universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

„On Modelling and Solving Green Collaborative Tactical Transportation Planning“

verfasst von / submitted by

Lukas Gosch, BSc

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2021 / Vienna, 2021

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	UA 066 910
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Masterstudium Computational Science
Betreut von / Supervisor:	Univ.-Prof. Mag. Dr. Karl Franz Dörner, Privatdoz.

ABSTRACT

This work presents a new mathematical model for tactical transportation planning in a horizontal collaboration defined by warehouse sharing and the joint organization of transport. The model enables sustainable planning by associating estimated CO_2 equivalent (CO_2e) emissions to each logistic operation and then allows to optimize for transportation costs, emissions or both objectives through carbon pricing. Furthermore, it features intermodal transport, handling and storage capacities, diverse products and realistic tariff structures with volume discounts. Graph-structures to linearize these non-linear tariffs are developed and a mixed-integer formulation is derived, which is provably strengthened by multiple sets of valid inequalities.

To solve large-scale instances, a hybrid heuristic composed of two parts is developed. The first part corresponds to a slope scaling matheuristic, which is generalized to non-negative integer variables. Due to this generalization, a new slope scaling design principle based on monotonicity is introduced. The second part consists of a local search, which reroutes flow of multiple products at once along lowest-cost paths in the network.

Results obtained from simulating collaboration in the Danube Region using the regional available transportation infrastructure including railway and shipping networks reveal significant saving potentials in both costs and CO_2e emissions. Cost minimizing solutions always lead to reductions of the carbon footprint. However, minimizing for emissions can significantly further this reduction, but requires a minimum size of the collaboration to be cost-efficient.

ZUSAMMENFASSUNG

Diese Arbeit präsentiert ein neues mathematisches Modell zur taktischen Transportplanung in einer horizontalen Kollaboration, definiert durch gemeinschaftlich genützte Warenlager und einer gemeinsamen Organisation des Transportes. Das Modell erlaubt nachhaltige Planung indem jeder logistischen Operation ein geschätzter Emissionswert in CO_2-Äquivalent (CO_2e) zugeordnet und danach eine Optimierung der Transportkosten, Emissionen oder beider Werte durch CO_2-Bepreisung ermöglicht wird. Zudem berücksichtigt es intermodalen Verkehr, Umschlag- und Lagerkapazitäten, Diversität von Produkten und realistische Tarifstrukturen mit Mengenrabatten. Es werden Graphstrukturen, um diese nicht-linearen Tarife zu linearisieren, entwickelt und eine gemischt-ganzzahlige Formulierung hergeleitet, welche nachweislich durch gültige Ungleichungen gestärkt wird.

Um große Instanzen zu lösen, wird eine zweiteilige, hybride Heuristik entwickelt. Der erste Teil entspricht einer Slope Scaling Matheuristik, welche für nicht-negative ganzzahlige Entscheidungsvariablen generalisiert wird. Aufgrund dieser Generalisierung wird ein neues Slope Scaling Designprinzip basierend auf Monotonie eingeführt. Der zweite Teil besteht aus einer Lokalen Suche, welche im Netzwerk Fluss von mehreren Waren gleichzeitig entlang von Pfaden niedrigster Kosten umleitet.

Ergebnisse, erzielt durch die Simulation von Kollaboration in der Donauregion unter Einbezug der regional verfügbaren Transportinfrastuktur einschließlich des Bahn- und Schiffsnetzwerkes, zeigen signifikantes Sparpotential bezüglich Kosten und CO_2e Emissionen. Kostenminimierende Lösungen führen immer zu einer Reduktion des CO_2-Fußabdrucks. Eine Emissionenminimierung kann diese Reduktion noch signifikant verbessern, benötigt aber eine Mindestgröße der Kollaboration, um kosteneffizient zu sein.

ACKNOWLEDGMENTS

This work came into being through my employment in the Integrated Transport Optimisation research group at AIT Austrian Institute of Technology in Vienna, Austria. I want to especially thank the group leader Dr. Matthias Prandtstetter for giving me this opportunity and providing great support and supervision in all stages of this thesis.

I also want to especially thank Prof. Karl F. Dörner for providing excellent supervision of this thesis at the University of Vienna.

Additionally, I want to thank Georg Brandstätter for always being open for questions and providing compute cluster support. I also want to thank all my other colleagues at AIT.

Finishing my master studies would have been far more difficult, were it not for the great study colleagues from the computational science master's program, thank you!

My life would not be the same without Florian Hechenberger, Werner Temper and all my other great friends in Vienna and around the globe. Thank you for being there when I need you and keeping me sane in the face of all the problems life throws up. It is the people who make life worth living.

Lastly, I want to give a special thank you to my parents Gerhard Gosch and Michaela Gosch as well as my sister Kathrin Gosch, who always support me in good as well as bad times and with their help have enabled my studies.

To conclude, this work received funding by the Austrian Federal Ministry for Climate Action, Environment, Energy, Mobility, Innovation and Technology (BMK) in the research program "Mobilität der Zukunft" under grant number 877710 (PhysICAL).

PREFACE

Results from this thesis were presented in a talk at the 8th International Physical Internet Conference (IPIC2021) held in virtual form on the 15th and 16th of June, 2021. As a result, parts of this work will appear in the proceedings of this conference (see Gosch et al. [1]). This concerns parts of all chapters of this thesis except Chapter 2 and parts of the abstract. Therefore, in the beginning of each chapter, a short notice is included about which sections will appear in altered or unaltered form in Gosch et al. [1].

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

ACRONYMS

BKP	Bounded Knapsack Problem
CO_2e	Carbon Dioxide Equivalent
DAG	Directed Acyclic Graph
DFS	Depth-First Search
GTTP	Generalized Tactical Transportation Problem
IP	Integer Program
ILPH	Iterative Linear Programming Heuristic
LS	Local Search
LP	Linear Program
MBKP	Multidimensional Bounded Knapsack Problem
MBKP′	MBKP-Variant Introduced in Section 4.2.2.3
MCFP	Minimum Cost Flow Problem
MCND	Multicommodity Capacitated Fixed-Charge Network Design Problem
MIP	Mixed-Integer Program
SSC	Slope Scaling Heuristic
SSP	Subset Sum Problem
TTP	Tactical Transportation Problem
TU	Transport Unit
VI	Valid Inequalities

Das Klima der Kontinente [hängt ab von den Veränderungen], welche der Mensch [...] durch die Entwicklung großer Dampf- und Gasmassen an den Mittelpunkten der Industrie hervorbringt.

– Alexander von Humboldt, 1844

1 INTRODUCTION

This chapter including Section 1.1 and a shortened version of Section 1.2 will appear in Gosch et al. [1].

The transport sector is responsible for over 14% of the total anthropogenic greenhouse gas emissions. Approximately 43% of these emissions can be attributed to freight transportation [4]. Furthermore, the OECD predicts a tripling of the global goods traffic until 2050 [5, 6] and when continuing with the status-quo, transport emissions are predicted to increase at the fastest rate compared to any other energy-related end-use sector [4]. As a result, urgent measures must be taken to reduce the rising emissions in freight logistics.

Central solution proposals for a more sustainable transport of goods are: modal shifts, increasing freight load factors for example through consolidation, and better optimized and integrated transport networks [4, 5, 7]. A recent paradigm in logistics addressing all these points at once is collaboration between competitors taking the form of sharing warehouses and jointly organizing freight transportation. This so called horizontal collaboration [8] is a major stepping stone of the EU's strategy to achieve climate-neutrality by 2050 [9, 10] and an integral part of the Physical Internet vision [11].

Therefore, this thesis introduces and solves a new mathematical model to optimally and sustainably plan and use logistic resources in a horizontal collaboration. Planning and decision making in logistics and supply chain management is structured into three different levels, strategic (long-term), tactical (medium-term) and operational (short-term) [12, 13]. The model introduced here assumes that the relevant transportation infrastructure is already in place and is concerned with tactical planning which sets the general conditions for operational decisions.

1.1 PROBLEM DESCRIPTION

Competing enterprises with diverse sets of products want to collaborate to reduce logistic costs and emissions. This collaboration is characterized by sharing transportation and opening up warehouses for mutual usage, both enabled by packing goods in modular and standardized load units. The companies together have a network of warehouses spread over different geographical regions with differing product demands, but not every enterprise on its own has a warehouse in every demand region. Furthermore, different companies can have

product demands in similar geographical regions. Consequently, they could share certain transport routes.

Now, to deliver goods into a demand region on time, the companies want to devise a shared transportation strategy which makes effective use of new consolidation and storage potentials and existing intermodal infrastructure whose usage is unlocked through higher product volumes. As these planning issues require some lead time, they are interested in developing a tactical freight plan.

Hence, the model's main focus is in finding optimal paths freight should take through the existing network. This includes tariff, transport mode and storage choices making effective use of spatial and temporal consolidation potentials and of opportunities for economics of scale. The model is not concerned with concrete vehicle routing, packing problems or similar as these decisions are part of operational planning.

Key Model Aspects

Sustainable planning is achieved by associating estimated CO_2 equivalent (CO_2e) emissions to each logistic operation and optionally pricing them allowing for *internalization*. Internalization refers to estimating costs of wider effects of business activities on the community and ecosystem and integrating them into the companies budgets [14]. As a result, the developed model allows to optimize for transportation costs, CO_2e emissions or both.

The ***diversity of products*** is met with a holistic commodity-modelling approach. This includes if necessary, special transportation conditions such as cooling. Then, through adding a time dimension, the model allows for expiry-aware shared routing of perishable and non-perishable goods.

The model considers a network of facilities which can be warehouses, factories or transshipment points, connected by (capacitated) transportation relations of arbitrary types in space and time. As a result, ***intermodal transportation*** possibilities are integrated capturing their full implications on emission, cost and delivery time.

Additionally, each node in the network can be endowed with ***handling capacity limits***. ***Storage*** possibilities are represented by transport relations in time between the same facility.

Finally, the model incorporates ***realistic tariff structures*** with all-unit volume discounts often found in practice [15]. We design graph-structures to linearly model these tariffs. These allow us to formulate the problem as a ***capacitated network design problem*** [16].

1.2 OVERVIEW

This work starts with an overview of the relevant theoretical background in Chapter 2. In Chapter 3 the linear mixed-integer programming formulation of the model is introduced. Due to the model being NP-hard (see Section 3.2.6), to successfully solve large instances two different types of heuristics are developed and combined:

- In Section 4.1 the slope scaling heuristic [17, 18] is generalized to non-negative integer variables making it applicable to the current model. Slope scaling is a matheuristic originally developed for network design problems with binary variables and constitutes an integral part of state-of-the-art hybrid heuristics for this problem type [19, 20]. It consists of an iterative solution process based on solving an approximate polynomial-time problem exactly and using the gathered solution to update the approximation. Section 4.1 and the results of Chapter 5 challenge the main slope scaling design paradigm that the approximation has to match the costs of the original problem and put forward a more fundamental design paradigm based on monotonicity.
- In Section 4.2 a local-search based approach - using the slope scaling heuristic as a fast and effective construction heuristic - is developed. Based on ideas from Harks et al. [21], it jointly reroutes flow of multiple commodities using lowest-cost paths in the network.

The aforementioned heuristics as well as a commercial MIP-solver are applied to generated problem instances based on the real intermodal transportation infrastructure in the Danube Region. This includes real ports along the Danube and the actual train and street network in the looked upon industrial regions. The results of which can be found in Chapter 5 and show that optimizing for transportation costs in general results in significant savings in costs and CO_2e emissions. These emission reductions can be increased when only optimizing for the carbon footprint. However, then a certain minimum critical size of the collaboration is necessary to simultaneously result in a cost saving solution.

A conclusion of this work including future research directions can be found in chapter 6.

2

BACKGROUND

This thesis is fundamentally concerned with optimization [22, 23]. In an optimization problem one is usually given a function $f : X \to \mathbb{R}$ and a set of constraints restricting all possible arguments $x \in X$ called *solutions* to a *feasible region* $\Omega \subseteq X$. Then, the goal is to find a *feasible solution* $x^* \in \Omega$ with the smallest[1] function value. As a result, an optimization problem can formally be stated as:

$$\min_{x \in X} f(x) \qquad \text{subject to } x \in \Omega \tag{2.1}$$

with f often called the *objective* or *cost function*.

Depending on the choice of f, X and Ω different areas of optimization are distinguished [24]. In *discrete* or *combinatorial optimization* Ω is a finite or countably infinite set. A special case of combinatorial optimization arises when Ω is a subset of $\mathbb{Z}^n$ called *integer programming*. However, if Ω is uncountable (e.g. an infinite subset of $\mathbb{R}^n$) one speaks of *continous optimization*. The main optimization problem considered in this thesis defines a solution as a pair of integer and real variables and thus is called a *mixed-integer program* (MIP). In general having $\Omega \subseteq \mathbb{Z}^p \times \mathbb{R}^{n-p}$ with $1 \leq p \leq n-1$ defines the area of *mixed-integer programming*.

The term programming for optimization traces back to the US military around World War II. It used the term "program" to refer to logistical supply plans or training and deployment schedules of combat units. In the post war period the US military tasked a group of researchers including George Dantzig with programming - the planning of such programs [25].

To solve this thesis' optimization problem techniques from continuous and integer optimization are used, which have been developed for linear objective functions. If f is linear one speaks of *linear programming* or *integer linear programming*, respectively. Section 2.1 introduces the main concepts and algorithms from the field of linear programming used in this work. Section 2.2 gives on overview of the relevant concepts from integer linear programming.

Solution techniques presented in Section 2.1 and 2.2 guarantee to find an optimal solution. However, many optimization problems are NP-hard and finding an optimal solution to large instances is infeasible. Polynomial time methods, which guarantee some closeness to an optimal solution are called *approximation algorithms* [26]. They are a special case of a general class of algorithms known as *heuristics*, which provide solutions without guaranteeing optimality [27]. While approximation

1 Without loss of generality the text is only concerned with minimization problems. Every maximization problem can be transformed into a minimization problem by defining a new objective function $\tilde{f}(x) = -f(x)$.

algorithms are often specifically developed for a given problem *metaheuristics* are general procedures to derive problem-specific heuristics, but mostly come without any approximation guarantees [28]. Section 2.3 introduces concepts from metaheuristics used to develop a heuristic for the optimization problem in this work. Approximation algorithms are not used in this work and hence, their presentation omitted.

Last but not least, Section 2.4 introduces concrete optimization problems relevant for this work.

2.1 LINEAR PROGRAMMING

The following presentation of linear programming, including the developments of the different simplex methods to solve linear programs in Sections 2.1.1 to 2.1.3, is mainly based on the book from Bertsimas and Tsitsiklis [2].

Linear programming refers to an optimization problem (2.1) where both the cost function f and constraints are linear. A linear program (LP) can be written as follows:

$$\begin{aligned} \min \quad & c^T x \\ \text{subject to} \quad & Ax \geq b \end{aligned} \tag{2.2}$$

$c \in \mathbb{R}^n$ refers to the cost vector and $x \in \mathbb{R}^n$ are the decision variables. $A \in \mathbb{R}^{m \times n}$ is the constraint matrix, which together with $b \in \mathbb{R}^m$ defines m constraints for the n variables.

