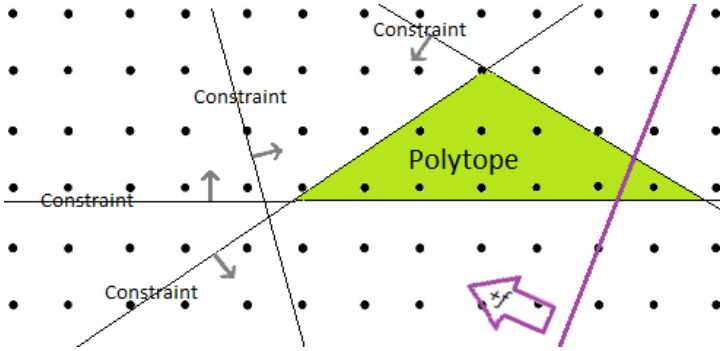


Figure 1.3: Polytope



The black lines denote the (linear) constraints. A gray arrow depicts which side of the line is allowed by the constraint. The yellow area shown is a part of the feasible region. The purple line is an equation $f(x_1, x_2) = y$ for a value of y. The purple arrow shows in which direction f increases.

If the feasible region is a (non-empty) polytope, then there is an optimal solution.

If the variables are required to be integral, then we add the constraints $x_1 \in \mathbb{Z}$ and $x_2 \in \mathbb{Z}$ to the problem. The resulting problem is an integer linear program (ILP). The LP we obtain from removing the integrality constraints is the corresponding LP relaxation. The black points in the green polytope in Figure 1.3 are the feasible solutions of the corresponding ILP.

This theory can be extended to higher dimensions to obtain a general definition of a standard form ILP. Let $n \in \mathbb{Z}_{>0}$. An ILP is defined as:

$$\max \quad \sum_{i=1}^{n} c_i x_i \tag{1.5}$$

subject to

$$\sum_{i=1}^{n} a_{i,j} x_i \leq b_j \qquad \forall j \in Z_m \tag{1.6}$$

$$x_i \geq 0 \qquad i \in Z_n \tag{1.7}$$

$$x_i \in \mathbb{Z} \qquad i \in Z_n. \tag{1.8}$$

An ILP can also be represented in matrix form:

$$\max \quad \mathbf{c}^T \mathbf{x} \tag{1.9}$$

subject to

$$A\mathbf{x} \leq b \tag{1.10}$$

$$\mathbf{x} \geq 0 \tag{1.11}$$

$$\mathbf{x} \in \mathbb{Z}^n. \tag{1.12}$$

An (I)LP can also have equality constraints. Equality constraints can namely be generated by inequality constraints:

$$\left.\begin{array}{l} \sum_{i=1}^{n} a_{i,j} x_i \leq b_j \\ -\sum_{i=1}^{n} a_{i,j} x_i \leq -b_j \end{array}\right\} \Rightarrow \sum_{i=1}^{n} a_{i,j} x_i = b_j.$$

It is also allowed to minimize instead of maximize, because:

$$\min \mathbf{c}^T \mathbf{x} \equiv -\max -\mathbf{c}^T \mathbf{x}.$$

2

Introduction

Each day numerous products are transported in containers. The transportation cost is a significant part of the cost of such a product [21]. Therefore developing algorithms and heuristics to transport them in a cost-friendly and robust way is of dire importance. Problems involving only trucks can be modelled with versions of the vehicle routing problem (VRP) [56]. Different solution methods [82] exist for the NP-hard [68] VRP [18]. For the transportation of containers vehicles belonging to different *modes*[1], such as trains and barges can also be used. Container transport with multiple modes is normally called *intermodal transport*. Even routing problems with customers that are not adjacent to a waterway are considered in this field, because the containers of such a customer might still be transported by barge on a part of its route. Problems in this field can become complex, because they consider different vehicle types. These vehicle types can namely have different infrastructure networks, travel times, capacities and restrictions.

Intermodal transport allows for many different options of container transportation. A container can be trucked from its origin to its destination, but (part of) the route can also be done by barge, plane or train. A logistics service provider (LSP) [45] is responsible for managing the flow of containers from their origins to their destinations. Especially in large logistics networks LSPs may require assistance from algorithms to make good routing choices. There algorithms can be an tool for helping reach economic and emission-reduction targets by offering decision support for optimizing the intermodal transport chain [49]. Additionally, the container transport is also less reliant on one type of mode. If there is a disturbance on the rail one could opt for trucking the container. If routes or schedules are changed because of current events such as disturbances, then we are entering the field of synchromodal transport. In synchromodal transport all sorts of real-time information, such as the traffic congestion, is used to improve the routes and schedules. Furthermore, in synchromodal transport information and resources of transporters are shared to allow a logistics service provider to find a good way to fulfill all transport requests. Definitions may vary somewhat depending on the source [45].

In this thesis we will look at inland intermodal transport, but some techniques can also be used for certain instances of long distance transport. With inland intermodal transport we mean the transportation of containers by rail, road and river/canal. The goal of this project is finding a good way to simultaneously route vehicles and transport containers. Firstly, a time-space graph is constructed to take into account the points in time that need to be in the model and the terminals of the problem. Then an integer linear program (ILP) is defined. This represents the constraints and objectives of the optimization problem. Two optimization problems of those defined by my predecessors [25, 45, 70] within the project are looked into (see Table 2.1). In Problem **1** (Table 2.1), we assume the routes of the vehicles are known. We are in this case only optimizing the schedules of the containers. In Problem **3** (Figure 2.1), we are addressing the main goal of the project: simultaneously optimizing the routes of the vehicles and the schedules of the containers. We will only look into uncertainty Problem 2/4 very briefly. The ILP we define for Problem **1** is the integral multi-commodity minimum-cost flow (MCMCF) problem [37, 45, 70]. The main contribution of this project is the integral multi-commodity network design (MCND) problem applied to Problem **3** with its extensions, reductions and

[1] A *mode* (of transport) is a type of transportation. Each mode has a different infrastructure, technology and vehicles. Examples of modes are air, water, rail and road.

Table 2.1: Problems in synchromodal transport

	deterministic	uncertain
container scheduling	**1**	**2**
container scheduling and vehicle routing	**3**	**4**

solution techniques. This ILP can be seen as a generalization of the integral MCMCF problem and a variation of the design-balanced service network design problem/design-balanced multi-commodity capacitated fixed charge network design problem [17, 61]. Finally, methods from the field of optimization to solve the ILPs are looked into: branch & cut, heuristics, column generation and other. Besides branch & cut more solution methods for the integral MCND problem are implemented that are inspired by the α cut-and-fix heuristic of [17]. Different features are incorporated in the models that can be useful in practice. Where possible the structure of the problems is used to reduce the computation time of the algorithms and heuristics. Specifically, different ways to reduce the number of variables and to add cutting planes are considered. Many of these techniques presented in this thesis to reduce the running time are especially valuable considering that some of the solution methods presented here may be used as a subroutine in models with uncertain parameters from synchromodal transport.

This work has been carried out within the project 'Complexity Methods for Predictive Synchromodality' (Comet-PS), supported by NWO (the Netherlands Organisation for Scientific Research), TKI-Dinalog (Top Consortium Knowledge and Innovation) and the Early Research Program 'Grip on Complexity' of TNO (The Netherlands Organisation for Applied Scientific Research).

3

Literature Overview

The factors that can be optimized in freight transport can belong to one of the different levels defined in literature [40]. The *strategic* level contains elements of the logistics network with a long-lasting effect. One can think of the locations of the terminals, number of barges and trains that are available and their sizes/capacities. These are called long term decisions. The *tactical* level is about medium term decisions, such as the layout of resources at a terminal. The *operational* level comprises the day-to-day decisions. In this category we have the scheduling of the containers and the rescheduling in case of disturbances [76]. The routing of the vehicles is located around the borderline of the tactical and operational level. Some weeks in advance a (rough) planning can be made for the vehicles. This planning can be further optimized once the bookings for the day are known. Some companies may only be interested in a planning that works reasonably well in general, while other companies might want to let their vehicle routing depend on the bookings of the day. In logistics, research has been done on all three levels [67]. Elements from different levels are included in the integral MCND problem (incl. additional features) from this thesis.

There are loads of interesting problems and/or methods related to intermodal transport. This thesis will focus more on intermodal inland container transport. Closely related research includes network design models for intermodal transport [4, 83], an intermodal/synchromodal model with compatibility clustering heuristic and expected value iteration [45], a continuous time routing model with a matheuristic solution technique [79], a routing model including maritime and road transport with Lagrangian relaxation based solution techniques [6] and more related research can be found in [23].

Other related papers and theses have a different focus:

International transport
In [16] an international container transport model is researched. Here also transportation by sea is considered. The structure and size of the networks of those instances, the chosen objectives and the structure of the problem are used to create a model. The label-setting algorithm from dynamic programming is used on their weighted constrained shortest path problem model. Dynamic programming can be used, because the problem satisfies the overlapping subproblems property [31] and the optimal substructure property [30, 32].

Ship routing
In the thesis [69] an approximate dynamic programming approach is implemented on a Markov decision process model, which can be used for small-scale barge planning problems with uncertainty. In [54] we can find a ferry routing problem including a two phase heuristic.

Empty container repositioning
In [80] the empty container repositioning problem is investigated. This problem is interesting, because a reduction of empty container movements has economic and environmental benefits. In [74] an inland container transport model for trucking is researched. It is a path-based model that is extended by adding constraints to take into account the distribution of empty containers among the terminals. Column generation is used to solve it.

Shared information routing
In [25] the focus is shared information routing and contains an extensive literature research as well. Heuristics and an iterative method are used to model among others the traffic on the roads.

Aircraft routing
An air network design problem is formulated and solved with column generation in [10].

Berthing allocation
Terminal operators [40] are responsible for many things within the terminal, such as allocating a berth position and berthing time interval. Solving this tactical problem is especially difficult considering in practice vessels may arrive at a different time or other unexpected events may occur. We refer to [39] for more information about the berth allocation problem.

Train routing
A train routing model can be found in [90]. A matheuristic that uses slope scaling and ellipsoidal search is used to solve the problem.

Looking at a specific vehicle type (train/aircraft/ship) or network (international) can be useful, because in these more specific models certain vehicle type/network attributes can be taken into account when building the model. Shared information routing and empty container repositioning are problems with objectives different from Problem **3**. The berthing allocation problem is a completely different problem within intermodal transport.

4

Problem Description

As mentioned in the introduction the main goal of this thesis is solving Problem **3**: finding a good way to simultaneously route vehicles and transport containers. We will start by looking at the easier Problem **1**: If the routes of the vehicles are known, how can we optimally allocate the containers to the vehicles?

In order to solve both problems, we require the following input: We have an LSP (logistics service provider). The LSP has different bookings, that all have an origin location, a destination location, a release time and a deadline. A booking consists of one or more containers. Every container in the booking has the same characteristics: start location, end location, release time, deadline. We call the number of containers in a booking, the demand of the booking. All the containers have the same size. The only available modes are water and road. More modes and vehicles can be added to the model. We have some barges and many trucks. Trucking is in general more expensive but quicker, than shipping [87].

We assume:

- It is always possible to truck a container from its origin to its destination before its deadline.
- There are always sufficiently many trucks and chassis available at every terminal (so the trucks are an infinite source).

Furthermore, not all locations are connected to waterways where barges can arrive. So only a subset of the locations is accessible by barge. Every barge has a route for the time span of the model that is known in advance in Problem **1**. In Problem **3** the routes of the vehicles are part of the optimization. There is a known finite number of barges available. The barges have a capacity denoting the number of containers they can carry at once, while the trucks can always only carry one container at a time. A container can be transported by different vehicle(s) (types) at different parts of the route.

It is also necessary to define what is meant by *optimal*. Unless stated otherwise, we try to minimize the total cost of all the elements that we assigned cost to in the models. This is normally only a cost per hour that a container is trucked. Other options that can be added to the models are minimizing the *robustness*, *flexibility* and other attributes. More information about including different objectives in the models can be found in previous work within this project [70]. A robust solution can withstand a bit of delay or other small disturbances. A solution is called flexible, if it is possible to effectively reroute/reschedule if unfavorable events (such as traffic congestion) occur. So especially this last concept is relevant for synchromodal transport.

5

Time-Space Graph

To build a model for intermodal transport, we need a way to model time. In literature [79] we find that models in intermodal transport have two ways to model time: continuous time or discrete time. We choose to look at a discrete time approach, because these models normally require fewer **types** of constraints and **types** of variables. Though this does not necessarily mean discrete time approaches always have less variables and/or constraints and/or are easier to solve. For modeling the discrete time we use a *time-space graph*.

A multigraph is a triple $G := (V, E, Q)$, where V, Q are sets and $E \subseteq V \times V \times Q$. The set V is the set of nodes. The set E is the set of edges (we consider directed edges also known as arcs unless stated differently) and Q is used to differentiate arcs with the same begin and end node. The elements $e \in E$ are written in square bracket notation; $e := [v_0, v_1, q_0]$ for certain $v_0, v_1 \in V, q_0 \in Q$.

A multigraph $G = (V, E, Q)$ is called *finite* if and only if $|V|, |Q| < \infty$. This implies $|E| \leq |V \times V \times Q| = |V|^2 |Q| < \infty$.

Let X, Q be finite sets. The set X represents the locations. Let $T \subset \mathbb{Z}_{\geq 0}$ with $|T| < \infty$ be a set of time stamps. Let $V := T \times X$ be a set of nodes. Let $E \subseteq \{[(t_0, x_0), (t_1, x_1), q] | (t_0, x_0), (t_1, x_1) \in V$ such that $t_0 < t_1, q \in Q\}$ be the set of (directed) arcs, we use square brackets for the (directed) arcs for clarity. Then the multigraph $G := (V, E, Q)$ is a *time-space graph*.

We will now start creating time-space graphs for problems in intermodal transport. If we consider a container and try to minimize the cost to get it to its destination, then we need to build the structure of the network first. The way we will build a time-space graph is by discretizing time; we can for example take a time step of one hour. Then the set $T = \{0, 1, 2, 3, 4, 5, 6, \ldots, n\}$ for a $n \in \mathbb{Z}_{\geq 0}$ is a set of time stamps. If we take $n = 20$, then we can model 20 hours for example from 00:00 to 20:00. Let $t \in T$, then 00:00 belongs to $t = 0$, 01:00 to $t = 1$ etc. The set X is the set of (terminal) locations. It is known in advance which vehicle types can be handled at a certain terminal. If some terminals are located around the same location, then they can be put together in one $x \in X$. This is useful if there are many locations and/or the time steps are too large to model them accurately separately. The set $T \times X$ is the set of nodes of the time-space graph V. The nodes $v = (t, x) \in T \times X$, are time-space couples. With the notation $\tau(v)$ and $\chi(v)$ we mean the corresponding time t and location x of the time-space node v respectively. The paths of the barges are decomposed in arcs/links and put into the time-space graph. For example if barge 0 goes from location 3 at 00:00 to location 1 where it arrives at 10:00 and then goes to location 5 where it arrives at 20:00. The path is $(0,3) \rightarrow (10,1) \rightarrow (20,5)$. This is decomposed in arcs $[(0,3), (10,1), \text{barge0}]$ and $[(10,1), (20,5), \text{barge0}]$ and these arcs are put in the time-space graph with capacity equal to the capacity of that barge. For the next barge we do the same. When we are done with the arcs for every barge, we look at the trucks. We assume that we have enough trucks. We add for every time stamp $t \in T$ for every location $x \in X_{\text{truck}} := \{\text{locations accessible by truck}\} \subseteq X$, truck arcs $[(t, x), (s, y), \text{truck}]$ for all $y \in X_{\text{truck}}$, with if $s \in T$, such that $s - t$ is the travel time from x to y. If $x = y$, then we take $s - t = 1$. These horizontal arcs are sometimes referred to as waiting arcs. Note that between some time-space nodes we will have multiple arcs for example a truck and a barge 0 arc. To make a distinction between those arcs the set of vehicles W is used. All trucks correspond with the element truck $\in W$. All other vehicles each have their own element in $W = \{\text{truck}, \text{barge0}, \text{barge1}, \text{etc.}\}$. The set E is the set of all arcs in the graph. All arcs go forward in time so by definition it is a time-space graph. With the notation E_w we mean the set of all arcs in the graph

belonging to vehicle (type) w and X_w is the set of all locations accessible by vehicle (type) w. The next lemma and theorem show some interesting properties of time-space graphs.

Lemma 1. *Let $G = (V, E, Q)$ be a finite directed multigraph without directed cycles. Then $\exists v_0 \in V$ with $\nexists v_1 \in V, q_0 \in Q$ such that $[v_1, v_0, q] \in E$.*

Proof. Suppose the statement is false. (†)

Then we recursively construct a path P.

1. Take $v_0 \in V, i = 0$ and define $P := (v_0)$.
2. Then take $v_{i+1} \in V, q_i \in Q$, such that $[v_{i+1}, v_i, q_i] \in E$. (†) implies that this is possible.
3. If $v_{i+1} \in P$, then $v_{i+1} + [v_{i+1}, v_i, q_i] + P$ is a directed cycle and we have a contradiction.
   If $v_{i+1} \notin P$, then we update $P := v_{i+1} + [v_{i+1}, v_i, q_i] + P, i := i + 1$ and go back to the previous step.

There are a finite number of nodes in the graph, thus the algorithm finds a directed cycle in a finite number of iterations. □

Theorem 1. *Let G=(V,E,Q) be a time-space graph, then G is a finite directed multigraph without directed cycles. Let G=(V,E,Q) be a finite directed multigraph without directed cycles, then it is isomorphic to a time-space graph.*

Proof. First part: By definition G is a directed multigraph. The sets Q, T, X are finite by definition a time-space graph. So $V := T \times X$ is finite as well. It remains to be shown that G has no directed cycles.

Suppose the G has a directed cycle. Take such a directed cycle H. Take an arc $[(t_i, x_i), (s_i, y_i), w_i]$ in that directed cycle with t_i minimal. The arc is in a directed cycle so there is an arc with $(s_j, y_j) = (t_i, x_i)$ in that directed cycle. By definition of time-space graph it follows $t_j < s_j < t_i$. This is in contradiction with the minimality of t_i. So G has no directed cycles.

Second part: Let $G = (V, E, Q)$ be a directed multigraph without directed cycles. Let $X := \{0\}, T := \{i : 0 \leq i \leq |V| - 1\}, U := \{T \times X\}$, $W := Q$.

Let $f : V \to U$ be bijective function defined in the following steps:

1. Let $V_0 := \emptyset, i = 0, V' := V$.
2. Take a $v_1 \in V'$ with $\nexists v_0 \in V', q \in Q$ with $[v_0, v_1, q] \in E$. (Lemma 1)
   We set $f(v_1) = (i, x), V' := V' \setminus \{v_1\}$ and $i = i + 1$.
3. If $V' = \emptyset$ go to the next step otherwise repeat the previous step.
4. Define $F := \{[f(v_0), f(v_1), q] | [v_0, v_1, q] \in E\}$.

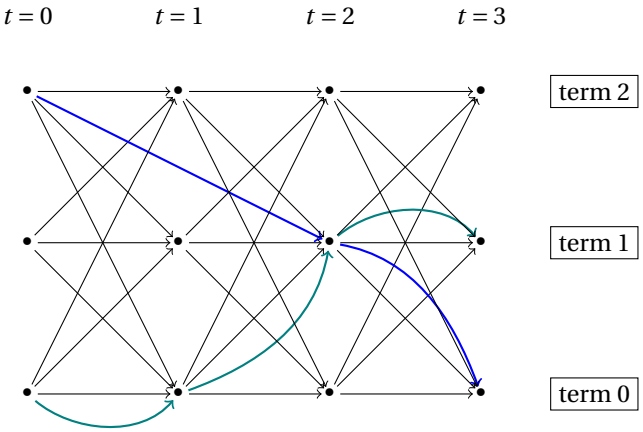
We have $|V|, |Q| < \infty$, so the algorithm will end in a finite number of steps.
We have $f(v) \in U \forall v \in V$ and $q \in W \forall q \in Q$, so $H = (U, F, W)$ is a multigraph.
For all $[f(v_0), f(v_1), q] = [(t_0, x_0), (t_1, x_1), q] \in F$, we have that $[v_0, v_1, q] \in E$. Consequently, v_0 was removed sooner from V', than v_1 and $t_0 < t_1$.

So it follows that G is a time-space graph.

□

An example of a time-space graph can be seen in Figure 5.1. There the blue arcs correspond with barge 0, the teal arcs belong to barge 1 and the black arcs to the trucks.

Figure 5.1: A time-space graph for MCMCF



6

Minimum Cost Flow Problem

Problem **1** can be modelled as an integral multi-commodity minimum cost flow problem on the deterministic time-space graph. The multi-commodity minimum cost flow (MCMCF) problem is a generalization of the (single commodity) min(imum) cost flow problem. This chapter is intended to get the reader familiar with flow problems, to explain how they can be applied to problems from practice. Moreover, minimum cost flow problems are solved with some solution methods for the integral multi-commodity minimum cost flow problem and both problems have many interconnections.
In the min cost flow problem we have a directed graph of which several nodes are sinks and some of the other nodes are sources. To further sketch the idea we look at an application; an oil network. We have a directed graph that represents the oil network. At some nodes in the graph, locations, the oil enters the network. For example locations where oil is produced. The oil needs to be transported to some destination nodes. They are called sources and sinks respectively. We know the amount of oil that enters the network via the sources and the amount of oil that has to arrive at the sinks. Intuitively, exactly the total amount of oil that enters the network at the sources has to leave the network via the sinks. Suppose different routes have prices depending on the amount of oil flowing through the route. The problem can then be modelled as min cost flow problem. Let the nodes in the graph $G = (V, E)$ be the crossroads, sources and sinks in this oil network. Let these nodes be connected with arcs, the links through which these nodes are connected in reality. If node v is a source, then the net amount of flow that leaves this source we call d_v, by definition of a source $d_v > 0$. Meaning more oil leaves, than enters the node. If node v is a sink, then the net amount of flow that leaves this sink is $d_v < 0$. Note that it is negative because the amount of oil that arrives at the sink is larger than the amount of oil that leaves it. If a node v in neither a source nor a sink, then the net flow that leaves the node is $d_v = 0$. Assume $(v, w) \in E$. Then $x_{v,w}$ is the amount of flow that leaves v and enters w via arc (v, w). Before we go on we need some notation. Let directed graph $G := (V, E)$ have a set of nodes V and a set of arcs $E \subseteq V^2$. In this thesis an arc is always directed and an edge is always undirected. We define $n := |V|$ and $m := |E|$. For a node $v \in V$ we define $\delta^-(v) := \{e \in E | e := (w, v) \text{ for a } w \in V\}$ and $\delta^+(v) := \{e \in E | e := (v, w) \text{ for a } w \in V\}$. We define $N^+(v) := \{w \in V | (v, w) \in E\}$ and $N^-(v) := \{w \in V | (w, v) \in E\}$. With the help of this notation we introduce the constraints

$$\sum_{w \in \delta^+(v)} x_{v,w} - \sum_{w \in \delta^-(v)} x_{w,v} = d_v \forall v \in V.$$

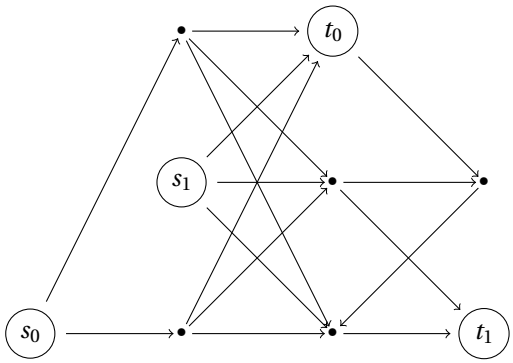
They model the conservation of the oil flow and the excess for sources and the slack for the sinks. Another important type of constraint is the non-negativity constraints

$$x_{v,w} \geq 0 \qquad \forall (v, w) \in E.$$

These constraints are added because the oil flow can never be negative. We also have capacity constraints, for every arc (v, w) in the model we have a certain capacity $c_{v,w}$. This is the capacity of the vehicle or pipeline that the arc belongs to. This gives rise to the constraints

$$x_{v,w} \leq c_{v,w} \forall (v, w) \in E.$$

Figure 6.1: Min cost flow on oil network



Lastly, an objective function is added for every arc. We multiply the flow on the arc with a cost per flow unit $f_{v,w}x_{v,w}$ and add it to the objective function. The whole problem becomes

$$\min \sum_{e \in E} f_e x_e \tag{6.1}$$

subject to

$$\sum_{e \in \delta^+(v)} x_e - \sum_{e \in \delta^-(v)} x_e = d_v \qquad \forall v \in V \tag{6.2}$$

$$x_e \leq c_e \qquad \forall e \in E \tag{6.3}$$

$$x_e \geq 0 \qquad \forall e \in E. \tag{6.4}$$

In Figure 6.1 we see a possible oil network. The s_i are sources and the t_i are sinks. It is irrelevant if oil from source s_0 or s_1 is transported to sink t_1. Every solution to the problem contains the routes from the sinks to the sources and the amount of oil that needs to be transported on those routes has to satisfy the demand.

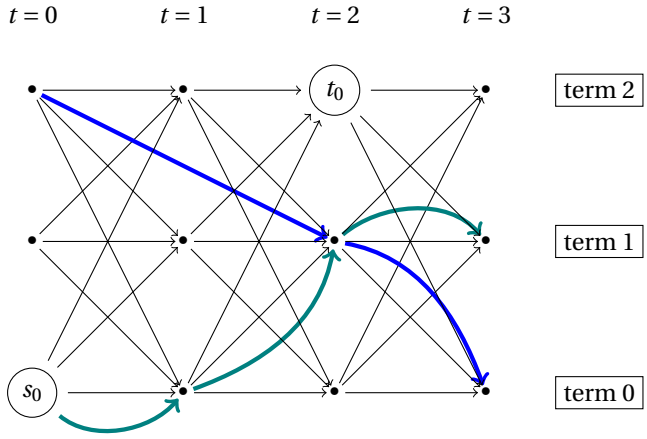
Many different algorithms for the min cost flow problem exist. One option is to solve the linear programming (LP) problem with one of the LP solution methods. There are many different LP solvers available that can use general techniques such as the simplex algorithm to solve it. In this case, the structure of the min cost flow problem can be utilized to solve the problem faster with the network simplex algorithm [51]. Although many methods with better theoretical worst case computation time bounds exist [78], the network simplex algorithm has very low computation time in practice [20, 52]. Some other methods [78] include cycle cancelling, successive shortest path and capacity scaling.

In case of an undirected graph it is also possible to solve the problem, by replacing each edge in the undirected graph with arcs in both directions [77]. Without loss of generality, one may assume for the min cost flow problem that there is only one sink and one source. If we have a graph with multiple sources and/or sinks, we can reduce it to such a problem by adding two additional nodes to the graph a super source s and a super sink t. Initially we set $d_s := 0$. Super source s is connected with every original source s_i with arcs (s, s_i). Then we set $c_{s,s_i} := d_{s_i}$ and $d_s := d_s + d_{s_i}$. After which we set $d_{s_i} := 0$. So in the model s_i is not a source anymore. This way the net flow from v (not counting the flow through the arc (s, s_i)) is still d_{s_i}. This process is repeated for every source. A similar tactic can be used for the sinks.

It is also possible to allow multiple arcs between two nodes if we add an index to the variables, so we get $x_{v,w,k}$ where k determines which arc between v and w is meant. The time-space graph we saw before has multiple arcs between some pairs of nodes. In case of oil, the variables can be fractional, but in other cases the variables may have to be integers. An example is a network (without directed cycles) of terminal locations in which a container has to be transported from a time-space node s_0 we call the source, to another time-space node t_0 the sink. We get $d_{s_0} = 1$ and $d_{t_0} = -1$. The variables are one if the corresponding arc is used in the route of the container and zero otherwise. So in this case the variables are not solely integral, they are even binary. Integrality conditions have to be added to the problem formulation

$$x_e \in \mathbb{Z} \qquad \forall e \in E, \tag{6.5}$$

Figure 6.2: One container min cost flow on time-space graph



then we get an integer linear program (ILP).

We are working over the time-space graph in Figure 5.1 and added a source and sink node for the container. The resulting network can be seen in Figure 6.2.