Each problem in general form (2.2) can be transformed into standard form (2.3) and vice versa.

$$\begin{aligned} \min \quad & c^T x \\ \text{subject to} \quad & Ax = b \\ & x \geq 0 \end{aligned} \tag{2.3}$$

Therefore, the feasible region for formulation (2.2) is the set $\{x \in \mathbb{R}^n | Ax \geq b\}$ and for formulation (2.3) $\{x \in \mathbb{R}^n | Ax = b, x \geq 0\}$. Both sets are *polyhedra*, see Definition 1.

Definition 1. [2] A **polyhedron** P is a set that can be described in the form $\{x \in \mathbb{R}^n | Ax \geq b\}$, where A is an $m \times n$ matrix and b is a vector in $\mathbb{R}^m$.

Studying the geometry of polyhedra is important for both devising solutions algorithms for LPs (see Section 2.1.1) as well as solving integer linear programming problems (see Section 2.2).

Important about a polyhedron are its *corner points*. While there are intuitive geometric definitions of a corner point, here an equivalent algebraic definition is given, which is important for the algorithmic development of the simplex algorithm in Section 2.1.1. The definition is based on constraints being *active*. A constraint i is called active if a given vector $x \in \mathbb{R}^n$ fulfils it with equality, i.e. $a_i^T x = b_i$ with a_i being the i-th row of matrix A.

Definition 2. [2] Consider a polyhedron P defined by linear equality and inequality constraints, and let x^* be an element of $\mathbb{R}^n$.

a) The vector x^* is a **basic solution** if:

   i) All equality constraints are active;

   ii) Out of the constraints that are active at x^*, there are n of them that are linearly independent.

b) If x^* is a basic solution that satisfies all of the constraints, we say that it is a **basic feasible solution**.

One can show that an $x \in P$ is a corner point of P if and only if x is a basic feasible solution. Furthermore, two basic solutions are called *adjacent* if they share $n - 1$ active and linear independent constraints.

Assume one is given the polyhedron P of a linear optimization problem for which an optimal solution exists. An important result [2] about P is that if it has at least one corner point it has an optimal solution, which is a corner point. Now, assume the polyhedron P stems from an LP in standard form. Then, it can be shown that if the LP in standard form has at least one solution, P has at least one corner point and hence, an optimal solution to the LP can be found at the corner points of P. Therefore, in the search for an optimal solution to an LP in standard form one can restrict oneself to only search at the corners of P.

However, a polyhedron can have exponentially many corners with respect to the problem size.

A local optimal solution has no improving solutions in its vicinity, but in general is not guaranteed to be the globally best solution. However, linear programs have per definition linear cost functions and hence, always convex objectives. Additionally, P can also be shown to be a convex set. These facts imply that each local optimal solution for a linear program is also globally optimal [2].

2.1.1 *Primal Simplex Method*

The main idea behind the (primal) simplex method [2] is to move from one basic feasible solution to another adjacent basic feasible solution with reduced costs until a local optimum is reached. That such an algorithm indeed terminates with a globally optimal solution is motivated in the previous Section 2.1. It showed that for each feasible LP in standard form the search for optimal solutions can be restricted to corner points, i.e. basic feasible solutions. Furthermore, it showed that for LPs a locally optimal solution always corresponds to a globally optimal one.

One can show the full-row rank assumptions can be made without loss of generality.

Assume an LP in standard form is given. Furthermore, assume the m rows of A are non-redundant and hence, linearly independent. Then, one can show that for each basic solution $x \in \mathbb{R}^n$ so called *basic-indices* $B(1), \ldots, B(m)$ exist for which the following holds: the matrix columns $A_{B(1)}, \ldots, A_{B(m)}$ are linearly independent, and if $i \notin \{B(1), \ldots, B(m)\}$ it follows that $x_i = 0$.

B as a subscript denotes the basic indices.

Indices $i \notin \{B(1), \ldots, B(m)\}$ are called *non-basic*. The *basic columns* $A_{B(1)}, \ldots, A_{B(m)}$ span $\mathbb{R}^m$ and are brought together to form a $m \times m$ dimensional *basis matrix* B. Likewise, the elements $x_{B(1)}, \ldots, x_{B(m)}$ in x form the vector x_B. Due to $x_i = 0$ for all $i \notin \{B(1), \ldots, B(m)\}$, $Ax = Bx_B = b$. Therefore, a unique basic feasible solution can be obtained from B by solving $x_B = B^{-1}b$.

As mentioned in the beginning, the main idea behind the simplex method is to move from one basic feasible solution to another. For this a *direction* d and scalar θ have to be chosen to yield another basic feasible solution $x + \theta d$. This is achieved by choosing a non-basic index j and move by θ in the direction of x_j, while letting the solution values associated to the other non-basic indices unchanged. Therefore, $d_j = 1$ and $d_i = 0$ for all $i \notin \{B(1), \ldots, B(m)\}$ except $i = j$.

When moving into the direction of the j-th non-basic variable, the basic variables x_B have to change by $x_B + \theta d_B$. One can show that choosing $d_B = -B^{-1}A_j$ results into a new feasible solution given θ is small enough.

Now, all elements of the direction d to move into to generate a new basic feasible solution have been defined. The chosen direction d changes the cost function by $c^T d = c_j + c_B^T d_B = c_j - c_B^T B^{-1} A_j$.

The question remains how far one has to move along d to yield another basic feasible solution. This is answered in Algorithm 1 representing one iteration of the simplex method. Its steps are taken and adapted from Bertsimas and Tsitsiklis [2]. The algorithm assumes that a basic feasible solution x and the associated basic matrix B are given. Fur-

thermore, it assumes global access to the data of the LP, i.e. A and b.

Note that due to the possibility of exponentially many basic feasible solutions, the simplex method may need exponentially many iterations to terminate. However, in practice, it is observed that the simplex method is efficient in calculating solutions to linear programming problems. In general, the linear programming problem is solvable in polynomial time. The first polynomial time algorithm for linear programming, proofing the polynomial time solvability of LPs, was the ellipsoid method. However, practically more useful polynomial time algorithms are interior-point methods [2].

Algorithm 1: One Iteration of the Simplex Method [2]

Function `simplex_iteration(`x, B`):`
foreach *non-basic index* j **do**
  $\tilde{c}_j \leftarrow c_j - c_B^T B^{-1} A_j$ `// Compute the change in cost`
if *all* $\tilde{c}_j$ *are non-negative* **then**
  Terminate algorithm `//` x `is an optimal solution`
else
  Choose a j with $\tilde{c}_j < 0$
$d_B \leftarrow -B^{-1} A_j$
if *all elements in* d_B *are non-negative* **then**
  Terminate algorithm with optimal costs $-\infty$
$\theta \leftarrow \min_{\{i=1,\dots,m | d_{B(i)} < 0\}} \left(-\frac{x_{B(i)}}{d_{B(i)}}\right)$
Let l be such that θ equals $-\frac{x_{B(l)}}{d_{B(l)}}$
Update B by replacing column $A_{B(l)}$ with A_j
$x_j \leftarrow \theta$
foreach *basic index* $i \neq l$ **do**
  $x_{B(i)} \leftarrow x_{B(i)} + \theta d_{B(i)}$
return x, B `// x is a new basic feasible solution with associated basis B`

2.1.2 *Dual Simplex Method*

If one is given a minimization problem (2.1), a feasible solution x, which is an upper bound to the optimal solution x^*, i.e. $x^* \leq x$, is called a *primal bound*. A lower bound on x^* is called a *dual bound*.

For each linear programming problem, a so called *dual problem* can be defined, which searches for the maximal lower bound on the optimal solution of the original problem. In this context, the original LP is called the *primal problem*. A solution to the primal problem is always greater or equal to a solution of the associated dual problem, this

property is known as *weak duality*. For linear programming problems in particular, it can be shown that the optimal solution to a primal problem and the optimal solution to the associated dual problem coincide. This property is called *strong duality* [2].

The simplex method of Section 2.1.1 can be seen as moving from one primal basic feasible solution to another one, until the solution is also feasible for the associated dual problem. The *dual simplex method* works similarly. However, it starts from a basic feasible solution to the dual problem and moves from one basic feasible solution for the dual problem to another until primal feasibility is reached.

2.1.3 *Network Simplex Method*

Network flow problems (see Section 2.4.1) can be formulated as linear programming problems and hence, solved using the simplex or dual simplex method. However, it is possible to develop a simplex method specialized to the underlying network structure called the *network simplex algorithm*. It has been shown to lead to significantly faster solution times compared to general simplex methods when applied to solve network flow problems [2].

2.2 INTEGER LINEAR PROGRAMMING

The sections concerning integer linear programming closely follow the books Wolsey [3] and Nemhauser and Wolsey [22].

Integer linear programming refers to an optimization problem (2.1) where both the cost function f and constraints are linear, but the feasible solutions are restricted to integer values. An integer linear program (IP) can be formulated as follows:

$$\begin{aligned} \min\ & c^T x && && (2.4)\\ & Ax \leq b \\ & x_j \geq 0 \text{ and integer} && \forall j = 1, \ldots, n \end{aligned}$$

where x is n-dimensional. Furthermore and similar to the linear programming case (see problem (2.2)), c is an n-dimensional cost vector, b is a vector of m dimensions and A is the constraints matrix with m rows and n columns. If only some of the variables in problem (2.4) are restricted to be integer values one speaks of a linear mixed-integer programming problem (MIP).

Note that P corresponding to $\{x \in \mathbb{R}^n : Ax \leq b\}$ is a polyhedron (see Definition 1). $Ax \leq b \equiv -Ax \geq -b$

Definition 3. [3] A polyhedron $P \subseteq \mathbb{R}^n$ is a *formulation* for a set $X \subseteq \mathbb{Z}^n$ if and only if $X = P \cap \mathbb{Z}^n$.

Given two optimization problems (P1) $\min\{f(x) : x \in X' \subset \mathbb{R}^n\}$ and (P2) $\min\{c(x) : x \in X \subset \mathbb{R}^n\}$. P1 is called a *relaxation* of P2 if $X \subseteq X'$ and $f(x) \leq c(x)$ for all $x \in X$. It follows that an optimal solution to P1 is always smaller or equal to an optimal solution of P2.

A formulation P is called *ideal* if it corresponds to the convex hull of all points in X. As argued in Section 2.1, the linear program $\min\{c^T x : x \in conv(X)\}$ has an optimal solution x^* at one of its corner points defined by X. However, the LP is a relaxation of $\min\{c^T x : x \in X\}$. Therefore, by solving the LP, one can solve $\min\{c^T x : x \in X\}$.

Definition 4. [3] Given a set $X \subseteq \mathbb{R}^n$, and two formulations P_1 and P_2 for X. P_1 is a *better formulation* than P_2 if $P_1 \subset P_2$.

However, if neither $P_1 \subseteq P_2$ nor $P_2 \subseteq P_1$, the formulations are said to be *incomparable*. Obtaining a better formulation can be achieved by adding valid inequalities to a formulation.

Definition 5. [22] The inequality $\pi x \leq \pi_0$ is called a *valid inequality* for X if it is satisfied by all points in X.

If adding a valid inequality to a formulation P for X results in a better formulation, the valid inequality is said to *strengthen* the formulation.

Note that X is a subset of $conv(X)$, which again is a subset of all possible formulation for X. However, for NP-hard problems, obtaining an explicit description for $conv(X)$ is usually not feasible unless $P = NP$.

Especially relevant for integer programming is the following type of relaxation:

Definition 6. [3] For the integer program $\min\{c^T x : x \in P \cap \mathbb{Z}^n\}$ with formulation $P = \{x \in \mathbb{R}^n_+ : Ax \leq b\}$, the *linear programming relaxation* is the linear program $x^{LP} = \max\{c^T x : x \in P\}$.

As mentioned in Section 2.1.2, a solution x, which is an upper bound to an optimal solution x^* for a given optimization problem, is called a *primal bound*. A lower bound on x^* is called a *dual bound*. Therefore, the solution x^{LP} from a linear programming relaxation of an IP provides a dual bound to the optimal solution x^* to the IP.

2.2.1 *LP-Based Branch & Bound*

IPs can be solved by the linear programming based branch & bound algorithm. Its pseudocode is given in Algorithm 2, which is adapted

from Wolsey [3]. The algorithm assumes to solve an initially provided optimization problem $\min\{c^T x : x \in X\}$ with formulation P. In the following, problem X refers to minimizing $c^T x$ over X using an associated formulation P.

Algorithm 2: LP-Based Branch & Bound [3]

Function `lp_branch_and_bound(`c, X, P`):`
Add problem X with formulation P to empty list L.
Initialize primal-bound $\overline{c}$ to $+\infty$
Set incumbent solution x^* to *None*
while *L is not empty* **do**
  Choose problem X^i with formulation P^i from L.
  Solve the LP relaxation over P^i to obtain solution $x^{LP,i}$ and dual bound value c^{LP}.
  if *X^i is empty* **then**
    Prune X^i by infeasibility
  else if $c^{LP} \geq \overline{c}$ **then**
    Prune X^i by bound
  else if $x^{LP,i} \in X$ **then**
    `//` $x^{LP,i}$ `is feasible for the original IP.`
    if $c^{LP} \leq \overline{c}$ **then**
      $\overline{c} \leftarrow c^{LP}$
      $x^* \leftarrow x^{LP,i}$
    Prune X^i by optimality
  else
    Choose an index j for which $x_j^{LP,i}$ is fractional.
    $X_1^i \leftarrow X^i \cap \{x : x_j \leq \lfloor x_j^{LP,i} \rfloor\}$
    $X_2^i \leftarrow X^i \cap \{x : x_j \geq \lceil x_j^{LP,i} \rceil\}$
    $P_1^i \leftarrow P^i \cap \{x : x_j \leq \lfloor x_j^{LP,i} \rfloor\}$
    $P_2^i \leftarrow P^i \cap \{x : x_j \geq \lceil x_j^{LP,i} \rceil\}$
    Add problem X_1^i with formulation P_1^i and X_2^i with formulation P_2^i to problem list L.

2.2.2 *Cutting Plane Algorithm*

The cutting plane algorithm is based upon iteratively strengthening a given formulation P for an integer programming problem $\min\{c^T x : x \in X\}$ by adding valid inequalities. The added valid inequalities are chosen so as to cut away the solution obtained through solving the linear programming relaxation over P. More formally, the following problem is solved either in an exact or heuristic fashion:

Definition 7. [3] The *Separation Problem* associated with $\min\{c^T x : x \in X\}$ is the problem: Given $x^* \in \mathbb{R}^n$, is $x^* \in conv(X)$? If not, find an inequality $\pi x \leq \pi_0$ satisfied by all points in X, but violated by point x^*.

The cutting plane method is given in Algorithm 3, which is taken and adapted from Wolsey [3]. It assumes that to solve the arising separation problem a known family of valid inequalities $\mathcal{F}$ is provided. If the separation problem can't be solved through a valid inequality in $\mathcal{F}$ and no integer solution has been found, the algorithm terminates. However, the strengthened formulation can be used in a subsequent branch & bound solution approach.