The ILP (6.1)-(6.5) can also be written in matrix form with the help of the $n \times m$ node-arc incidence matrix B of the graph.

$$\min \mathbf{f}^T \mathbf{x} \tag{6.6}$$

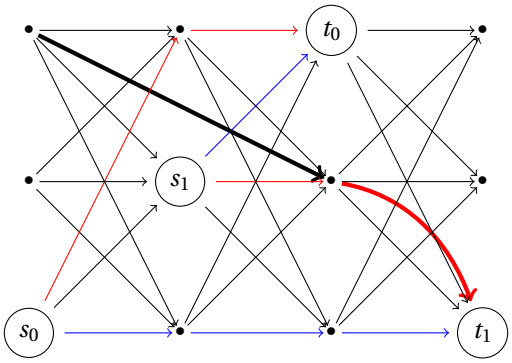
subject to

$$B\mathbf{x} = \mathbf{d} \tag{6.7}$$
$$\mathbf{0} \leq \mathbf{x} \leq \mathbf{c} \tag{6.8}$$
$$\mathbf{x} \in \mathbb{Z}^m. \tag{6.9}$$

A matrix A is called *total unimodular* if the determinant of each square submatrix of A is $-1, 0$ or 1 [15].
Theorem 2. *If all entries of a matrix are elements of the set* $\{-1, 0, 1\}$ *and at most one* -1 *and one* 1 *in each column or in each row, then the matrix is total unimodular (TUM) [35].*

Matrix B is TUM, because it has exactly one 1 and one -1 in each column and the other elements are zeros (Theorem 2). Also the right hand side in matrix form is integral, because in the ILP we take integer demands and capacities. So now by a theorem in [84] we can conclude that the feasible region of the ILP only has integral vertices. That means that if we solve the LP relaxation of the ILP with methods like the simplex

Figure 6.3: Multiple containers example



algorithm, then we get an integral solution! An LP relaxation of an ILP is the LP you get from the ILP by removing the integrality constraints. With solution we do not mean the optimal objective function value, but the values of the variables in an optimum. This all means that we can use LP solving techniques to solve the ILP. Additionally because LPs can be solved in polynomial time, this ILP can also be solved in polynomial time.

Unfortunately, the integral min cost flow problem cannot be used for all problems involving multiple containers. Suppose we have two containers, one needs to be transported from time-space node s_0 to t_0 and the other one from s_1 to t_1. The instance is visualized in Figure 6.3. There the bold arcs are the arcs of a barge and the others are truck arcs. The blue arcs are one in the optimal solution of the ILP. The red arcs are one in the actual optimal solution of the problem. The reason the min cost flow formulation is not suitable for this problem with two containers is the following: We have two types of commodities that we want to transport. In the optimal solution of the ILP container 0 is transported to the sink of container 1 and container 1 is transported to the sink of container 0. It does not take into account that the containers are distinguishable. Thus that explains that it does not find the real optimal solution, but it finds an infeasible solution.

7

Multi-Commodity Minimum Cost Flow Problem

In this chapter we will show how Problem **1** can be portrayed as an integral multi-commodity min cost flow (MCMCF) problem (Section 7.1). There is an arc-based model (Subsection 7.1.1) and a path-based model (Subsection 7.1.2) for this problem [81]. In the arc-based model the variables are based on the arcs in the network, while in the path-based model the variables are based on paths. Moreover, some solution methods (Section 7.2) from literature for these models are presented. Other related mathematical problems can be found in [29] including a combinatorial algorithm to solve them.

7.1. Problem Formulation

7.1.1. Arc-Based Model

The integral multi-commodity minimum cost flow (MCMCF) problem is a generalization of the integral min cost flow problem. The ILP (7.1)-(7.5) looks similar to the min cost flow problem (6.1)-(6.5), but it does contain some noteworthy differences. We add an extra layer to the min cost flow problem; now we can have multiple items that share the resources in the network. Each of those items is called a commodity $k \in K$. The idea is that these commodities compete for the capacity of available resources and are distinguishable. This can be seen in (7.3); together the commodities must satisfy the shared capacity constraints of a link/arc (7.3). The other constraints in the model are pretty much the same as in the min cost flow problem (see Chapter 6). We will go through everything briefly.

We have a directed graph $G = (V, E)$ (or multigraph) that contains all the nodes and arcs in the network. We have a set of commodities K. Every commodity $k \in K$ has without loss of generality one source node s_k and one sink node t_k. The parameters c_e for $e \in E$ are the capacities of the arcs. The parameters $f_{e,k}$ for $e \in E, k \in K$ determine the per unit cost of commodity k on arc e. The parameters $d_{v,k} = \begin{cases} d_k \text{ if } v = s_k \\ -d_k \text{ if } v = t_k \\ 0 \text{ else} \end{cases}$, where d_k is the demand/size of commodity k. The variables $x_{e,k}$ depict the magnitude of the flow of commodity k on arc e. The objective function (7.1) minimizes the cost using the $f_{e,k}$. The flow conservation constraints (7.2) make sure that the total amount of a commodity that enters the node also leaves the node, except for the sources and sinks. As mentioned before (7.3) are the capacity constraints of the arcs. The sum of the flows of all commodities on an arc should be below the capacity of the arc. The non-negativity constraints (7.4) disallow negative flow. The integrality constraints ensure every variable is integral.

The arc-based integral MCMF ILP:

$$\min \sum_{k} \sum_{e \in E} f_{e,k} x_{e,k} \tag{7.1}$$

subject to

$$\sum_{e \in \delta^+(v)} x_{e,k} - \sum_{e \in \delta^-(v)} x_{e,k} = d_{v,k} \qquad \forall v \in V, \forall k \in K \tag{7.2}$$

$$\sum_{k} x_{e,k} \leq c_e \qquad \forall e \in E \tag{7.3}$$

$$x_{e,k} \geq 0 \qquad \forall e \in E, \forall k \in K \tag{7.4}$$

$$x_{e,k} \in \mathbb{Z} \qquad \forall e \in E, \forall k \in K. \tag{7.5}$$

This all allows us to model Problem **1** completely instead of only one container, namely we can let every container be a different commodity in the problem. It is even possible to combine all the containers of one booking and model them together as a single commodity $k \in K$, this way the model contains less commodities and thus variables, so we will do so. We will have a model in which every commodity corresponds with one booking. That means that every commodity has one sink and one source and the demand of the commodity is the number of containers in the booking. The underlying graph that is used is the time-space graph that was defined in Chapter 5. If a booking starts at a certain place in time-space in the network, then that time-space node becomes a source for the corresponding commodity. Similarly for the sink. The variables $x_{e,k}$ represent the number of containers of commodity k that use link e. The parameters c_e are the capacities of the vehicle (type) that the arc corresponds to. The parameters $f_{e,k}$ represent the per container cost of transporting a container of commodity k on arc e. We can express the ILP in terms of the time-space nodes $v \in V$

$$\min \sum_{k} \sum_{(v_1,v_2,w) \in E} f_{v_1,v_2,w,k} x_{v_1,v_2,w,k} \tag{7.6}$$

subject to

$$\sum_{(v_1,v_2,w) \in E} x_{v_1,v_2,w,k} - \sum_{(v_2,v_1,w) \in E} x_{v_2,v_1,w,k} = d_{v_1,k} \qquad \forall v_1 \in V, \forall k \in K \tag{7.7}$$

$$\sum_{k} x_{v_1,v_2,w,k} \leq c_{v_1,v_2,w} \qquad \forall (v_1, v_2, w) \in E \tag{7.8}$$

$$x_{v_1,v_2,w,k} \geq 0 \qquad \forall (v_1, v_2, w) \in E, \forall k \in K \tag{7.9}$$

$$x_{v_1,v_2,w,k} \in \mathbb{Z} \qquad \forall (v_1, v_2, w) \in E, \forall k \in K. \tag{7.10}$$

Remember that the arcs of the time-space graph in E are of the form $[v_1, v_2, w]$. The variables $x_{v_1,v_2,w,k}$ represent the number of containers from booking/commodity k that are transported by vehicle w from time-space node v_1 to time-space node v_2. Suppose we have a directed time-space graph $G = (V, E)$, where V is the set of $n = |V|$ time-space nodes and E the set of $m = |E|$ arcs. We define a *capacity function* $c : E \to \mathbb{Z}_{\geq 0}$ on the arcs and a *cost function* $f : E \times K- > \mathbb{Z}_{\geq 0}$. We also define the *set of commodities* K, each commodity $k \in K$ has an origin node s_k, a destination node t_k and a demand d_k. If $v_1 \in V$ is the source of commodity k, then we have $d_{v_1,k} = d_k$, for its sink v_2 we have $d_{v_2,k} = -d_k$. For nodes $v_3 \in V$ that are neither sinks nor sources for commodity k, we have $d_{v_2,k} = 0$. We need to find a multi-commodity minimum cost flow (MCMCF) on G; for each commodity k a flow on G of demand d_k from s_k to t_k needs to be found. Those flows need to satisfy the capacity on the arcs; together all the flows of all the commodities that run through an arc cannot exceed the capacity of the arc (7.8). Also the capacity constraints for when $w = truck$ are in the model, but those capacities are set to a high value. They can also be removed (we assume that we have enough trucks). Conservation of flow must hold; for every commodity the same amount of flow must enter the node as the amount of flow that leaves the node (7.7), except for sources and sinks where some excess or slack is added. The same amount of flow that leaves the origin of commodity of each k should enter its destination. There are also non-negativity constraints (7.9) and the integrality constraints (7.10). The objective function (7.6) is a per container per arc cost. The multi-commodity flow must be of minimum cost. Note if $|K| = 1$ this problem reduces to the min cost flow problem. Unless stated otherwise the cost of an arc that runs between two nodes in the same location, a waiting arc, is zero.

There are also variants of this problem. One of those variants is solved in [28]. There solution methods for the k-splittable Multi-Commodity Max Flow problem are discussed. Here for every commodity it is only allowed to use at most k paths. Furthermore, they are interested in maximizing the flow instead of calculating the min cost flow.

The previous ILP is the integral MCMCF problem, because we are working on a time-space graph the node index can be expressed in time and space indices. For the implementation it is useful to use those indices. An arc in the time-space graph is of the form $[(t, loc1), (s, loc2), w] \in (T \times X) \times (T \times X) \times W$ with $s > t$. To specify an arc we need the begin location $loc1$, end location $loc2$, the type of vehicle w and the begin time t. The end time s can be derived from the others using the known travel time for that trip with that type of vehicle.

$$\min \sum_{k} \sum_{[(t,loc1),(t+\mathrm{a}(loc1,loc2,w),loc2),w]\in E} f_{loc1,loc2,t,w,k} x_{loc1,loc2,t,w,k} \tag{7.11}$$

subject to

$$\sum_{[(t,loc1),(t+\mathrm{a}(loc1,loc2,t,w),loc2),w]\in E} x_{loc1,loc2,t,w,k} - \sum_{[(t-\mathrm{a}(loc1,loc2,w),loc2),(t,loc1),w]\in E} x_{loc2,loc1,t-\mathrm{a}(loc1,loc2,w),w,k} = d_{loc1,t,k} \qquad \forall (t,loc1) \in V, \forall k \in K \tag{7.12}$$

$$\sum_{k} x_{loc1,loc2,t,w,k} \leq c_{loc1,loc2,t,w} \qquad \forall [(t,loc1),(t+\mathrm{a}(loc1,loc2,w),loc2),w] \in E \tag{7.13}$$

$$x_{loc1,loc2,t,w,k} \geq 0 \qquad \forall [(t,loc1),(t+\mathrm{a}(loc1,loc2,w),loc2),w] \in E, \forall k \in K \tag{7.14}$$

$$x_{loc1,loc2,t,w,k} \in \mathbb{Z} \qquad \forall [(t,loc1),(t+\mathrm{a}(loc1,loc2,w),loc2),w] \in E, \forall k \in K. \tag{7.15}$$

The parameter a($loc1, loc2, w$) is the travel time from location $loc1$ to location $loc2$ by vehicle (type) w and it is known in advance. Either historical data, a truck routing algorithm or the euclidean distance of ($loc1, loc2$) can be used to estimate the travel time. The travel times are also important, because they can also be used to estimate the travel cost f for the trucks. The objective function of the ILP (7.11) only has a cost function dependent on the size of the container flows. Basically, we multiply some cost values with the arc flows for every commodity. The function f gives us the liberty to let the cost depend on the travel time, mode w and commodity k. It can happen that too many containers are trucked, in that case it is wise to add some capacity constraints for some truck arcs. Another option is to add a capacity restriction c for a group of truck arcs $E' \subseteq E_{\text{truck}}$

$$\sum_{e \in E'} \sum_{k \in K} x_{e,k} \leq c. \tag{7.16}$$

For example all arcs going from a certain location through time.

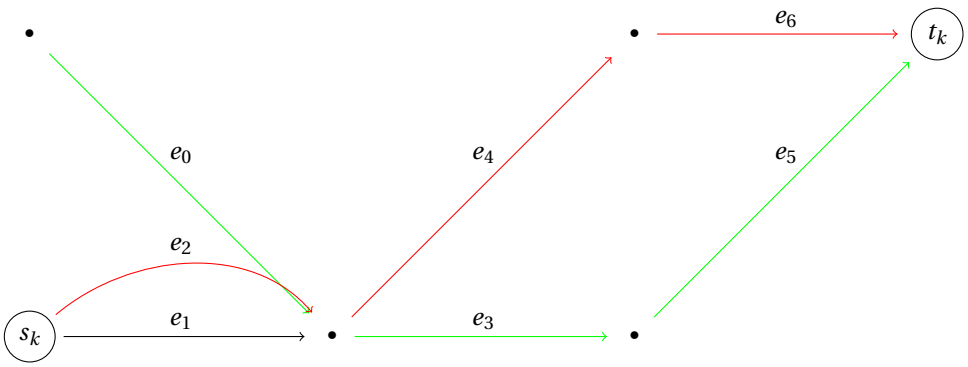
If we have solved the arc-based integral MCMCF problem ILP, then the values of the arc variables $x^*_{e,k}$ have to be used to construct paths for the containers, these may be referred to as *flow paths*. A way to do this, is to take the following steps for every commodity k (Corollary 1 for correctness):

1. Repeat the following steps for all commodities k
2. Repeat for all containers i in commodity k
3. Start at the source of the commodity k
4. Look at all the variables of that commodity that are nonzero in the optimal solution that correspond to the arcs leaving the node
5. Take such an arc†, save it and subtract 1 from its value
6. Repeat the previous two steps for the node that the arc goes to, until the sink of the commodity is reached
7. Make a path for container i of commodity k from the saved arcs (in the sequence they were saved)

† It might be that one is interested in minimizing the number of switches to different vehicles in the construction of the paths for the containers.

We consider Figure 7.1. We have a commodity k with $d_k = 200$. Let H_k be the set of containers in commodity k. The red arcs belong to the route of barge 0 and the green arcs belong to the route of barge 1 and the black arc is a truck arc. Both barges have capacity 100. The optimal solution is $x_{e_i,k} = 100$ for $i \in \{1,2,3,4,5,6\}$. The path-constructing method may find a solution that sends 100 containers over the path with arcs e_1, e_4, e_6 and the other 100 over the path with arcs e_2, e_3, e_5. However the solution where 100 containers are sent over the path with arcs e_2, e_4, e_6 and the other 100 over the path with arcs e_1, e_3, e_5 has less switches. To prioritize these solutions we check in step 5 if it is possible to let the container stay on the same vehicle. This greedy strategy may help, but does not necessarily always find the best solution. For instance if we put the first container in the example on e_1, then it might be put on e_4 afterwards without violating the greedy strategy resulting in a

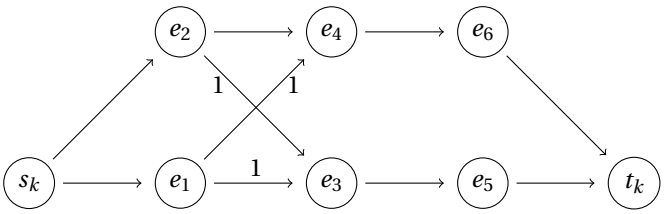
Figure 7.1: Path construction



nonoptimal path construction. If we really want to find an optimal solution to the path construction problem, then we need to resort to a different method.

We create an auxiliary graph $P := (U, O)$ using the arcs and nodes that are used by commodity k in the solution x^*.

Figure 7.2: Path construction auxiliary



For every node of the form e_i in the auxiliary graph we create constraints

$$\sum_{o \in \delta^-(e_i)} u_{o,h} \leq x^*_{e_i,k}$$

and

$$\sum_{o \in \delta^+(e_i)} u_{o,h} \leq x^*_{e_i,k},$$

where $u_{o,h} = 1$ if arc o in the auxiliary graph is used by container h. We define a cost function

$$g(o) = \begin{cases} 1 \text{ if } o := (e_\alpha, e_\beta) \text{ and } \nexists w \in W \text{ with } e_\alpha \in E_w \text{ and } e_\beta \in E_w \\ 0 \text{ otherwise} \end{cases}$$

on the arcs o of the auxiliary graph. The ILP that needs to be solved for every $k \in K$ is

$$\min \sum_{o \in O} g_o \sum_{h \in H_k} u_{o,h} \tag{7.17}$$

s.t.

$$\sum_{o \in \delta^-(e_i)} u_{o,h} \leq x^*_{e_i,k} \qquad \forall h \in H_k \tag{7.18}$$

$$\sum_{o \in \delta^+(e_i)} u_{o,h} \leq x^*_{e_i,k} \qquad \forall h \in H_k \tag{7.19}$$

$$u_{o,h} \geq 0 \qquad \forall o \in O, \forall h \in H_k \tag{7.20}$$

Basically, this is an integral multi-commodity flow problem with capacities on the nodes instead of on the arcs. We can easily transform the graph of Figure 7.2 by splitting all the nodes that represent arcs in the original graph of Figure 7.1. The integral multi-commodity flow problem on this new graph visible in Figure 7.3 is equivalent to the ILP problem.

Instead of (7.18) and (7.19) we have for arcs of the form (e_i^0, e_i^1) the capacity constraints

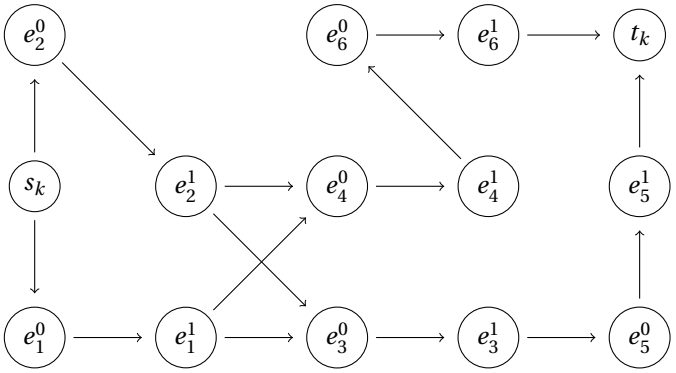
$$\sum_{h \in H_k} u_{(e_i^0, e_i^1),h} \leq x^*_{e_i,k}.$$

The rest of the ILP on this graph is very similar to the previous one.

Theorem 3. *The decision version of the integral multi-commodity min-cost flow problem (with two commodities) on time-space graphs is strongly $\mathcal{NP}$-hard.*

Proof. The decision version of the integral multi-commodity max flow problem with two commodities on directed graphs without directed cycles is strongly $\mathcal{NP}$-hard [12]. Every such decision problem instance is equivalent to an instance of the decision version of the integral multi-commodity min-cost flow problem (with two commodities) on time-space graphs taking $f \equiv \mathbf{0}$ (Theorem 1). □

Figure 7.3: Path construction auxiliary modified



7.1.2. Path-Based Model

The arc-based integral MCMCF problem has a matrix with a special structure. The (block) matrix is of the form

$$\begin{bmatrix} C_0 & C_1 & C_2 & \dots & C_{|K|-1} \\ D_0 & 0 & 0 & \dots & 0 \\ 0 & D_1 & 0 & \dots & 0 \\ 0 & 0 & D_2 & \ddots & 0 \\ 0 & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 0 & D_{|K|-1} \end{bmatrix},$$

where the rows with the C correspond to the capacity constraints (7.13) and the other rows to the other constraints of our ILP. In our case $K = \{0, 1, \dots, |K| - 1\}$ is the set of commodities, but it could also be some other set for other problems. For optimization problems with matrices of this form a decomposition method exists called the Dantzig-Wolfe decomposition. This can find another formulation of the ILP, which may be easier to solve. There are problems for which the Dantzig-Wolfe decomposition (in combination with column generation) can reduce the integrality gap significantly. Problems with a large integrality gap are often hard to solve, so the reformulation can be beneficial. In [42] a problem is shown for which the Dantzig-Wolfe decomposition makes an ILP a lot easier to solve. On matrices of the form

$$\begin{bmatrix} D_0 & 0 & 0 & \dots & 0 & C_0 \\ 0 & D_1 & 0 & \dots & 0 & C_1 \\ 0 & 0 & D_2 & \ddots & 0 & C_2 \\ 0 & \vdots & \ddots & \ddots & 0 & \vdots \\ 0 & 0 & 0 & 0 & D_{|K|-1} & C_{|K|-1} \end{bmatrix}$$

the very similar Benders decomposition can be applied.

The Dantzig-Wolfe decomposition has been applied to a problem similar to ours the multi-commodity max flow problem [73]. We will apply it to the integral MCMCF problem. Let $k \in K$ be fixed, then the rows corresponding to D_k, so some constraints of (7.2) and (7.4) in the model, define a polyhedron. The maximum value a variable can attain is d_k and with the nonnegativity constraints we have that zero is a lower bound. So we can speak of a polytope. A polytope is the convex hull of its vertices [19]. Remember that a time-space graph has no directed cycles. Furthermore, we define its *set of paths from s_k to t_k* as $P^k := \{0, 1, \dots, u\}$ for some $u \in \mathbb{Z}_{\geq 0}$. Then because there are no directed cycles, every solution $\mathbf{x}_k$ in the region defined by the rows corresponding to D_k is in the convex hull of the set of solutions of the type $\mathbf{x}_k^{j*}$ that send flow d_k over one of the paths $P_j^k \in P^k$ (see Corollary 1);

$$\mathbf{x}_k \in \text{convhull}(\{\mathbf{x}_k^{j*} | P_j^k \in P^k\}).$$

Here $\mathbf{x}_k^{j*}$ is defined by

$$x_{e,k}^{j*} = \begin{cases} d_k \text{ for all } e \in P_j^k \\ 0 \text{ else.} \end{cases}$$

Theorem 4. *Let $G = (V, E, W)$ be a finite directed multigraph. Let there be a single source - single sink flow on G, i.e.*

$$\exists! s_0, t_0 \in V,$$

such that

$$\sum_{e \in \delta^+(v)} x_e - \sum_{e \in \delta^-(v)} x_e = d_v \qquad \forall v \in V \tag{7.21}$$

$$x_e \geq 0 \qquad \forall e \in E \tag{7.22}$$

with

$$d_v = \begin{cases} 0 & \text{if } v \notin \{s_0, t_0\} \\ d & \text{if } v = s_0 \\ -d & \text{if } v = t_0 \end{cases}$$

for a value $d \in \mathbb{R}_{>0}$.
Additionally, we assume that there is no directed cycle C, such that $x_e > 0 \forall e \in E(C)$. (•)
Let P be the set of $s_0 - t_0$ paths in G. Then there is a set $Q \subseteq P$, such that $\exists \lambda \in [0,d]^{|P|}$ with

- $\lambda_j > 0 \Leftrightarrow P_j \in Q \qquad \forall j$ *such that* $P_j \in P$,
- $x_e = \sum_{j|e \in P_j} \lambda_j \qquad \forall e \in E$.

If in addition $x_e \in \mathbb{Z} \forall e \in E$, then we can even find $\lambda \in ([0,d] \cap \mathbb{Z})^{|P|}$ such that the above holds.

Proof. Suppose we have a flow $\mathbf{x}$.

1. Initialize $\lambda_j := 0 \forall j$, such that $P_j \in P$ and the set $Q := \emptyset$.
2. Find a $P_j \in Q$, with $x_e > 0 \forall e \in E(P_j)$. (*)
3. (Re)define $mine := \min_{e \in P_j} x_e$ and set $\lambda_j := mine$ and $Q := Q \cup \{P_j\}$.
4. Update $x_e := x_e - mine \forall e \in C_k$. Note that one of those x_e becomes zero, so P_j it no longer satisfies the condition in step 2. Also note that the new flow $\mathbf{x}$ now satisfies the flow conditions (7.21) and nonnegativity conditions (7.22) with $d := d - mine$.
5. Check if there still is a $P_j \in P$, with $x_e > 0 \forall e \in E(P_j)$, if so go back to step 2. (*)

(*) Suppose there is a $x_e > 0$, but there is no $s_0 - t_0$ path P_j with $x_e > 0 \forall e \in P_j$. By (•), we can and will take a longest directed path P in G that starts at s_0 with $x_e > 0 \forall e \in E(P)$ and it is a simple path. Call the last node of that path v_0, then by constraints (7.21) and (7.22), there is a $e \in \delta^+(v_0)$ with $x_e > 0$. So the path we chose was not the longest. This is a contradiction. Consequently, if there is a $x_e > 0$, then there is a directed $s_0 - t_0$ path P_j with $x_e > 0 \forall e \in E(P_j)$.

This algorithm ends in a finite number of steps, because $|V| < \infty$ and in every iteration one arc variable is set to zero (and that value is not increased afterwards).

We found a set Q for which the main statement from the theorem holds. Note that if $x_e \in \mathbb{Z} \forall e \in E$, then $mine \in \mathbb{Z}$ in every iteration, so then $\lambda \in ([0,d] \cap \mathbb{Z})^{|P|}$ follows. □

Theorem 5. *Let $G = (V, E, W)$ be a finite directed multigraph. Let there be a single source - single sink flow on G, i.e.*

$$\exists! s_0, t_0 \in V,$$

such that

$$\sum_{e \in \delta^+(v)} x_e - \sum_{e \in \delta^-(v)} x_e = d_v \qquad \forall v \in V \tag{7.23}$$

$$x_e \geq 0 \qquad \forall e \in E \tag{7.24}$$

with

$$d_v = \begin{cases} 0 & \text{if } v \notin \{s_0, t_0\} \\ d & \text{if } v = s_0 \\ -d & \text{if } v = t_0 \end{cases}$$

for a value $d \in \mathbb{R}_{>0}$. Let P be the set of $s_0 - t_0$ paths in G. Let C be the set of directed cycles in G. Then there is a set $Q \subseteq P$ and a set $B \subseteq C$, such that $\exists \lambda \in [0,d]^{|P|}$ and $\mu \in \mathbb{R}_{\geq 0}^{|C|}$ with

- $\lambda_j > 0 \Leftrightarrow P_j \in Q \qquad \forall j$ *such that* $P_j \in P$,
- $\mu_k > 0 \Leftrightarrow C_k \in B \qquad \forall k$ *such that* $C_k \in B$,
- $x_e = \sum_{j|e \in P_j} \lambda_j + \sum_{k|e \in C_k} \mu_k \qquad \forall e \in E$.

If in addition $x_e \in \mathbb{Z} \forall e \in E$, then we can even find $\lambda \in ([0,d] \cap \mathbb{Z})^{|P|}, \mu \in \mathbb{Z}_{\geq 0}^{|C|}$ such that the above holds.

Proof. Suppose we have a flow $\mathbf{x}$.

1. Initialize $\lambda_j := 0 \forall j$, such that $P_j \in P$, $\mu_k := 0 \forall k$, such that $C_k \in C$ and the sets $Q := \emptyset, B := \emptyset$.
2. Find (if one exists) a $C_k \in C$, with $x_e > 0 \forall e \in E(C_k)$.
3. (Re)define $mine := \min_{e \in C_k} x_e$ and set $\mu_k := mine$ and $B := B \cup \{C_k\}$.
4. Update $x_e := x_e - mine \forall e \in C_k$. Note that one of those x_e becomes zero, so C_k it no longer satisfies the condition in step 2. Also note that $\mathbf{x}$ still satisfies the flow conservation constraints (7.23) and nonnegativity constraints (7.24).
5. Check if there still is a $C_k \in C$, with $x_e > 0 \forall e \in E(C_k)$, if so go back to step 2.

This algorithm ends in a finite number of steps, because $|V| < \infty$ and in every iteration one arc variable is set to zero (and that value is not increased afterwards).

Together with theorem 4, now follows that we found B and can find Q for which the main statement from the theorem holds. Note that if $x_e \in \mathbb{Z} \forall e \in E$, then $mine \in \mathbb{Z}$ in every iteration, so then follows $\mu \in \mathbb{Z}_{\geq 0}^{|C|}$. Theorem 4 then additionally gives $\lambda \in ([0,d] \cap \mathbb{Z})^{|P|}$. □

Corollary 1. *Suppose we have a solution $\mathbf{x}^{\text{sol}}$ of the arc-based MCMCF problem on a time-space graph. Then for every $k \in K$ there is a set $Q^k \subseteq P^k$, such that $\exists \lambda^k \in [0, d_k]^{|P_k|}$ with*

- $\lambda_j^k > 0 \Leftrightarrow P_j^k \in Q^k \qquad \forall j$ *such that* $P_j^k \in P^k$,
- $x_{e,k}^{\text{sol}} = \sum_{j | e \in P_j^k} \lambda_j^k \qquad \forall e \in E$.

Proof. Follows from Theorem 4 and the fact that a time-space graph has no directed cycles (Theorem 1). □

So we have

$$\mathbf{x}_k = \sum_{j : P_j^k \in P^k} \lambda_j^k \mathbf{x}_k^{j*} \text{ for } \lambda_j^k \geq 0 \text{ with } \sum_j \lambda_j^k = 1.$$

The $\mathbf{x}_k^{j*}$ are the vertices of the polytope defined by the rows belonging to D_k. If we define $\mathbf{x}_k^{j\circ}$ as the solution that send flow one over path P_j^k, then we can equivalently write

$$\mathbf{x}_k = \sum_{j : P_j^k \in P^k} \lambda_j^k \mathbf{x}_k^{j\circ} \text{ for } \lambda_j^k \geq 0 \text{ with } \sum_j \lambda_j^k = d_k. \tag{7.25}$$

We substitute each arc variable in the ILP; $x_{e,k} = \sum_{j : P_j^k \in P^k} \lambda_j^k x_{e,k}^{j\circ}$. By construction the a solution $\mathbf{x}$ that satisfies (7.25) already satisfy constraints (7.2) and (7.4), so they become obsolete. We remove them from our new ILP. The $x_{e,k}^{j\circ}$ are parameters and the λ_j^k are the variables. The constraints on the λ_j^k are added to the model.

$$\min \sum_k \sum_{e \in E} f_{e,k} \sum_{j : P_j^k \in P^k} \lambda_j^k x_{e,k}^{j\circ} \tag{7.26}$$

subject to

$$\sum_k \sum_{j : P_j^k \in P^k} \lambda_j^k x_{e,k}^{j\circ} \leq c_e \qquad \forall e \in E \tag{7.27}$$

$$\sum_{j : P_j^k \in P^k} \lambda_j^k x_{e,k}^{j\circ} \in \mathbb{Z} \qquad \forall e \in E, \forall k \in K \tag{7.28}$$

$$\lambda_j^k \geq 0 \qquad \forall k \in K \forall P_j^k \in P^k \tag{7.29}$$

$$\sum_j \lambda_j^k = d_k \qquad \forall k \in K. \tag{7.30}$$

We rewrite the model with $u^j_{e,k} := x^{j\circ}_{e,k}$, because these parameters are binary we can also rewrite condition (7.28)

$$\min \sum_k \sum_{e \in E} f_{e,k} \sum_{j:P^k_j \in P^k} \lambda^k_j u^j_{e,k} \tag{7.31}$$

subject to

$$\sum_k \sum_{j:P^k_j \in P^k} \lambda^k_j u^j_{e,k} \leq c_e \qquad \forall e \in E \tag{7.32}$$

$$\lambda^k_j \in \mathbb{Z} \qquad \forall k \in K \forall P^k_j \in P^k \tag{7.33}$$

$$\lambda^k_j \geq 0 \qquad \forall k \in K \forall P^k_j \in P^k \tag{7.34}$$

$$\sum_j \lambda^k_j = d_k \qquad \forall k \in K. \tag{7.35}$$

The variables in this ILP represent paths, so we will call this ILP the path-based integral MCMCF problem. We see that the new constraints reintroduce the nonnegativity constraints (7.34) and introduce a constraint to enforce the demand of the commodities (7.35). On some instances this model is solved faster with simplex or some other algorithm, than the arc-based equivalent [70]. The path-based version has less constraints, than the arc-based version; because instead of all those flow conservation constraints we only have a few demand constraints. The path-based formulation, however, normally has a lot more variables than the arc-based formulation.

We define a *flow path* P^k_j of a solution for commodity k as a (s_k, t_k) path with $\lambda^k_j > 0$.

We call an arc $[(t, loc1), (t + a(loc1, loc2, w), loc2), w] \in E$ *saturated* with respect to a solution if in the solution the maximum capacity of the arc is reached. So in the path-based version that would mean $\sum_k \sum_{j:P^k_j \in P^k} \lambda^k_j u^j_{loc1,loc2,t,w,k} = c_{loc1,loc2,t,w}$. The corresponding constraint we then call *tight*.

7.2. Solution Methods

7.2.1. General ILP Solution Techniques

The ILPs can be solved with general (I)LP solving techniques. If one has a non integral optimal solution, it can be decided to add constraints to the LP relaxation to find an integral (optimal) solution. If we have a discrete variable $x_i \in \{0, 1\}$ that is non integral in the solution, then we can define two different subproblems the LP relaxation with the added inequality $x_i \leq 0$ and the LP relaxation with the added inequality $x_i \geq 1$. (because x_i is binary we can use equality) These are two branches. If we start to solve the first subproblem different things can occur.

- The solution could become integral, in that case it is not necessary to branch deeper in that direction. The integral solution and its cost is saved, if there was not an integral solution with a lower cost that was found before. Any possible solution with higher cost that was found before can be removed. Any possible subproblem in the branch & bound tree with higher cost also does not have to be explored any further.
- The solution could still be non integral. In this case it is necessary to branch deeper. So we again branch on a variable that is not integral in the solution, but should be integral.
- There could be no feasible solution for the subproblem. In that case we do not branch deeper.
- The cost of the solution could be higher, than the cost of the (best) integral solution that is saved. In that case we also do not search any deeper.

When we are done searching the tree with these rules an optimal integral solution is found. Rules are needed to specify in which sequence to search the variables and the tree.

There are several ways to speed up the branch & bound. One could for example save an integral solution found by a heuristic before beginning the process. Another common thing to do is to add cuts (additional constraints), see Chapter Cutting Planes. This method is called branch & cut. This can be done before the branching process starts or during the branching. It can also be decided to stop searching the tree at a certain point and take the best solution found up until that point. This way it cannot be guaranteed that the actual best solution is found. For example if the objective function value of the best found solution integral solution is close to a known lower bound it can be decided to stop the branch & bound process.

The LPs that are solved during the branch & bound can be solved with various solution methods for LPs such as the simplex method and interior point methods. There are even interior point methods that are specifically tailored to the problem, see [37].

7.2.2. Combinatorial Approximation Algorithm

Another method that divides the problem in single commodity min cost flows can be found in [37]. There a combinatorial approximation algorithm is introduced for the **fractional** MCMCF problem. The algorithm finds an $\epsilon-$optimal solution by roughly doing the following:

1. Choose a commodity k.
2. Compute a min cost flow for commodity k on an auxiliary graph, where the costs are exponential functions with base α of the current flow.
3. Reroute a fraction σ according to the min cost flow that is found.
4. Repeat from the start until an $\epsilon-$optimal solution is found.

More details about this method can be found in the cited paper. There it is also concluded that it is three times faster, than an exact approach with CPLEX for the instances considered.

7.2.3. Repeated Shortest Path Heuristic

An intuitive way to solve the MCMCF problem is to calculate shortest paths for the containers. A sequence is used to specify in which sequence the commodities can schedule their containers. For example if commodity 0 and 1 both with demand 100, want to put all of their containers on the same barge with capacity 100. Then if commodity 0 has higher priority in the sequence, it can put all its containers on the barge and commodity 1 looks for the next best route for its containers. A possible sequence to take is the sequence of ascending deadlines. In that case you give higher priority to commodities that have to be at their sink the earliest. See appendix A.1. for a *repeated shortest path* heuristic [70]. The solution found this way can be arbitrarily worse than the optimal solution value; see [70] for an example.

7.2.4. Repeated Min Cost Flow Heuristic

Another intuitive heuristic for the MCMCF problem that also works with a sequence is the repeated min cost flow heuristic [70] . The difference is that in this case the problems for the single commodities are modelled as a min cost flow problems and solved with one of the available techniques for that problem.

If we do not reduce the capacities, then we get a solution that is highly likely to be infeasible. Basically, the constraints (7.3) are replaced by

$$x_{e,k} \leq c_e \forall e \in E, \forall k \in K. \tag{7.36}$$

This is weaker, because

$$x_{e,k} \leq \sum_{k_1 \in K} x_{e,k_1} \leq c_e \forall e \in E, \forall k \in K. \tag{7.37}$$

So in that case the heuristic gives a lower bound for the integral MCMCF problem. There are no more constraints over multiple commodities, so the integral arc-based MCMCF problem ILP reduces to the ILPs

$$\min \sum_{e \in E} f_{e,k} x_{e,k} \tag{7.38}$$

subject to

$$\sum_{e \in \delta^+(v)} x_{e,k} - \sum_{e \in \delta^-(v)} x_{e,k} = d_{v,k} \qquad \forall v \in V \tag{7.39}$$

$$x_{e,k} \leq c_e \qquad \forall e \in E \tag{7.40}$$

$$x_{e,k} \geq 0 \qquad \forall e \in E \tag{7.41}$$

$$x_{e,k} \in \mathbb{Z} \qquad \forall e \in E \tag{7.42}$$

for all $k \in K$.

Theorem 6. *If the solution of the ILPs (7.38)-(7.42) happens to be feasible, then it is optimal for the integral arc-based MCMCF problem (7.1)-(7.5).*

Proof. Let $(\mathbf{x}_k^*)_{k \in K}$ be optimal solutions for (7.38)-(7.42) and $(\text{val}(\mathbf{x}_k^*))_{k \in K}$ the corresponding optimal solution values of the ILPs (7.38)-(7.42). Let x° be an optimal solution and val(x°) the corresponding optimal solution value of the integral arc-based MCMCF problem. Constraints (7.37) yield $\sum_{k \in K} \text{val}(\mathbf{x}_k^*) \leq \text{val}(\text{x}^\circ)$. Suppose $(\mathbf{x}_k^*)_{k \in K}$ is feasible for the integral arc-based MCMCF problem, then $\sum_{k \in K} \text{val}(\mathbf{x}_k^*) \geq \text{val}(\text{x}^\circ)$. Consequently $\sum_{k \in K} \text{val}(\mathbf{x}_k^*) = \text{val}(\text{x}^\circ)$, so $(\mathbf{x}_k^*)_{k \in K}$ is an optimal solution for the integral arc-based MCMCF problem. □

7.2.5. Resource-Directive Methods

For multi-commodity flow problems resource-directive methods [2] have been developed. These methods make the multi-commodity flow problem equivalent to multiple (single commodity) min cost flow problems

$$\min \sum_{e \in E} f_{e,k} x_{e,k} \tag{7.43}$$

subject to

$$\sum_{e \in \delta^+(v)} x_{e,k} - \sum_{e \in \delta^-(v)} x_{e,k} = d_{v,k} \qquad \forall v \in V \tag{7.44}$$

$$x_{e,k} \leq r_{e,k} \qquad \forall e \in E \tag{7.45}$$

$$x_{e,k} \geq 0 \qquad \forall e \in E \tag{7.46}$$

$$x_{e,k} \in \mathbb{Z} \qquad \forall e \in E \tag{7.47}$$

for all $k \in K$.

The way they do that is by assigning a fraction of the capacities on the arcs to each commodity. This way we get individual capacity constraints for every commodity: $x_{e,k} \leq r_{e,k}$, where $r_{e,k}$ is the part of c_e assigned to commodity k. The equations $\sum_{k \in K} r_{e,k} = c_e \forall e \in E$ holds. Normally an iterative method, such as a subgradient method is used to find a good allocation. Often these methods take too much time in comparison with other methods.

A heuristic approach is to first use the repeated min cost flow heuristic without reducing the capacities. Then if the solution is infeasible, we divide the capacities over the commodities on to arcs that exceed capacity according to the infeasible flow on the arcs. We can then solve the LPs of this chapter. This strategy is then repeated until a feasible flow is found.

7.2.6. Lagrangian Relaxation

Suppose we have a minimization problem:

$$z := \min c^T x$$

subject to

$$A_1 x \leq b_1$$
$$A_2 x \leq b_2$$
$$x \in \mathbb{Z}_{\geq 0}.$$

Then a Lagrangian relaxation of this problem $\sigma(\lambda)$ for Lagrange multipliers $\lambda \geq 0$ is:

$$\sigma(\lambda) := \min c^T x - \lambda^T (b_2 - A_2 x)$$

subject to

$$A_1 x \leq b_1$$
$$x \in \mathbb{Z}^n_{\geq 0}.$$

The idea is that we create a problem that is easier to solve by relaxing certain constraints. The Lagrangian relaxation puts a part of the constraints in the objective function. This way it is allowed to violate such a condition, but it comes at a price depending on the values of the Lagrangian multipliers. If the Lagrangian multipliers are too small, then many of the constraints are violated. If the Lagrangian multipliers are too large, then the optimal solution of the Lagrangian relaxation has a lot higher cost than the optimal solution of the original problem. Therefore normally iterative methods are used to find good Lagrangian multipliers.

The Lagrangian relaxation gives a lower bound for the problem [53]:

$$c^T \hat{x} \geq c^T \hat{x} - \tilde{\lambda}^T (b_2 - A_2 \hat{x}) \geq c^T \bar{x} + \tilde{\lambda}^T (b_2 - A_2 \bar{x}).$$

Here $\bar{x}$ is an optimal solution of the Lagrangian relaxation for $\lambda = \tilde{\lambda}$ and $\hat{x}$ is an optimal solution of the original problem. The first inequality is true, because $\hat{x}$ satisfies the constraints of the original problem and $\tilde{\lambda} \geq 0$. The second one because $\bar{x}$ is an optimal solution of the Lagrangian relaxation.

We rewrite the constraints of the Lagrangian relaxation to $x \in X$, where $X := \{x \in \mathbb{Z}^n_{\geq 0} | A_1 x \leq b_1\}$. The *Lagrangian dual* is defined as

$$\vartheta(\sigma) := \max_{\lambda \geq 0} \sigma(\lambda).$$

The following theorem [27] holds

$$\vartheta(\sigma) = \min\{c^T x | A_2 x \leq b_2, x \in \text{conv}(X)\}.$$

This means that in linear programming strong duality holds. However, in integer linear programming this is not always the case.

Let z^* be the optimal solution value of the original problem and z^*_{LP} the optimal solution value of the LP relaxation of the original problem then we get [27]

$$z^* \geq \vartheta(\sigma) \geq z^*_{LP}.$$

Lagrangian relaxation has already been applied on the integral MCMCF problem in [24, 89]. There the capacity constraints (7.3) are put in the objective function and an iterative strategy for solving the Lagrangian dual problem is developed. Relaxing the capacity constraints (7.3) is a good idea as it simplifies the ILP to shortest path problems. There are $|E|$ Lagrange multipliers, but only a subset of them become non zero during the iterations. In every iteration a Lagrangian relaxation is solved. Every Lagrangian relaxation is actually just t single commodity min cost flow problems. So they can be solved using the network simplex algorithm; an efficient implementation can be found in [38]. Lagrangian relaxation can in practice be used to get a lower bound for the optimal objective function value. This lower bound might be useful for branch & bound. A combination of the Lagrangian relaxation and a penalty method, called the augmented Lagrangian relaxation method is successfully applied to extremely large instances of the MCMCF problem in [57].

Different methods to solve the Lagrangian dual problem exist [8], such as the bundle method and the subgradient method. We will look at a subgradient method.

subgradient method adapted from [2]

1. Set $k = 0$ and choose λ_0. We can take $\lambda_0 = \mathbf{0}$.
2. Compute $\sigma(\lambda_k)$ and x_k, where it is achieved.
3. Choose subgradient $g_k = A_2 x_k - b_2$.
4. If $(g_k)_i \approx 0$ for all $(\lambda_k)_i > 0$ we stop, the solution is $\sigma(\lambda_k)$
5. Compute $\lambda_{k+1} = \max(0, \lambda_k + \theta_k^T g_k)$ where θ_k is the stepsize
6. Increment k and go to step 2

It is possible to use $\theta_{k+1} = \tau\theta_k, \theta_0 = 1, \tau \in (0,1)$ or to take $\theta_k = \frac{1}{k}$. Also more complicated step sizes like $\theta_k = \tau_k \frac{UB - \sigma(\lambda_k)}{||A_2 x_k - b_2||^2}$, where τ_k starts at 1 and halves if $\sigma(\lambda_k)$ stays the same for three iterations can be used [57, 89].

We will use the termination criterion $\max_{e \in E} |\frac{(g_k)_e}{c_e}|(\lambda_k)_e \leq \epsilon_t$ from [57] for solving the MCMCF problem with this method. Here $\epsilon_t > 0$ is a small number. Clearly if ϵ_t is small enough, then $(g_k)_i \approx 0$ for all $i \in E$.

A Lagrangian heuristic uses the Lagrangian relaxation to construct an approximation of the optimal solution (instead of only the optimal solution value) of the primal. In [57] an approach is discussed to get a feasible approximate solution for the primal. We lower the capacities in the graph according to $c_i := c_i(1 - \epsilon_p), \epsilon_p > 0$ and we add the additional termination criterion $\max \frac{(g_k)_i}{c_i} \leq \frac{\epsilon_p}{(1-\epsilon_p)}$. So we first run the Lagrangian relaxation method and then the Lagrangian heuristic. The first time we get a lower bound and the second time we get a feasible solution for the primal an thus an upper bound. Lagrangian relaxation method have relatively low computation time especially for large instances. The Lagrangian heuristic is trying to find an approximate solution to the integral MCMCF problem. The solution normally always sends the complete commodity over one path. Note that the upper and lower bound can also be used in the branch & bound tree of an exact method [44].

Many of the capacity constraints are not tight in the optimal solution. However to know which of those constraints are obsolete is difficult. In [7] an active set strategy is discussed that is combined with Lagrangian relaxation to solve the MCMCF problem. The basic idea for the active set strategy is that the arcs are split into two disjoint sets E_1, E_2 for which $E = E_1 \cup E_2$. Only for all the arcs in E_1 capacity constraints are added. An iterative solution method is used to solve the problem during the iterations there is a rule which decides which arcs should be in E_1 and which should not. A possible rule to move arcs e from E_2 to E_1 if in an optimal solution the capacity of e is violated. This active set strategy technique can be combined with Lagrangian relaxation, but can also be used with branch & bound. The MILP solver CPLEX can do a very similar thing by adding the capacity constraints as *lazy constraints* [48].

7.2.7. Column Generation

Column generation is a technique to solve LPs with a large number of variables and has been applied to the path-based MCMCF problem in [81]. It is applied to other flow problems in [73]. The idea is that only a small number of variables are added to the LP initially. Through iterations of the method it is checked if enough variables are added to proof optimality or if more variables should be added. To decide which variables should be added duality theory (see appendix A.2.) is used to find the dual of the fractional path-based MCMCF problem:

$$\max \sum_{e \in E} c_e z_e + \sum_{k \in K} y_k d_k \tag{7.48}$$

subject to

$$y_k + \sum_{e \in E} z_e a_{e,j}^k \leq \sum_{e \in E} f_{e,k} a_{e,j}^k \qquad \forall k \in K, \qquad \forall j \in P^k \tag{7.49}$$

$$z_e \leq 0, y_k \text{ unrestricted}, \qquad \forall e \in E, k \in K \tag{7.50}$$

where

$$a_{e,j}^k = \begin{cases} 1 \text{ if } e \in P_j^k \\ 0 \text{ else.} \end{cases}$$

Suppose the primal fractional path-based MCMCF LP has a bounded optimal solution. For linear programming strong duality holds, so then the dual has the same optimal solution value. Let us now describe a column generation method for the fractional path-based MCMCF problem. We call the primal the master LP and its dual the dual master LP. Let P^k be the master LP, then the primal restricted to $P'^k \subseteq P^k$ for every commodity k is called a restricted master LP. The dual of the restricted master LP is called the dual restricted master LP. If an optimal solution to the dual restricted master LP is a feasible solution for the dual master LP, then it is also optimal for the dual. This is because both dual problems have the same variables, only the constraints in the dual restricted master LP are a subset of the constraints of the dual master LP. Furthermore in that case the optimal solution to the restricted master LP is an optimal solution of the master LP by strong duality.

To start the column generation procedure an initial set of variables that make up the P'^k for every commodity $k \in K$ is required. One option to find such a set is to follow the phase I procedure described in [81], another option is to start from a feasible solution for example one obtained from a fast heuristic such as the repeated cheapest path heuristic (Appendix A.1.). In that case for all commodities all path variables corresponding to flow paths with positive cost of the feasible solution of the heuristic are added to the P'_k. All the slack variables corresponding to capacity constraints that are not tight are also added. Now we find an optimal solution and dual prices for each constraint to the restricted master LP with the simplex method. These dual prices satisfy the constraints in the dual restricted master LP. It is however necessary to check if the solution to the dual restricted master LP violates a condition of the dual master LP. If $z_e > 0$ for an $e \in E$, then condition (7.50) is violated. Consequently slack variable $s_e \geq 0$ is added to the restricted master LP and we start over. We look at condition (7.49). This condition can be equivalently be written as

$$\sum_{e \in E} (f_e - z_e) a^k_{e,j} \geq y_k \qquad \forall k \in K \forall j \in P^k.$$

Instead of checking if one of these constraints is violated one at the time, a better option exists. It can be checked relatively easily, if this condition is violated for a commodity and path by solving $|K|$ shortest path problems. Values $f_e - z_e$ are put on the arcs. Then for each commodity k, we calculate the shortest/cheapest $s_k - t_k$ path with respect to these values. Shortest paths can for example be found with *Dijkstra*, because the values put on the arcs are positive. If that value corresponding to that path is smaller than y_k, then the corresponding constraint is violated and the path should be added as variable to the restricted master LP and we should restart the steps. If for all commodities no such path can be found, then the optimal solution found to the restricted master LP is optimal for the master LP.

For ILPs such as the **integral** MCMCF problem the column generation method can be combined with branch & bound. The resulting method is called branch and price and can be seen in Figure 7.4. More information about branch and price can be found in [26, 65]. The branch and price framework is applied to the integral MCMCF problem in [11] and to similar problems in [59].

7.2.8. ILP-Based Heuristics

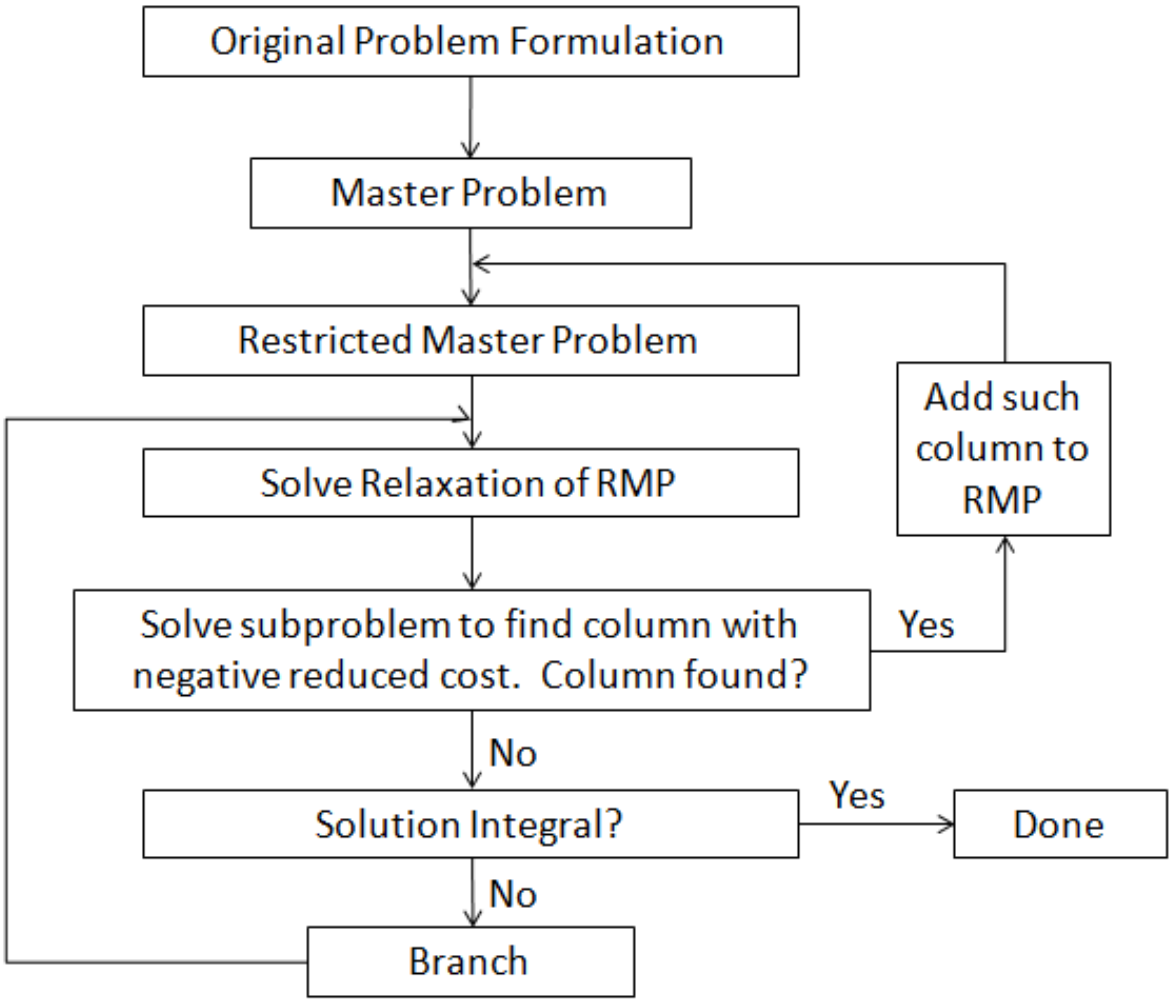
A way to solve the integral MCMCF problem on our time-space graphs for intermodal transport is with ILP-based heuristics.

Using Reductions

Reductions similar to reduction D are used to reduce the computation time of the integral arc-based MCMCF problem. For every **terminal** location z_1 we add the arcs $[(s-1, z_1), (s, x), \text{truck}]$ for every non-truck arc $[(s, x), (t, y), w]$. We call these class A truck arcs. Add for every origin destination pair (s_k, t_k), the truck arc $[(\tau(t_k) - 1, \chi(s_k)), t_k, \text{truck}]$. We call them class B truck arcs. Note if the travel times for trucks are taken into account we will use $s - \text{a}(\cdot, \cdot, \cdot)$ instead of $s-1$, for certain input for travel time function a. For every commodity k, we add an arc $[(\tau(t_k) - 1, z_3), t_k, \text{truck}]$ from every terminal location z_3 to the destination of the commodity. We also add arcs from s_k to every terminal location z_4, so $[s_k, (\tau(s_k) + 1, z_4), \text{truck}]$. Together we call these class C truck arcs. It is also possible to only add class C truck arcs to and from terminals that are relatively close to the customer location. We call these class C_r truck arcs, where r is the radius in kilometers in which the terminals are from the customer location.

In [70] only class B truck arcs are used in a path-based implementation. This is a heuristic suitable for inter-terminal transport, because terminals can often be reached by train and/or barge. Roughly the following steps are taken to find an integral MCMCF.

Figure 7.4: Branch and Price diagram adapted from [3]



Method 1

- For every commodity k find all paths satisfying theorem 1 for its origin-destination pair with Dijkstra in the graph with class B truck arcs.
- Use that information to solve the ILP with an ILP solver

If there are also customers, method 1 will truck everything that has to be delivered to or from a customer directly from its origin to its destination. When we also add class A and C truck arcs to the time-space graph, an exact solution to the integral MCMCF problem can be found, but the computation time might become too large. (Method 2) Consequently, we also look at another extension of method 1 that works when there are customers, but normally does not find an exact solution to the integral MCMCF problem. (Method 3)

Method 2

- For every commodity i find all paths satisfying theorem 1 and corresponding cost for its origin-destination pair with Dijkstra in the graph with class A, B and C truck arcs.
- Use that information to solve the ILP with an ILP solver

Method 3

- For every commodity i find all paths satisfying theorem 1 and corresponding cost for its origin-destination pair with Dijkstra in the graph with class B and C truck arcs.
- Use that information to solve the ILP with an ILP solver

Method 4 uses the solution of method 3 to to give a better approximation to the integral MCMCF problem. The computation time increases, but the cost of the solution decreases (or stays the same) in comparison with the solution of method 3.

Method 4

- For every commodity i find all paths satisfying theorem 1 and corresponding cost for its origin-destination pair with Dijkstra in the graph with class B and C truck arcs.
- Use that information to solve the ILP with an ILP solver
- For every commodity i find all paths with lower cost, than the previously found flow path for commodity i with the highest cost found in the previous step satisfying theorem 1 and corresponding cost for its origin-destination pair with Dijkstra in the graph with class A, B and C truck arcs.
- Use that information to solve the ILP with an ILP solver

The solution of method 4 is not necessarily optimal, but it does give an upper bound for the optimal solution value $f_{LB} \leq f_{opt} \leq f_{M4}$. Here also a lower bound f_{LB} is added. Solving the LP relaxation is a way to obtain a lower bound. Another lower bound was mentioned in Section 7.2.4. f_{opt} is the cost of an optimal solution and f_{M4} is the cost of the solution of method 4.

Using the LP Relaxation
An optimal solution of the LP relaxation of the integral MCMCF problem on time-space graphs can be calculated in polynomial time. This optimal solution however is not necessarily integral. This non integral optimal solution can be used to create a non feasible integral solution [13]. After which flow can be added to get a feasible integral not necessarily optimal solution:

- Calculate an optimal solution of the LP relaxation of the integral MCMCF problem
- If this solution is integral, we are done. Otherwise decompose the flow in this solution for every commodity in flow paths (for arc-based models see Section 7.1.1)
- Take the floor $\lfloor \cdot \rfloor$ of the flow of every flow path. The total removed flow of commodity k in this step is ∂_k
- Add the flow that was removed again to the respective commodities with a heuristic*

*A possible option for the heuristic is the repeated cheapest path heuristic (Appendix A.1.).

8

Multi-Commodity Network Design Problem

8.1. Problem Formulation

Recall that the main goal of this thesis is about simultaneously routing vehicles and scheduling containers; solving Problem **3**. The integral arc-based MCMCF problem can be extended to be able to solve the main problem of this thesis. Up until now we looked at models in which the paths of the vehicles were fixed in advance. We now allow the possibility to optimize the routes of the vehicles as well. In the next problem from literature an additional service/design layer is added to the MCMCF problem.