Algorithm 3: Cutting Plane Algorithm [3]

```
1  Function cutting_plane(P, F):
2      t ← 0
3      P^t ← P
4      while True do
5          Solve the linear program min{c^T x : x ∈ P^t} to obtain a
            solution x^{LP,t}.
6          if x^{LP,t} ∈ Z^n then
7              Terminate because x^{LP,t} is an optimal solution for IP.
8          else
9              Try to solve the separation problem for x^{LP,t} given F.
10             if ∃(π^t, π_0^t) ∈ F with π^t x^{LP,t} > π_0^t then
                   // A valid inequality cutting away x^{LP,t} has
                      been found.
11                 P^{t+1} ← P^t ∩ {x : π^t x ≤ π_0^t}
12                 t ← t + 1
13             else
14                 Terminate without an integer solution found.
```

2.2.3 *Branch & Cut*

The branch & cut algorithm is closely related to the linear programming based branch & bound algorithm introduced in Section 2.2.1. However, its main difference is that it applies a cutting plane algorithm to each subproblem (node in the branch & bound tree) to improve the corresponding LP relaxation.

Complex branch & cut algorithms underly state-of-the-art commercial MIP solvers such as CPLEX [29] or Gurobi [30].

2.3 METAHEURISTICS

Metaheuristics can be seen as high-level templates to develop problem specific heuristic optimization algorithms [31]. They are most prevalent for NP-hard optimization problems when exact solution methods fail [32].

The presentation in this section closely follows two good reference works written on the subject: Gendreau and Potvin [28], and Talbi [32].

Metaheuristics can be divided in *single-solution* based or *population* based:

- Single-solution based refers to a type of metaheuristic, which operates on one solution only and iteratively tries to move from its current solution to another one in the search space of possible solutions. The presentation here focuses on a specific single-solution based metaheuristic called *local search* (see Section 2.3.1).
- Population based methods maintain a set of solutions, which are iteratively improved. A famous example are *genetic algorithms* [33].

Let Ω be the set of all feasible solutions to an optimization problem. Then, Ω is called the *search space*.

Definition 8. [32] A neighbourhood function N is a mapping $N : \Omega \to 2^{\Omega}$ that assigns to each solution x of Ω a set of solutions $N(x) \subset \Omega$.

$N(x)$ is called the *neighbourhood* of x and a solution $x' \in N(x)$ a *neighbour* to x. Now, a *move* is defined as an operator m that takes as input a solution x and changes it to one of its neighbours $x' \in N(x)$.

2.3.1 *Local Search*

Local search is one of the most simple metaheuristics [28, 32]. It requires an initial solution x usually provided by a *construction* heuristic. Then, it generates the neighbourhood $N(x)$ of the current solution and selects a new improving solution $x' \in N(x)$ as the current one. This process is repeated until it reaches a local optimum or a termination criterion is fulfilled.

The pseudocode adapted from Talbi [32] is shown in Algorithm 4.

Gradient descent [34], which in its stochastic approximation is the state-of-the-art to train deep neural networks [35], can be seen as a local search algorithm on a continuous search space.

Algorithm 4: Local Search

```
1 Function LS():
2     x ← initial solution
3     while Termination criterion not fulfilled do
4         Generate N(x)
5         if there is no better neighbour then
6             break
7         else
8             Select a better neighbour x′ ∈ N(x)
9             x ← x′
```

2.3.2 *Hybrid Metaheuristics*

Hybrid metaheuristics are heuristics, which usually combine ideas from a metaheuristic with other algorithmic ideas from different fields or other metaheuristics. Combining multiple different strategies for optimization to exploit synergies gained traction with the no free lunch theorems for optimization [36, 37]. They imply that the improved performance of an optimization algorithm on a set of problems always comes from a drop of performance on other problem sets. Therefore, there can't exist one single algorithm outperforming all others on all possible problems.

An example of a hybrid heuristic combining strategies from two different metaheuristics are genetic algorithms applying a local search on new solutions obtained through recombination. In general, evolutionary algorithms employing a local search module are called *memetic algorithms* [28].

For this work especially relevant are hybrid heuristics, which combine ideas from metaheuristics with exact optimization techniques from the fields of linear programming (see Section 2.1) and integer linear programming (see Section 2.2). Such hybrid algorithms are termed *matheuristics*.

Exemplary, a MIP solver can be used to search through a neighbourhood of solutions defined by fixing a certain proportion of decision variables [38]. Another example for defining a neighbourhood is to add so called *local branching constraints* [39]. Given a bit-vector x representing a solution, they restrict the MIP to find new solutions having at most k bit-flips compared to x.

Recently, combining metaheuristics with machine learning algorithms has become popular. There are multiple approaches from learning certain decisions in optimization algorithms using reinforcement learning up to end-to-end learning using deep neural networks. For a general

survey of this emerging field see Bengio et al. [40]. In recent years graph neural networks [41, 42] often proofed to be key building blocks to effectively solve combinatorial optimization tasks with the help of machine learning. A survey focusing on graph neural networks for combinatorial optimization has recently been written by Cappart et al. [43].

There are a multitude of ways of how to exactly combine metaheuristics with other algorithmic techniques [28]. One of the most natural ways is to embed one algorithm into the other as done in memetic algorithms. This work can also be seen as following this approach. First a matheuristic (using techniques from linear programming, see Section 4.1) is employed to obtain an initial solution, which is then improved by a local search algorithm (see Section 4.2).

2.4 PROBLEM CLASSES

Mathematical models solvable using the previously introduced techniques are a standard tool for decision making in logistics and supply chain management [13, 44]. Crainic and Laporte [45] give an overview of common optimization problems arising in strategic, tactical and operational transportation planning.

In this section two related problem classes are introduced. First, the minimum cost flow problem [46] relevant for the slope scaling heuristic is introduced in Section 2.4.1. It also includes an important theorem about representing flow used in the local search heuristic from Section 4.2.

Section 2.4.2 introduces the general network design problem [16] of which the model introduced in Chapter 3 is a variation of. Network design problems are, among many other application areas, very prominent in freight planning [12].

2.4.1 *Network Flow Problems*

The following exposition closely follows Ahuja et al. [46].

Assume a directed graph $G = (\mathcal{N}, \mathcal{A})$ with $n = |\mathcal{N}|$ nodes and $m = |\mathcal{A}|$ arcs is given, in which each arc $(i, j) \in \mathcal{A}$ has costs c_{ij} and a maximal capacity u_{ij}. Furthermore, each node $i \in \mathcal{N}$ has a balance b_i, which when positive indicates supply of and when negative demand for flow. The sum over all supply and demand is assumed to be zero.

For simplicity, the formulation assumes $x_{ij} = 0$ if $(i,j) \notin \mathcal{A}$.

In the *minimum cost network flow problem* (MCFP) one searches flow-values x_{ij} defined on each arc $(i,j) \in \mathcal{A}$ as follows:

$$\min \sum_{(i,j)\in\mathcal{A}} c_{ij}x_{ij} \tag{2.5}$$

$$\sum_{j\in\mathcal{N}} x_{ij} - \sum_{j\in\mathcal{N}} x_{ji} = b_i \qquad \forall i \in \mathcal{N} \tag{2.6}$$

$$0 \leq x_{ij} \leq u_{ij} \qquad \forall (i,j) \in \mathcal{A} \tag{2.7}$$

with constraints (2.6) representing *flow conservation* and constraints (2.7) capacity restrictions.

The MCFP is solvable in polynomial time and in practice can be efficiently solved using the network simplex algorithm (see Section 2.1.3). Through certain choices of b_i, c_{ij} and u_{ij} other well-known problems such as the *shortest path problem* or the *maximum flow problem* can be obtained as special cases [46].

Flow Decomposition Theorem

In the MCFP flow has been defined on arcs. However, it is possible to define an equivalent flow on paths and cycles. To derive this, an *arc flow* is formally defined as a vector, which has an element x_{ij}, $\forall (i,j) \in \mathcal{A}$ and satisfies

$$\sum_{j\in\mathcal{N}} x_{ij} - \sum_{j\in\mathcal{N}} x_{ji} = -e_i \qquad \forall i \in \mathcal{N} \tag{2.8}$$

$$0 \leq x_{ij} \leq u_{ij} \qquad \forall (i,j) \in \mathcal{A} \tag{2.9}$$

If $e_i = -b_i \; \forall i \in \mathcal{N}$, the arc flow is feasible for the MCFP [46].

with $\sum_{i=1}^{n} e_i = 0$ [46]; e_i can but does not necesserily coincide with the supply or demand of a node and is called the *imbalance* of node i. An *excess node* is defined as a node with more inflow than outflow, hence $e_i > 0$ and a *deficit node* as one with more outflow than inflow, hence $e_i < 0$.

For an alternative flow definition, denote with $\mathcal{P}$ the set of all directed paths between any pair of nodes and with $\mathcal{W}$ the set of all directed cycles in G. Then, one defines a *path and cycle flow* by introducing flow variables $x_P \; \forall P \in \mathcal{P}$ and $x_W \; \forall W \in \mathcal{W}$.

The following theorem establishes that any path and cycle flow can be represented as an arc flow and vice versa.

Theorem 1 (Flow Decomposition Theorem)**.** [46] Every path and cycle flow has a unique representation as nonnegative arc flows. Conversely, every nonnegative arc flow can be represented as a path and cycle flow (though not necessarily uniquely) with the following two properties:

1. Every directed path with positive flow connects a deficit node to an excess node.
2. At most $n + m$ paths and cycles have nonzero flow; out of these, at most m cycles have nonzero flow.

One way to generalize this result to multicommodity network flow problems with $|K|$ commodities is to consider a separate flow decomposition for each commodity k. Then one can upper bound the number of nonzero paths and cycles by $|K| \cdot (n + m)$.

Due to the time-dependent structure of the graphs considered in this work, they do not have cycles. Thus, every flow decomposition is entirely made up of flow paths.

2.4.2 *Network Design Problems*

The following exposition closely follows the survey on network design problems by Magnanti and Wong [16] as well as the surveys by Gendron et al. [47] and by Crainic [12].

In network design problems, one is given a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ with a set of nodes $\mathcal{V}$ and arcs $\mathcal{A}$. Additionally, a commodity set $\mathcal{K}$ is given with each commodity $k \in \mathcal{K}$ having source nodes $O(k) \subseteq \mathcal{V}$ and demand nodes $D(k) \subseteq \mathcal{V}$ supplying o_i^k or demanding d_i^k commodities, depending on if $i \in O(k)$ or $i \in D(k)$, respectively. Now, one wishes to find a minimum cost selection of arcs and commodity flows on these arcs to satisfy the demands for each commodity.

The presentation focuses on the directed formulation, but the problem can be similarly modelled on an undirected graph [16].

Therefore, there are two sets of decision variable for each arc $(i, j) \in \mathcal{A}$: integer design variables y_{ij} with associated fixed-costs f_{ij} and real-valued flow variables x_{ij}^k with linear costs c_{ij}^k. In general, network design problems can have arbitrary (non-linear) cost functions with respect to the design and flow variables. However, then the problem can't be formulated as a linear mixed-integer program.

The general linear network design problem takes the following form:

$$\min \sum_{(i,j)\in\mathcal{A}} f_{ij}y_{ij} + \sum_{k\in\mathcal{K}}\sum_{(i,j)\in\mathcal{A}} c_{ij}^k x_{ij}^k \tag{2.10}$$

$$\sum_{j\in\mathcal{V}} x_{ij}^k - \sum_{j\in\mathcal{V}} x_{ji}^k = \begin{cases} o_i^k & \text{if } i \in O(k) \\ -d_i^k & \text{if } i \in D(k) \\ 0 & \text{otherwise} \end{cases} \quad \forall i \in \mathcal{V}, k \in \mathcal{K} \tag{2.11}$$

$$\sum_{k\in\mathcal{K}} x_{ij}^k \leq B_{ij}y_{ij} \qquad \forall (i,j) \in \mathcal{A} \tag{2.12}$$

$$(x,y) \in \mathcal{S} \tag{2.13}$$

$$x_{ij}^k \geq 0 \qquad \forall (i,j) \in \mathcal{A}, k \in \mathcal{K} \tag{2.14}$$

$$y_{ij} \in \mathcal{Y} \qquad \forall (i,j) \in \mathcal{A} \tag{2.15}$$

Constraints (2.11) are classic flow conversation constraints and inequalities (2.12) capacitate the flow with respect to arc choices. If B_{ij} is larger than the possible flow in the system, the problem is called *uncapacitated*.

Equation (2.13) and the set $\mathcal{S}$ refer to any problem specific side constraints such as topology restrictions. The set $\mathcal{Y}$ in constraints (2.15) could be $\{0,1\}$ and hence, an arc can either be chosen or not chosen, or $\mathcal{Y}$ can also be the set of natural number $\mathbb{N}_0$. Then it represents *units of facility installed* (quote from Crainic [12]).

For network design problems in sustainable planning see McKinnon et al. [14].

A prominent example of the network design problem common in transportation planning [12] and especially relevant for this work is the (capacitated) *fixed-charged network flow* problem (MCND). It arises when there are no problem specific constraints 2.13 and $\mathcal{Y} = \{0,1\}$.

The MCND is NP-hard and difficult to solve in practice. State-of-the-art heuristics mostly combine heuristic and exact solution methods and among the most recent works are Gendron et al. [19] and Akhavan Kazemzadeh et al. [20]. For a recent exact solution approach based on a cutting-plane algorithm see Chouman et al. [48] and for another recent work based on branch and prize see Gendron and Larose [49].

It is known that the LP-relaxation bounds of the MCND are very weak [47]. Therefore, one often adds the following valid inequalities (VI) [12, 19, 47]:

$$x_{ij}^k \leq d^k y_{ij} \qquad \forall (i,j) \in \mathcal{A}, k \in \mathcal{K} \tag{2.16}$$

with d^k representing the sum over all commodities k in the system. With respect to the integer solutions the valid inequalities (2.16) are

redundant with constraints (2.12), however, they are known to significantly strengthen the MIP formulation. Therefore, and as they can be shown to be facet defining for a relaxed version of the MCND, the inequalities (2.16) are called *strong linking constraints* and equations (2.12) *weak linking constraints*. Even though the strong linking constraints are probably the most simple and common VI, there are other known types of VI and Chouman et al. [48] give an overview of four other classes of valid inequalities for the MCND.

This section is closed by noting that network design formulations have impressive modelling power and the above presented model includes many famous problems such as the Steiner tree, traveling salesman, vehicle routing or facility location problem as special cases [16]. These problems are known to be computationally hard [50]. Therefore, the general network design problem inherits their (and the MCND's) difficulty and is again NP-hard.

3

MATHEMATICAL MODEL

The introductory paragraphs for Chapter 3 will appear in shortened form in Gosch et al. [1]. Section 3.1 with all its subsections and Section 3.2 with all its subsections except 3.2.5 will appear nearly identically in Gosch et al. [1].

This chapter presents the formal model developed and used in this work. Fundamentally two stages of the underlying network are differentiated.

First, Section 3.1 presents a *time-expanded network*. With Section 3.1.1, it includes a description of how to model a diverse set of commodities. Section 3.1.2 formally introduces the non-linear transportation tariffs with all-unit discounts used in this work. It derives cost functions covering these tariff structures extending the list of tariff cost functions presented in Harks et al. [21]. However, these cost functions are inherently non-linear and hence, result in a non-linear objective.

Therefore, in a second stage arcs representing transport relations between nodes can be replaced with more complex graph-structures to obtain a linear mixed-integer programming (MIP) formulation called the *tariff-expanded network* (see Section 3.2). These graph-structures are called *graph-gadgets* [21] and Section 3.2.1 develops new graph-gadgets for the tariff introduced in Section 3.1.2. On this basis Sections 3.2.2 to 3.2.4 develop the linear MIP formulation allowing to apply exact MIP-solvers to the problem. Last but not least, Section 3.2.5 introduces multiple strong valid inequalities for the MIP formulation.