We define design variables (8.4) $y_e \forall e \in E$, that are one if the service at link e is active and zero otherwise. In intermodal transport these design variables are normally one if and only if the corresponding vehicle travels the corresponding link. Many possible arcs to travel are added for a vehicle and after optimization process, it is decided which design variables are one; ergo which routes the vehicles should travel. The graphs for these models are often time-space graphs, but other graphs can also be used [79].

One of the models that could be used as model is the service network design problem/capacitated fixed charge network flow problem from [34, 41, 63, 75]:

$$\min \sum_{k \in K} \sum_{e \in E} f_{e,k} x_{e,k} + \sum_{e \in E} g_e y_e \tag{8.1}$$

subject to

$$\sum_{e \in \delta^+(v)} x_{e,k} - \sum_{e \in \delta^-(v)} x_{e,k} = d_{v,k} \qquad \forall v \in V, \forall k \in K \tag{8.2}$$

$$\sum_{k} x_{e,k} \leq c_e y_e \qquad \forall e \in E \tag{8.3}$$

$$x_{e,k} \geq 0, y_e \in \{0,1\} \qquad \forall e \in E, \forall k \in K. \tag{8.4}$$

The first part of the objective function (8.1) is identical to (7.2). The second part is a link cost, if a vehicle travels a certain link $e \in E$, then a certain cost g_e is added. The flow conservation constraints (8.2) are identical to (7.2). The capacity constraints (8.3) are different from before (7.3). If an arc in the network is not travelled by a vehicle, then no commodity flow may be on that arc. If a vehicle does travel an arc, then the container flow on that arc can be at most the capacity of the vehicle.

Per commodity capacity constraints are sometimes [5] added:

$$x_{e,k} \leq u_{e,k} y_e \qquad \forall e \in E, \forall k \in K, \tag{8.5}$$

where $u_{e,k}$ is the per commodity per arc upper bound.

This model is in some papers [17, 72, 85, 86] extended to include what are called design-balanced constraints:

$$\sum_{e \in \delta^+(v)} y_e - \sum_{e \in \delta^-(v)} y_e = 0 \qquad \forall v \in V. \tag{8.6}$$

These constraints make sure that everywhere a vehicle arrives, it also leaves. This means that the routes for the vehicles are directed cycles. So the graph where we are working over should contain directed cycles. When working over a time-space graph an additional arc should be added from the sink of vehicle type w to the source of vehicle type w to make sure it is possible to have a directed cycle.

In [79] a similar continuous time ILP is proposed. This model also has time variables and some more types of constraints. An extension of the design-balanced service network design problem is given in [60]. The model takes into account the usage of vehicles and the opening of corridors. In [5] another extension is derived and in [50] a model that shares some resemblances is applied to freight car distribution in scheduled railways. A completely different ILP that can handle the same sort of problem is given in [45].

The problem we will consider is similar, though contains some key differences:

- The design variables are integral, because there can be multiple trucks on an arc.
- Only the first part of the objective function of the service network design problem is used.
- The flow conservation constraints for the vehicles work slightly differently, than the design-balanced constraints:
  1. In our model the number of non-truck vehicles is prespecified.
  2. Only for the non-truck vehicles, vehicle flow conservation constraints are added.

We call our problem the multi-commodity network design (MCND) problem. For the model we need a time-space graph for intermodal transport, that is a bit different than the one for intermodal transport that we defined before. The routes of the non-truck vehicles are not known in advance. So the time-space graph will be changed first. The time-space graph still has the same nodes and truck arcs, but $\forall w \in W \setminus \{\text{truck}\} \forall t \in T \forall loc1 \in X$, we add arcs $[(t, loc1), (t + \text{a}(loc1, loc2, w), loc2), w] \forall loc2 \in X$ if $t + \text{a}(loc1, loc2, w) \in T$. In other words, for all other vehicles we make sure all the links of all the possible routes they can take are included in the time-space graph. The (arc-based) integral MCND problem is:

$$\min \sum_k \sum_{e \in E} f_{e,k} x_{e,k} \tag{8.7}$$

subject to

$$\sum_{e \in \delta^+(v)} x_{e,k} - \sum_{e \in \delta^-(v)} x_{e,k} = d_{v,k} \qquad \forall v \in V, \forall k \in K \tag{8.8}$$

$$\sum_{e \in \delta^+(v) \cap E_w} y_e - \sum_{e \in \delta^-(v) \cap E_w} y_e = b_{v,w} \qquad \forall v \in V, \forall w \in W \setminus \{\text{truck}\} \tag{8.9}$$

$$\sum_k x_{e,k} \leq c_e y_e \qquad \forall e \in E \setminus E_{\text{truck}} \tag{8.10}$$

$$\sum_k x_{e,k} \leq c_e \qquad \forall e \in E_{\text{truck}} \tag{8.11}$$

$$x_{e,k} \geq 0, y_e \geq 0 \qquad \forall e \in E, \forall k \in K \tag{8.12}$$

$$x_{e,k}, y_e \in \mathbb{Z} \qquad \forall e \in E, \forall k \in K. \tag{8.13}$$

For all non-truck arcs e in the graph, we have created a discrete design variable $y_e \in \mathbb{Z}_{\geq 0}$, determining if the arc is used (8.13), (8.12). The y_e are binary variables unless Reduction C: Same Vehicle Type (Section 9.2.1) is applied. The container flows are again modelled with the variables $x_{e,k}$ and still have to be integral (8.13) and non-negative (8.12). For example barge 0 travels arc $e \in E_w$ with w = barge0, then we get $y_e = 1$. We assume that trucks do not necessarily need to be used the whole day, whereas barges do have to be used the whole day. For the paths of the barges to make sense we should add constraints, that disallow the barge to teleport or travel multiple links at the same time. Flow conservation constraints for $w \in W \setminus \{\text{truck}\}$ (8.9) can

do exactly this in the same the flow conservation constraints (8.8)/(7.2) work for the commodities. This is normally equivalent with (8.6), but can be more effective if reduction C is used. In these constraints we have $b_{v,w}$ which describe the time-space nodes that are the sink and source for a $w \in W \setminus \{\text{truck}\}$. In more detail for such w we have

$$b_{v,w} = \begin{cases} b_w \text{ if } v \text{ is the source node of } w \\ -b_w \text{ if } v \text{ is the sink node of } w \\ 0 \text{ else,} \end{cases}$$

where b_w is the number of vehicles of type $w \in W \setminus \{\text{truck}\}$. This is normally 1 unless the vehicle reduction from Section 9.2.1 has been applied. The capacity of an arc is dependent on the number of vehicles that travel the arc (8.10). For the truck arcs we have the same capacity constraints as in the MCMCF problem (8.11). Normally a vehicle has the same capacity the whole day. So we can then replace the c_e in the capacity constraints for the arcs, by capacity constants for the vehicle type, c_w. Where c_w is the capacity of a vehicle belonging to w.

We assume a truck can carry exactly one container. Thus, the number of trucks that travel an arc $e \in E_{\text{truck}}$ is equal to the number of containers that are trucked on that arc $\sum_{k \in K} x_{e,k}$. The integral MCND problem has integral commodity flow variables, no link cost for the vehicles and only design variables and vehicle flow conservation constraints/design-balanced constraints for a subset of the arcs in the network unlike the service network design problem.

Similar as for the arc-based MCMCF problem we again need to construct the paths for the commodities. Now however we also need to construct paths for the non-truck vehicles. Suppose we have a solution (x^*, y^*).

1. Repeat the following steps for all $w \in W \setminus \{\text{truck}\}$
2. Repeat for all vehicles $l \in \{0, 1, \ldots, b_w - 1\}$
3. Start at the source v of vehicle l
4. Look at all the y_e^* variables with $e \in E_w$ that are nonzero in the optimal solution that correspond to the arcs leaving the node
5. Take such an arc e, save it and subtract 1 from its value
6. Repeat the previous two steps for the node that the arc goes to, until a sink of w is reached
7. Make a path for vehicle l of vehicle type w from the saved arcs (in the sequence they were saved)

The capacity constraints (8.11) for the truck arcs E_{truck} can also often be taken out completely. This also allows the ILP to be written in a path-based form. Here we assume that every vehicle of the same type has the same source and sink.

$$\min \sum_k \sum_{e \in E} f_{e,k} \sum_{j : P_j^k \in P^k} \lambda_j^k \mathrm{u}_{e,k}^j \tag{8.14}$$

subject to

$$\sum_k \sum_{j : P_j^k \in P^k} \lambda_j^k \mathrm{u}_{e,k}^j \leq c_e \sum_{i : Q_i^w \in Q^w} \zeta_i^w r_{e,w}^i \qquad \forall w \in W \setminus \{\text{truck}\}, \forall e \in E_w \tag{8.15}$$

$$\sum_j \lambda_j^k = d_k \qquad \forall k \in K \tag{8.16}$$

$$\sum_i \zeta_i^w = b_w \qquad \forall w \in W \setminus \{\text{truck}\} \tag{8.17}$$

$$\lambda_j^k \geq 0, \zeta_i^w \geq 0 \qquad \forall k \in K, \forall j \text{ with } P_j^k \in P^k, \forall w \in W \setminus \{\text{truck}\}, \forall i \text{ with } Q_i^w \in Q^w \tag{8.18}$$

$$\lambda_j^k, \zeta_i^w \in \mathbb{Z} \qquad \forall k \in K, \forall j \text{ with } P_j^k \in P^k, \forall w \in W \setminus \{\text{truck}\}, \forall i \text{ with } Q_i^w \in Q^w, \tag{8.19}$$

where Q^w is the set of paths for vehicles of type w and $r_{e,w}^i := \begin{cases} 1 \text{ if } e \in Q_i^w \\ 0 \text{ else.} \end{cases}$

It is also possible to only consider path variables for the non-truck vehicles. This is a good idea, because the vehicle variables are what makes the problem challenging. In [86] this is done for the service network design problem and column generation and slope scaling are used to solve that problem.

8.2. Solution Methods

8.2.1. ILP based heuristics

Besides branch & cut and column generation other solution methods are available. A very effective ILP-based heuristic is the α *cut-and-fix* heuristic from [17] that has been applied to the service network design problem with design-balanced constraints. The outline of the heuristic can be used for all types of ILPs, but is only effective if most variables are zero in feasible solutions of an ILP. The α cut-and-fix heuristic roughly comes down to

1. Take an $\alpha \in \mathbb{Z}_{\geq 0}$.
2. Solve the LP relaxation of the DBSND problem.
3. Save which variables are used in the solution (so non-zero).
4. Look if there are cutting planes (more info about cutting planes can be found in Chapter Cutting Planes) that are violated, if so add them.
5. Solve the problem with the added cutting planes.
6. Repeat from step 2, unless the solution is integral or the solution value did not improve in the last iteration.
7. Solve the DBSND problem restricted to the variables that were non-zero in at least α LP relaxations (and possibly some additional variables for the connectivity feasibility).

This algorithm has a relatively low computation time and good solution quality. Consequently, this is a motivation for investigating a few variations of this heuristic for the implementation.

The first variation is the α *B&C-and-fix* heuristic. It is very similar to the previous heuristic, but it does branch & cut instead of only adding cutting planes. The advantage of this method is that one can let the first phase run as long as is desired.

1. Take an $\alpha \in \mathbb{Z}_{\geq 0}$.
2. Do branch & cut for the integral MCND problem, while saving which variables are used in every node of the branch & cut tree (so non-zero). After a certain time is elapsed or if the solution did not improve sufficiently for some time, then the branch & cut process is terminated.
3. Solve the DBSND problem restricted to the variables that were non-zero in at least α nodes in the branch & cut tree (and possibly some additional variables for the connectivity feasibility).

If the computation time of the α cut-and-fix is too high, then some steps can be omitted. The resulting variation we call the *relax-and-fix* heuristic:

1. Solve the LP relaxation of the integral MCND problem.
2. Save which variables are used in the solution (so non-zero).
3. Solve the MCND problem with only the variables that were non-zero in the LP relaxation (and possibly some additional variables for the connectivity feasibility).

In the relax-and-fix heuristic only a few variables are selected in the first phase. More variables can be selected in phase one by repeating phase one with (slightly) changed capacities and/or costs. Another option is to only use the selected design variables to decide which variables to fix in phase two. A variation of the α cut-and-fix heuristic is combined with a Lagrangian heuristic to solve the service network design problem with per commodity capacity constraints in [43]. There the flow conservation constraints for the commodities are relaxed.

Two other ILP based approaches are:

- solving the ILP with a smaller set T (a coarser time grid),
- repeatedly solving the problem with a subset of the commodities (and possibly also vehicles).

This first approach will be further explained in reduction L (Section 9.6.2) and the second approach works similar as the repeated min cost flow heuristic (Section 7.2.3).

8.2.2. Metaheuristics

General ILP techniques might not always be able to solve the MCND ILPs fast enough. Therefore metaheuristics exist that can be used to (approximately) solve the MCND problem. We start off with a general introduction. Metaheuristics are higher-level problem-independent strategies to build heuristics [36]. Local Search and similar metaheuristics require an initial solution. That solution is used to find a new solution with lower cost. From that new solution the next solution is obtained. This process is iterated. In order to use Local Search you need to define *neighbourhoods* to find those new solutions. Every solution has a neighbourhood of solutions that are normally in some way similar. After a certain amount of iterations a solution is obtained that has the lowest cost of its neighbourhood. That final solution is called a local optimum. For a given problem there are multiple things that influence the process:

- the way a next solution is picked from a neighbourhood,
- the sizes of the neighbourhoods,
- the structures of the neighbourhoods,
- the initial solution.

If there are multiple solutions with lower cost in a neighbourhood, then there are several options to pick the next solution. Options include choosing the first solution evaluated in the neighbourhood that has lower cost that the current solution or choosing the solution with the lowest cost. It is clear that with that first strategy there are generally less solutions that need to be evaluated every iteration, but there could be more iterations needed to find a local optimum.

Normally if the neighbourhoods are larger, the computation time will increase, but the final solution is better. This trade-off should be taken into account when using local search in practice. Note: If you define all neighbourhoods to be the set of all possible solutions and you pick the strategy of choosing the best solution from your neighbourhood, then you will get the optimal solution in one iteration, but you need to calculate the cost of all possible solutions.

Clearly, if we define our neighbourhoods differently, the local solutions found and the amount of iterations can change. This is because other solutions might be chosen during the iterations when there are different solutions in the neighbourhoods.

The initial solution influences the computation time, which local optimum is found etc.. A more or less randomized initial solution can be used, by then repeating local search a broad part of the feasible region can be explored. Local search can also be used to improve an existing solution that is not a local optimum (for example a solution of a heuristic).

To avoid local search getting stuck in a local optimum too quickly, metaheuristics derived from local search are used. Tabu Search allows going back to a worse solution when a local optimum is attained and keeps a Tabu list containing the k last visited solutions. This is to avoid visiting any of the last k solutions again, but even with this Tabu list it is possible to get stuck in a loop. When k is larger that chance is smaller. The best found solution is also memorized. To make sure the metaheuristic stops, the number of iterations can be fixed. It can also be decided that if a new best solution is not found for a certain amount of iterations the algorithm stops.

In order to apply local search or similar metaheuristics to improve a solution of the integral MCND problem on time-space graphs, it is essential to define neighbourhoods. The most challenging part of the integral MCND problem is the integrality of the arc design variables; y_e. If these variables are known, then the problem reduces to the integral MCMCF problem, which is a lot easier to solve. Therefore some of the methods that use metaheuristics for these kind of problems in literature focus on those types of variables. Metaheuristics are often combined with exact methods into what is known as a matheuristic. In [72] a two-step matheuristic

that uses Tabu search is applied to a service network design model. The neighbourhoods for the Tabu search are defined by closing (or opening) an arc design variable y_e. In [85] a three-step matheuristic that uses Tabu search with a cycle based neighbourhood and path relinking metaheuristics is used. The three-step method shows better results, than the two-step method. The matheuristic of [41] uses both arc-based and path-based formulations. In [79] a metaheuristic approach for a continuous time service network design problem is introduced. Here neighbourhoods are based on the scheduled routes of one, two or three vehicles. In [61] a matheuristic is introduced that is based on the LP relaxation and Lagrangian relaxation. A Tabu search approach with a cycle-based neighbourhood is implemented in [60]. In [88] two metaheursitics namely simulated annealing and genetic algorithms are used for a container allocation problem. A matheuristic that uses a neighbourhood based on local branching is used in [75]. The scatter search metaheuristic is used in [22] and it is concluded that path relinking approach is a better option.

We will use a local search metaheuristic which can be described by:

1. Obtain an initial feasible solution to the integral MCND problem.
2. Pick $t_{\min}, t_{\max} \in T$ with $t_{\min} < t_{\max}$ and $w \in W \setminus \{\text{truck}\}$
3. Pick the best solution from the chosen neighbourhood with respect to $t_{\min}, t_{\max}$ and w of that solution.
4. Move to that solution.
5. Go to step 2, unless stopping criterion is satisfied.

To be more specific some options for the initial solution are:

- A feasible solution obtained during branch & bound
- A feasible solution of a heuristic [1]

We define neighbourhood NEIGH-$(t_{\min}, t_{\max}, w)$ we can use, where $[t_{\min}, t_{\max}]$ is a time frame, for a feasible solution $(\mathbf{x}^*, \mathbf{y}^*)$ by:

1. Deselecting all design variables belonging to arcs of non-truck vehicle type w that start at $t_{\min}$ or later and end before $t_{\max}$ or earlier.
2. Deselecting all container flow variables of all arcs that start at $t_{\min}$ or later and end before $t_{\max}$ or earlier.
3. Deselecting for every commodity k all waiting and truck arc variables that are on the path that corresponds with waiting as long as possible at the origin location of k and finally trucking it to the destination location of k.
4. Reconstruct paths for containers and vehicles, while keeping every variable that was not deselected fixed at its former value.

Even when we construct a really 'bad' route for vehicles of type w within $[t_{\min}, t_{\max}]$, then it is always possible to truck containers to their destination because of the third item. Furthermore, the second item makes it possible for new containers to be transported by w. For our metaheuristic, a best solution from this neighbourhood is selected. Such a solution is found by solving the integral MCND problem with an ILP solver with every variable that is not deselected fixed to its former value.

The parameters $(t_{\min}, t_{\max}, w)$ are chosen according to a strategy: Let Δ_w be the maximum travel time of a vehicle of type w for travelling a link. Then in the first part of the strategy we choose we explore neighbourhood for the parameters

$$(\min T, \min T + \Delta_w, w) \text{ and } (\max T - \Delta_w, \max T, w) \qquad \forall\, w \in W \setminus \{\text{truck}, \text{fixed}\},$$

we call $[\min T, \min T + \Delta_w]$ and $[\max T - \Delta_w, \max T]$ begin time-frame and end time-frame for w respectively. We take Δ_w, because we also want to allow the vehicle to travel 'longer' arcs. A larger value can also be chosen,

[1] Some options to find an initial solution with a heuristic are the greedy gain heuristic and compatibility clustering heuristic described in [45] and the ILP based heuristics of the previous section. We will use the ILP based heuristics to find initial solutions and see how they compare in computation time and solution quality in the Chapter Results. The metaheuristic we described can already be called a matheuristic, because exact ILP solution methods are used in finding a best solution from the neighbourhood. There is even more reason to call it a matheuristic now we are solving ILPs to find an initial solution.

but taking a too large value can make this subproblem too difficult to solve quickly. In the second part we draw a random $w \in W \setminus \{\text{truck}, \text{fixed}\}$ and $t_{\min} \in [\min T, \max T - \Delta_w] \cap \mathbb{Z}$ and use

$$(t_{\min}, t_{\min} + \Delta_w, w).$$

We add the stopping criterion: If for every $w \in W \setminus \{truck, fixed\}$ iterations with the begin time-frame and end time-frame part are completed and in the last three iterations the solution improved less than κ percent, then it stops.

Local Branching
Different options for neighbourhoods do exist. In [75] a neighbourhood based on local branching is introduced. If the non-truck vehicles in the model are all modelled separately, then the non-truck vehicle $w \in W \setminus \{\text{truck}\}$ arc variables y_e are binary variables. An option then is to use the ρ-OPT for a $\rho \in \mathbb{Z}_{>0}$ neighbourhood defined by the local branching operator

$$\Theta(\mathbf{y}, \bar{\mathbf{y}}) := \sum_{e \in E \setminus E_{\text{truck}} | \bar{y}_e = 1} (1 - y_e) + \sum_{e \in E \setminus E_{\text{truck}} | \bar{y}_e = 0} y_e. \tag{8.20}$$

So for a feasible solution of (8.7) – (8.13) $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ and a $\rho \in \mathbb{Z}_{>0}$ the $\rho - OPT$ neighbourhood is:

$$\mathcal{N}(\mathbf{y}, \bar{\mathbf{y}}) := \{(\mathbf{x}, \mathbf{y}) | (\mathbf{x}, \mathbf{y}) \text{ is a feasible solution of the ILP and } \Theta(\mathbf{y}, \bar{\mathbf{y}}) \leq \rho\}. \tag{8.21}$$

9

Variable Reductions

Reducing the number of variables of an ILP may reduce the computation time needed to solve it. For that reason we will look at several ways to remove variables from the MCMCF and MCND ILPs. A simple approach is to somewhat arbitrarily remove arc (or path) variables from one of the models. In this case there is no guarantee that no variables are removed that are used in a optimal solution. So in this chapter we try to introduce variable reductions that do not change the optimal solution value too much. Some of these variable reductions are more effective than others for reducing the computation time in practice. The influence on the computation time can be found in the Chapter Results. The variables are indexed over the locations, time stamps, vehicles and commodities in the arc-based MCND problem. In the path-based problems variables are indexed over the origin-destination paths of the commodities. One way to reduce the number of variables in the model is to decrease the size of these sets of indices. This will result in a reduced number of variables in the model. In Figure 9.1 a table for the compatibility of the variable reductions with the models is presented.

*Variable reduction K does not reduce the number of path variables, but can help to find all the $s_k - t_k$ paths faster.

Table 9.1: Compatibility chart variable reductions

	arc MCMCF	path MCMCF	arc MCND	path MCND
Variable Reduction A	x		x	
Variable Reduction B	x		x	
Variable Reduction C			x	
Variable Reduction D	x	x	x	x
Variable Reduction E	x	x		
Variable Reduction F		x		com. paths only
Variable Reduction G path		x		com. paths only
Variable Reduction G arc	x		x	
Variable Reduction H			x	x
Variable Reduction I	x	x	x	x
Variable Reduction J	x	x	x	x
Variable Reduction K	x	*	x	*
Variable Reduction L	x	x	x	x

9.1. Commodity Reductions

9.1.1. Reduction A: Same Sink/Source

Already all the containers in one booking are combined in one commodity, but it is possible to do more reductions. If multiple bookings have the same sink, then these bookings can be combined in one commodity [7]. (Similarly it is possible to do this if they share the same source, see Figure 9.1 and Figure 9.2)

The problem with combining them if they have different sinks and sources, is that a container of booking 0 can be transported to the sink of booking 1, if they are put in the same commodity. We have seen an example before in Figure 6.3. There the red solution is an optimal solution and the blue infeasible solution is found when the bookings are combined in one commodity.

If two bookings o_0 and o_1 have similar sinks for example if they have to be transported to the same destination location, then the same reduction is possible. If o_0 has to be there one time stamp earlier, then we can set the sink of o_1 to the same sink as that of o_0. Note that by doing this the optimal solution may become worse. After this we combine them in a single commodity.

The algorithm to find the paths of the containers of the bookings, when the values of the arc variables are known from Section 7.1.1 changes slightly with this reduction.

1. Repeat the following steps for all commodities k
2. Repeat the following steps for all bookings i in k
3. Repeat the following steps for all containers j in i
4. Start at the source of booking i
5. Look at all the variables of k that are nonzero in the optimal solution that correspond to the arcs leaving the node
6. Take such an arc, save it and subtract 1 from its value
7. Repeat the previous two steps for the node that the arc goes to, until the sink of the commodity is reached
8. Make a path for container j of booking i from the saved arcs (in the sequence they were saved)

Figure 9.1: Bookings shared source

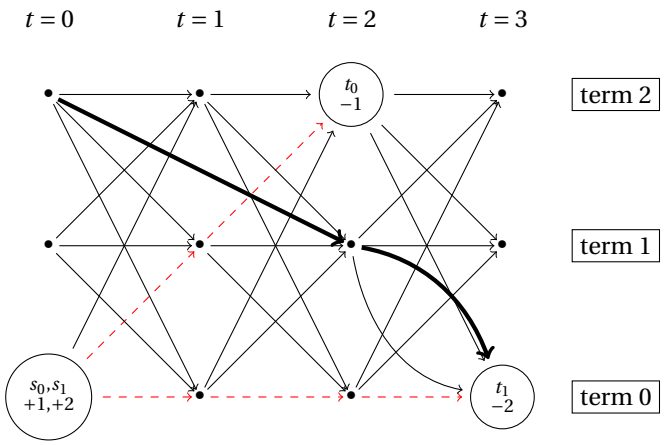
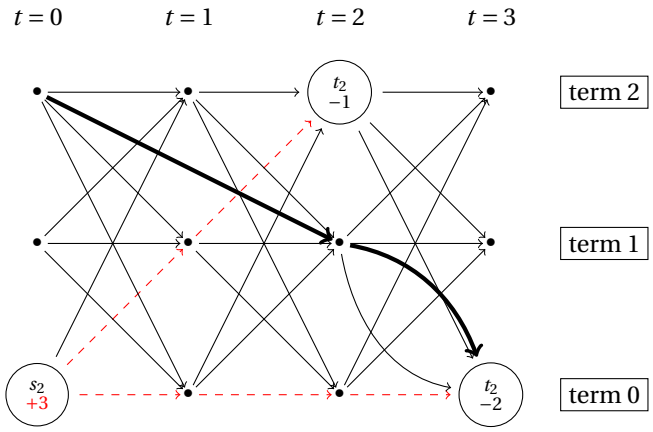


Figure 9.2: Combined bookings shared source



9.1.2. Reduction B: Disjoint Time Frame Bookings

In the implementation of the arc-based ILP for Problem **3** arc variables are defined for a booking for every vehicle arc in the time-space graph. Even those that start at times before a booking is released or past its deadline. Suppose we have two commodities of which one has its deadline before the release time of the other one in this case the bookings can be combined. The will be put together in one commodity. If the release time is before the deadline or if the release time equals the deadline, then there are examples where it is not possible to combine them. Namely, there is a danger that a container is shipped to the sink of a different booking. (blue solution in Figure 6.3)

We can combine bookings in a greedy way: We start with the first one that is released and we add to the same commodity the first booking that is available after its deadline. We repeat this until it is no longer possible to add more bookings to the same commodity. After which we repeat this process for the next commodity. Clearly these bookings in the same commodity will not use the same arcs because there is no time stamp for which they are simultaneously available in the network. Every booking is available during a certain time frame.

Theorem 7. *This greedy algorithm actually finds an optimal way to combine the bookings (minimizing the number of commodities), such that their time frames do not overlap.*

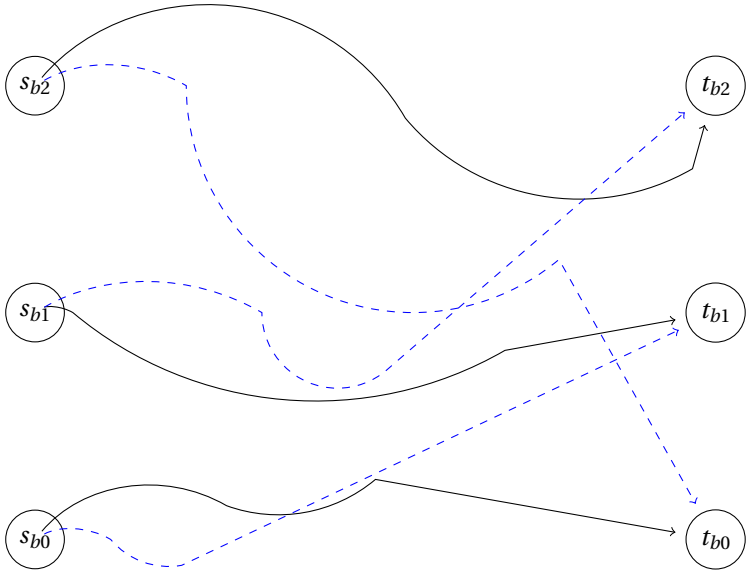
Proof. For readability we assume that none of the release time are equal, but the proof can be extended for cases when there are bookings with equal release times. Suppose we have an allocation that minimizes the number of commodities by combining bookings in a certain way. In that allocation we start with examining the first booking b_{i_1} that is released. If the next (in time) booking in the same commodity b_{i_2} is not the first one released after the first booking, then we swap the first booking released after it; b_{i_3} and everything in the same commodity as b_{i_3} later in time, with b_{i_2} and everything released after b_{i_2} on the same commodity as b_{i_2}. Then we repeat this process for b_{i_3} etc. Until we are done for the commodity and then we move to the first booking released that is not on a commodity that we already handled and do the same for that commodity, but we do not move the bookings that are on a commodity that is already 'done'. It it is clear that the solution remains feasible and the number of commodities that are used does not increase. When we are done with all the commodities, we have found an allocation that is found by the greedy algorithm. □

9.2. Vehicle Reductions

9.2.1. Reduction C: Same Vehicle Type

In the MCND problem we have a set of vehicles $W = \{\text{truck}, \text{barge0}, \text{barge1}, \ldots\}$. The trucks are already combined in the model, it is also possible to combine the barges in the model assuming they all have the same travel times and capacities. So then we get $W = \{\text{truck}, \text{barge}\}$. If there are barges of types A and B we take $W = \{\text{truck}, \text{typeAbarge}, \text{typeBbarge}\}$. In the MCND model the y_e variables are no binary variables anymore,

Figure 9.3: Two path combinations



but more general discrete variables. They keep track of how many barges take arc e. In the capacity constraints these variables are multiplied with the capacities per barge to model the total barge capacity for a certain link.

If the barges are modelled individually, then for every barge a source and sink has to be given. With the reduction, it is fortunately possible to add multiple sources and sinks. So the barges still have the freedom to start from or end at different locations. Though, we can only specify the number of barges that has to arrive at a certain sink, so not which individual barge has to arrive there, if there are multiple sinks. Furthermore if too many sinks and sources are added the number or possible paths for the barges might increase a lot, also increasing the size of the feasible region. For an example see Figure 9.3, in that figure three barges all have different sources and sinks and two of the possible path combinations are drawn.

Even for the standard MCMCF models a vehicle reduction is possible. If multiple barges with the same travel times take the same link then they can be combined in one variable in the MCMCF models.

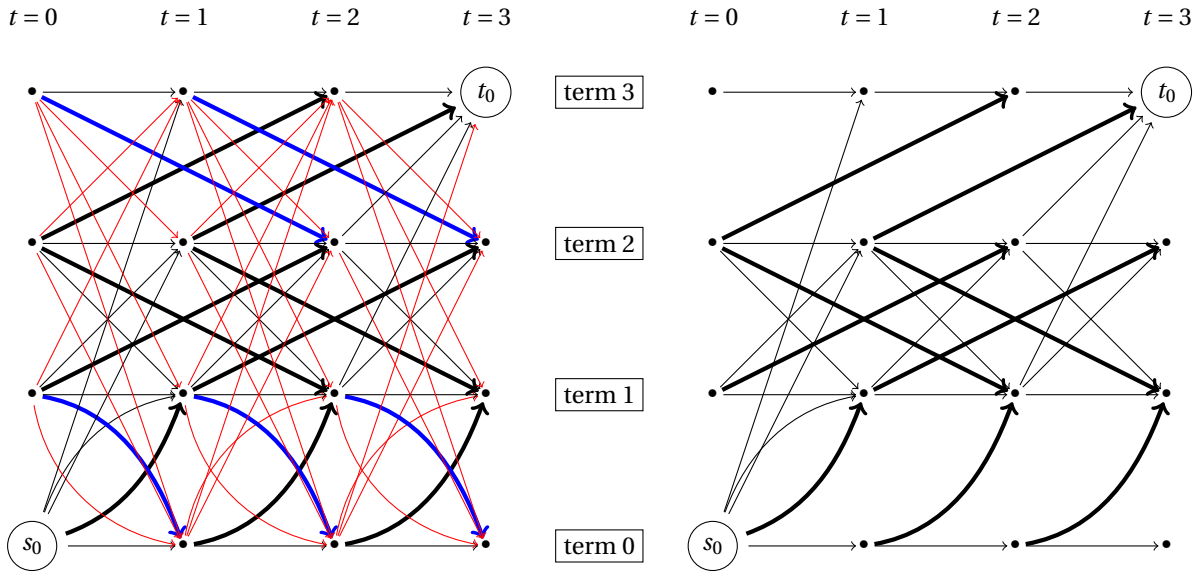
An advantage of bundling different vehicles in the model together into a single vehicle type, is that this a way to avoid problems with symmetry and reduce the number of variables in the model. If the individual vehicles are modelled separately, many different solutions that are equivalent in practice have different variable allocations in the model. For instance, if there are ten identical barges that start at location 0 and one container needs to be moved from location 0 to location 1, then barge 0 can transport the container or barge 1 can transport it etc. As the barges are identical, these solutions are equivalent in practice. If these vehicles are put together in one index w in the model, then the container is not allocated to a specific barge in the model. That happens afterwards.

9.3. Arc Reductions

9.3.1. Reduction D: Source/Sink Location

The number of truck arc variables in MCMCF problem on the time-space graph can be reduced without compromising the optimal solution. This is also done in [45]. We add truck arc variables for every commodity from its source to every location. Furthermore, we add at truck arc variables to every location for every commodity at the end of every non-truck arc. Here it is used that if some part of the route it needs to be trucked. It suffices to do that as soon as its possible to do so. We also use the fact that it is always shorter to truck directly to a location, than through another location.

Figure 9.4: Reduction D for MCND



In the MCND problem on a time-space graph some non-horizontal truck arc variables adjacent to a source location of a booking can be removed. It suffices to only add truck arc variables for a commodity from its origin location to every other location at its release time. The other non-horizontal truck arc variables adjacent to the source location can be removed. Similar things can be done for its sink location, though the arc from the source location to the sink location at the release time is never removed. Additionally, non-truck vehicle arc variables that go to the source location or leave from the sink location can be removed. In Figure 9.4 we apply reduction D. The truck arcs are the thin arcs, the barge arcs the thick arcs. The truck arc variables that are removed for booking 0 are in red and the barge arc variables that are removed in blue. Note: These truck arc variables are only removed for booking 0.

9.3.2. Reduction E: Waiting Arcs

If at a time-space node no non-truck arcs arrive and also no non-truck arcs leave, then we can merge the waiting arc going in and the one going out. In addition we remove the node and its non waiting arcs from the graph.

9.4. Path Reductions

9.4.1. Reduction F: No Repeated Location Paths

The set of paths P_k for a commodity k can potentially have an enormous cardinality in the path-based multi-commodity min cost flow problem. In that case there will be an enormous number of variables. In order to decrease the amount of variables/paths we use theorem 1, that is based on an insight informally discussed in [70]. Basically terminal handling and cost for letting a container stay at a terminal is not taken into account in a model, then it is never a good idea to let a container return to a location before reaching its destination.

Theorem 8. *Consider a time-space graph with zero cost on the horizontal arcs* $[(\tau_1,x),(\tau_2,x),w]$ *and positive cost on the non-horizontal arcs. Let* F_j^k *be a flow path of an optimal solution for some commodity* k *of a MCMCF. Then* $(s,x) \in F_j^k$ *and* $(t,x) \in F_j^k$ *with* $s < t$*, yields* $(s,x),[(s,x),(s+j_1,x),w_0],(s+j_1,x),\ldots,(t-j_q,x),[(t-j_q,x),(t,x),w_q],(t,x) \in F_j^k$*. For certain* $j_1,\ldots,j_q \in \mathbb{Z}_{\geq 1}$*. (horizontal arcs are called waiting arcs)*

Proof. Suppose not. Take such a flow path F_j^k for which it does not hold. Replace the sub path in F_j^k starting with (s,x) ending at (t,x) by $(s,x),[(s,x),(s+j_1,x),w],(s+j_1,x),\ldots,(t-j_q,x),[(t-j_q,x),(t,x),w],(t,x)$ and call

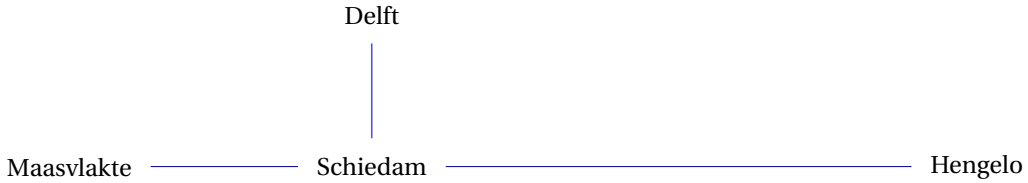
it G_j^k, because the horizontal arcs have zero cost G_j^k has lower cost. If we replace F_j^k by G_j^k in the MCMCF, we get a feasible solution with lower cost. So the original MCMCF was not minimal, contradiction. □

Flow paths that do not satisfy theorem 8 can be ignored. This drastically reduces the amount of variables, considering that flow paths cannot return to a previously visited location.

9.4.2. Reduction G: Minimal Paths

We call a $(loc1, loc2)$ path in a graph minimal if the path has no sub path that is a $(loc1, loc2)$ path. For every commodity k we have that every path that is not a minimal (s_k, t_k) path in the space network can be removed. In the arc-based models we can use that every location that is not on a minimal (s_k, t_k) path in the space network can be removed for commodity k. In Figure 9.5 we see a space network with the waterway connections of several locations. Let $k \in K$ be a commodity with source location $\chi(s_k)$ =Maasvlakte and sink location $\chi(t_k)$ =Hengelo, then we can conclude that location Delft does not have to be added for commodity k, if this reduction is applied.

Figure 9.5: Reduction G example



9.4.3. Reduction H: Idle Time Restriction

Non-truck vehicles are allowed to stay idle at a terminal for as long as they want. If the link cost or other cost for the design variables are not used, then it does not seem beneficial to let a barge stay idle at terminals for a long time. This can be used to reduce the number of variables in the integral path-based MCND problem. Let $\omega \in \mathbb{Z}_{>0}$ be a parameter denoting the maximum total number of time steps that a vehicle of type $w \in W \setminus \{\text{truck}\}$ is allowed to stay idle at terminals. Then for the integral path-based MCND problem all paths for non-truck vehicles with more than ω horizontal arcs are removed from the problem. For the integral arc-based MCND problem additional constraints can be added that impose the same conditions:

$$\sum_{e:=[(s,x),(t,x),w]\in E_w} y_e \leq b_w \omega \qquad \forall\, w \in W \setminus \{\text{truck}\}. \tag{9.1}$$

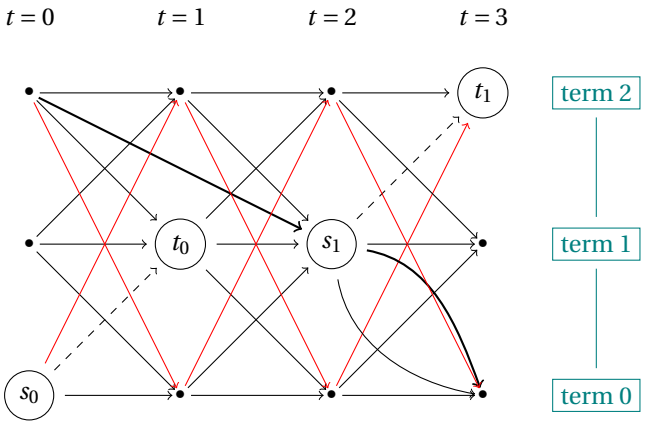
It is also possible to take different values of ω for different vehicle types.

9.5. Location Reductions

9.5.1. Reduction I: Direct Connection

The network of the locations, waterways and roads has a lot of structure. This structure can be used. If shipping from location $loc0$ to location $loc2$ means shipping through location $loc1$. Then no arcs from $loc0$ to $loc2$ have to be added. It suffices to have arcs from $loc0$ to $loc1$ and from $loc1$ to $loc2$. See Figure 9.6 for an example. We recommend taking a dense time grid with this reduction, because larger time steps may adversely affect the accuracy of the travel times between certain locations.

Figure 9.6: Direct connection reduction



9.5.2. Reduction J: Locations In-between

Every commodity k has an origin location $\chi(s_k)$ and a destination location $\chi(t_k)$. Let $d(\chi(s_k), \chi(t_k))$ be the euclidean distance between $\chi(s_k)$ and $\chi(t_k)$, then we set all variables going to or from a location loc with $d(\chi(s_k), loc) > d(\chi(s_k), \chi(t_k)) + \delta$ and/or $d(\chi(t_k), loc) > d(\chi(s_k), \chi(t_k)) + \delta$ to zero. The δ should be chosen large enough to include locations that could be interesting for commodity k. Instead of the distance as the crow flies, it is also possible to use the trucking time for every pair of locations. **Beware: This reduction can cut away optimal solutions.** It is possible namely that first trucking a container further away from your destination, before shipping it to the destination gives the optimal solution.

9.6. Time Reductions

9.6.1. Reduction K: Obsolete Time Reduction

Let $v \in V$ be a time-space node, then we define $\tau(v)$ to be its *time* and $\chi(v)$ its *location*. Clearly a commodity can never take arcs that begin before its origin node time or end after its deadline. Instead of combining bookings in a commodity as in reduction B, it is more effective to remove those obsolete arcs entirely from the model. For every commodity $k \in K$ we remove all variables with $t + \mathrm{a}(loc1, loc2, w)$ bigger than the time of its sink node $\tau(t_k)$, so $t + \mathrm{a}(loc1, loc2, w) > \tau(t_k)$. We also remove all arcs with $t < \tau(s_k)$.

This reduction can even be enhanced some more by also removing variables with:

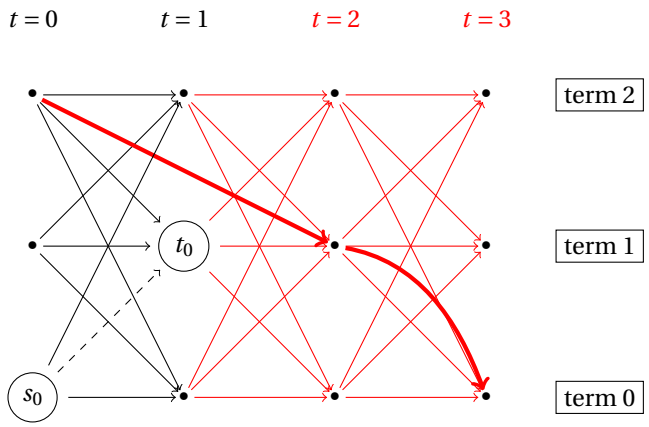
$$t + \mathrm{a}(loc1, loc2, w) = \tau(t_k) \text{ and } loc2 \neq \chi(t_k)$$

or

$$t = \tau(s_k) \text{ and } loc2 \neq \chi(s_k).$$

In Figure 9.7 we see an example of the arcs the basic reduction removes for commodity 0 in red. The dashed line represents the optimal solution.

Figure 9.7: Time reduction



9.6.2. Reduction L: Time Slot Reduction

It is not always possible to go to a terminal. If the available time slots are known, then that information can be used to severely reduce the number of variables in the model. For every arc $e := [(t, loc1), (t+\mathrm{a}(loc1, loc2, w), loc2), w] \in E$ we remove all variables with e as an index if $loc1$ is not available at time t and/or $loc2$ is not available at time $t + \mathrm{a}(loc1, loc2, w)$.

If the set of time stamps T is large. A similar but more drastic reduction for the MCMCF problem, that might influence the solution value a lot is to take a coarser time grid $T' \subset T$. All vehicles that have no fixed schedule in MCMCF problem have duplicates of its arcs for every time stamp. For those vehicles we remove every arc $e := [(t, loc1), (t + \mathrm{a}(loc1, loc2, w), loc2), w] \in E$ leaving at a time stamp in $t \in T \setminus T'$. For other vehicles we transform the arc by taking μ as small as possible, such that $(t - \mu) \in T'$. The new modified arc then becomes $e := [(t - \mu, loc1), (t + \mathrm{a}(loc1, loc2, w), loc2), w] \in E$. Note if $t < \min T'$ then there is no such μ and we remove the arc completely. After this process for every arc $e := [(t, loc1), (t + \mathrm{a}(loc1, loc2, w), loc2), w] \in E$ that ends at a time stamp in $T \setminus T'$ we transform the arc similarly. We take ν as small as possible, such that $(t + \mathrm{a}(loc1, loc2, w), loc2) + \nu) \in T'$. The new modified arc becomes $e := [(t - \mu, loc1), (t + \mathrm{a}(loc1, loc2, w) + \nu, loc2), w] \in E$. If $t > \max T'$ then we remove the arc completely. It is not difficult to see that every solution of this new graph corresponds to a solution in the original graph with the denser time grid.

10

Cutting Planes

The feasible region of an LP may be unnecessary large, which may lead to a large computation time. For instance: There may be many equivalent solutions satisfying the constraints of the LP. Furthermore, for ILPs in general the feasible region defined without the integrality conditions, contains many non-integral solutions. In practice constraints are often added or changed to refine the feasible region making it easier to solve the problem. These constraints are called cutting planes or cuts. Specifically, in this thesis we define cuts as constraints that are added before or during the optimization process, such that the optimal solution value of the problem does not change. An interesting heuristic for the integral MCND problem that uses cuts is discussed in [17].

Table 10.1: Compatibility chart cutting planes

	arc MCMCF	path MCMCF	arc MCND	path MCND
General Cuts	x	x	x	x
Avoiding Symmetry	x	x	x	x
Optimal Solution Structure		x		x
No Repeated Locations (weak)	x		x	
Arc Residual Capacity Cut			x	
Cutset Cut			x	
Strong Cut			x	

In contrary to the (single commodity) min cost flow problem, the MCMCF problem does not always give an integral solution if the demands and capacities are integral. In the experiments performed in this thesis and in [13] it does happen often that the solution that is found is integral or mostly integral. In that paper the flow found in the path-based MCMCF problem is rounded down to find a solution for the integral problem. In practice this means that if it occurs that a non integral solution is found, then for some of the bookings some containers will not be scheduled. It can be decided not to schedule those containers, but another option is to truck them.

We will now show an example, in which there is no integral optimal solution. In Figure 10.1 all arcs have unit capacities and costs and the demand of the commodities is one. Then the optimal solution of the MCMCF problem on this time-space graph is fractional. This solution can be obtained by sending 0.5 flow over the two cost 5 source-sink paths for commodity 1 and the two cost 3 source sink paths for commodity 0 and 2. The optimal solution satisfies the demand for each commodity and has cost $0.5*5+0.5*5+0.5*3+0.5*3+0.5*3+0.5*3=11$. The flow paths of the commodities overlap in some arcs e_i. All these arcs e_i are saturated in the optimal solution and even no other feasible solution is possible in this case. The capacity constraints of the arcs e_i and the constraints that for every commodity the sum of the flow through its flow paths must be one, is visualized in the auxiliary graph (Figure 10.2). Here the nodes correspond with the flow paths.

If there is a feasible integral solution in the time-space graph, then there is a independent set of size $|K|$ in the auxiliary graph. Also each flow path has a certain cost, these costs can be added to the nodes in the auxiliary graph. Then if we want to find an optimal solution that is integral, we want to find a minimum weight size $|K|$ independent set. Note that this set is always a maximum independent set. Finding a maximum independent set is NP-hard [14]. If we remove the integrality conditions we get an LP relaxation, we are left with constraints $x_i + x_j \leq 1$ with $i \in N(j)$ for all j. Now we are able to find the size 3 independent set that corresponds with the optimal solution we found in the time-space graph. It is possible to tighten the LP relaxation by adding the clique constraints, for example $x_{k0f0,k0f1} + x_{k0f1,k1f0} + x_{k0f0,k1f0} \leq 1$. This cuts away some non integral solutions, such as the non integral optimal solution we found before. Therefore it is also called a cut. In our example there is no size 3 independent set, adding this cut leads to the feasible region being empty. For problems with higher capacities and demands these clique cuts generally do not work.

10.1. General Cuts

For ILPs different general cuts exist. The cuts only remove non-integral solutions from the feasible region defined by the constraints of the problem (without integrality constraints). These general cuts are automatically added by solvers like CPLEX [46]. A well-known cut is the Gomory mixed integer cut (GMIC) for ILPs [1, 58]:

$$\sum_{f_j \leq f} f_j x_j + \sum_{f_j > f} \frac{f(1-f_j)}{1-f} x_j \geq f$$

where $f_j = a_j - \lfloor a_j \rfloor$.

A GMIC is often based on an inequality/row of the simplex tableau, but can also be based on other valid inequalities. The cut is stronger than the fractional Gomory cut [1, 58]

$$\sum_{j=1}^{n} (a_j - \lfloor a_j \rfloor) x_j \geq a_0 - \lfloor a_0 \rfloor.$$

A zero-half cut [47] is basically adding two inequalities dividing by two and taking the floor. Example: Suppose we have the inequalities $x_1 + 2x_2 \leq 3$ and $x_1 \leq 2$. Then we can combine them to $2x_1 + 2x_2 \leq 5$. We divide both sides by two $x_1 + x_2 \leq \frac{5}{2}$. Now we take the floor at both sides and get $x_1 + x_2 \leq 2$.

10.2. Avoiding Symmetry

If we have barge 0 and barge 1 in the model, then they correspond to different variables in the model. Suppose barge 0 and barge 1 are identical and they have the same start node and end node. In that case a solution with barge 0 taking path P_{j_1} and barge 1 taking path P_{j_2} is in practice equivalent with a solution in which barge

Figure 10.1: Counterexample

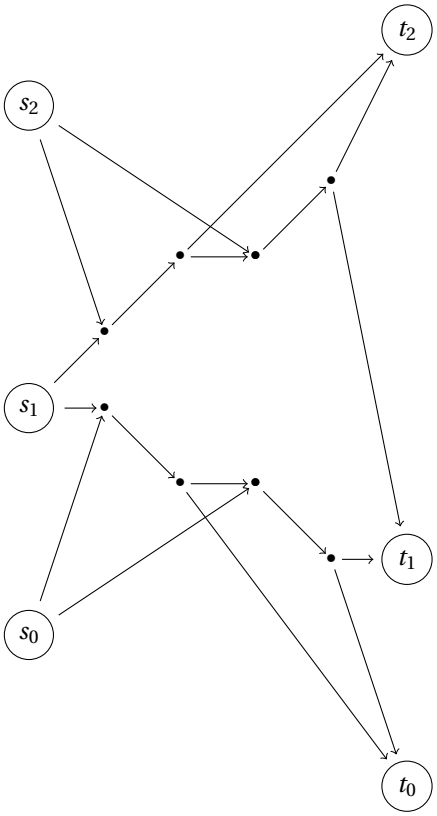
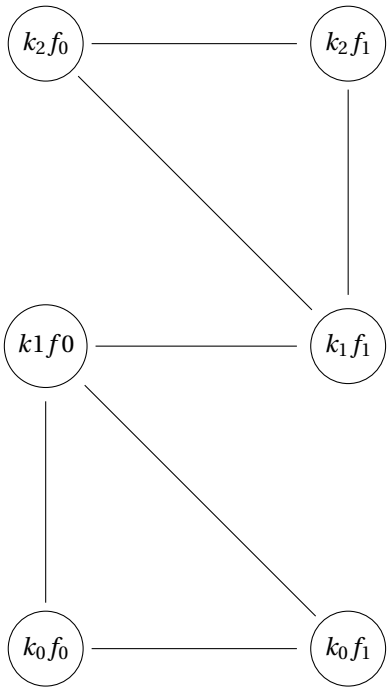


Figure 10.2: Counterexample auxiliary



0 takes path P_{j_2} and barge 1 path P_{j_1} ceteris paribus. The symmetry leads to an unnecessary large feasible region and may result in a lengthy branch & bound process. There are ways to deal with the symmetry. An option is to reformulate the problem. In the case of the models in this thesis applying reduction C is such an approach. Another option is to add symmetry breaking constraints to the model in advance. Adding too many cuts might be detrimental to the computation time of the algorithms, motivating our choice of only adding $|W|-2$ path-length cuts for the arc-based problems, where $W = \{\text{truck}, \text{barge0}, \text{barge1}, \text{barge2}, \dots\}$.

We use the fact that if we sum up all the y variables belonging to barge 0, we get the total path length of barge 0. The chosen numbering of the barges is then used to avoid some symmetry

$$\sum_{e \in E_{\text{barge0}}} y_e \leq \sum_{e \in E_{\text{barge1}}} y_e$$

$$\sum_{e \in E_{\text{barge1}}} y_e \leq \sum_{e \in E_{\text{barge2}}} y_e$$

$$\vdots$$

In these cuts only the length of the paths is calculated, do if in the last example P_{j_1} and P_{j_2} would be paths of the same length, then the cuts will not block this symmetry. If the paths have different lengths, then the cuts will help out. Although normally optimal solution will be removed from the feasible region by adding these cuts, at least one optimal solution will always remain.

For the path-based problems it is relatively easy to avoid symmetry. If $Q^{\text{barge0}} = Q^{\text{barge1}}$, then we number the paths of both barges the same way. Symmetric solutions can then be broken by adding constraints:

$$\sum_i i\zeta_i^{\text{barge0}} \leq \sum_i i\zeta_i^{\text{barge1}}.$$

10.3. Optimal Solution Structure

Theorem 9. *Suppose we have an optimal solution of the integral (path based) MCMCF problem. Then in that optimal solution, for every commodity k it holds that if a s_k-t_k path is used in an optimal solution, then all shorter $s_k - t_k$ paths have at least one saturated arc in the optimal solution. (Here s_k is the source node of commodity k and t_k its sink node)*

Proof. Suppose it does not hold for some path, then a flow of 1 can be moved from that path to a shorter path, decreasing the objective function so the solution was not optimal. □

This property can be made explicit with the constraints of the type

$$0 = \min(\lambda_i^k, \sum_{1 \leq q < i} \min_{e \in P_q^k}(c_e - \sum_{r,l | P_l^r \cap e \neq \emptyset} \lambda_l^r)).$$

Here P_j^k are the paths for commodity $k \in K$, numbered in such a way that the cost of $P_{j_1}^k$ is less or equal than the cost of $P_{j_2}^k$ if $j_1 \leq j_2$. Although these constraints could be added to the path-based MCMCF problem in theory to reduce the feasible region of the problem, these constraints will not be added. This is because it is non-linear and because it introduces numerous new constraints to the problem.

10.4. No Repeated Location Paths (weak)

In the arc-based MCMCF ILP on a time-space graph with horizontal arcs of zero cost, a reduction that is weaker than the one stemming from theorem 8 can be added: For every commodity k on every location

$loc1 \in X \setminus \chi(t_k)$ the combined flow of this commodity going out to a different location must be at most the demand of the commodity d_k and equal to zero for the location of the sink $x(t_k)$;

$$\sum_{\substack{loc2,t,w|[(t,loc1),(t+\mathrm{a}(loc1,loc2,w),loc2),w]\in E,\\ loc2\neq loc1}} x_{loc1,loc2,t,w,k} \leq d_k \qquad \forall k \in K, \forall loc1 \neq \chi(t_k)$$

$$\sum_{\substack{loc2,t,w|[(t,loc1),(t+\mathrm{a}(loc1,loc2,w),loc2),w]\in E,\\ loc2\neq loc1}} x_{loc1,loc2,t,w,k} = 0 \qquad \forall k \in K, loc1 = \chi(t_k).$$

For this cut it is essential that the terminal handling is not taken into account and that it is free for a container to stay idle at a terminal. If that is the case, then it is clearly never beneficial for a container to leave a location and later come back to the same location (before it arrives at its destination). If no container is allowed to come back to a location, then the total number of container that leave any location for a different location is always at most the demand of the commodity. This last notion is modelled with the cut above.

Especially for commodities with a high demand this is a lot weaker than the identically named variable reduction for the path-based ILP, because they are more likely to have many flow paths. One of those flow paths of low value, that corresponds to a small number of containers of a commodity, could return to some location without violating the constraints. A way to avoid this is to model every container as separate commodity with demand one. This will increase the number of variables by a lot, however.

10.5. Arc Residual Capacity Cut

In [64, 66] two cuts for multi-commodity problems are described that can be used for our MCND problem. One of them is the *arc residual capacity cut*:

$$\sum_{k\in K'} x_{e,k} \leq \sum_{k\in K'} d_k - r'(\mu' - y_e), \tag{10.1}$$

with $\mu' = \lceil \frac{\sum_{k\in K'} d_k}{c_e} \rceil$ and $r' = \sum_{k\in K'} d_k - (\mu' - 1)c_e$ for $K' \subseteq K$ for $e \in E \setminus E_{\text{truck}}$. In the MCND model constraints (8.10) hold, so for all $K' \subseteq K$ we have

$$\sum_{k\in K'} x_{e,k} \leq c_e y_e \qquad \forall e \in E \setminus E_{\text{truck}}. \tag{10.2}$$

A time-space graph has no cycles (theorem 1), together with constraints (8.10), (8.12) and (8.8) we know that

$$\sum_{k\in K'} x_{e,k} \leq \sum_{k\in K'} d_k. \tag{10.3}$$

Theorem 10. *Let $e \in E \setminus E_{\text{truck}}$ and $K' \subseteq K$. Suppose y_e satisfies the integrality constraints (8.13) and constraints (10.2), (10.3) are satisfied for e and K' then (10.1) holds for e and K'.*

Proof. Let y_e be such that (8.13),(10.2) and (10.3) hold.
If $y_e \geq \mu'$, then

$$\sum_{k\in K'} x_{e,k} \leq \sum_{k\in K'} d_k \leq \sum_{k\in K'} d_k - r'(\mu' - y_e).$$

So in this case if (10.3) holds, then (10.1) holds.
If $y_e < \mu'$, then by (8.13) we have $y_e = \mu' - s$ where $s \geq 1$. It then follows that

$$\sum_{k\in K'} x_{e,k} \leq c_e y_e = c_e\mu' - c_e s \leq \sum_{k\in K'} d_k - r's,$$

where the last inequality holds, because for $s = 1$ we have equality and

$$\begin{aligned} r' &= \sum_{k\in K'} d_k - (\mu' - 1)c_e = \sum_{k\in K'} d_k - \left(\lceil \sum_{k\in K'} \frac{d_k}{c_e} \rceil - 1 \right) c_e \\ &= \sum_{k\in K'} d_k - \lfloor \sum_{k\in K'} \frac{d_k}{c_e} \rfloor c_e = \sum_{k\in K'} \frac{d_k}{c_e} c_e - \lfloor \sum_{k\in K'} \frac{d_k}{c_e} \rfloor c_e \\ &= \left(\sum_{k\in K'} \frac{d_k}{c_e} - \lfloor \sum_{k\in K'} \frac{d_k}{c_e} \rfloor \right) c_e < c_e. \end{aligned}$$

□

Theorem 11. *We replace $\sum_{k\in K'} d_k$ by $\sum_{k\in K'} d_k + c_e$ in the arc residual capacity cut, then we get*

$$\sum_{k\in K'} x_{e,k} \leq \sum_{k\in K'} d_k + c_e - r''(\mu'' - y_e), \tag{10.4}$$

with $\mu'' = \lceil \frac{\sum_{k\in K'} d_k + c_e}{c_e} \rceil$ and $r'' = \sum_{k\in K'} d_k + c_e - (\mu''-1)c_e$ for $K' \subseteq K$. Then (10.4) is strictly dominated by (10.1) if $\frac{\sum_{k\in K'} d_k}{c_e} \notin \mathbb{Z}$. Furthermore (10.4) is equivalent to (10.1) if $\frac{\sum_{k\in K'} d_k}{c_e} \in \mathbb{Z}$.