If one only optimizes for emissions the time- and tariff-expanded networks coincide as the time-expanded network only consists of arcs with linear cost factors (representing the emissions). Furthermore, the heuristic solution approaches also operate on the time-expanded network due to being able to handle the non-linear cost functions induced by the employed tariffs.

This chapter is concluded with Section 3.2.6 showing that the model presented here includes the tactical transportation problem (TTP) introduced by Harks et al. [21] as a special case. Therefore, the current problem is called the generalized tactical transportation problem (GTTP) and inherits the NP-hardness of the TTP.

3.1 TIME-EXPANDED NETWORK

The time-expanded network $G_{\mathcal{T}}$ is constructed for a certain time horizon T (e.g. 7, 14 or 30 days) with the individual time periods summarized in the set $\mathcal{T} = \{1, ..., T\}$. $G_{\mathcal{T}}$ is built up from a set of base nodes copied T-times resulting into a time-expanded node set

$\mathcal{V}_\mathcal{T}$. Base nodes are either: physical facilities (e.g. warehouses, factories or transshipment points) with each facility being part of a demand region, demand nodes used to represent demand regions, or bin nodes to remove unused or expired goods form the system. There is only one bin node in the set of base nodes.

Then, the arc set $\mathcal{A}_\mathcal{T}$ consists of transport relations between these nodes in different time periods. Transport relations can be of arbitrary types and we distinguish transportation modes $\mathcal{M} = \{L, R, S\}$ consisting of lorry (L), rail (R) and ship (S), storage arcs connecting the same facility at two consecutive time periods denoted by type C, and artificial arcs connecting each facility to its regional demand node as well as to the bin node. These use the type-symbol Ω.

Network Structure

The facilities are connected among themselves based on physical realities. If for example two facilities $i, j \in \mathcal{V}_\mathcal{T}$ are connected by a road network, then there exists an arc $a = (i, j, L) \in \mathcal{A}_\mathcal{T}$ with corresponding non-zero distance d_a and non-zero travel time τ_a. Storage arcs have a distance of zero but a travel-time of one. For each time-period, there is one bin node and each facility is connected to it with distance and travel time of zero.

Facilities in a demand region are connected to the associated demand node with distance and travel times of zero using mode Ω. Hence, demand in the region the warehouse is located in can be satisfied directly out of stock. This is due to the fact that the model is only concerned with successful transportation into a region to fulfil its demand and the inner-regional distribution to customers should be addressed by an operational model.

Furthermore, each facility is connected to every other demand region except its own using mode L with non-zero distances and travel times. These connections represent direct deliveries by lorries into a demand region without first delivering into a regional facility or using any form of intermodal transport.

3.1.1 *Commodities*

All facilities are supplied with a certain stock of products also called commodities each time period acting as *source* nodes. Demand nodes have associated commodity demands for each time period acting as *sinks*. The set of commodities is denoted by $\mathcal{K}$.

In general, products can be highly diverse in their volume and weight henceforth called the products *properties* (see also Harks et al. [21]). As

an example, it does make a non-negligible impact on the transportation costs and available resources if one transports a certain number of styrofoam sheets compared to the same number of steel beams. Therefore, each commodity $k \in \mathcal{K}$ has an associated extent p_{kl} for each property $l \in \mathcal{P}$.

Additionally, products have different transportation and storage characteristics. As an example, some goods may require cooling to specific temperatures or can be perishable. To model different transportation requirements, we introduce a *product type* concept and associate to each commodity a specific type from a set of types Σ. Only goods of the same type can be transported together. In the instances generated for this work two types of goods are separated, those whose storage-containers require electricity and those whose storage-containers don't.

Additionally, to address perishability commodities $k \in \mathcal{K}$ can have varying *lifetimes* Δt_k. As a result, a commodity is defined as perishable if its lifetime is smaller than the number of time-periods in the network. Consequently, commodities can be grouped into a set of perishable ones $\mathcal{K}_\Delta$ and a set of non-perishable ones $\mathcal{K}_\mathcal{T}$.

It is possible to associate production timestamps to non-perishable goods and in that simplify formal presentation. However, this leads to an unnecessary model blowup.

In order to correctly handle perishability in the model, each perishable commodity has an associated *production timestamp* $t \in \mathcal{T}$ and is counted as expired in time-periods greater than $t + \Delta t_k$. Therefore, with $x_{k_t} \in \mathbb{R}_0^+$ the *flow* of perishable commodity k produced at time t is denoted and with $x_k \in \mathbb{R}_0^+$ the flow of non-perishable commodity k. These can be pooled together into a *flow vector* $x \in \mathbb{R}_0^{|\mathcal{K}_\mathcal{T}|+|\mathcal{K}_\Delta||\mathcal{T}|}$.

The summed-up extent of property $l \in \mathcal{P}$ over all commodities of type $\sigma \in \Sigma$ in x is given by the flow-sum function $P_l^\sigma(x) := \sum_{k \in \mathcal{K}_\mathcal{T}^\sigma} p_{kl} x_k + \sum_{t \in \mathcal{T}} \sum_{k \in \mathcal{K}_\Delta^\sigma} p_{kl} x_{k_t}$. The sets $\mathcal{K}_\mathcal{T}^\sigma$ and $\mathcal{K}_\Delta^\sigma$ comprise all non-perishable or perishable goods of type σ.

3.1.2 *Tariffs*

Multiple available tariffs can be modelled by parallel arcs.

Each transport relation has one associated tariff for each product type. Harks et al. [21] give a good overview of possible tariffs and their resulting cost functions. In this work, we are especially concerned with *all-unit quantity discounts* as they are the most prevalent [15].

Corresponding tariffs usually have a set of cost levels $\mathcal{N} = \{1, ..., N\}$ defining different cost rates depending on the shipped volume or weight. Therefore, Harks et al. [21] develop a cost function with different linear cost rates depending on the sent flow extent in some property $ł \in \mathcal{P}$ as follows.

Each cost level $n \in \mathcal{N}$ is applied starting from transport volume β_n and has the linear cost rate c_n. As a result, the cost function by Harks et al. [21] for flow vector x and a specific product type σ is given as

Harks et al. [21] do not have a product type concept and aggregate the property-extent over all products.

$$C(x) = \min_{n \in \mathcal{N}} \{c_n \cdot \max\{P_l^\sigma(x), \beta_n\}\} \tag{3.1}$$

However, only linearly pricing transport volumes does not adequately capture the diversity of all-unit volume discount tariffs employed by freight transportation companies. Especially, it is not applicable to constant cost levels for different transport weight or quantity intervals or a combination of a fixed-cost rate with volume-dependent linear costs resulting in piecewise-linear cost levels (see Figure 3.1). An overview of such tariffs with more complex cost structures is given in Kempkes and Koberstein [51]. Below, we formalize these tariffs used in our work.

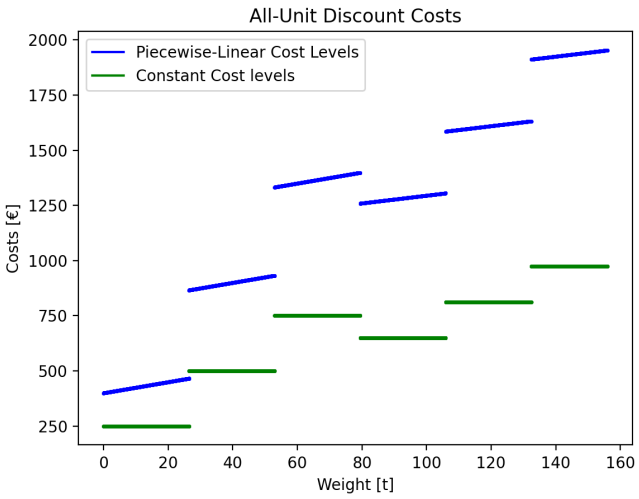


Figure 3.1: In this work constant cost levels are used for rail and ship connections and piecewise-linear cost levels for lorry tariffs. The displayed tariffs have one cost level for the first three transport units (TUs) and a second discounted cost level starting with the fourth ordered TU. In this work, one TU always corresponds to one 40-feet ISO container.

3.1.2.1 *Constant Cost Levels with All-Unit Discounts*

Constant cost levels for different weight or quantity intervals are, among others, often encountered in rail cargo shipments. In them for each cost level $n \in \mathcal{N}$ a different fixed-cost rate f_n is assumed.

In theory these cost levels can have arbitrary interval lengths, but here a fixed property- and type-dependent interval length of B_l^σ is assumed and interpreted as the capacity in direction $l \in \mathcal{P}$ of one installed *transport unit (TU)* on this transport relation. Exemplary, for rail or ship cargo shipments one transport unit could be one ISO container (see

Figure 3.1). Then, the number of installed TUs for product type σ is given by $\tilde{y}^\sigma = \max_{l \in \mathcal{P}} \{ \lceil \frac{P_l^\sigma(x)}{B_l^\sigma} \rceil \}$.

In the mixed-integer problem installed TUs are additional integer decision variables (see Section 3.2.2). Now, β_n denotes the minimum number of TUs at which cost level n starts to be applicable. The starting cost of the n-th level is b_n and represents the base cost of shipping β_n containers. This yields the cost function (3.2).

$$C(x) = \min_{n \in \mathcal{N}} \{ f_n \cdot \max\{\tilde{y}^\sigma - \beta_n, 0\} + b_n \} \tag{3.2}$$

Exemplary, b_n can represent the price of β_n TUs with the new cost rate f_n, i.e. $b_n = f_n \beta_n$. If $f_n \leq f_{n-1}$, this represents an all-unit discount structure. However, if b_n represents the costs of $\beta_n - 1$ TUs to the previous price rate f_{n-1} plus the costs of one TU to the current rate f_n, it would represent incremental discounts. Note that to formulate the cost function not in terms of installed TUs, but in term of single-property levels (as for example weight levels), replace $\tilde{y}^\sigma$ with $\tilde{y}_l^\sigma = \lceil \frac{P_l^\sigma(x)}{B_l^\sigma} \rceil$.

3.1.2.2 *Piecewise-Linear Cost Levels with All-Unit Discounts*

Tariffs with different piecewise linear cost levels are very similar to the constant cost levels case except that for each discount level n, we additionally assume variable costs c_n for the actual volume or weight of flow. The resulting cost function writes

$$C(x) = \min_{n \in \mathcal{N}} \{ f_n \cdot \max\{\tilde{y}^\sigma - \beta_n, 0\} + c_n \cdot \max\{P_l^\sigma(x) - B_l^\sigma(\beta_n - 1), 0\} + b_n \} \tag{3.3}$$

Again, setting $b_n = f_n \beta_n + c_n B_l^\sigma (\beta_n - 1)$ with $f_n \leq f_{n-1}$ and $c_n \leq c_{n-1}$ results in all-unit discounts. Note that the linear costs only depend on one property. This cost function exemplary arises when modelling transportation costs with lorries. A fixed costs part arises for each commissioned truck due to various singular factors such as driver wages. Linear costs arise due to increasing fuel consumption based on its actual weight.

3.2 TARIFF-EXPANDED NETWORK

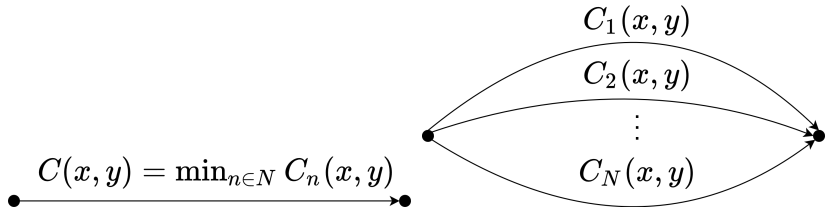
Due to the complex tariff structures employed if one would define a mixed-integer model on the time-expanded graph $G_\mathcal{T}$, the arcs representing transport relations would have highly non-linear cost functions. This would make it difficult to apply established and pow-

erful solution techniques for integer linear programs. To address this problem, two specific concepts are used.

First, we introduce a second set of decision variables y representing the number of installed *transport units (TUs)* on each arc. A TU represents a certain amount of capacity bought for product flow on this arc and is a non-negative integer variable. These TUs can be containers, trucks or similar. This concept naturally leads to a capacitated network design formulation but alone is not enough to result into a linear formulation. A similar concept is used by Harks et al. [21] and more general *units of facility installed* (taken from Crainic [12]) is common in the network design literature.

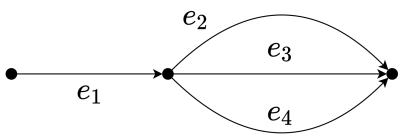
Secondly, each simple arc in $G_{\mathcal{T}}$ representing a transport relation is replaced by a more complex graph structure known as *graph gadgets*. These gadgets are again made up of arcs and nodes and - together with the above decision variables - allow to linearly model the non-linear cost structures induced by the employed tariffs.

The resulting network $G = (\mathcal{V}, \mathcal{A})$ constructed from $G_{\mathcal{T}}$ is called *tariff-expanded*. Employing tariff-expansion allows to derive a linear mixed-integer model defined on G which is a variant of the fixed-charge network flow problem.

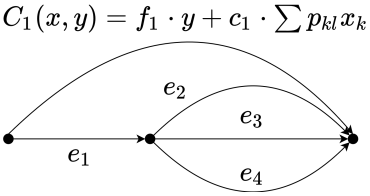


(a) Transport relation with non-linear cost function.

(b) Add parallel arcs for each cost level.



(c) Replace each parallel arc with four arcs and one gadget node.



(d) Simplify graph structure of first tariff level.

Figure 3.2: Graph gadget construction.

3.2.1 *Graph Gadgets*

Graph gadgets were first introduced by [21] for their employed transportation tariffs. In this section, we extend their approach by deriv-

ing graph gadgets for tariffs with constant cost levels t_{const} and for tariffs with piecewise-linear cost levels t_{lin}. It is assumed that the cost levels are discount structures, i.e. have monotonic cost levels $C_n(x,y) \leq C_{n-1}(x,y)$. The description starts with the graph gadget construction for the piecewise-linear case.

First, the number of transport containers depending on the flow $\tilde{y}$ is replaced by the decision variable y leading to a cost function $C(x,y)$ depending on both x and y. Therefore, it is possible and done in practice to book more TUs than necessary to get a higher discount (see Figure 3.1).

For the second step assume one t_{lin} is active on a transport relation. Then, the graph gadget is constructed using the steps outlined in Figure 3.2. At first, the transport relation is replaced by one arc for each cost level $n \in \mathcal{N}$ with cost function $C_n(x,y)$ (see Figures 3.2a and 3.2b). Then, each of these arcs is replaced by the graph structure shown in Figure 3.2c.

Now, the decision variables associated with the arcs in the graph gadget in Figure 3.2c have to be capacitated and priced in the following way to model t_{lin}. Variable y_1 associated to e_1 is set to be binary, i.e. $y_1 \leq 1$. The TU-capacity of e_1 is set to infinity. The costs of e_1 are set to the starting cost of the tariff level this graph structure represents, i.e. $c(e_1) = b_n$. This means any flow over this graph structure has to pay at least the starting costs of this tariff. As these already include the costs of $\beta_n - 1$ fully filled TUs, $c(e_2) = 0$ with $y_2 \leq \beta_n - 1$. As b_n also includes the price of the empty β_n-th TU, costs on e_3 are set to the linear costs only $c(e_3) = c_n \cdot P_l^\sigma(x_3)$ and e_3 is restricted to be chosen only once $y_3 \leq 1$. The capacity in property $l \in \mathcal{P}$ of each TU on e_2 to e_4 is set to the physical capacity B_l^σ of one real TU employed on this transport relation. Lastly, the costs of e_4 are set to the cost rate of the new tariff level $c(e_4) = f_n \cdot y_4 + c_n \cdot P_l^\sigma(x_4)$.

An optimal solution will automatically first fill arc e_2 then e_3 and only then use e_4. If each tariff level is replaced by such a graph structure, again by an argument of optimality, only this graph structure will be chosen resulting into the minimum cost tariff level for a given flow x.

However, the graph structure for the first tariff level can be simplified to only one arc with $C_1(x,y) = f_1 \cdot y + c_1 \cdot P_l^\sigma(x)$ as $b_1 = \beta_1 = 0$. If only a small number of tariff-levels exists, this significantly reduces model size. The graph gadget developed above includes a graph gadget for constant cost tariff levels as special case when $c_n = 0$.

From a pure mathematical perspective, it is possible to model the employed tariffs using only two outer edges. However, this would not preserve the number of employed TUs for counting purposes necessary for handling capacity restrictions (see Section 3.2.4).

3.2.2 *Decision Variables*

To formulate the objective and constraints, we formally define the decision variables on the network $G = (\mathcal{V}, \mathcal{A})$. Note that the node set $\mathcal{V}$ decomposes into a set of facilities $\mathcal{F}$, demand nodes $\mathcal{D}$, bin nodes $\mathcal{B}$, and gadget nodes $\mathcal{G}$ and that each node is uniquely identified by in index-time pair (i, τ). As a result, an arc $a \in \mathcal{A}$ connecting (i, τ) with (j, τ') is identified by a five tuple (i, j, τ, τ', m) with m representing the type of the transport relation. Arcs connecting to and from a gadget node inherit the transport mode m of the original transport relation.

The *flow of non-perishable goods* $x_a^k \in \mathbb{R}_0^+$ is defined on all arcs $a \in \mathcal{A}$ and for each non-perishable commodity $k \in \mathcal{K}_{\mathcal{T}}$. The *flow of perishable goods* $x_a^{k_t} \in \mathbb{R}_0^+$ is defined on all arcs $a = (i, j, \tau, \tau', m) \in \mathcal{A}$ with $\tau' \leq t + \Delta t_k$ and for each perishable commodity $k \in \mathcal{K}$.. The second decision variable $y_a^\sigma \in \mathbb{N}_0$ describing the *number of transport units* of type $\sigma \in \Sigma$ installed is defined for all $a \in \mathcal{A}$ and $\sigma \in \Sigma$.

All flow-variables on an arc a can be collected into a flow vector x_a. These can be again stacked to one big flow vector x. Analogously a TU vector y is constructed.

3.2.3 *Objective*

The objective consists of a weighted sum of costs and emissions (for a review on green network design see McKinnon et al. [14], for green planning techniques in general see Bektaş et al. [52]). The (linear) cost term includes system-wide transportation, handling and storage costs as given by equation (3.4).

Handling costs are incorporated into the transportation arcs and dependent on the connected facilities and mode of transportation.

$$C(x, y) = \sum_{a \in \mathcal{A}} \sum_{\sigma \in \Sigma} \left[f_a^\sigma y_a^\sigma + c_a^\sigma P_{l(a)}^\sigma (x_a) \right] \tag{3.4}$$

Fixed-costs per transport unit of type σ on arc a are f_a^σ. Flow extent in property $l \in \mathcal{P}$ of type σ on arc a is linearly priced by c_a^σ. The priced flow-extent property $l \in \mathcal{P}$ depends on the employed tariff. Therefore, it is dependent on the looked upon arc a and written as a function thereof $l(a)$.

In this work artificial connections incident to the bin node as well as connections to a regional demand node are assumed to have zero associated costs.

The emissions term (3.5) looks very similar except it replaces fixed-costs with fixed CO_2e emissions Δ_a^σ per TU and variable CO_2e emissions δ_a^σ per flow extent.

$$\Delta(x, y) = \sum_{a \in \mathcal{A}} \sum_{\sigma \in \Sigma} \left[\Delta_a^\sigma y_a^\sigma + \delta_a^\sigma P_{l(a)}^\sigma (x_a) \right] \tag{3.5}$$

The combined objective reads

$$\min_{x,y} \lambda_1 C(x,y) + \lambda_2 \Delta(x,y) \tag{3.6}$$

Different choices of λ_1 and λ_2 allow different CO_2e pricing schemes. CO_2e pricing can be ignored by setting $\lambda_2 = 0$. Minimizing for emissions only is enabled by setting $\lambda_1 = 0$.

3.2.4 *Constraints*

The model is presented in a cut-set formulation and $\delta^-(v)$ and $\delta^+(v)$ have their usual meaning of all incoming or outgoing arcs, respectively, from node $v \in \mathcal{V}$. Additionally, mode-specific cut-sets are defined for each facility $v \in \mathcal{F}$ with $\delta_m^-(v)$ representing the set of all incoming arcs of type m.

One could define $\delta_m^+(v)$ analogously. But to formulate handling capacity constraints in the model, we need to capture the correct number of outgoing transport units of a specific type using the arcs in $\delta_m^+(v)$. In its analogous definition, it will be distorted due to the graph-gadgets. To correct this distortion, in $\delta_m^+(v)$ the arcs connecting node v to graph-gadget nodes are not included and in return the outgoing arcs of graph-gadget nodes of type m are added.

To formalize the mode-specific cut-sets, note that the arc set $\mathcal{A}$ can be decomposed into mode-specific arc-sets $\mathcal{A}_m := \{(i,j,\tau,\tau',m') \in A | m' = m\}$. This allows to define mode-specific cut-set $\delta_m^-(v)$ as $\delta^-(v) \cap A_m$ for all $v \in \mathcal{F}$.

As mentioned above, to correctly count outgoing TUs using $\delta_m^+(v)$, we don't include the arcs connecting node v to graph-gadget nodes and in return add the outgoing arcs of graph-gadget nodes of type m.

Denote by $\mathcal{G}(v) = \{v' | (v,v',\tau,\tau',m) \in A, v' \in \mathcal{G}\}$ the set of graph-gadget nodes which are direct successors to node v. Remember that for $\delta_m^+(v)$ one wants to exclude arcs going into graph-gadget nodes but include arcs going out of graph-gadget nodes. This leads to the following formal definition:

Add outgoing arcs from associated gadget nodes to the outgoing arc-set ...

$$\delta_m^+(v) := \Big(\overbrace{\underbrace{\big(\delta^+(v) \cup \bigcup_{v' \in \mathcal{G}(v)} \delta^+(v')\big) \cap A_m}_{\text{... but only choose arcs of mode m}} \Big) \setminus \bigcup_{v' \in \mathcal{G}(v)} \delta_m^-(v')}^{\text{... and exclude arcs connecting node v to associated gadget nodes.}}$$

.

Having defined the used cut-sets, the constraints read as follows:

$$\sum_{a\in\delta^+(v)} x_a^k - \sum_{a\in\delta^-(v)} x_a^k = b_v^k \qquad \forall k \in \mathcal{K}_\mathcal{T}, v \in \mathcal{V}\setminus\mathcal{B} \tag{3.7}$$

$$\sum_{a\in\delta^+(v)} x_a^{k_t} - \sum_{a\in\delta^-(v)} x_a^{k_t} = s_v^{k_t} \qquad \forall k \in \mathcal{K}_\Delta, t \in \mathcal{T}, v \in \mathcal{F}\cup\mathcal{G} \tag{3.8}$$

$$\sum_{t=0}^{\tau}\sum_{a\in\delta^-(v)} x_a^{k_t} = \omega_v^k \qquad \forall k \in \mathcal{K}_\Delta, v = (i,\tau) \in \mathcal{D} \tag{3.9}$$

$$\sum_{k\in\mathcal{K}_\mathcal{T}^\sigma} p_{kl}x_a^k + \sum_{k\in\mathcal{K}_\Delta^\sigma}\sum_{t=0}^{\tau} p_{kl}x_a^{k_t} \le B_{al}^\sigma y_a^\sigma \qquad \forall l \in \mathcal{P}, \sigma \in \Sigma, a \in \mathcal{A} \tag{3.10}$$

$$y_a^\sigma \le u_a^\sigma \qquad \forall \sigma \in \Sigma, a \in \mathcal{A} \tag{3.11}$$

$$\sum_{\sigma\in\Sigma}\sum_{a\in\delta_m^\circ(v)} y_a^\sigma \le h_v^{m\circ} \qquad \forall \circ \in \{+,-\}, m \in \mathcal{M}, v \in \mathcal{F} \tag{3.12}$$

$$\sum_{\sigma\in\Sigma}\sum_{a\in\delta_m^-(v)\cup\delta_m^+(v)} y_a^\sigma \le h_v^m \qquad \forall m \in \mathcal{M}, v \in \mathcal{F} \tag{3.13}$$

The constraints can be grouped into *flow-conservation constraints* (3.7)-(3.9) and *capacity constraints* (3.10)-(3.13).

Constraint (3.7) defines flow-conservation for non-perishable goods with a positive b_v^k to represent a source for commodity k and negative to represent demand for commodity k. The following two constraints handle the more complex case of perishable goods.

Equation (3.8) concerns the conservation of flow through facility and gadget nodes. For gadget nodes $s_v^{k_t} = 0$ as they have no supply of goods. If a facility acts as a source for commodity k in time period t, $s_v^{k_t} > 0$.

The third flow-conservation constraint (3.9) regards demand satisfaction of perishable goods. Each region has a time-dependent demand ω_v^k for perishable commodity k which must be exactly fulfilled by its incoming flow. Demand for a commodity k can be satisfied by any commodity k_t independent of its production timestamp t.

Constraint (3.10) links the flow of goods with the necessary transport units. B_{al}^σ refers to the maximal extend of property l a TU on arc a for commodities of type σ can transport. For this study on arcs of mode $m \in \mathcal{M}$, B_{al}^σ always corresponds to one 40-ft ISO container except in the case when a graph-gadget requires adjusting the capacity.

Constraint (3.11) establishes a maximum number of TUs for type σ installable on arc a. Together with (3.10) it capacitates the flow on a given arc. Storage volume is capacitated by setting $B_{al}^\sigma = 1$ and u_a^σ to the real world warehouse capacity for the respective property l.

This work assumes warehouses are only capacitated in volume and uncapacitated in all other properties.

Equation (3.12) capacitates incoming (−) and outgoing (+) handling operations separately. This captures the operational truth in cross-docks with separated incoming and outgoing docks or conceptually similar architectures like transshipment-points with mode-switches.

Lastly, constraint (3.13) capacitates the sum of containers handled both for incoming and outgoing operation. This is relevant for facilities like a classic warehouse with a specific number of docks for lorries not strictly split into incoming and outgoing docks. During handling commodity types are not distinguished (exemplary, a container crane is oblivious to the fact that a specific container has a cooling module or not).

3.2.5 *Strengthening Valid Inequalities*

In this section three different types of valid inequalities (VI) are developed. These VI are specifically devised to exploit the problem structure induced by the employed graph-gadgets. They are compatible with one another and strengthen the LP relaxation of the tariff-expanded MIP formulation.

A complete characterization of the polyhedral hierarchy induced by the strengthening constraints is given in Section 5.2.2.1 and based upon empirical results as well as the theoretical investigation presented here.

3.2.5.1 *Strengthened Capacity Constraints*

Equation (2.16) in Section 2.4.2 defines strengthening inequalities for the fixed-charge network flow problem (MCND). To remind the reader they are stated again in equation (3.14):

$$x_a^k \leq d^k y_a \qquad \forall k \in \mathcal{K}, a \in \mathcal{A} \tag{3.14}$$

In the MCND the integer variables are binary and d^k refers to the sum over all commodities k in the system.

To adapt these VI to the GTTP to strengthen constraints (3.10) one has to incorporate product dimensions as follows:

$$x_a^k \leq b_a^k y_a^\sigma \qquad \forall \sigma \in \Sigma, k \in \mathcal{K}_\mathcal{T}^\sigma, a \in \mathcal{A} \tag{3.15}$$

$$x_a^{k_t} \leq b_a^{k_t} y_a^\sigma \qquad \forall \sigma \in \Sigma, k \in \mathcal{K}_\Delta^\sigma, t \in \mathcal{T}, a \in \mathcal{A} \tag{3.16}$$

Constants b_a^k or $b_a^{k_t}$ are the sums over all supply of commodity k or k_t, which is present in the system up to the time period arc a starts in and, which can reach arc a based on the network topology.

However, VI (3.15) and (3.16) are only effective if the volume and weight (the property extents) of the summed commodities is less than a transport unit on the looked upon arc can carry. Mathematically speaking, if $B^{\sigma}_{al} > p_{kl} b^{k}_{a}$, $\forall l \in \mathcal{P}$ or $B^{\sigma}_{al} > p_{kl} b^{k_t}_{a}$, $\forall l \in \mathcal{P}$, respectively. This is very likely not the case as in most time periods the supply of products in the system has a much larger volume and weight than a transport unit B^{σ}_{al}, $l \in \mathcal{P}$ provides capacity for (see Harks et al. [21] for a similar argument).

Nonetheless, selectively applying these VI can have great effects as argued in the following. Notice that in the tariff-expanded network most transport relations are replaced with graph-gadgets and each tariff-level has a graph-structure as shown in Figure 3.3 (for a general explanation of the graph-gadgets see Section 3.2.1).

Arc e_1 in such a graph-structure connects a physical node with a gadget-node and always represents a tariff-level choice. It has infinite capacity and the associated design variable is binary. Hence, the associated linking constraints (3.10) take the form of classic big-M constraints. The big-M value can be calculated based on the available commodities of the respective type and their extent in the system at the time period arc e_1 starts in (see Appendix A.1 on how available commodities are counted for the instances in this work). For this arc-type the above argument of ineffectiveness does not apply.

Applying constraints (3.15) and (3.16) only to these type of arcs empirically proofs to significantly improve the quality of the lower bound obtained by the LP relaxation (see results in Section 5.2.2.1). Furthermore, they aid the branch and cut process in commercial MIP-solvers (see Table 5.2 in Section 5.2.2).

When employing the strengthening constraint not only to the above binary arcs but all arcs for which the effectiveness criterion holds. One can still observe a slight strengthening of the model mostly driven by VI (3.16). This is due to the sum of available perishable goods produced at a certain time period is significantly smaller than the sum of available non-perishable goods over all time periods and hence, the effectiveness criterion way more often fulfilled. However, this very slight increase in the LP relaxed solution comes at the cost of having multiplied the number of strengthening constraints. This leads to a model blow-up, which hinders the solution process due to among others disproportionately slowing down the LP solving step. Therefore, this approach is not used.