Proof. We have

$$\mu'' = \lceil \frac{\sum_{k\in K'} d_k + c_e}{c_e} \rceil = \lceil \frac{\sum_{k\in K'} d_k}{c_e} \rceil + 1 = \mu' + 1$$

and

$$r'' = \sum_{k\in K'} d_k + c_e - (\mu''-1)c_e = \sum_{k\in K'} d_k - (\mu''-2)c_e = \sum_{k\in K'} d_k - (\mu'-1)c_e = r'.$$

Then (10.4) is equivalent to

$$\sum_{k\in K'} x_{e,k} \leq \sum_{k\in K'} d_k + c_e - r'(\mu' + 1 - y_e) = \sum_{k\in K'} d_k - r'(\mu' - y_e) + c_e - r'.$$

In the case $\frac{\sum_{k\in K'} d_k}{c_e} \notin \mathbb{Z}$ we find

$$\begin{aligned} r' = \sum_{k\in K'} d_k - (\mu'-1)c_e &= \sum_{k\in K'} d_k - \left(\lceil \frac{\sum_{k\in K'} d_k}{c_e} \rceil - 1 \right) c_e \\ &< \sum_{k\in K'} d_k - \left(\frac{\sum_{k\in K'} d_k}{c_e} - 1 \right) c_e \\ &= \sum_{k\in K'} d_k - \left(\frac{\sum_{k\in K'} d_k}{c_e} \right) c_e + c_e = c_e. \end{aligned}$$

So with the above we conclude that the arc residual cut (10.1) strictly dominates (10.4):

$$\sum_{k\in K'} x_{e,k} \leq \sum_{k\in K'} d_k - r'(\mu' - y_e) < \sum_{k\in K'} d_k - r'(\mu' - y_e) + c_e - r'.$$

If $\frac{\sum_{k\in K'} d_k}{c_e} \in \mathbb{Z}$, then we get

$$\begin{aligned} r' = \sum_{k\in K'} d_k - (\mu'-1)c_e &= \sum_{k\in K'} d_k - \left(\lceil \frac{\sum_{k\in K'} d_k}{c_e} \rceil - 1 \right) c_e \\ &= \sum_{k\in K'} d_k - \left(\frac{\sum_{k\in K'} d_k}{c_e} - 1 \right) c_e \\ &= \sum_{k\in K'} d_k - \left(\frac{\sum_{k\in K'} d_k}{c_e} \right) c_e + c_e = c_e. \end{aligned}$$

So in that case we conclude that the arc residual cut (10.1) is equivalent to (10.4):

$$\sum_{k\in K'} x_{e,k} \leq \sum_{k\in K'} d_k - r'(\mu' - y_e) = \sum_{k\in K'} d_k - r'(\mu' - y_e) + c_e - r'.$$

□

Corollary 2. *The right-hand side of the arc residual capacity cut (10.1),*

$$\sum_{k\in K'} d_k - \left(\sum_{k\in K'} d_k - (\lceil \frac{\sum_{k\in K'} d_k}{c_e} \rceil - 1)c_e \right) (\lceil \frac{\sum_{k\in K'} d_k}{c_e} \rceil - y_e),$$

is not (monotonically) increasing in $\sum_{k\in K'} d_k$.

Proof. We first write the right-hand side as a function of $\sum_{k\in K'} d_k$

$$g : \mathbb{Z}_{\geq 0} \to \mathbb{R}$$
$$g(\sum_{k\in K'} d_k) = \sum_{k\in K'} d_k - \left(\sum_{k\in K'} d_k - (\lceil \frac{\sum_{k\in K'} d_k}{c_e} \rceil - 1)c_e\right)(\lceil \frac{\sum_{k\in K'} d_k}{c_e} \rceil - y_e).$$

Let $A \in \mathbb{Z}_{\geq 0}$ such that $\frac{A}{c_e} \in \mathbb{Z}$ and define $B := A - s$ for a $s \in \mathbb{Z} \cap [1, c_e - 1]$. We have

$$A - c_e < B < A < B + c_e < A + c_e.$$

Suppose g is monotonically increasing, then we would have

$$g(A - c_e) \geq g(B) \leq g(A) \leq g(B + c_e) \leq g(A + c_e). \tag{10.5}$$

From theorem 11 follows

$$g(A - c_e) = g(A) = g(A + c_e) \tag{10.6}$$
$$g(B) < g(B + c_e). \tag{10.7}$$

From (10.5) and (10.6) follows

$$g(A - c_e) = g(B) = g(A) = g(B + c_e) = g(A + c_e),$$

but this contradicts with (10.7). So we conclude the right-hand side of the arc residual capacity cut is not monotonically increasing in $\sum_{k\in K'} d_k$. □

The structure of the time-space graph can be used to reduce the right-hand side of the arc residual capacity cut (10.1). Although this does not always help to find a stronger cut (corollary 2), it still finds a stronger cuts very often (theorem 11). For an arc $e \in E$ no commodities should be included in the $K' \subseteq K$ for the arc residual capacity cut, that consist of bookings that cannot use that link. If we exclude these commodities, then the left-hand side does not change as $x_{e,k} = 0$ if commodity k cannot use arc e. The right-hand side is likely to decrease (theorem 11), so if this is the case the cut will become stronger. Examples of arcs that are not used by commodities can be found in Section 9.6 Time Reductions.

10.6. Cutset Cut

For every truck arc $e \in E_{\text{truck}}$ we add the variable $y_e \in \mathbb{Z}_{\geq 0}$ and the constraint

$$\sum_{k\in K} x_{e,k} = y_e \qquad \forall e \in E_{\text{truck}}. \tag{10.8}$$

The second cut suitable for the MCND problem described in [64, 66] is the cutset cut. Let $k \in K$ be a commodity that consists of one booking. Then it has a source node s_k. For the node we can add the cutset cut

$$\sum_{e\in\delta^+(s_k)} y_e \geq 1. \tag{10.9}$$

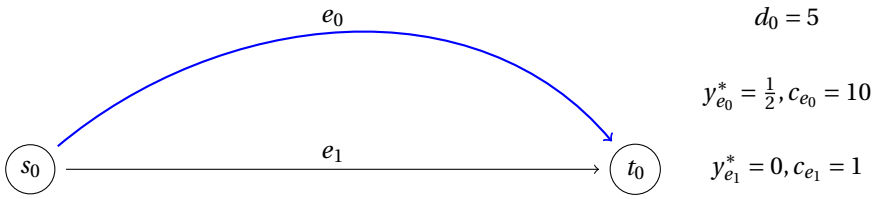
In Figure 10.3 an example is visualized, where only half a barge is deployed to transport the container of a commodity at the start. In scenarios like that the cutset cut can be used to avoid the fractional solution that is found for the LP relaxation.
Constraints (8.10) imply

$$x_{e,k} \leq \sum_{k_1\in K} x_{e,k_1} \leq y_e c_e \qquad \forall e \in E, \forall k \in K.$$

Consequently

$$\sum_{e\in\delta^+(s_k)} y_e c_e \geq \sum_{e\in\delta^+(s_k)} x_{e,k} \qquad \forall k \in K$$

Figure 10.3: cutset cut example and LP relaxation solution y^*



Constraints (8.8) and (8.12) imply

$$\sum_{e \in \delta^+(s_k)} x_{e,k} = d_k \qquad \forall e \in E, \forall k \in K.$$

If we substitute that and use the maximum arc capacity $c_{\max} := \max_{e \in E} c_e$ we get

$$c_{\max} \sum_{e \in \delta^+(s_k)} y_e \geq \sum_{e \in \delta^+(s_k)} y_e c_e \geq d_k \qquad \forall k \in K$$

or equivalently

$$\sum_{e \in \delta^+(s_k)} y_e \geq \frac{d_k}{c_{\max}} \qquad \forall k \in K.$$

Constraints (8.12), (8.13) imply

$$y_e \in \mathbb{Z}_{\geq 0} \qquad \forall e \in E.$$

We can use this to get the enhanced cutset cut

$$\sum_{e \in \delta^+(s_k)} y_e \geq \lceil \frac{d_k}{c_{\max}} \rceil \qquad \forall k \in K. \tag{10.10}$$

The demand d_k and the maximum capacity $c_{\max}$ are always integral so (10.10) is stronger than (10.9). This cut can be generalized.

Let $G = (V, E)$ be a directed graph. Suppose we have a subset of the nodes $S \subseteq V$. We denote the complement of S as $\overline{S} := V \setminus S$. The partition of the nodes $C := (S, \overline{S})$ is called a *cut*, to avoid ambiguity we will call this a *graph cut* in this thesis. Then $\delta(S, \overline{S}) = \{(i, j) \in E | i \in S, j \in \overline{S}\}$ is called the *cutset* of the graph cut $(S, \overline{S})$. The cuts in this section are named after it.

This allows us to define the general cutset cut

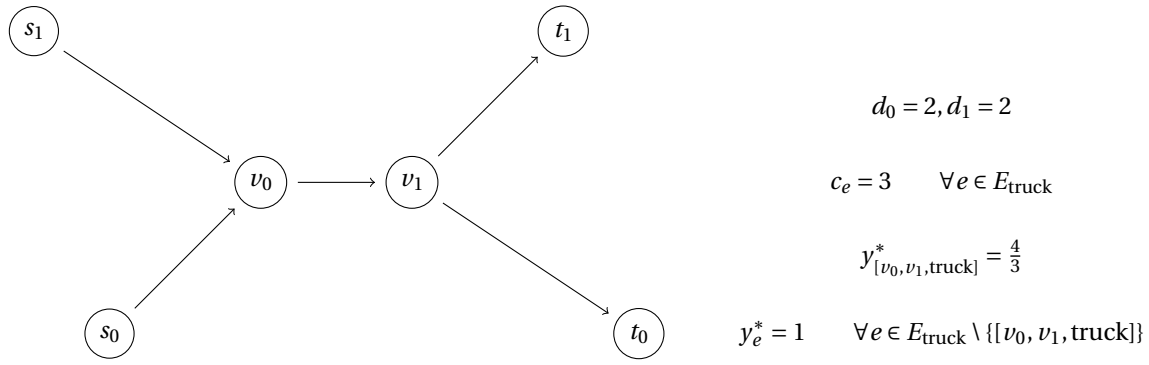
$$\sum_{e \in \delta(S, \overline{S})} y_e \geq \lceil \frac{\sum_{k | s_k \in S, t_k \notin S} d_k}{c_{\max}} \rceil \qquad \forall S \subseteq V. \tag{10.11}$$

If we take $S = \{s_{k_1}\}$ for a $k_1 \in K$, then we get an enhanced cutset cut back (10.10) (if there is no $k_2 \in K$ with $k_2 \neq k_1$ and $s_{k_1} = s_{k_2}$). If there are multiple commodities with the same source, then sufficient capacity needs to be installed to handle the sum of their demands. So then (10.11) is stronger than (10.10). Suppose we have a graph cut $(S, \overline{S})$, then one could even say that sufficient capacity needs to be installed on the arcs of $\delta(S, \overline{S})$ to be able to transport at least the sum of the demands of the commodities that have their source in S and sink in $\overline{S}$. Every container of those commodities has to be transported from a node in S to a node in $\overline{S}$. Consequently, every such container is transported by at least one arc in $\delta(S, \overline{S})$. This reasoning gives some intuition why (10.10) can be generalized to (10.11). If the design variables y_e are binary variables, then we will use the flow cover cuts

$$\sum_{e \in Q} y_e \geq 1 \qquad \forall Q \subseteq \delta(S, \overline{S}) \text{ with } \sum_{e \in Q} c_e < \sum_{k | s_k \in S, t_k \notin S} d_k, \forall S \subseteq V. \tag{10.12}$$

There are a huge number of node subsets $S \subseteq V$. So adding all cuts (10.11) is in general impractical. We can add some general cutset cuts during the branch & cut process. Finding violated cuts is called the *separation*

Figure 10.4: Undiscovered violated cutset cut example



problem. An $s_k - t_k$ *cut* for a $k \in K$ is a graph cut $C = (S, \overline{S})$ with $s_{k_1} \in S$ and $t_{k_1} \in \overline{S}$. Suppose we have a non-integral solution (x^*, y^*) found in one of the nodes of the branch & cut tree. To find violated cuts we take the following steps:

1. For all commodities $k_1 \in K$ repeat the following steps
2. Set $S := \{s_{k_1}\}$ and $T := \{t_{k_1}\}$
3. Put the values y^*_e as capacity on the arcs of the time-space graph
4. Find a minimal $s_{k_1} - t_{k_1}$ cut by solving the the max flow problem[19] on the time-space graph with s_{k_1} the source and t_{k_1} the sink (according to the max flow - min cut theorem[19])
5. Check for the S constructed if the corresponding cut (10.11) is violated, if so add it to the ILP

This way at a node in the branch & cut tree at most $|K|$ max flow problems need to be solved. The max flow problem is similar to the min cost flow problem.

Although the method can discover many violated cutset cuts, there are also some it cannot discover. The step-by-step plan above gives two minimum graph cuts in the graph of Figure 10.4 with five truck arcs. The graph cut with $S = \{s_0\}$ and the graph cut with $S = \{s_1\}$. The corresponding general cutset cuts are not violated, because

$$y_{[s_1,v_0,\text{truck}]} \geq 1,$$
$$y_{[s_0,v_0,\text{truck}]} \geq 1$$

hold (with equality). Although the general cutset cut with $S = \{s_0, s_1, v_0\}$,

$$y_{[v_0,v_1,\text{truck}]} \geq 2,$$

is violated.

Repeatedly solving max flow problems might take too much time in practice. So better options may be to add cuts (10.10) to the model. This can be done in advance or if they are violated during the branch & cut. Other useful cuts may be (10.11) for $S := \{s_k, (\tau(s_k)+1, \chi(s_k)), \ldots, (\tau(s_k)+r, \chi(s_k))\}$ for all $k \in K$ for some small $r \in \mathbb{Z}_{\geq 0}$. If S and $\bar{S}$ are both large sets then the cut is normally not effective, because in that case there is a high chance that too many vehicles go from nodes in S to nodes in $\bar{S}$ rendering the corresponding cut (10.11) useless.

10.7. Strong cut

Strong cuts from [17] are defined as:

$$x_{e,k} \leq d_k y_e \qquad \forall e \in E, \forall k \in K. \tag{10.13}$$

These are only useful if all containers of a commodity are going through an arc $e \in E \setminus E_{\text{truck}}$. Then the corresponding strong cut ensures that at least one vehicle should be used for that link. These cuts are less effective,

when used in combination with vehicle and commodity reductions. With Reduction C: Same Vehicle Type (Subsection 9.2.1) the design variables are non binary making the cuts weaker. In case commodity reductions are used and/or commodities have high demands, then the chance is smaller that all containers of the commodity take the same arc. So then the cut is also less effective.

We have general integral design variables when vehicle reductions are used. In that case, we can make more effective cuts. If $x_{e,k} = d_k$, then we want to have that $y_e \geq \lceil \frac{d_k}{c_e} \rceil$. This means that we need to have $\lceil \frac{d_k}{c_e} \rceil d_k$ for the LHS (left-hand side). Furthermore, if $x_{e,k} = \lfloor \frac{d_k}{c_e} \rfloor c_e$, then we want to have $y_e \geq \lfloor \frac{d_k}{c_e} \rfloor$; so that means $\lfloor \frac{d_k}{c_e} \rfloor d_k$ for the LHS. Suppose $\frac{d_k}{c_e} \notin \mathbb{Z}$. Then we can define a linear function for the LHS that goes through the two points described:

$$ax_{e,k} + b \leq d_k y_e \qquad \forall e \in E, \forall k \in K, \tag{10.14}$$

with

$$a = \frac{\lceil \frac{d_k}{c_e} \rceil d_k - \lfloor \frac{d_k}{c_e} \rfloor d_k}{d_k - \lfloor \frac{d_k}{c_e} \rfloor c_e} = \frac{d_k}{d_k - \lfloor \frac{d_k}{c_e} \rfloor c_e}$$

and

$$b = \lceil \frac{d_k}{c_e} \rceil d_k - a d_k.$$

We divide by d_k to get:

$$ax_{e,k} + b \leq y_e \qquad \forall e \in E, \forall k \in K, \tag{10.15}$$

with

$$a = \frac{1}{d_k - \lfloor \frac{d_k}{c_e} \rfloor c_e}$$

and

$$b = \lceil \frac{d_k}{c_e} \rceil - a d_k.$$

For the substitution we use $\mu' = \lceil \frac{d_k}{c_e} \rceil$ and $r' = d_k - (\mu' - 1)c_e$ and we simplify to get:

$$x_{e,k} \leq d_k - r'(\mu' - y_e) \qquad \forall e \in E, \forall k \in K. \tag{10.16}$$

Theses are arc residual capacity cuts. This has a few implications. We do not have to show that this enhanced cut does not cut away integral solutions, because that is already shown for arc residual capacity cuts. It is not necessary to add strong cuts, because if they are effective, $x_{e,k} = d_k$, then the arc residual capacity cut forces the same bound for y_e and the arc residual capacity cut can be even stronger for integral design variables. These arc residual capacity cuts seems to be useful if $x_{e,k} \in \{d_k, \lfloor \frac{d_k}{c_e} \rfloor c_e\}$, but for most other values of $x_{e,k}$ it gives a non-integral lower bound for y_e. For the interested reader we refer to [17] for another cut; the flow pack cut.

11

Additional Features

There are various additional features that can be added to the different models of this thesis. We will look into some that can be useful for the implementation in practice.

Table 11.1: Compatibility chart additional features

	arc MCMCF	path MCMCF	arc MCND	path MCND
Link Cost			x	x
Due Time	x	x	x	x
Container Refusal	x	x	x	x
Terminal Handling Time (paragraph 1)	x	x	x	x
Terminal Handling Time (remainder)	x		x	
Container Sizes	x	x	x	x
Booking Bias	x	x	x	x
Vehicle Usage Penalty			x	x
Location Constraints (main)	x	x	x	x
Location Constraints (last paragraph)	x	x	x	x
Customers	x	x	x	x
Minimal Capacity Usage	x		x	
Fixed Paths	x	x	x	x
Unknown Vehicle End Location	x	x	x	x

11.1. Link Cost

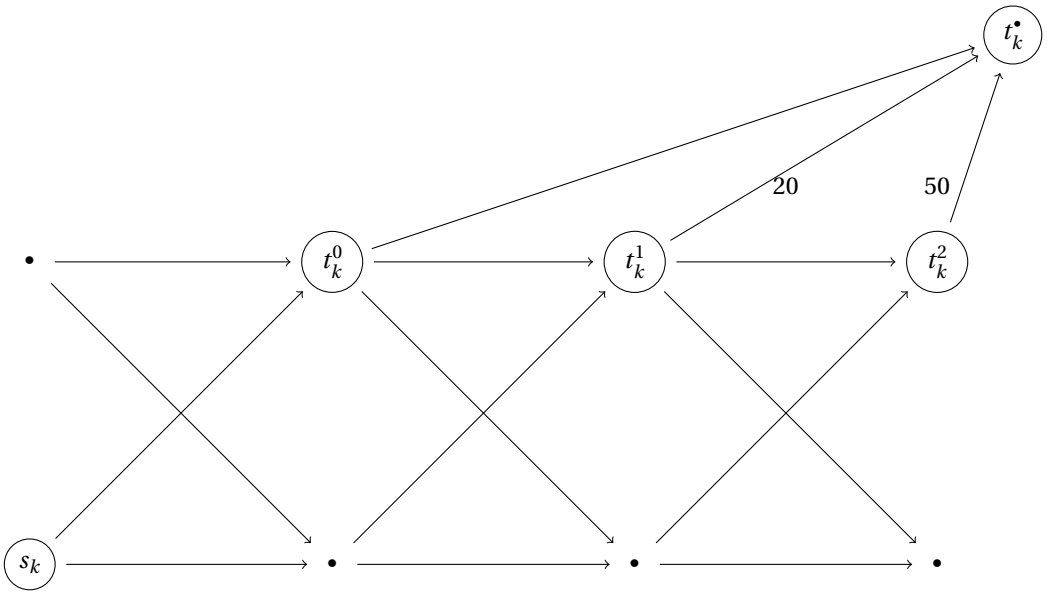
The objective function in the models has cost dependent on the number of containers that are being transported with a certain arc. In the MCND problem more terms can be added to the objective function. A cost can be added, that is dependent on the links that a barge travels on its route. For instance, cost can be added if a barge is idle at a certain terminal. The added terms $\sum_{e \in E} g_e y_e$ are very similar to the terms for the x variables already in the objective function.

11.2. Due Time

Instead of deadlines there are often due times in practice. Violating the due time is allowed, but in that case some penalty has to be paid. The theses [70] and [45] both have a different approach for the inclusion of due times. The approach from [45] does not introduce directed cycles to the time-space graph, so we will use that.

For every commodity k with a due date instead of a deadline, a virtual sink $t_k^\bullet$ is added. This node is connected with all the possible sink nodes for this commodity. Arriving at some sink nodes may incur a penalty. That penalty can be put as cost on the arc connecting the sink node and the virtual sink. In Figure 11.1 penalties are incorporated with costs 20 and 50 on arcs. The virtual sink becomes the new sink in the model for commodity k in the model. The ILP is modified to reflect the changes in the graph.

Figure 11.1: Due time example



11.3. Container/Booking Refusal

In practice there can be containers or whole bookings that are very expensive to transport or that are difficult to deliver in time. Then it can be decided to refuse those containers. This concept is applied on a routing problem in [13]. We can implement it on our arc-based and path-based models. We change constraint (7.12) for the source and sink nodes. For the sources s_k we instead put in the model

$$\sum_{e \in \delta^+(s_k)} x_{e,k} - \sum_{e \in \delta^-(s_k)} x_{e,k} \leq d_{s_k,k} \qquad \forall k \in K$$

and

$$\sum_{e \in \delta^+(s_k)} x_{e,k} - \sum_{e \in \delta^-(s_k)} x_{e,k} \geq 0 \qquad \forall k \in K.$$

For the sinks we replace it by

$$\sum_{e \in \delta^+(s_k)} x_{e,k} - \sum_{e \in \delta^-(s_k)} x_{e,k} \geq d_{s_k,k} \qquad \forall k \in K$$

and

$$\sum_{e \in \delta^+(s_k)} x_{e,k} - \sum_{e \in \delta^-(s_k)} x_{e,k} \leq 0 \qquad \forall k \in K.$$

This is does not suffice. With only these constraints no containers will be transported, because of the transportation cost. For this reason a penalty will be added if a container is not transported. For every commodity k the term

$$+h_k(d_k - \sum_{e \in \delta^+(s_k)} x_{e,k})$$

is added to the objective function. (Note that with the arc e we mean the four indices $loc1, loc2, t, w$ that specify the arc and remember $\delta^+(s_k)$ are all the arcs leaving s_k.) A good value for h_k would be the profit that is gained from transporting one container of commodity k.

For the path-based problems we modify the ILP in a similar way. The demand constraint (7.35) is replaced by

$$\sum_j \lambda_j^k \leq d_k$$

and

$$\sum_j \lambda_j^k \geq 0.$$

Again we add a penalty term for not delivering a container:

$$+h_k(d_k - \sum_j \lambda_j^k).$$

These modifications to the ILPs are useful, but make to feasible region a lot larger because also solutions in which containers are not delivered are allowed.

11.4. Terminal Handling Time and Cost

The *terminal handling time and cost* can be added to the models in multiple ways. For all terminal locations q_1, q_2 in the time-space graph with a (un)loading times r_1, r_2 and costs u_1, u_2 respectively, we add that terminal handling time to the model by the following process: We replace all arcs in the time-space graph of the form $[(s,q_1),(t,q_2),w]$ with $s < t$ and $q_1 \neq q_2$ by $[(s-r_1,q_1),(t+r_2,q_2),w]$. This approach is also useful for finding more robust solutions. The terminal handling cost is modelled by increasing the cost with a fixed terminal handling cost. The new cost function f_1 is defined by $f_1([(s-r_1,q_1),(t+r_2,q_2),w]) := f([(s,q_1),(t,q_2),w]) + u_1 + u_2$. This means that every time a barge arrives at a terminal a terminal handling time is taken into account, that unfortunately is not contingent on the number of containers that need to be (un)loaded at that time-space node. It is possible to make the terminal handling time and cost depend on the time of the day, vehicle type and terminal location. It is simple to implement and does not require additional constraints or variables in the ILP. A barge will normally not go to a terminal where it does not have to (un)load anything. Only with location reductions barges may for example be forced to go to locations where they do not load or unload. Therefore location reductions are incompatible with this approach.

For every booking it is obligatory to load a container at the origin location and unload at the destination location. We assume the loading time and cost does not rely on w. Consequently, the loading at the start location and unloading at the destination location, do not have to and will not be taken into account in the following approaches.

There is also another way to implement the unloading time and cost that does take into account the number of containers. For the arc-based MCND problem we can add additional constraints. For every time-space node v for every commodity k we have a flow conservation constraint already in the model;

$$\sum_{e \in \delta^+((t,loc))} x_{e,k} - \sum_{e \in \delta^-((t,loc))} x_{e,k} = d_{(t,loc),k} \text{ for all } k \in K, (t,loc) \in V.$$

For all time-space nodes with $T \ni t \geq 1$ for all commodities $k \in K$ for all vehicle (types) $w \in W$ we can add some more constraints

$$\sum_{e \in \delta_w^+((t,loc))} x_{e,k} - \sum_{e \in \delta_w^-((t,loc)) \setminus [(t-1,loc),(t,loc),w]} x_{e,k} \geq d_{(t,loc),k}.$$

Here $\delta_w^+(v)$ are all the arcs leaving time-space node v of vehicle (type) w and $\delta_w^-(v)$ all the arcs entering time-space node v of vehicle (type) w. These constraints make sure that if a container wants to switch from $w_1 \in W$ to $w_2 \in W$ it has to wait at the terminal for one time step. If in the set W all the barges are modelled as individual elements, then this constraint also forces the barge to wait one time step if a container of his wants to go to a different element of W. Aside from that, if a barge does not unload any containers but does load some containers, no loading time is taken into account. If they are not modelled individually, it is still possible for a container to go to a barge with the same w in the model without terminal handling time. In that case it can be decided to instead put the constraint

$$x_{[(t,loc),(t+1,loc),w],k} - \sum_{e \in \delta_w^-((t,loc)) \setminus [(t-1,loc),(t,loc),w]} x_{e,k} \geq d_{(t,loc),k}$$

in. This forces forces every container and barge/truck to have terminal handling at every location the barge arrives. On the horizontal arcs we put the terminal handling cost for the terminal concerned. This can be done per vehicle by putting the cost on the horizontal y variables or per container by putting it on the x variables. Additional horizontal waiting arcs are added to the graph and put in the model with w = waiting instead on w = truck. This way it is still possible for a container to stay at a terminal for a longer time without continuously paying the terminal handling cost. This last-mentioned constraint also forces terminal handling if it

goes to a location where nothing is (un)loaded, so this is also not compatible with location reductions. These approaches only introduce a terminal handling time of one time step, adding more and longer horizontal arcs allows longer terminal handling time but introduces too many variables to be used in practice.