3.2.5.2 *Intra-Tariff-Level Strengthening Constraints*

Transport relations usually have multiple tariff levels and hence, multiple graph-structures as shown in Figure 3.3 in parallel. Note that

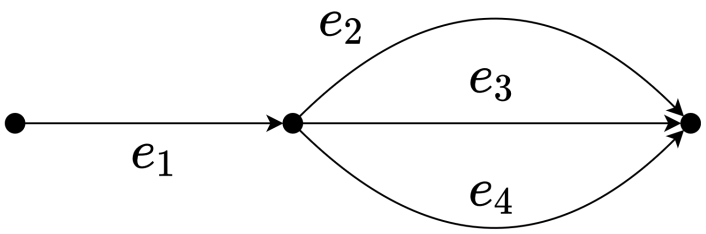


Figure 3.3: Graph-gadget structure for one cost level for a transport relation with an all-unit volume discount tariff.

the first (starting) tariff level can be represented by a single arc (see Figure 3.2d).

The main idea behind *intra-tariff-level strengthening constraints* is the insight that an optimal integer solution would never choose multiple tariff-levels together. However, an LP relaxed solution might use multiple tariff-levels fractionally. To combat this possibility a set of constraints are developed, which reduce the solution-space with respect to tariff-level combinations.

In the current work transport relations usually have three tariff levels. Hence, the following presentation focuses on this case but it can be generalized to arbitrary numbers of tariff levels. Therefore, the number of containers of the first tariff-level arc is denoted y_1. For the other two tariff levels each, a graph-structure as shown in Figure 3.3 is inserted. The design variable corresponding to arc e_1 for the second tariff level is denoted y_2 and the design variable corresponding to arc e_1 in the third tariff level is denoted y_3.

The added constraints read as follows:

$$y_1 \leq UB_1(1 - y_2) \tag{3.17}$$
$$y_1 \leq UB_1(1 - y_3) \tag{3.18}$$
$$y_2 \leq 1 - \frac{y_1}{UB_1} \tag{3.19}$$
$$y_2 \leq 1 - y_3 \tag{3.20}$$
$$y_3 \leq 1 - \frac{y_1}{UB_1} \tag{3.21}$$
$$y_3 \leq 1 - y_2 \tag{3.22}$$

Here, UB_1 refers to the upper bound on the number of TUs choosable in the first tariff level. In the investigated instances, this number is usually a small positive one-digit integer. Remember that y_2 and y_3 are binary choice variables.

These constraints are no ordinary valid inequalities in the sense that they do cut away sub-optimal integer solutions. This is easy to see as

for example choosing the second and third tariff levels at once is a valid solution for both the MIP and its LP-relaxation. However, having chosen y_2 inequality (3.22) makes it infeasible to also choose y_3 (or a fraction thereof). That an optimal integer solution never uses two tariff levels at once is explained in Section 3.2.1 and is connected to the fact of monotonic cost levels and the structure of the graph-gadgets.

The above argument proofs that the polyhedron of the LP relaxation is made smaller and hence, these constraints indeed strengthen the formulation. However, experimental results from Section 5.2.2.1 show that these constraints to not increase the lower bound found by the LP relaxation. This can be explained by the conjecture that monotonic cost levels not only imply no optimal integral solution with two chosen cost levels at once but also that no optimal LP relaxed solutions with more than one cost level chosen exist.

Even though the lower bound is not improved, the constraints introduced in this section show superior performance in the branch and cut solution process of commercial MIP solvers (see Table 5.2 in Section 5.2.2). This can be explained by the significant reduction of the feasible solution space to explore by cutting away sub-optimal integer solutions.

3.2.5.3 *Inner-Tariff-Level Strengthening Constraints*

The main idea behind *inner-tariff-level strengthening constraints* is to make choosing arcs e_2, e_3 and e_4 in Figure 3.3 dependent on choosing the relevant tariff level, i.e. on choosing e_1.

Denote by y_1 up to y_4 the design variables corresponding to the above-mentioned arcs. Then the strengthening constraints take the following form:

$$y_2 \leq UB_2 y_1 \tag{3.23}$$
$$y_3 \leq UB_3 y_1 \tag{3.24}$$
$$y_4 \leq UB_4 y_1 \tag{3.25}$$

The constants UB_i refer to upper bounds for y_i. These are set in the graph-gadget construction (see Section 3.2.1). UB_4 can be set to the maximum number of ordered TUs left (when y_2 and y_3 are at their upper bounds) before reaching the next tariff level. For the last tariff-level UB_4 has no natural small upper bound. Therefore, one can omit constraints (3.25) for the last tariff level.

The empirical results in Section 5.2.2.1 show that inner-tariff-level strengthening constraints indeed cut away optimal LP-relaxed solutions and hence, strengthen the formulation.

As these VI only concern the choice of design variables inside one tariff cost level, they still allow solutions choosing multiple tariff levels at once. This proofs the incomparability of the polyhedra induced by intra-tariff-level compared to inner-tariff-level strengthening constraints.

Note that inner-tariff-level strengthening constraints also cut away feasible but sub-optimal integer solutions, which unnecessarily choose some y_2, y_3 or y_4 but not y_1. These can never be optimal solutions as these choices incur costs but can't transport any flow due to the flow-conservation constraints (3.7) and (3.8), together with y_1 not chosen.

3.2.6 *Hardness Result*

Harks et al. [21] show that the TTP is NP-hard. The following theorem proofs the NP-hardness of the GTTP.

Theorem 2. The GTTP generalizes the TTP introduced in Harks et al. [21].

Proof. It is shown that the TTP arises as a special case of the GTTP.

Consider instances with $\sum_{v \in V} b_v^k = 0$ for all $k \in \mathcal{K}_\mathcal{T}$, only one commodity type and no perishable goods. Then there is no flow to the bin node and there are no constraints (3.8) and (3.9). Furthermore, the second term on the left-hand side of constraint (3.10) drops out. Lastly, only consider those instances with non-restricting (or unbounded) handling capacities. Therefore, the handling capacity constraints (3.12) and (3.13) have no effect and can be omitted. Now the problem formulation takes the general form of the TTP. Therefore, the GTTP solves the TTP which arises as a special case from the GTTP. □

4

HEURISTICS

The introductory text in Chapter 4, and Section 4.1 including all subsections of Section 4.1, will appear in significantly shortened form in Gosch et al. [1]. The same is true for Section 4.2 with all subsections except Section 4.2.2.3 to appear in significantly shortened form and without pseudocode in Gosch et al. [1].

In this chapter a hybrid heuristic composed of two parts is developed. The first part consists of a slope scaling heuristic (SSC) presented in Section 4.1 to construct a good solution in minimal time. SSC has original been developed for the MCND [17, 18], however, in this work it is generalized to non-negative integer variables and hence, applicable to a much broader range of network design problems including the model from Chapter 3. Additionally, Section 5.2.3 shows that in the general integer regime, SSC's prime design paradigm of how to scale the slope is challenged and a more fundamental property of monotonicity uncovered.

In Section 4.2 a local search (LS) algorithm is developed, which comprises the second part of the hybrid heuristic and in its design is heavily inspired by Harks et al. [21]. It operates on an initial solution generated by the slope scaling mechanism. The LS is based upon a path-decomposition of flow and removes individual paths to reroute freed products on more optimal routes. However, Section 4.2 introduces a new rerouting operator, which as evidenced in the results in Section 5.2.4 significantly outperforms the rerouting schemes proposed by Harks et al. [21].

Note that this work also experimented with the iterative linear programming heuristic (ILPH) [19] as its nice theoretical properties such as finite convergence (see Wilbaut and Hanafi [53] and Hanafi and Wilbaut [54]) can be generalized to non-negative integer variables. However, it was found that ILPH does not scale well to larger instances.

4.1 SLOPE SCALING

Slope Scaling (SSC) is an iterative solution process based on solving an approximate linear program $\mathcal{LP}$ exactly. The resulting commodity-flow solution $\tilde{x}$ is used to construction a feasible solution $(\tilde{x}, \tilde{y})$ to the original problem $\mathcal{MIP}$. Then, $(\tilde{x}, \tilde{y})$ is used to update the coefficients in the objective of $\mathcal{LP}$ yielding $\mathcal{LP}'$ in such a way that the objective function value $v(\mathcal{MIP}(\tilde{x}, \tilde{y}))$ equals $v(\mathcal{LP}'(\tilde{x}))$. Now, this process is repeated with $\mathcal{LP}'$.

Therefore, the main idea is to incorporate the fixed-costs occurred by the integer variables $\tilde{y}$ through linearization into the cost-coefficients

of $\tilde{x}$. This additional set of linear cost coefficients at iteration t of the heuristic is denoted $\rho(t)$ and the corresponding linear program as $\mathcal{LP}(\rho(t))$. Algorithm 5 presents the pseudocode of this solution process.

Algorithm 5: Slope Scaling Heuristic

```
Function SSC(T_max):
v* ← ∞
memory ← empty list
(x*, y*) ← (None, None)
t = 0
Calculate ρ(0) using equation (4.8)
while CpuTime() < T_max do
    Solve LP(ρ(t)) to obtain a solution x̃*
    Obtain feasible ỹ* using equation (4.5)
    ṽ ← v(MIP(x̃*, ỹ*))
    if ṽ in memory then
        break
    else
        Add ṽ to memory
    if ṽ < v* then
        v* ← ṽ
        (x*, y*) ← (x̃*, ỹ*)
    Calculate ρ(t + 1) based on x̃* using equations (4.6) or
     (4.7) t ← t + 1
```

4.1.1 *An Approximate Linear Program*

$\mathcal{LP}(\rho(t))$ takes the form of a network flow problem (see Section 2.4.1) and its objective reads:

$$v(\mathcal{LP}(\rho(t))) = \min_{x} \sum_{a \in \mathcal{A}} \sum_{\sigma \in \Sigma} \left[\left(\lambda_1 c_a^\sigma + \lambda_2 \delta_a^\sigma + \rho_a^\sigma(t) \right) \cdot P_{l(a)}^\sigma(x_a) \right] \quad (4.1)$$

the used decision variables and coefficients other than $\rho_a^\sigma(t)$ are defined as in the original problem (compare with the $\mathcal{MIP}$-objective in Section 3.2.3).

The linear program uses the same flow-conservation constraints (3.7) - (3.9) as the original model (see Section 3.2.4). The other constraints are adapted as follows:

$$\sum_{k\in\mathcal{K}^\sigma_\mathcal{T}} p_{kl}x^k_a + \sum_{k\in\mathcal{K}^\sigma_\Delta}\sum_{t=0}^{\tau} p_{kl}x^{k_t}_a \leq B^\sigma_{al}u^\sigma_a \quad \forall l\in\mathcal{P}, \sigma\in\Sigma, a\in\mathcal{A} \tag{4.2}$$

$$\sum_{\sigma\in\Sigma}\sum_{a\in\delta^\circ_m(v)} P^\sigma_l(x_a) \leq B^\sigma_{ml}h^{m\circ}_v \quad \forall l\in\mathcal{P}, \circ\in\{+,-\}, m\in\mathcal{M}, v\in\mathcal{F} \tag{4.3}$$

$$\sum_{\sigma\in\Sigma}\sum_{a\in\delta^-_m(v)\cup\delta^+_m(v)} P^\sigma_l(x_a) \leq B^\sigma_{ml}h^m_v \quad \forall l\in\mathcal{P}, m\in\mathcal{M}, v\in\mathcal{F} \tag{4.4}$$

Constraints (4.2) merge (3.10) and (3.11). The adapted handling constraints (4.3) and (4.4) (compare to the original ones (3.12) and (3.13)) use the mild assumption of same TU property-extents B^σ_{ml}, $l\in\mathcal{P}$ on one mode m and commodity type σ, valid for all the used input instances.

An exact solution $\tilde{x}$ of $\mathcal{LP}(\rho(t))$ can be calculated efficiently by linear programming algorithms (see Section 2.1). If an instance has no capacitating handling constraints, $\mathcal{LP}(\rho(t))$ takes the form of a multi-commodity minimum-cost flow problem and for this case the *network simplex* algorithm [2] outperforms more general simplex algorithms (see Section 5.2.3.1).

4.1.2 *Constructing a Feasible Solution*

After solving $\mathcal{LP}(\rho(t))$ to obtain $\tilde{x}$, a heuristic solution $(\tilde{x},\tilde{y})$ to $\mathcal{MIP}$ is obtained by choosing feasible $\tilde{y}^\sigma_a$-values based on $\tilde{x}_a$ for all $a\in\mathcal{A}$ and $\sigma\in\Sigma$ as follows:

$$\tilde{y}^\sigma_a = \max_{l\in\mathcal{P}}\{\lceil\tfrac{P^\sigma_l(\tilde{x}_a)}{B^\sigma_{m(a)l}}\rceil\} \tag{4.5}$$

with $m(a)$ denoting the mode used by arc a.

Implicitly estimating the number of TUs by summing up flow-extent over multiple arcs as done in (4.3) and (4.4) could lead to minor counting differences compared with $\tilde{y}_a$ obtained through equation (4.5).

To see this, think about two incoming arcs to a transshipment node each transporting as much flow as to fill up one third of a TU. In summation the flow does require only one TU even though it actually requires two. Thus, if in instances with handling constraints some are slightly violated, the solution is repaired by rerouting flow exceeding handling capacities using a variant of the local search moves introduced in Section 4.2.

4.1.3 *Updating the Approximate Linear Program*

Having obtained a feasible solution $(\tilde{x}, \tilde{y})$, the linear cost coefficients $\rho(t)$ are updated resulting into $\rho(t+1)$ and an updated linear program $\mathcal{LP}(\rho(t+1)))$ to solve in the next iteration.

In this section two alternative updating schemes are developed. The first one termed *cost-matching* (see Section 4.1.3.1) follows the general design paradigm behind previous slope scaling mechanisms [17, 18]. That is, having the property that the cost function of the original problem and the updated approximation have the same value.

Section 4.1.3.2 introduces an alternative updating scheme based on *monotonicity*.

4.1.3.1 *Cost-Matching Update*

Using $(\tilde{x}, \tilde{y})$, $\rho(t)$ can be updated as follows:

$$\rho_a^\sigma(t+1) = \begin{cases} (\lambda_1 f_a^\sigma + \lambda_2 \Delta_a^\sigma)\tilde{y}_a^\sigma / P_{l(a)}^\sigma(\tilde{x}_a) & \text{if } P_{l(a)}^\sigma(\tilde{x}_a) > 0 \\ \rho_a^\sigma(t) & \text{otherwise} \end{cases} \tag{4.6}$$

with $\lambda_1 f_a^\sigma$ referring to the weighted costs and $\lambda_2 \Delta_a^\sigma$ to the weighted emissions of one TU. Denote with $v(\mathcal{MIP}(\tilde{x}, \tilde{y}))$ the objective value of $\mathcal{MIP}$ given the solution pair $(\tilde{x}, \tilde{y})$. Furthermore, denote with $v(\mathcal{LP}(\rho(t), \tilde{x}))$ the objective value of $\mathcal{LP}(\rho(t))$ given a solution $\tilde{x}$.

The updating scheme (4.6) results in $v(\mathcal{MIP}(\tilde{x}, \tilde{y})) = v(\mathcal{LP}(\rho(t+1)))$. This can be seen by plugging in the terms on the right hand side of equation (4.6) into the LP-objective (4.1), which yields the objective of the original MIP (see Section 3.2.3).

Figure 4.1 shows that this cost-matching updating scheme results in a zigzag shaped cost-coefficient value with respect to increasing flow extent. This is due to the fact that starting at certain product volumes an additional ordered TU is needed, which then is barely filled and hence, sharply increases costs.

Another property of the updating scheme (4.6) is that for each tariff level the minimum cost-coefficient value is reached when all ordered transport units are fully filled independent of how many TUs are ordered. Exemplary, Figure 4.1 shows that when filling up the first, second or third TU, the blue line always converges to the same cost-coefficient value before spiking due to ordering an additional TU.