There are also ways to only include the terminal handling cost in the models. In [79] we find transshipment constraints

$$\sum_{e \in \delta^-_{w_1}(v)} x_{e,k} = \sum_{w_2 \in W} q_{v,w_1,w_2,k} \qquad \forall k \in K \forall w_1 \in W \forall v \in V \setminus \{t_k\},$$

$$\sum_{e \in \delta^+_{w_1}(v)} x_{e,k} = \sum_{w_2 \in W} q_{v,w_2,w_1,k} \qquad \forall k \in K \forall w_1 \in W \forall v \in V \setminus \{s_k\}$$

and

$$\sum_{w_2 \in W} q_{v,w_1,w_2,k} = 0 \qquad \forall k \in K \forall w_1 \in W \forall v \in \{s_k, t_k\}.$$

These constraints are added to the models and the new variables $q_{v,w_1,w_2,k}$ represent the transshipment of containers of commodity k from vehicle (type) w_1 to w_2 at time-space node v. The terminal handling time can then be obtained by putting

$$+ \sum_{v,w_1,k} \sum_{w_2 \neq w_1} g_{v,w_1,w_2,k} q_{v,w_1,w_2,k}$$

in the objective function. This way staying in the same w is not penalized. Here g represents the per container cost for the handling.

The terminal handling cost can also be added to the model a bit differently. We can also add

$$\sum_{e \in \delta^-_{w_1}(v)} x_{e,k} - \sum_{e \in \delta^+_{w_1}(v)} x_{e,k} \leq q_{v,w,k} \qquad \forall k \in K \forall w \in W \forall v \in V \setminus \{t_k, s_k\},$$

$$\sum_{e \in \delta^+_{w_1}(v)} x_{e,k} - \sum_{e \in \delta^-_{w_1}(v)} x_{e,k} \leq q_{v,w,k} \qquad \forall k \in K \forall w \in W \forall v \in V \setminus \{t_k, s_k\}.$$

After we add those constraints and variables we add

$$+ \sum_{v,w,k} g(v,w,k) q_{v,w,k}$$

to the objective function. In this case $q_{v,w,k}$ is the number of containers of commodity k that switching from w to some other vehicle (type) at time-space node v and g is similar as before.

11.5. Container Sizes

Most containers are of the standardized 1 TEU = 20 feet or 2 TEU = 40 feet size. So normally containers of 1 TEU can be modelled by linking them to another 1 TEU container in the same booking. After which they are seen as one 2 TEU container in the model. If none other exist an additional 1 TEU container can be created. This method has some limitations, though. The linked containers, namely, have to use the same path. Furthermore, although around 95% [9] of the containers is 1 TEU or 2 TEU, there are still containers of different sizes. We can model containers of other sizes by putting them in separate commodities. Each commodity can at most contain one type of container. We assume that trucks can only transport one container at a time. Then we only need to make some changes for the barges. Suppose we have a barge that can transport 156 TEU = 78 containers of 2 TEU. Then we can model the other containers by adding weights w_k to the capacity constraints for a barge arc.

$$\sum_k w_k x_{e,k} \leq c_{e,w} \qquad \forall e \in E \qquad \text{for arc MCMCF}$$

$$\sum_k w_k x_{e,k} \leq c_{e,w} y_e \qquad \forall e \in E \qquad \text{for arc MCND}$$

We calibrate the capacity constraints to the size of a 2 TEU container to make it identical to the models discussed in this paper initially. So in our example the barge arc e gets a capacity of $c_e := 78$. We have

$$\sum_{k \in K} w_k x_{e,k} \leq 78 \qquad \text{for arc MCMCF}$$

$$\sum_{k \in K} w_k x_{e,k} \leq 78 y_e \qquad \text{for arc MCND,}$$

here the weights are w_k. If we would set $w_k = 1$ for all the $k \in K$, we would get our standard capacity constraint (7.3) back. We set $w_k = 1$ for containers of 2 TEU. For the commodities in general we set $w_k := \frac{O_k}{2}$, where O_k is the size of the containers in commodity k expressed in TEU. So for the common 1 TEU container we get $w_k = \frac{1}{2}$. If we put this in ILP, then the Dantzig-Wolfe decomposition can actually help in some cases to reduce the integrality gap, see [42].

11.6. Booking Bias

The models find optimal solutions in which the total cost is minimized. However, it does occur that the route for some individual booking (that is in commodity k) is extremely expensive. If the LSP charges with regard to the cost in the solution, then the customer paying for the booking might become dissatisfied. A better solution for the booking concerned can be found by multiplying the cost of the variables of commodity k with $\mu_k > 1$. So we for commodity k we set $f_{e,k} := \mu_k f_{e,k}$. If there are also other bookings in commodity k as result of a commodity reduction, then these other bookings should first be put in a separate commodity. If we modify the objective function this way, then the optimal solution value is not the cost anymore. Though, the cost can be calculated afterwards. It is likely that the booking has a relatively low demand, otherwise its cost would have spiked the total cost in the optimal solution. For that reason it is less likely to be a problem if we give it a very high priority; μ_k.

11.7. Vehicle Usage Penalty

It is useful to determine the number of barges that need to be purchased and the number of barges that need to be deployed on a certain day. Both can be taken into account. Suppose reduction C is used and each barge of type A is modelled by the same $w \in W$. Then in the original model parameter b_w would be the number of barges that will be used. Instead of setting a fixed value b_w, we let b_w be a variable in the model with

$$\beta_w \leq b_w \leq \alpha_w \tag{11.1}$$

$$b_w \in \mathbb{Z}_{\geq 0}, \tag{11.2}$$

where α_w is a parameter denoting the upper bound for the number of vehicles of type w and β_w a lower bound.

Another option is to modify the model with the constraints (8.6) and add arcs for the sinks of the vehicles t_w to the sources of the vehicles s_w. Bounds on the number of vehicles of type w can be added by adding:

$$\beta_w \leq y_{[t_w, s_w, w]} \leq \alpha_w. \tag{11.3}$$

If we want to know the number of barges that need to be deployed, then we set α_w to the number of available barges, for the other purpose it needs to be set higher. In order to optimize the number of barges, a barge usage cost needs to be added to the model, an option is the link cost from the beginning of this chapter. It is also possible to add some base cost to the model for using a barge. This can be modelled by adding

$$+ g_w b_w \qquad \forall w \in W \setminus \{\text{truck}\},$$

where g_w is the base cost of using a barge (belonging to variable w) to the objective function. If we allow the number of barges to be optimized, then the feasible region will become a lot bigger. This is because also solutions in which fewer and/or more barges are used, are allowed. This feature is not suitable for the trucks, because they are included in the model in a different way.

11.8. Location Constraints

We can add an extra layer to the models to know where to build terminals or which terminals to use (in case there is a long term contract to be able to use a terminal). Let X be our set of possible terminal locations or

terminal locations, depending on if we attempt to model the first or second feature. We define E_{loc} for $loc \in X$ as all arcs that leave from or go to location x. Then for the MCMCF we get

$$\sum_{e \in E_{loc}} x_{e,k} \leq M\beta_{loc} \qquad \forall loc \in X \forall k \in K$$

and for the MCND

$$\sum_{e \in E_{loc} \cap E_{\text{truck}}} x_{e,k} \leq M\beta_{loc} \qquad \forall loc \in X \forall k \in K,$$

$$\sum_{e \in E_{loc}} y_e \leq M\beta_{loc} \qquad \forall loc \in X \forall k \in K,$$

where M is a very large number and β_{loc} is a binary variable modeling if the terminal is opened/used. We add to the objective function

$$+g_{loc}\beta_{loc},$$

where g_{loc} is the price for opening/using the location. If a model with this layer is tested out on multiple instances, then it can be used to conclude where terminals should be opened.

It may be that within a certain time frame only v trips to and from a terminal $term$ are allowed. This is modelled with the constraints

$$\sum_{e \in E_{term} | (t,term) \notin e \forall t \in T_{term}} y_e \leq v,$$

where T_{term} is the set of time stamps that belong to the prespecified time frame. A use of this feature has been shown in an earlier chapter, see (7.16).

11.9. Customers

There are different ways to add customers to the models discussed in this thesis. In models we can add customer locations to the model, the same way terminal locations are added. Of course one should take into account that it is not allowed to have barge arcs going to locations that are not accessible by barge. Also a booking that starts at a customer location without water connectivity is always trucked if the distance to the closest terminal is bigger or about the same as the distance to its destination. This method will introduce many new variables and constraints on time-space graphs. A better option is to add customers nodes for every booking. If the start location of a booking is a customer, then we add one additional customer node. This becomes the source node of the commodity. If the end location of a booking is a customer, then we also add one additional node. This becomes the sink node of the commodity. If the start and end location of a booking are customers, then we add both additional nodes. We connect those additional nodes with truck arcs to all[1] terminal locations and take into account the time it takes to get to the terminal locations. In specific cases customers may be accessible by barge, then we additionally can add barge arcs to and from the customer node. We also add a truck arc from the source directly to the sink. This last arc need not be added if the source or/and the sink is a terminal location, but we are interested to see if it reduces the computation time (see Figure 11.2, Figures 11.3, Figure 11.4). We also add arc variables and add and modify flow conservation constraints to model the modifications in the graph.

[1] Instead of all terminal locations, a subset of the terminal locations can be chosen. Optimality can then normally not be guaranteed however.

Figure 11.2: Customer source

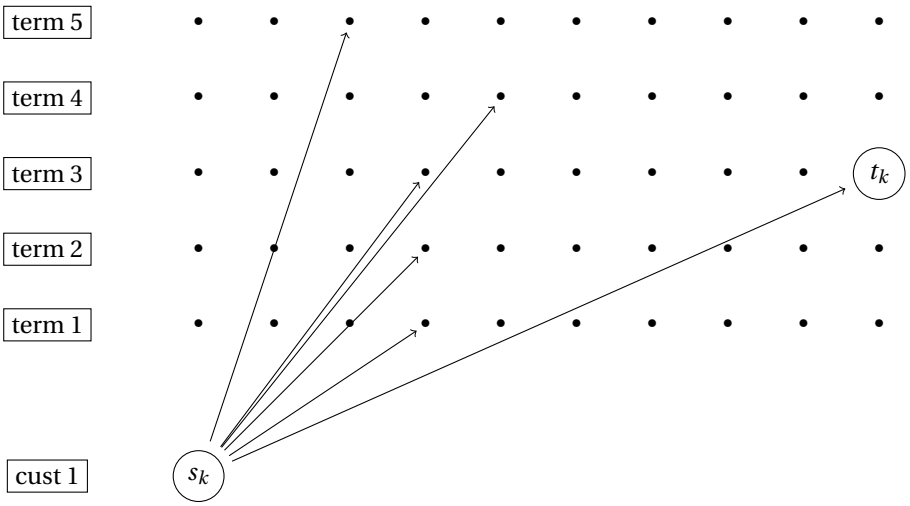


Figure 11.3: Customer sink

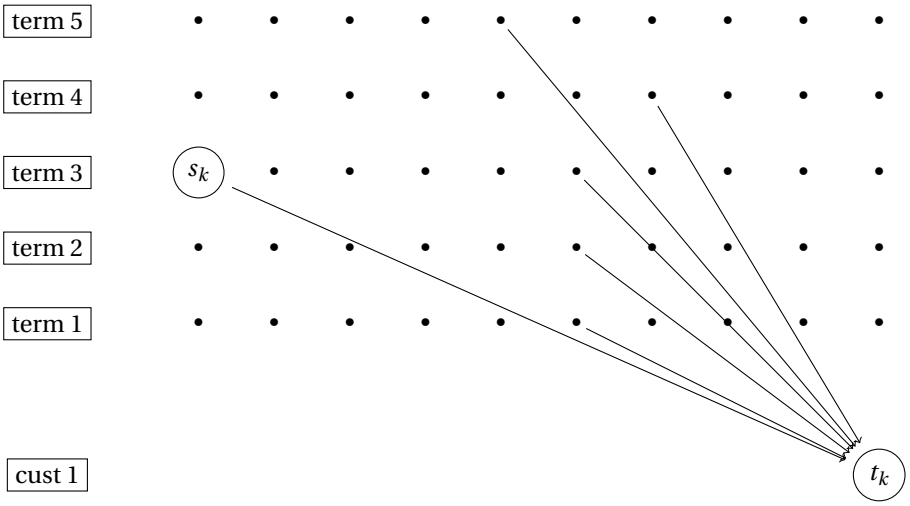
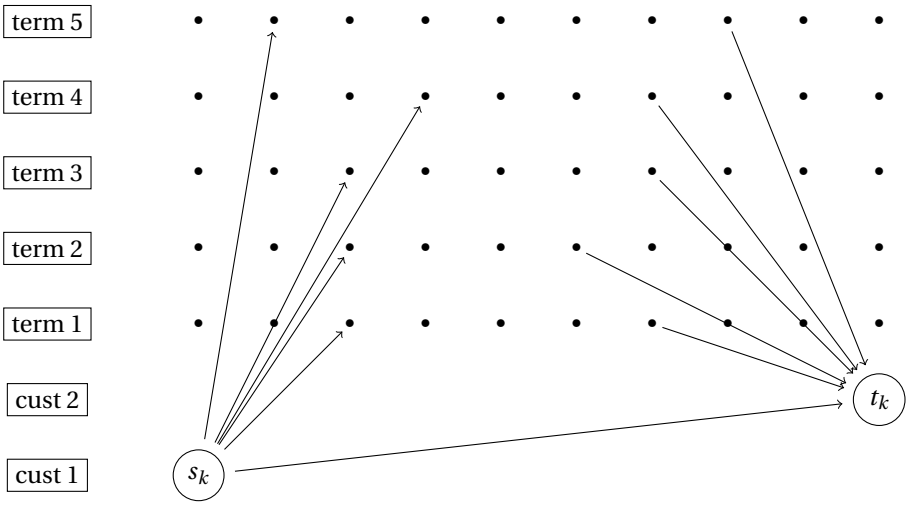


Figure 11.4: Customer source and customer sink



11.10. Minimal Capacity Usage

Recall condition (8.10):

$$\sum_k x_{e,k} \leq c_e y_e \qquad \forall e \in E.$$

It can happen that a vehicle travels some link with no or only a small number of containers. This may not be desirable. If we want every vehicle to travel a link if it is at least a γ fraction of its capacity is used, then constraints enforcing this condition can be added. For some or all non-horizontal arcs e we can add the condition

$$\sum_k x_{e,k} \geq \gamma c_e y_e. \tag{11.4}$$

This condition is also used in [88]. A possible motivation for adding these constraints is that the feasible region of the problem becomes smaller. This may make the problem easier to solve. It may also be that one wants that on average a certain fraction γ of the capacity of w denoted by c_w is used.

$$\gamma c_w \sum_{[(loc1,t),(loc2,t+\mathrm{a}(w,i,j,t)),w] \in E_w | loc1 \neq loc2} y_e \leq \sum_{k \in K} \sum_{[(loc1,t),(loc2,t+\mathrm{a}(w,i,j,t)),w] \in E_w | loc1 \neq loc2} x_{e,k} \tag{11.5}$$

11.11. Fixed Paths/Transfers

If it is already decided which path some container(s) will take, then we call the container(s) fixed. Fixed containers are not added to the model. We should make sure in the MCND problem, however, that the vehicles that need to transport the containers actually travel the links that are required to transport it. Let $\beta_e \in \mathbb{Z}_{>0}$ be the number of fixed containers on an arc. Then we add

$$y_e \geq \lceil \frac{\beta_e}{c_e} \rceil,$$

so we ensure that the minimal required capacity is installed on e in order to be able to transport the fixed containers on that link. In addition we reduce the right-hand side of the capacity constraint for $e \in E$ by adding $-\beta_e$.

We can also similarly fix a whole path for a certain non-truck vehicle. This can be done by adding constraints

$$y_e \geq 1$$

for all e on the fixed path of the vehicle. If reduction C from Section 9.2.1 is not used, then it is better to model the barge arcs for a barge that has a fixed path as done for the MCMCF problem.

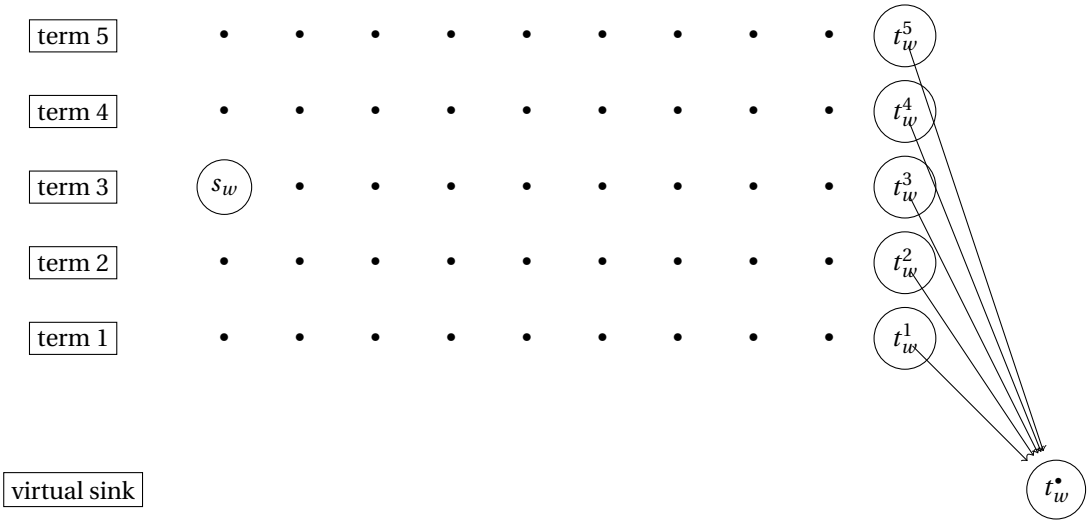
Because of labor regulations and such it might be necessary for a vehicle $w \in W \setminus \{\text{truck}\}$ to take a break for one time step at $t = \xi$. This condition can be added as constraint to the model:

$$\sum_{x \in X} y_{[(\xi,x),(\xi,x),w]} \geq 1. \tag{11.6}$$

11.12. Unknown Vehicle End Location

If the end location of a non-truck vehicle (type) w is not known. Then it does not have to be specified to be able to use the model. We can connect every possible sink t_w^i of w in the time-space graph with a virtual sink $t_w^\bullet$. See Figure 11.5 for a visualization.

Figure 11.5: Unknown Vehicle end location



12

Online Optimization

The problems we looked at up until now had fixed travel times, terminal handling times, capacities etc. In reality some of those parameters are not known exactly, because they rely on a lot of factors of which not all are known in advance. A good way to model those factors is with stochastic variables. There are methods that allow parameters to be stochastic variables. Some of those techniques can be read in [45]. In that thesis exact methods like stochastic programming and Markov decision processes are crossed off, because they take too much time to solve for real-life size instances. Parameters like travel times and terminal handling times that can vary a lot from time to time can be modelled to have a certain probability distribution. So they are not fixed values, but stochastic variables. In [45] a heuristic is described, that can (approximately) solve problems involving stochastic elements when probability distributions for the stochastic parameters are known:

Expected Future Iteration

- Define a deterministic problem by replacing all the stochastic elements by their expected values (rounding if necessary).
- Solve the deterministic problem.
- Run that solution for one time step.
- Look at the realization of all stochastic variables at this time stamp.
- Go back to step 1 to solve this new state, until every commodity is at its destination.

In this (iterative) heuristic the solution is adapted to the new state. The current situation is used to improve the solution. This is called online optimization. It is even possible for new bookings to be included during the iterations. This idea is also generalized using $1-\alpha$ percentiles instead of expected values. The aforementioned heuristic is a way to extend the models in this paper to models suitable for synchromodal transport. The heuristic can be adapted somewhat, such that it is not imperative to do an iteration every time step or to recalculate routes for every container. For example if the route of a container or booking is not obstructed, then it is not crucial to reschedule that container or booking. The deterministic methods described in this paper may also be convenient for use in methods that compare solutions over a set of different scenarios. This is done for a problem similar to our MCND problem with a progressive hedging-based metaheuristic in [55].

13

Results

In order to test out the integral arc-based MCND model, instances have to be generated. We generate own instances based on data from practice. Because of time considerations we will only generate one complex instance, but more instances should be generated to test out the implementation of the integral arc-based MCND problem with some extensions.

Our instance has eight terminals locations in the Netherlands: Nijmegen, Hengelo, Gorinchem, Alblasserdam, Maasvlakte, Schiedam/Rotterdam, Botlek and Delft. We have two groups of six barges. All barges that belong to the same group have the same capacities and travel times (and begin location and end location). Therefore, all barges in the same group are modelled with one variable $w \in W$ in the model (see 9.2.1 Reduction C) unless stated otherwise. We assume we have an infinite number of trucks at every terminal, but restrictions may be added if that is desired (Constraints 8.11 and last paragraph of Section 11.8 Location Constraints). The travel times of the vehicles are loosely based on data from practice and the truck travel times on data from Google Maps [71]. Every truck can carry exactly one container. All the containers in the model are 2 TEU = 40 feet. The capacity of the barges in the model is 200 TEU = 100 containers (the capacity of the barges was a bit less in the data).We choose to look at a time span of 36 hours with time steps of one hour. We chose 36 hours, because the travel times of the barges were high for certain pairs of locations and these trips should be possible. On the other hand, if we took a time span of more hours this may lead to a very long computation time. Furthermore, some relevant bookings may in reality not be known yet when planning to far ahead. In the instance 50 bookings need to be scheduled and 12 barges are available. We let the cost on the non-horizontal truck arcs be directly proportional to travel time of the arc. Initially, we even let the cost on those arcs even equal the travel time for those arcs. The other arc variables have no cost associated. This models a company that owns barges and has to pay additional cost when trucking containers. The barges are always utilized and the more expensive truck transport is minimized. We make sure that it is possible to truck a booking to its destination, so its deadline should be (at least one time stamp) later than its release date. Besides, the terminal locations, two customer locations in the model (see Section 11.9 Customers). The customer locations are not reachable by barge. So in the time-space graph none of the barge arcs are incident to customer locations (that are not accessible by barge). Trains or other (types of) vehicles are initially not in the model, but could easily be added.

For the generation of other instances it would be wise to take into account the following characteristics that describe the situation in practice. Around 90% of all bookings contain at most 10 containers [9]. The demands of the bookings in our instance are larger. The remainder of the bookings normally has at most demand 100.

We solved our ILPs with IBM's CPLEX solver [46]. This solver allows the user to experiment with different parameters and it was able to solve our ILPs a lot faster than CBC and GLPK. We tried out different sets of parameters, to find out which parameters work well for our discrete time MCND problem. In Tables 13.1,13.2 we see some different choices of parameters and how they effect the gap between the lower bound and the minimal integral solution value found after 300 seconds for one specific instance. We also gave a priority to branching on the design variables (y_e variables), because if they are integral the $x_{e,k}$ variables are then often also integral (as mentioned in Chapter 10 Cutting Planes).

Table 13.1: CPLEX Options

branching direction	default	up	up	up
search strategy	dynamic	dynamic	dynamic	traditional
node select	best bound	best bound	best estimate	best bound
Gap (after 300 *s*)	7.59%	3.05%	4.16%	4.65%

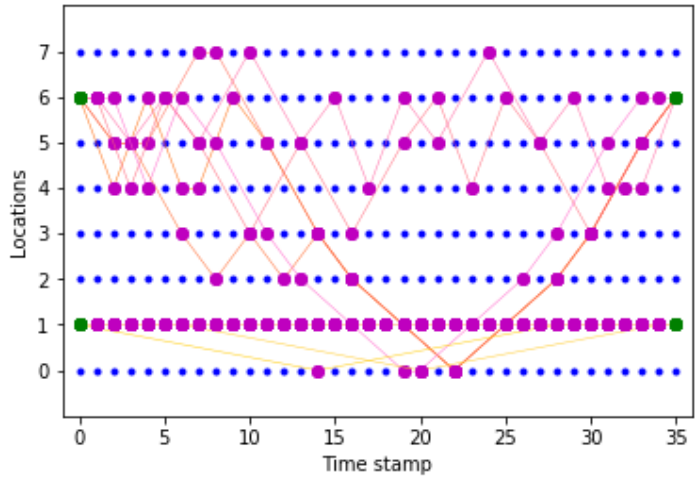
Table 13.2: CPLEX Options

branching direction	up	up	up	up	up
variable select	default	min infeas	strong	pseudo cost	pseudo red
Gap (after 300 *s*)	3.05%	15.27%	21.46%	6.20%	7.95%

CPLEX can recognize structures in the ILP and add cuts. For our problems CPLEX adds mixed integer rounding cuts, Gomory fractional cuts and zero-half cuts. CPLEX solves ILPs with branch & cut. There are also heuristics [62] such as local branching that are used to find solutions faster. For the remaining results this option is turned off in order to have a better idea how the reductions and such influence the branch & cut process.

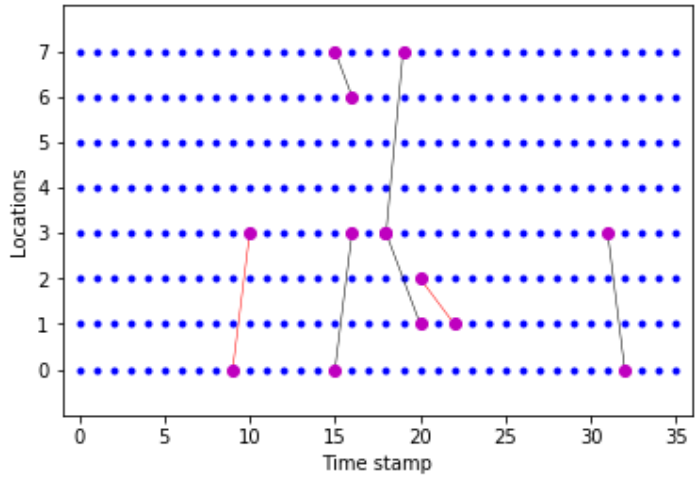
Visualizations of the routes of the barges, trucks and booking 3 in an optimal solution can be found in Figures 13.1, 13.2 and 13.3 respectively. The route of booking 3 begins at a customer location denoted by k0 in Figure 13.3. Note that the used truck (arcs) from and to customer locations are not visualized in Figure 13.2 to make sure the plot is clear even if there are many bookings that go to and/or from customers.

Figure 13.1: Routes of barges



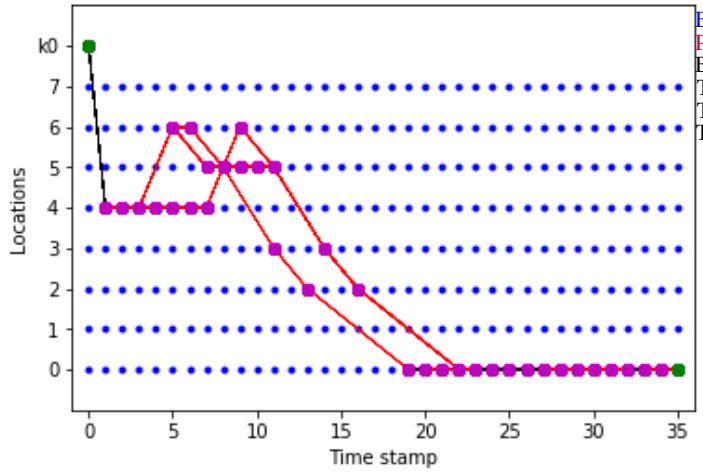
Blue time-space nodes are not visited by barges.
Purple time-space nodes are visited by at least one barge.
There are 6 barges that start at the upper left green node.
There are 6 barges that start at the lower left green node.
There are 6 barges that end at the upper right green node.
There are 6 barges that end at the lower right green node.
Each barge has edges of a (slightly) different color.

Figure 13.2: Routes of trucks



Blue time-space nodes are not visited by trucks.
Purple time-space nodes are visited by at least one truck.
The edges that are travelled by less than 10 trucks are red.
The edges that are travelled by at least 10 trucks are **black**.

Figure 13.3: Routes of the containers of booking 3



Blue time-space nodes are not visited by booking 3.
Purple time-space nodes are visited by booking 3.
Booking 3 is released at the upper left green node.
The lower right green node is the destination of booking 3.
The **black** edges represent truck arcs.
The red edges represent barge arcs.

In all the experiments reduction K is used, because this proved to be a very good reduction. Some of the other reductions and cuts we will turn on and off to see how they influence the computation time. Not all variable reductions are compatible with all the additional features, so it is good to investigate how they influence the computation time.

In Table 13.3 we look at the effect of using or not using reduction I from 9.5.1 with all other things being equal.

Table 13.3: Waterway Connectivity (Reduction I)

Direct Connection Reduction	Computation Time	Best Integer Solution Value
Yes	61.16s	3760 (optimal)
No	1122.50s	3760 (optimal)

Our set of vehicle types $W = \{\text{truck}, \text{typeAbarge}, \text{typeBbarge}\}$ is obtained by applying reduction C from Section 9.2.1. If we subdivide the six barges of type A in two groups of two barges and two groups of one barge in the model, then we get a larger set W. A comparison in computation time between the larger set $|W| = 6$, the smaller set $|W| = 3$ and other sizes of W can be found in Table 13.4.

Table 13.4: W with Reduction C

$\lvert W\rvert$	Computation Time	Best Integer Solution Value
3	61.16s	3760 (optimal)
4	183.51s	3760 (optimal)
5	628.58s	3760 (optimal)
6	1667.61s	3760 (optimal)

Reduction A: Same Sink from 9.1.1 is used to reduce the initial number of commodities $|K| = 50$ to $|K| = 39$. In Table 13.5 we compare the ILP with this reduction and without and we also split some commodity into sub commodities to investigate the effect of taking a larger set $|K|$ on the computation time.

Table 13.5: K with Reduction A

Same Sink Reduction	$\lvert K\rvert$	Computation Time	Best Integer Solution Value
Yes	$25 \rightarrow 20$	5.86s	2600 (optimal)
No	25	7.12s	2600 (optimal)
Yes	$50 \rightarrow 39$	61.16s	3760 (optimal)
No	50	67.45s	3760 (optimal)
No	100	$> 3000s$	3900

Reduction H from 9.4.3 limits the number of horizontal arcs a vehicle can take it is applied to our problem for $\omega = 8$ the results for the performance of that reduction can be seen in Table 13.6.

Table 13.6: Idle Vehicle Reduction (Reduction H)

Idle Vehicle Reduction	Computation Time	Best Integer Solution Value
No	61.16s	3760 (optimal)
Yes ($\omega = 8$)	65.58s	3760 (optimal)

Results for the symmetry breaking cut can be found in Table 13.7. Again vehicles of type A are split in two or three groups and the cut is applied to those groups.

Table 13.7: Symmetry Breaking cut

Symmetry Breaking Cut	$\|W\|$	Computation Time	Best Integer Solution Value
Yes	4	$292.32s$	3760 (optimal)
No	4	$259.27s$	3760 (optimal)
Yes	5	$641.11s$	3760 (optimal)
No	5	$665.19s$	3760 (optimal)

In Table 13.8 we see the effect of adding per commodity arc residual capacity cuts, $K' = \{k\} \forall k \in K$ and arc residual capacity cuts where K' represents the commodities that have their release time before and deadline after the arc. The latter is for convenience written as $K' = K$. The solver CPLEX [46] allows us to create a pool of user cuts and/or lazy constraints. This can in some cases be useful. For vehicle routing problems in some cases subtour elimination constraints need to be added. There are so many of them that it is better to input them as lazy constraints. CPLEX will only add a lazy constraint to the model if it is violated. User cuts work similarly the only difference is that cuts as defined in CPLEX may not cut off integral solutions and do not necessarily need to be added to get an optimal solution. For the arc residual capacity cuts and truck capacity constraints (next paragraph) we tried these options.

Table 13.8: Arc Residual Capacity Cut

$K' = K$	$K' = \{k\}$	Computation Time	Best Integer Solution Value
Standard	Standard	$61.16s$	3760 (optimal)
No	Standard	$54.58s$	3760 (optimal)
Standard	No	$> 300s$	3850
Standard	User Cuts	$> 300s$	3840
No	No	$> 300s$	3790

For some problems in practice adding truck arc capacity constraints may be useful. In Table 13.9 we see the effect of those constraints on the computation time. Note, that very large values are taken for the capacities on the truck arcs.

Table 13.9: Truck Capacity Constraints

Truck Capacity Constraints	Computation Time	Best Integer Solution Value
No	$61.16s$	3760 (optimal)
Standard	$86.58s$	3760 (optimal)
Lazy Constraints	$45.14s$	3760 (optimal)

The results of the cutset cuts can be found in Table 13.10. The counters determine the number of cutset cuts we take and which cutset cuts. If the counter is zero we only take the cutset based on the sink of the commodity. If the counter is one, we also take the cutset cut with the sink and the time-space node with the same location as the sink one time stamp earlier etc.

Table 13.10: Cutset Cut

Counters	Computation Time	Best Integer Solution Value
0	$61.16s$	3760 (optimal)
1	$58.42s$	3760 (optimal)
2	$47.83s$	3760 (optimal)
3	$100.97s$	3760 (optimal)

In Table 13.11 we try out different values of γ in the minimal (average) barge usage constraints.

Table 13.11: Minimal Barge Usage Constraints

Barge Usage Constraints	Computation Time	Best Integer Solution Value
No	61.16s	3760 (optimal)
$\gamma = 0.1$	79.33s	3760 (optimal)
$\gamma = 0.2$	44.70s	3760 (optimal)
$\gamma = 0.3$	99.25s	3860 (optimal)

We look if removing the two customers from the model has some effect in Table 13.12.

Table 13.12: Customers

Customers	Computation Time	Best Integer Solution Value
2	61.16s	3760 (optimal)
No	98.33s	2350 (optimal)

Reduction B and D are applied to the MCND problem, results can be found in Tables 13.13 and 13.14 respectively. In the caption of the tables short descriptions of the reductions can be found.

Table 13.13: Disjoint Time Frame Bookings (Reduction B)

Reduction B	Computation Time	Best Integer Solution Value
No	61.16s	3760 (optimal)
Yes	43.35s	3760 (optimal)

Table 13.14: Sink/Source Location Non-Horizontal Arcs Reduction (Reduction D)

Reduction D	Computation Time	Best Integer Solution Value
No	117.61s	3760 (optimal)
Sink Incoming	61.16s	3760 (optimal)
Sink Incoming/Outgoing	64.58s	3760 (optimal)
Complete	58.50s	3760 (optimal)

The strong arc cut is compared to the $K' = \{k\}$ arc residual capacity cut in Table 13.15.

Table 13.15: Strong Cut

Cut	Computation Time	Best Integer Solution Value
Strong	101.22s	3760 (optimal)
Arc Residual Capacity $K' = \{k\}$	61.16s	3760 (optimal)

The no repeated locations constraints (weak) are also implemented. The results can be found in Table 13.16.

Table 13.16: No Repeated Locations (weak)

Cut	Computation Time	Best Integer Solution Value
No	61.16s	3760 (optimal)
Yes	$> 300s$	3840 (optimal)

The results from the minimal path reduction can be found in Table 13.17.

Table 13.17: Minimal Path Reduction (Reduction G)

Reduction G	Computation Time	Best Integer Solution Value
Yes	61.16s	3760 (optimal)
No	129.98s	3760 (optimal)

Our last experiments can be found in Tables 13.18, 13.19. We subdivide the type A barges again in the second instance to make the problem more difficult to solve. We compare several solution methods:

1. branch & cut,
2. branch & cut with CPLEX heuristics such as local branching ,
3. α B & C-and-fix with 200s for the branch & cut phase and $\alpha = 0$,
4. relax-and-fix,
5. relax-and-fix with where the capacities of the vehicles are halved in the first stage,
6. relax-and-fix where the fixing is based on the design variables (all truck arcs are added),
7. relax-and-fix + NEIGH-($t_{\min}, t_{\max}, w$) search for begin time frames and end time frames of all $w \in W \setminus \{\text{truck}\}$.

Table 13.18: Solution Times of Solution Methods

	Method 1	Method 2	Method 3	Method 4	Method 5	Method 6	Method 7
Instance 1	61.16s	140.06s	146.90s	23.38s	27.72s	25.53s	29.97s
Instance 2	500.20s	$> 700s$	380.13s	68.18s	27.57s	58.22s	110.27s

Table 13.19: Solution Values of Solution Methods

	Method 1	Method 2	Method 3	Method 4	Method 5	Method 6	Method 7
Instance 1	3760 (optimal)	3760 (optimal)	3760 (optimal)	4290	5100	3760 (optimal)	3830
Instance 2	3760 (optimal)	3890	3760 (optimal)	4390	5100	3760 (optimal)	3900

14

Conclusions and Recommendations

In the first part of this thesis we looked at a way to effectively solve Problem **1**. We used the integral MCMCF problem on a time-space graph to model this problem. Different solution methods for this problem were discussed. One of the most promising methods from literature is the column generation method. It stands out, because it can find the optimal solution and can be solved relatively quickly, because the subproblems that need to be solved are shortest path problems. It is however not the fastest method for all types of instances [24]. If a quick method is the main concern, then the repeated cheapest path and repeated min cost flow method will probably stand out [70]. Though more experiments should be done to confirm this.

Also for the main problem of this thesis Problem **3**, where we want to schedule containers **and** route vehicles, a suitable model has been built. The integral MCND ILPs on time-space graphs (with extensions) are suitable models for the problem. Customers can be added to the models without adding to many additional variables and constraints (see Section Customers). Due times can be taken into account (see Section Due Time). Even different container sizes pose no problem to the models. The models can handle different vehicle types.

The models still have a few disadvantages: The models have a set of discrete times, models with continuous time will always be able to make use of the time more efficiently. The trucks are modelled differently than the other vehicles. This means that we look at the number of trips between two locations that need to be done by truck, though in some cases the total number of trucks used in the time frame is more important. This can however be remedied by modeling the trucks the same way as the other vehicles. In that case the problem will become nearly identical to the service network design problem and may be more difficult to solve. Furthermore, a continuous time model such as [79] may be better at modeling conditions like terminal handling time and cost. Though, continuous time models also have their disadvantages. For instance, they often require additional variables and constraints and duplicated location nodes.

Next up conclusions based on the results will be given. Though they should be tread with caution, as essentially only one instance has been investigated. Therefore we first and foremost recommend that the model be tested on more instances preferably from practice. From Figure 13.1 and Figure 13.2 we can conclude that most of the parameter options CPLEX chooses by default seem to work best: dynamic search, best bound search etc. Specifying an upwards branching direction does help. This can be explained. The LP relaxation of the models will almost always return non integral values for some of the y_e variables, due to the fact that usually not every vehicle that is used is constantly at full capacity. For example if only 5 containers want to take a barge arc e_1; $\sum_k x_{e_1,k} = 5$. Together with that the barge (arc) has capacity $c_{e_1} = 10$, then it suffices to take $y_{e_1} = \frac{1}{2}$ looking at the concerned capacity constraint

$$\sum_k x_{e_1,k} \leq y_{e_1} c_{e_1}.$$

This way another arc e_2 can be partially opened, allowing some containers to use e_2 and to possibly get a better solution. A boat cannot be subdivided, so solutions with fractional y_e are not useable in practice. Therefore, the branch & cut process serves a purpose. First looking at the upwards branch might help, because if containers use an arc in the fractional solution, then apparently it is likely that the arc is opened in the integral optimal solution.

Reduction K is one of the most essential reductions. Although no research into the effect of this reduction is done in this thesis, it is clear that it removes loads of variables and constraints from the model. Reduction I and C are both also very powerful tools. Reduction C does require a company to have many identical vehicles, though. Reduction A seems to help a little, but clearly is more effective if there are many bookings with the same/similar sinks. We have seen that doubling the number of commodities in the model can ruin the solution time. So the reduction is beneficial if the number of commodities wreaks havoc on the computation time. Reduction H seems utterly useless. Reduction B surprisingly leads to better results, though it increases the number of variables because it was implemented together with reduction K. More experiments should be performed to see if this is also the case for other instances. Reduction D helps to reduce the computation time to a certain extend. This reduction reduces the number of feasible solutions, but mainly the number of optimal solutions. Reduction G is very helpful, this reduction does require some precalculations to find paths on a space-graph though. For that reason, it may be better to omit this reduction if the space graph is large.

The arc residual capacity cuts are very effective. This may be because many commodities send all of their containers over one path. As expected from theory the cut is stronger, than the strong cut. The cutset cut only helps a little. More experiments with different types of cutsets an other instances should be done to see the full impact of this cut. The no repeated locations constraints are not very effective on our instance. The minimal barge usage constraints do seem to help somewhat, but it is difficult to choose a good value of γ in advance. Truck capacity constraints increase the computation time a little.

During the experiments with the different solution methods several things stood out. branch & cut works well when the right reductions and cuts are added. The standard CPLEX heuristics may for some instances help in the branch & cut process to find a reasonably good feasible solution faster. For very large instances however branch & cut is not an option anymore because the runtime increases too rapidly. In those cases branch & cut already took a very long time to find a first integral feasible solution. The design variable based relax-and-fix has good results in runtime and solution value even for the larger instance. Thus, we recommend to do design variable based fixing. More experiments on different instances should be performed to decide in which cases adding a metaheuristic layer to fine-tune the solution can be useful and if in some cases it may be worth it to solve multiple (I)LPs to decide which design variables to select in the first phase. Other (I)LPs that can be solved for the design variable selection include:

- Solving α cut LP relaxations.
- Solving the ILP with a subset of the time stamps.
- Solving the ILP with a subset of the commodities and/or vehicles.
- Solving the relaxation with a slightly different capacity and/or cost function.

In the design variable based relax and fix none of the truck arc variables are fixed. This means that the number of variables in the model may still grow too quickly to use it for extremely large instances. Three possible options to fix/remove some of the truck arc variables in the second phase of the design variable based relax-and-fix are:

- Only adding truck arcs to every location at the end of every barge arc that is not removed/fixed and at the source of the booking (see Reduction D for the integral MCMCF problem).
- For every truck arc that is used by a commodity k, add the corresponding truck arc variable for every commodity. The other truck arc variables are fixed/removed.
- For every truck arc that is used by a commodity k, add the corresponding truck arc variable for commodity k. The other truck arc variables are fixed/removed.

It may also be a good idea to compare or combine these methods with a column generation approach. We recommend to also take a look at the iterative solution method for the service network design problem from [33], because this was left out of scope.

Not all reductions and additional features discussed in this thesis were implemented. Therefore, it is advisable to do so in future work to see how these additions influence the runtime of the solution methods. In reality many of the parameters in this thesis are uncertain. Hence, it may be worthwhile to investigate if some of the techniques in this thesis can be useful in synchromodal transport.

A

Appendix

A.1. Repeated Cheapest Path Heuristic

[70]

1. Choose a commodity k according to some sequence.
2. Find a cheapest $s_k - t_k$ path for commodity k with no zero capacity arcs with for example Dijkstra; say path p_j^k.
3. Send flow $\mu := \min(\min_{i \in p_j^k} c_i, d_k)$ through the p_j^k.
4. Set $c(e) := c(e) - \mu$ for all $e \in p_j^k$ and $d_k := d_k - \mu$
5. Repeat from step 2 until $d_k = 0$.
6. Repeat from step 1 until all commodities are handled.

A.2. Calculating the Dual

$$\max -c^T x$$

subject to

$$Ax \leq b$$
$$Dx \leq e$$
$$-Dx \leq -e$$
$$x \geq 0$$

$$\min c^T x$$

subject to

$$Ax \leq b$$
$$Dx = e$$
$$x \geq 0$$

$$\min b^T y + e^T z^+ - e^T z^-$$

subject to

$$A^T y + D^T z^+ - D^T z^- \geq -c$$
$$y, z^+, z^- \geq 0$$

$$\min b^T y + e^T z$$

subject to

$$A^T y + D^T z \geq -c$$
$$y \geq 0, z \text{ unrestricted}$$

$$\max -b^T y - e^T z$$

subject to

$$A^T y + D^T z \geq -c$$
$$y \geq 0, z \text{ unrestricted}$$

$$\max -b^T y - e^T z$$

subject to

$$-A^T y - D^T z \leq c$$
$$y \geq 0, z \text{ unrestricted}$$

$$\max b^T y + e^T z$$

subject to

$$A^T y + D^T z \leq c$$
$$y \leq 0, z \text{ unrestricted}$$

Bibliography

[1] Karen Aardal. Advanced discrete optimization, 2016.

[2] Ravindra K. Ahuja, Thomas H. Magnanti, and James B. Orlin. *Network Flows*. Elsevier, 1988.

[3] Mohan Akella, Sharad Gupta, and Avijit Sarkar. Branch and price. *Omega*, 40(5):672–679, 2012.

[4] Jardar Andersen, Teodor Gabriel Crainic, and Marielle Christiansen. Service network design with management and coordination of multiple fleets. *European Journal of Operational Research*, 193(2):377–389, 2009.

[5] Jardar Andersen, Teodor Gabriel Crainic, and Marielle Christiansen. Service network design with asset management: Formulations and comparative analyses. *Transportation Research Part C: Emerging Technologies*, 17(2):197–207, 2009.

[6] Burak Ayar. Multimodal multicommodity routing problem with scheduled services. Master's thesis, Bilkent University, 2008.

[7] F. Babonneau, O. du Merle, and J.P. Vial. Solving large scale linear multicommodity flow problems with an active set strategy and proximal accpm. *Operations Research*, 54(1):184–197, 2006.

[8] Frédéric Babonneau. *Solving the multicommodity flow problem with the analytic center cutting plane method*. PhD thesis, Université de Genève, 2006.

[9] Lina Baranowski. Development of a decision support tool for barge loading, 2013.

[10] Cynthia Barnhart and Rina R. Schneur. Air network design for express shipment service. *Operations Research*, 44(6):852–863, 1996.

[11] Cynthia Barnhart, Christopher A. Hane, and Pamela H. Vance. Using branch-and-price-and-cut to solve origindestination integer multicommodity flow problems. *Operations Research*, 48(2):318–326, 2000.

[12] Cédric Bentz. The maximum integer multiterminal flow problem in directed graphs. *Operations research letters*, 35(2):195–200, 2007.

[13] Berit Dangaard Brouer, David Pisinger, and Simon Spoorendonk. Liner shipping cargo allocation with repositioning of empty containers. *INFOR*, 49(2):109–124, 2011.

[14] Chandra Chekuri. Approximation algorithms, 2011.

[15] Marco Chiarandini. Totally unimodular matrices, 2009.

[16] Jae Hyung Cho, Hyun Soo Kim, and Hyung Rim Choi. An intermodal transport network planning algorithm using dynamic programming—a case study: from busan to rotterdam in intermodal freight routing. *Applied Intelligence*, 36(3):529–541, 2012.

[17] Mervat Chouman and Teodor Gabriel Crainic. Mip-based matheuristic for service network design with design-balanced requirements. *CIRRELT*, 2012.

[18] William Cook. World tsp, 2003. URL `http://www.math.uwaterloo.ca/tsp/world/`.

[19] William J. Cook, William H. Cunningham, William R. Pulleyblank, and Alexander Schrijver. *Combinatorial Optimization*. Wiley Interscience, 1998.

[20] Kamiel Cornelissen and Bodo Manthey. Smoothed analysis of the minimum-mean cycle canceling algorithm and the network simplex algorithm. *International Computing and Combinatorics*, 2015.

[21] Teodor Gabriel Crainic. Service network design in freight transportation. *European journal of Operations Research*, 122(2):272–288, 2000.

[22] Teodor Gabriel Crainic and Michel Gendreau. A scatter search heuristic for the fixed-charge capacitated network design problem. *Metaheuristics*, pages 25–40, 2007.

[23] Teodor Gabriel Crainic and Kap Hwan Kim. Intermodal transportation. *Handbooks in operations research and management science*, 14:467–537, 2007.

[24] Weibin Dai, Jun Zang, and Xiaoqian Sun. On solving multi-commodity problems: An experimental evaluation. *Chinese Journal of Aeronautics*, 30(4):1481–1492, 2017.

[25] Myrthe de Juncker. Optimising routing in an agent-centric synchromodal network with shared information. Master's thesis, TU Eindhoven, 2017.

[26] Jacques Desrosiers and Marco E. Lübbecke. Branch-price-and-cut algorithms. *Wiley encyclopedia of operations research and management science*, 2011.

[27] Marina A. Epelman. Lagrangian relaxations and duality, 2012.

[28] M. Gamst. A local search heuristic for the multi-commodity k-splittable maximum flow problem. *Optimization Letters*, 8(3):919–937, 2014.

[29] Naveen Garg and Jochem Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. *SIAM Journal on Computing*, 37(2):630–652, 2007.

[30] GeeksforGeeks. Dynamic programming set 2 (optimal substructure property), 2017. URL `https://www.geeksforgeeks.org/?p=12819`.

[31] GeeksforGeeks. Dynamic programming set 1 (overlapping subproblems property), 2017. URL `https://www.geeksforgeeks.org/?p=12635`.

[32] GeeksforGeeks. Dynamic programming set 6 (min cost path), 2017. URL `https://www.geeksforgeeks.org/dynamic-programming-set-6-min-cost-path/`.

[33] Bernard Gendron, Saïd Hanafi, and Raca Todosijević. An efficient matheuristic for the multicommodity fixed-charge network design problem. *IFAC*, 49(12):117–120, 2016.

[34] Ilfat Ghamlouche, Teodor Gabriel Crainic, and Michel Gendreau. Path relinking, cycle-based neighbourhoods and capacitated multicommodity network design. *Annals of Operations research*, 131(1-4): 109–133, 2004.

[35] Dion Gijswijt. lecture notes of the course discrete optimization, 2015.

[36] Fred Glover and Kenneth Sörensen. Metaheuristics, 2015. URL `www.scholarpedia.org/article/Metaheuristics`.

[37] Andrew Goldberg, Jeffrey D. Oldham, Cliff Stein, and Serge Plotkin. An implementation of a combinatorial approximation algorithm for minimum-cost multicommodity flow. *International Conference on Integer Programming and Combinatorial Optimization*, 1998.

[38] M. D. Grigoriadis. An efficient implementation of the network simplex method. *Netflow at Pisa*, 26: 83–111, 1986.

[39] Y. Guan and R.K. Cheung. The berth allocation problem: Models and solutions methods. *OR Spectrum*, 26(1):75–92, 2004.

[40] Maarten P. M. Hendriks. *Multi-step Optimization of Logistics Networks: Strategic, Tactical and Operational Decisions*. PhD thesis, TU Eindhoven, 2009.

[41] Mike Hewitt, George L. Nemhauser, and Martin W. P. Savelsbergh. Combining exact and heuristic approaches for the capacitated fixed charge network flow problem. *Journal on Computing*, 22(2):314–325, 2010.

[42] Lê Nguyên Hoang. Column generation and dantzig-wolfe decomposition, 2012. URL http://www.science4all.org/article/column-generation/.

[43] Kaj Holmberg and Di Yuan. A lagrangian heuristic based branch-and-bound approach for the capacitated network design problem. *Operations Research*, 48(3):461–481, 1998.

[44] Kaj Holmberg, Martin Joborn, and Kennet Melin. Lagrangian based heuristics for the multicommodity network flow problem with fixed cost on paths. *European Journal of Operations Research*, 188(1):101–108, 2008.

[45] Dylan Huizing. General methods for synchromodal planning of freight containers and transports. Master's thesis, TU Delft, 2017.

[46] IBM. Cplex optimizer, 2017. URL https://www.ibm.com/analytics/data-science/prescriptive-analytics/cplex-optimizer.

[47] IBM. Zero-half cuts, 2017. URL https://www.ibm.com/support/knowledgecenter/SSSA5P_12.5.0/ilog.odms.cplex.help/CPLEX/UsrMan/topics/discr_optim/mip/cuts/38_zerohalf.html.

[48] IBM. Differences between user cuts and lazy constraints, 2017. URL https://www.ibm.com/support/knowledgecenter/en/SSSA5P_12.6.2/ilog.odms.cplex.help/CPLEX/UsrMan/topics/progr_adv/usr_cut_lazy_constr/04_diffs.html.

[49] Milan Janic. Modelling the full costs of an intermodal and road freight transport network. *Transportation Research Part D: Trnasport and Environment*, 12(1):33–44, 2007.

[50] Martin Joborn, Teodor Gebriel Crainic, Michel Gendreau, Kaj Holmberg, and Jan T. Lundgren. Economies of scale in empty freight car distribution in scheduled railways. *Transportation Science*, 38 (2):121–134, 2004.

[51] Dieter Jungnickel. *Graphs, Networks and Algorithms*, volume 5. Springer, 2007.

[52] Damian J. Kelly and Garrett M. O'Neill. The minimum cost flow problem and the network simplex solution method, 1991.

[53] G. W. Klau. Lagrangian relaxation: An overview, 2007.

[54] M. F. Lai and Hong K. Lo. Ferry service network design: optimal fleet size, routing, and scheduling. *Transportation Research Part A*, 38(4):305–328, 2004.

[55] Giacomo Lanza, Teodor Gabriel Crainic, Walter Rei, and Nicoletta Ricciardi. Service network design problem with quality targets and stochastic travel times. *CIRRELT*, 2017.

[56] Gilbert Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(3):345–358, 1992.

[57] Torbjörn Larsson and Di Yuan. An augmented lagrangian algorithm for large scale multicommodity routing. *Computational Optimization and Applications*, 27(2):187–215, 2004.

[58] Adam N. Letchford and Andrea Lodi. Strengthening chvátal–gomory cuts and gomory fractional cuts. *Operations Research Letters*, 30(2):74–82, 2002.

[59] Xiangyong Li, Y. P. Aneja, and Jiazhen Huo. Using branch-and-price approach to solve the directed network design problem with relays. *Omega*, pages 672–679, 2012.

[60] Xiangyong Li, Kai Wei, Y.P. Aneja, and Peng Tian. Design-balanced capacitated multicommodity network design with heterogeneous assets. *Omega*, 67:145–159, 2017.

[61] Xiangyong Li, Kai Wei, Y.P. Aneja, Peng Tian, and Youzhi Cui. Matheuristics for the single-path design-balanced service network design problem. *Computers and Operations Research*, 77:141–153, 2017.

[62] Ricardo Lima. Ibm ilog cplex what is inside of the box?, 2010.

[63] T. L. Magnanti and R. T. Wong. Network design and transportation planning: Models and algorithms. *Transportation Science*, 18(1):1–55, 1984.

[64] Thomas. L. Magnanti, Prakash Mirchandani, and Rita Vachani. Modeling and solving the two-facility capacitated network loading problem. *Operations Research*, 43(1):142–157, 1995.

[65] John W. Mamer and Richard D. McBride. A decomposition-based pricing procedure for large- scale linear programs: An application to the linear multicommodity flow problem. *Management Science*, 46 (5):693–709, 2000.

[66] Hugues Marchand, Alexander Martin, Robert Weismantel, and Laurence Wolsey. Cutting planes in integer and mixed integer programming. *Discrete Applied Mathematics*, 123(1-3):397–446, 2002.

[67] Farahanim Misni and Lai Soon Lee. A review on strategic, tactical and operational decision planning in reverse logistics of green supply chain network design. *Journal of Computer and Communications*, 5 (08):83, 2017.

[68] Snežana Mitrović-Minić and Ramesh Krishnamurti. The multiple tsp with time windows: vehicle bounds based on precedence graphs. *Operations Research Letters 34*, 34(1):111–120, 2006.

[69] Christel Monrooij. Barge planning with uncertainty of routing and container availability. Master's thesis, Erasmus University Rotterdam, 2017.

[70] Max Roberto Ortega del Vecchyo. Container-to-mode assignment on a synchromodal transportation network: a multi-objective approach. Master's thesis, TU Delft, 2017.

[71] Larry Page and Sundar Pichai. Google maps, 2018. URL `https://www.google.nl/maps`.

[72] Michael Berliner Pedersen, Teodor Gabriel Crainic, and Oli B.G. Madsen. Models and tabu search metaheuristics for service network design with asset-balance requirements. *Transportation Science*, 43(2): 158–177, 2009.

[73] Marc Pfetsch. Multicommodity flows and column generation, 2006.

[74] Line Blander Reinhardt, David Pisinger, Simon Spoorendonk, and Mikkel M. Sigurd. Optimization of the drayage problem using exact methods. *INFOR*, 54(1), 2016.

[75] Inmaculada Rodríguez-Martín and Juan José Salazar-González. A local branching heuristic for the capacitated fixed-charge network design problem. *Computers and Operations Research*, 37(3):575–581, 2010.

[76] Günter Schmidt and Wilbert Wilhelm. Strategic, tactical and operational decisions in multi-national logistics networks: A review and discussion of modeling issue. *International Journal of Production Research*, 38(7):1501–1523, 2000.

[77] Jonatan Schroeder, André L. Pires Guedes, and Elias P. Duarte Jr. Computing the minimum cut and maximum flow of undirected graphs. Technical report, Universidade Federal do Paraná, 2004.

[78] Robert Sedgewick. Data structures and algorithms, 2003.

[79] Kristina Sharypova. *Optimization of Hinterland Intermodal Container Transportation*. PhD thesis, Eindhoven University of Technology, 2014.

[80] Dong-Ping Song and Jonathan Carter. Empty container repositioning in liner shipping. *Maritime Policy and management*, 36(4):291–307, 2009.

[81] J.A. Tomlin. Minimum-cost multicommodity network flows. *Operations Research*, 14(1):45–51, 1966.

[82] Erica van der Sar. Multi-attribute vehicle routing problems. Master's thesis, TU Delft, 2018.

[83] Bart van Riessen, Rudy R. Negenborn, Rommert Dekker, and Gabriel Lodewijks. Service network design for an intermodal container network with flexible transit times and the possibility of using subcontracted transport. *International Journal of Shipping and Transport Logistics*, 7(4):457–478, 2015.

[84] Lieven Vandenberghe. Network flow optimization, 2013.

[85] Duc Minh Vu, Teodor Gabriel Crainic, and Michel Toulouse. A three-stage matheuristic for the capacitated multi-commodity fixed-cost network design with design-balance constraints. *CIRRELT*, 2012.

[86] Duc Minh Vu, Teodor Gabriel Crainic, Michel Toulouse, and Mike Hewitt. Service network design with resource constraints. *Transportation Science*, 50(4):1380–1393, 2014.

[87] Bart Wiegmans and Rob Konings. Intermodal inland waterway transport: Modelling conditions influencing its cost competitiveness. *The Asian Journal of Shipping and Logistics*, 31(2):273–294, 2015.

[88] Yan Xu, Chengxuan Cao, Bin Jia, and Guangzhi Zang. Model and algorithm for container allocation problem with random freight demands in synchromodal transportation. *Mathematical Problems in Engineering*, 2015, 2015.

[89] Masoud Yaghini. Network flows, 2010.

[90] Endong Zhu, Teodor Gabriel Crainic, and Michel Gendreau. Air network design for express shipment service. *Operations Research*, 44(6), 2013.