The monotonic updating scheme in Section 4.1.3.2 is designed to overcome these possibly disadvantageous properties through putting an emphasis on smoothness.

4.1.3.2 *Monotonic Update*

It is possible to omit $\tilde{y}_a^\sigma$ from the update equation (4.6) yielding

$$\rho_a^\sigma(t+1) = \begin{cases} (\lambda_1 f_a^\sigma + \lambda_2 \Delta_a^\sigma)/P_{l(a)}^\sigma(\tilde{x}_a) & \text{if } P_{l(a)}^\sigma(\tilde{x}_a) > 0 \\ \rho_a^\sigma(t) & \text{otherwise} \end{cases} \tag{4.7}$$

Then $v(\mathcal{MIP}(\tilde{x}, \tilde{y})) \neq v(\mathcal{LP}(\rho(t+1)))$, but for $t \geq 1$ the update scheme becomes *monotonic* on each tariff level with respect to increasing flow extent. This property can be seen in Figure 4.1 and interpreted as favouring higher transport volumes on an arc no matter the filling rate of the last TU, playing into the economics of scale.

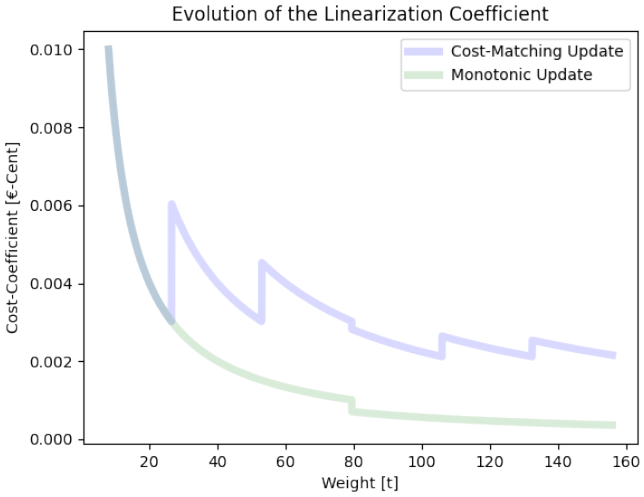


Figure 4.1: Updated cost-coefficient value based on simulated product flow and the all-unit discount costs tariff with linear-cost levels shown in Figure 3.1. About every 26.5t a new TU has to be ordered. At 80t a new discounted tariff level starts.

Note that the original slope scaling updating scheme for binary variables is both cost-matching and monotonic [18]. Therefore, one can view the original cost-matching updating function as the best monotonic one from the space of all monotonic updating functions.

In this view generalizing to general non-negative integer variables poses the important updating scheme design question:

Should the space of cost-matching functions or the space of monotonic function be explored?

The monotonic updating scheme introduced here is an attempt to explore this alternative function space. The results in Section 5.2.3 indicate that the monotonic update scheme outperforms the cost-

matching one. For a detailed discussion of these results the reader is referred to Section 5.2.3.

4.1.4 *Initializing the Approximate Linear Program*

Initial cost estimates are set to the linearized costs of one full TU on the respective connection independent of the used updating scheme. Equation (4.8) represents the initialization formula.

$$\rho_a^\sigma(t=0) = (\lambda_1 f_a^\sigma + \lambda_2 \Delta_a^\sigma) / B_{al(a)}^\sigma \tag{4.8}$$

4.1.5 *Termination Criterion*

The slope scaling heuristic starts at $t = 0$ and runs for as many iterations until a specific time limit is reached.

However, it is observed that SSC sometimes gets stuck in a loop always producing the same solution sequence over and over again. This is due to the fact that the updating schemes are deterministic producing the same cost-coefficients given the same inputs.

As this loop can be of arbitrary iteration size. The costs of all generated solutions are stored and if a new solution is generated with already seen costs, it is assumed to be identical leading to a solution loop. Then SSC terminates.

4.1.6 *Time-Expanded vs Tariff-Expanded Network*

The slope scaling mechanism introduced here can be applied on the much more memory-efficient time-expanded network by replacing $a \in \mathcal{A}$ with $a \in \mathcal{A}_\mathcal{T}$ in the approximate linear program and in each iteration adapt the cost factors c_a^σ and f_a^σ to the best available tariff level for the chosen $\tilde{y}_a^\sigma$.

All results involving slope scaling in Chapter 5 are produced by SSC operating on the time-expanded network.

4.2 LOCAL SEARCH

The presented local search (LS) is inspired by Harks et al. [21] and based upon the *flow decomposition theorem* [46] presented in Section 2.4.1, which states that every non-negative arc flow can be represented as a path and cycle flow. Thus, given an initial solution, the basic idea of the local search is to construct a path-decomposition of flow on the acyclic time-expanded graph $\mathcal{G}_\mathcal{T}$. Then, randomly dissolve paths and reroute the flow. This results in a new path-decomposition of flow and the process can be repeated until no improvements are possible.

Section 4.2.1 formally introduces the concept of a path decomposition on $\mathcal{G}_\mathcal{T}$ and presents two different ways to calculate it. Section 4.2.2 introduces two different rerouting schemes, which can be applied after a flow has been released by dissolving one or multiple paths. Based on these two sections, Section 4.2.3 introduces the neighbourhoods and moves used by the local search. With all these definitions in hand, Section 4.2.4 presents the local search algorithm including a detailed pseudocode.

Note that the LS can operate on the time-expanded graph rather than on the tariff-expanded one, as it can handle non-linear arc-costs. Furthermore, the LS is fundamentally concerned with commodity flows and assumes flow is always accompanied by corresponding transport unit choices y using equation (4.5) from Section 4.1.2, even though y may not be explicitly mentioned.

4.2.1 *Path Decompositions*

A path-decomposition $\mathcal{P}$ consists of a set of tuples (P, x_P). Each tuple consists of a directed path P in $\mathcal{G}_\mathcal{T} = (\mathcal{V}_\mathcal{T}, \mathcal{A}_\mathcal{T})$ and the transported commodities $x_P \in \mathbb{R}_0^{|\mathcal{K}_\mathcal{T}|+|\mathcal{K}_\Delta||\mathcal{T}|}$. Paths carry only commodities of a specific type and due to the topology of the network, either end in a demand or bin node. Therefore, a path is either called a demand-path or a bin-path.

Note that for a given flow, $\mathcal{P}$ is not necessarily unique. This inspires multiple different calculation schemes for $\mathcal{P}$ based on depth-first search (DFS) as presented in and adapted from Harks et al. [21].

4.2.1.1 *Unidirectional Construction*

To calculate a path decomposition in a unidirectional manner similar to Harks et al. [21], first select the subset of source nodes $\mathcal{S}$ from $\mathcal{V}_\mathcal{T}$ and define all flow on each arc from a given solution as unassigned. Then, for each $s \in \mathcal{S}$ taken in chronological order, iterate through all

product types $\sigma \in \Sigma$ and apply the following algorithmic scheme until all ougoing flow is associated to flow-paths:

Starting at s, for each incident arc, the flow of commodities of type σ that could be assigned to a path traversing this arc is calculated. Then, the arc whose assignable flow has maximal weight is chosen. Next, this process is repeated for the newly connected node until a bin or demand node is reached. Finally, add the calculated path P and associated flow-vector x_P to the path-decomposition and declare on each arc $a \in P$ the flow x_P as assigned.

The pseudocode for this construction can be found in Appendix A.2.

Chronologically going through the source nodes ensures that each flow-path indeed ends in a bin or demand node. To see this, assume a source node s' is chronologically before another source s and sends products to s. Furthermore, assume that starting from s products sourced by s' and products sourced by s share the same path through the network to a destination node d. Then, applying the above flow-path construction scheme on node s before processing s' results in a flow-path having associated all products flowing from s to d including those sourced by s'. However, afterwards, applying the flow-path construction scheme on node s' will find a flow-path which ends at s as the flow outgoing from s is already associated to a different flow-path.

4.2.1.2 *Bidirectional Construction*

An alternativ path decomposition construction scheme again inspired and adapted from Harks et al. [21] repeats the following steps for each product type σ:

1. Sort all arcs with respect to highest savings potential. Highest savings potential is defined as having the highest impact on the objective function when the unassigned flow of type σ is removed.
2. Apply a bidirectional depth-first search. Starting from the arc who currently has the highest savings potential, choose arcs in both directions with maximal savings potential with respect to the possible flow of type σ carried by the flow-path in construction.
3. As in the unidirectional construction, add the calculated path P and associated flow-vector x_P to the path-decomposition and declare on each arc $a \in P$ the flow x_P as assigned. Then, go back to the second step until all flow of type σ is associated to a flow path.

See Appendix A.2 on how to adapt the given pseudocode for the bidirectional construction.

Due to also applying a depth-first search in a backward direction it is not necessary to perform the iteration in a chronological manner.

4.2.2 *Rerouting Schemes*

If one or multiple paths are removed from the path decomposition $\mathcal{P}$, the now freed commodities have to be routed anew to fulfil all demands. Here, two rerouting schemes are introduced: *heaviest-first rerouting* in Section 4.2.2.1 taken and adapted from Harks et al. [21], and *cheapest-first rerouting* in Section 4.2.2.2 developed independently in this work to overcome shortcoming of heaviest-first rerouting. In both rerouting schemes an algorithmic subproblem related to knapsack problems [55] arises due to storage and capacity constraints. How to optimally solve this problem is discussed in Section 4.2.2.3.

The presentation in the following subsections assumes that s refers to a source node and t to a demand node. Furthermore, the current commodity-flow in $\mathcal{G}_\mathcal{T}$ is denoted x. Then, $\Delta_k(s,t,x)$ refers to the maximal quantity of commodity k sourced by s and deliverable to demand node t with respect to the demand of t for k and the existing flow x (i.e. not already in x). For all commodities $k \in \mathcal{K}$ these $\Delta_k(s,t,x)$ can be assembled into a vector $\Delta(s,t,x) \in \mathbb{R}_0^{|\mathcal{K}|}$. It is not necessary to use a separate $\Delta_{k_t}(s,t,x)$ as k_t is specified by k and the time-period of s. With $W(\Delta(s,t,x))$ the total weight of all commodities in $\Delta(s,t,x)$ is denoted.

The algorithms below also depend on shortest path calculations in $\mathcal{G}_\mathcal{T}$. For this, weights $w(a)$ are defined on all met arcs $a \in \mathcal{A}_\mathcal{T}$ as follows: $w(a) = c(a, x+\Delta) - c(a,x)$, which represents the change in costs for arc a when Δ commodities are added to the existing flow in the system.

Note that shortest path calculations in $\mathcal{G}_\mathcal{T}$ can be made in linear time $\Theta(|\mathcal{V}_\mathcal{T}| + |\mathcal{A}_\mathcal{T}|)$ due to the fact that $\mathcal{G}_\mathcal{T}$ is a dicrected acyclic graph (DAG) [26].

Assuming a given fixed-size DAG. $\mathcal{G}_\mathcal{T}$'s graph-size itself scales pseudopolynomial due to the dependency on T.

4.2.2.1 *Heaviest-First Rerouting*

In heaviest-first rerouting one searches a (s,t)-pair allowing for the highest delivery weight. Then, one calculates a cheapest path from s to t and adds the found path to the path decomposition. These two steps are repeated until all demand has been satisfied or infeasibility is detected.

Heaviest-first rerouting is presented in Algorithm 6 and assumes a flow-vector x is given, which represents an earlier calculated solution with flow of type σ along some paths removed (see Section 4.2.3).

Algorithm 6: Heaviest-First Rerouting

```
1  Function reroute_heaviest_first(x):
2  while not all demand has been satisfied do
3      Let s, t ∈ V_T such that t reachable by s and W(Δ(s,t,x)) is
         maximum
4      if no such (s,t) pair exists then
5          break
6      Δ ← Δ(s,t,f)
7      Compute shortest path P in G_T from s to t w.r.t.
         w(a) = c(a, x+Δ) − c(a,x)
8      Adapt Δ if handling or storage capacity constraints
         violated
9      Augment x along P by Δ
10 if not all demand satisfied then
11     return infeasibility detected
12 else
13     Route unused commodities directly to bin nodes and
         update x.
14 return x
```

Infeasibility is possible when rerouting perishable commodities and having at least two demand nodes with unsatisfied demand. To see this, assume that a demand node d_1 in time-period t could get its products from source node s_1 and s_2. Another demand node d_2 in time-period $t+1$ can only get its products from source s_2 because those from source s_1 will have perished in time-period $t+1$. However, line 3 could choose the pair (s_2, d_1) depleting the available commodities from s_2. Empirically it is found that this case happens only rarely.

Adapting Δ in line 8 leads to a relaxed multidimensional bounded knapsack problem (MBKP) with a special structures, which allows to derive an optimal solution in closed form as shown in Section 4.2.2.3. Note that the shortest path calculation is implemented in a way to only consider arcs with left-over handling or storage capacity.

4.2.2.2 *Cheapest-First Rerouting*

The *cheapest-first* or more accurately the *cheapest-relative-cost rerouting* differs to heaviest-first rerouting by choosing the (s,t)-pair that results in the cheapest path relative to the transported weight. The idea is to

make better use of existing transport routes by "filling them up". The pseudocode of which is shown in Algorithm 7.

Harks et al. [21] describes the difficulty of their heaviest-first rerouting scheme to effectively exploit consolidation potentials. The results in Section 5.2.4.2 show that cheapest-first rerouting consistently outperforms heaviest-first rerouting, which in part can be explained by the results from Section 5.2.4.4 empirically showing that cheapest-first rerouting is superiour at finding and exploiting consolidation opportunities.

Algorithm 7: Cheapest-Relative-Cost Rerouting

```
1  Function reroute_cheapest_first(x):
2  while not all demand has been satisfied do
3      t ← demand region with heaviest unsatisfied demand
4      S ← set of all source nodes with unrouted products
         reaching t
5      if S = ∅ then
6          break
7      P_best ← empty path
8      P_best.c ← ∞
9      foreach s ∈ S do
10         Δ ← Δ(s, t, x)
11         Compute shortest path P in G_T from s to t w.r.t.
             w(a) ← c(a, x + Δ) − c(a, x)
12         Adapt Δ if handling or storage capacity constraints
             violated
13         c(P) ← additional costs of sending Δ over P given x
14         P.c ← c(P)/W(Δ)
15         if P.c < P_best.c then
16             P_best ← P
17             P_best.Δ ← Δ
18     Augment x along P_best by P_best.Δ
19 if not all demand satisfied then
20     return infeasibility detected
21 else
22     Route unused commodities directly to bin nodes and
         update x.
23 return x
```

4.2.2.3 *Adapting Transported Goods to Capacity Constraints*

In both rerouting schemes it could be necessary to adapt (reduce) the carried products $\Delta \in \mathbb{R}_0^{|\mathcal{K}|}$ on a path P to $\tilde{\Delta}$ to satisfy handling or

storage capacities available on the arcs along P. $\tilde{\Delta}$ represents the largest feasible shipment on P, where largest is defined as the shipment with the highest delivery-weight.

As shown in the following, adapting Δ to fulfil storage capacities leads to a relaxed bounded knapsack problem, whereas adapting Δ to fulfil handling capacities leads to a relaxed multidimensional bounded knapsack problem. However, due to its special structure not only the first but also the second problem can be solved in closed form as shown in Theorem 4.

The individual entries of an adapted product vector $\tilde{\Delta}$ are denoted $\delta_k,\ k \in \mathcal{K}$. For easy readability denote with W_k the weight-property and with V_k the volume-property of commodity k. The commodities in all instances used in this work have exactly these two properties.

Storage capacity exceeded

Assume the products Δ exceed the storage capacity V_{max} provided by a specific arc. Note that in the GTTP storage capacity is only restrictive in volume. Therefore, one wants to maximize the delivery weight with respecting the maximal possible delivery volume. This leads to the following relaxed *bounded knapsack problem* (BKP) [55]:

$$\max \sum_{k=1}^{|\mathcal{K}|} W_k \delta_k \tag{4.9}$$

$$\text{subject to} \sum_{j=1}^{|\mathcal{K}|} V_k \delta_k \leq V_{max} \tag{4.10}$$

$$0 \leq \delta_k \leq \Delta_k,\ \delta_k \in \mathbb{R}\ \forall k \in \mathcal{K} \tag{4.11}$$

If δ_k would be restricted to integer values, the original BKP formulation would be obtained. However, for the current work, only the optimal solution vector of the LP-relaxation is of interest. Using the concepts (taken from Kellerer et al. [55]) of *efficiency* $e_k := \frac{W_k}{V_k}$ and of a *split item* s defined by

$$\sum_{k=1}^{s-1} V_k \Delta_k \leq V_{max} \text{ and } \sum_{k=1}^{s} V_k \Delta_k > V_{max}$$

one can compute the optimal LP-solution in a straight forward manner, using the following lemma:

Lemma 3. [55] If the item types are sorted by decreasing efficiencies such that

$$e_1 \geq e_2 \geq \cdots \geq e_{|\mathcal{K}|}$$

then the optimal solution vector $\delta^{LP} := (\delta_1^{LP}, \ldots, \delta_{|\mathcal{K}|}^{LP})$ of the LP-relaxation of BKP is given by

$$\begin{aligned} x_k^{LP} &:= \Delta_k && \text{for } k = 1, \ldots, s-1 \\ x_s^{LP} &:= \frac{1}{V_s}\left(V_{max} - \sum_{k=1}^{s-1} V_k \Delta_k\right) \\ x_k^{LP} &:= 0 && \text{for } k = s+1, \ldots, |\mathcal{K}| \end{aligned}$$

Handling capacity exceeded

Assume the products Δ exceed the handling capacities V_{max} or W_{max} of a node incident to an arc in P. V_{max} and W_{max} follow from the maximal number of TUs the node can handle and the dimensions of the respective TUs.

Similar to the storage case, one wants to maximize the delivery weight respecting V_{max} and W_{max}. This leads to the following relaxed *multidimensional bounded knapsack problem* (MBKP) [55]:

$$\max \sum_{k=1}^{|\mathcal{K}|} W_k \delta_k \tag{4.12}$$

$$\text{subject to} \sum_{k=1}^{|\mathcal{K}|} V_k \delta_k \leq V_{max} \tag{4.13}$$

$$\sum_{k=1}^{|\mathcal{K}|} W_k \delta_k \leq W_{max} \tag{4.14}$$

$$0 \leq \delta_k \leq \Delta_k,\ \delta_k \in \mathbb{R}\ \forall k \in \mathcal{K} \tag{4.15}$$

It differs from the general MBKP by not only relaxing δ_k from integer values to reals, but in the objective coinciding with one of the constraints. In the following, the problem arising from the above one when requiring all δ_k to be integer is called MBKP'. Notice, if one removes equation (4.13) and changes the equations (4.15) to $\delta_k \in \{0, 1\}$ one obtains the subset sum problem (SSP) [55].

The property of the objective coinciding with one of the constraints allows to derive a simple proof for a theorem similar to Lemma 3 for the MBKP'. For this, define again efficiencies e_k as $\frac{W_k}{V_k}$, but split items have to be defined for each dimension of the knapsack problem:

Split item s_V defined by

$$\sum_{k=1}^{s_V-1} V_k \Delta_k \leq V_{max} \text{ and } \sum_{k=1}^{s_V} V_k \Delta_k > V_{max}$$

Split item s_W defined by

$$\sum_{k=1}^{s_W-1} W_k\Delta_k \leq W_{max} \text{ and } \sum_{k=1}^{s_W} W_k\Delta_k > W_{max}$$

Now, the following theorem derived in this work provides a closed form solution on how to calculate an optimal $\tilde{\Delta}$ from Δ satisfying the handling constraints.

Theorem 4. If the item types are sorted by decreasing efficiencies such that

$$e_1 \geq e_2 \geq \cdots \geq e_{|\mathcal{K}|}$$

then the optimal solution vector $\delta^{LP} := (\delta_1^{LP}, \ldots, \delta_{|\mathcal{K}|}^{LP})$ of the LP-relaxation of the MBKP' is given by

$$\delta_k^{LP} := \Delta_k \qquad \text{for } k = 1, \ldots, s-1 = \min(s_V, s_W) - 1$$

$$\delta_s^{LP} := \frac{1}{V_s}\left(V_{max} - \sum_{k=1}^{s-1} V_k\Delta_k\right) \qquad \text{if } s = s_V < s_W$$

$$\delta_s^{LP} := \frac{1}{W_s}\left(W_{max} - \sum_{k=1}^{s-1} W_k\Delta_k\right) \qquad \text{if } s = s_W < s_V$$

$$\delta_k^{LP} := 0 \qquad \text{for } k = s+1, \ldots, |\mathcal{K}|$$

However, if $s = s_V = s_W$ then

$$\delta_s^{LP} := \min\left\{\tfrac{1}{V_s}\left(V_{max} - \textstyle\sum_{k=1}^{s-1} V_k\Delta_k\right), \tfrac{1}{W_s}\left(W_{max} - \textstyle\sum_{k=1}^{s-1} W_k\Delta_k\right)\right\}.$$

The author is convinced similar results have been derived in other works, but is not aware of them. Hence, an own proof is given.

Proof. The solution δ^{LP} is by construction feasible for the relaxed MBKP'. Furthermore, notice that there are exactly three distinct cases, either $s_V < s_W$, $s_V > s_W$ or $s_V = s_W$.

Case $s_V < s_W$:
By Lemma 3 the constructed δ^{LP} is an optimal solution to the relaxed BKP. However, it is obvious that the relaxed BKP is a relaxation of the relaxed MBKP' obtained by removing the bound on the maximal weight (constraint (4.14)). Therefore, and as δ^{LP} is an optimal solution to the relaxed BKP and feasible for the relaxed MBKP', it is an optimal solution to the relaxed MBKP'.

Case $s_V > s_W$:
By construction δ^{LP} has maximal weight W_{max}. As δ^{LP} is feasible and the objective is to maximize weight, δ^{LP} is an optimal solution.

Case $s_V = s_W$:
For this case note that the above proofs do not rely on the fact that $s_V < s_W$ or $s_V > s_W$, but on a solution δ^{LP} having either maximal volume V_{max} or maximal weight W_{max}. Therefore, if $\delta_s^{LP} = \frac{1}{V_s}\left(V_{max} - \sum_{k=1}^{s-1} V_k\Delta_k\right)$, then the argument of Case $s_V < s_W$ proofs optimality. Otherwise, the argument given for Case $s_V > s_W$ proofs optimality for δ^{LP}. □

4.2.3 *Neighbourhoods*

Two types of neighbourhoods are distinguished. They are defined as the set of all solutions constructible

1. by removing an arbitrary demand-path;
2. by removing a group of demand-paths sharing the same transport relation and carried flow type σ;

and all bin-paths and then, repairing the solution by rerouting the flow. In the second case, the removal and rerouting are repeated for all $\sigma \in \Sigma$.

Therefore, the neighbourhoods are characterized by a given path-decomposition and the employed rerouting scheme R and are denoted $\mathcal{N}_1(\mathcal{P}, R)$ and $\mathcal{N}_2(\mathcal{P}, R)$, respectively.

A *move* is defined by randomly choosing a solution from $\mathcal{N}_1(\mathcal{P}, R)$ or $\mathcal{N}_2(\mathcal{P}, R)$ and accepted, if it improves upon the current solution. To apply such a move it is not necessary to generate the whole neighbourhood: for $\mathcal{N}_1$ remove a randomly choosen path from $\mathcal{P}$ and then apply the rerouting scheme R; for $\mathcal{N}_2$ select a random arc with at least one path and for all $\sigma \in \Sigma$ repeat: remove all paths carrying flow of type σ over this arc from $\mathcal{P}$ and apply R.

4.2.4 *Algorithm*

The local search works by in the beginning constructing a solution using slope scaling, followed by choosing an initial neighbourhood $\mathcal{N}_i(\mathcal{P}, R)$ and then, repeating the following steps until convergence or a time limit:

1. Construct a path-decomposition for $\mathcal{N}_i(\mathcal{P}, R)$.
2. Follow randomly chosen improving neighbouring solutions until the improvement over a certain number of iterations falls below a threshold.
3. Change R and go to step 1. If all R have been used with N_i in the previous iterations, change i instead.

Algorithm 8 presents the detailed pseudocode. The pseudocode assumes that an initial solution x with objective value $v(x)$ is given. Furthermore, a set of path decomposition schemes $\mathcal{P}$ (see Section 4.2.1) and rerouting schemes $\mathcal{R}$ (see Section 4.2.2) are given.

Each flow-solution x is accompanied by y choices given by equation (4.5), but y is omitted for brevity.

Line 6 constructs a path decomposition of x and chooses a computation scheme from $\mathcal{P}$ depending on N_i. In this work $\mathcal{P}$ either has only one element and hence, the computation scheme is clear or two elements, a unidirectional computation scheme A and bidirectional one B. Then,

if $N_i = N_1$ scheme A is used and if $N_i = N_2$ then B is used. This strategy is taken from Harks et al. [21] introducing move dependent path decomposition calculations.

Besides a time limit, there are two other termination criterions in lines 9 and 22. The one in line 22 is triggered, when all neighbourhoods have been searched unsuccessfully for improving solutions, hence the LS has found a local optimum.

The termination criterion in line 9 is chosen so as to be fulfilled if over the course of 100 iteration, the objective could not be improved by at least 10% of what it had improved over in the previous 100 iterations.

Algorithm 8: Local Search for the GTTP

```
1  Function local_search(x, 𝒫, ℛ, T_max):
2  v* ← v(x)
3  c ← 0
4  i ← 1
5  while CpuTime() < T_max do
6      𝒫 ← construct_path_decomposition(x, N_i, 𝒫)
7      for R ∈ ℛ do
8          v_old ← v*
9          while not termination criterion do
10             Choose random x̃ and associated 𝒫̃ from N_i(𝒫, R)
11             if v(x̃) < v* then
12                 x ← x̃
13                 𝒫 ← 𝒫̃
14         if v_old == v* then
15             c ← c + 1
16         else
17             c ← 0
18     if i%2 == 0 then
19         i = 1
20     else
21         i = 2
22     if c == 2 · |ℛ| then
23         break
```

Note that the LS presented here differs from Section 2.3.1 by using multiple neighbourhoods. To see the effects of using different neighbourhoods and rerouting schemes the reader is referred to the results in Section 5.2.4.2.

5

RESULTS

This chapter presents a detailed summary of the results obtained when applying the introduced solution algorithms (see Chapters 3 and 4) to the data used to simulate horizontal cooperation in the Danube region. It starts with introducing these problem instances and their generation process in Section 5.1. In the subsequent Section 5.2 the results of applying a state-of-the-art MIP solver and the developed heuristics are shown and the algorithmic aspects of these results discussed. The managerial implications are discussed in Section 5.3.

Selected parts of this results chapter will appear partly shortened in Gosch et al. [1]. Especially the data description in Section 5.1, the introduction to the experimental subsections given in Section 5.2, and most parts of Section 5.3 including its subsections.

5.1 DATA

5.1.1 *Description*

The instances are based on a dataset introduced by Wolfinger et al. [56] including locations of major cities, train stations and Danube ports in Austria, Slovakia, Hungary, Romania, Serbia, and Bulgaria. For this work, each city region has been categorized as small (population (P) $<$ 100.000), medium (100.000 $\leq P <$ 1.000.000) or large ($P \geq$ 1.000.000). Then, nine random locations have been added in each city and large ones additionally equipped with a cross-dock. All road-distances have been recalculated using Ariadne [57]. Distance and time-matrices of other transport modes were taken from Wolfinger et al. [56]. Moreover, each city is interpreted as a demand region and we randomly distribute commodity demands and stocks (see Section 5.1.2.2).

Generated instances range in size from two collaborating warehouses in two different regions up to the exhaustion of our memory limits during the solution process. More precisely, input instances can be grouped into having two ($R2$), fife ($R5$) or ten regions ($R10$). $R2$-instances are generated with either one or four warehouses per region. $R5$-instances are generated with two or four warehouses per region. For $R10$-instances the five more populated regions have four warehouses and five lower populated ones have two. Warehouses are again grouped representing association to different companies (see Section 5.1.2.1). For each selected region the closest cross-dock, train station and port in the dataset by Wolfinger et al. [56] is included. All instances have a time period of one day and are generated in three different time horizon (T) versions of 7, 14 or 30 days with halve of

their commodities being perishable. Lastly, each instance has a version with active handling constraints (tight t) and without (loose l). Therefore, instance groups are identified using $R<\#>\_T<\#>\_t/l$ and using a $*$ instead of $<\#>$ to indicate averages over all respective instance groups. In each instance half of the commodities are perishable.

In total 60 instances have been generated (see Appendix B) leading to 180 optimization problems considering optimizing for costs, emissions or both. The input instances and the code to generate them are available on https://github.com/sapero/gttp-data.

5.1.2 *Generation*

The data-generation process described in Sections 5.1.2.1 - 5.1.2.4 also appeard slightly altered in the above linked Git repository.

In the following sections, the data generation process is described.

5.1.2.1 *Warehouse-Groups*

Given C groups (for example companies) indexed using set $\mathcal{C}$ and N warehouses indexed using set $\mathcal{N}$. Assume $C \leq N$. Each warehouse is assigned to a group using the following algorithm:

1. Associate to each $c \in \mathcal{C}$ a unique random warehouse $n \in \mathcal{N}$ preferably from a new location.
2. Remove associated warehouses from $\mathcal{N}$.
3. Remove $\lfloor C/3 \rfloor$ groups from $\mathcal{C}$
4. If $\mathcal{N} \neq \emptyset$, go to step 1.

5.1.2.2 *Commodities*

Half of the products (non-perishable and perishable) are evenly associated to the groups. The other half of products are randomly assigned to a group in such a way that the assignment probability is directly proportional to the group size. Furthermore, each product is requested in between 25% and 100% of all demand regions (but at least one) in which the group has no associated warehouses. A product is twice as likely to be demanded in a middle-sized demand region and four times as likely to be demanded in a large-sized demand region compared to a small-sized region. The product dimension is drawn from a heavy-tailed Lévy-distribution with location $\mu = 0$ and scale $c = 0.2$ to emphasize smaller products but not exclude larger ones (see Figure 5.1). The dimension[1] is rounded up to the next 0.01 m^3 (interpretable as a minimum modular-container size) and redrawn if exceeding 2.16

1 2.16 m^3 roughly correspond to the maximal dimensions of a fully loaded Euro-pallet.

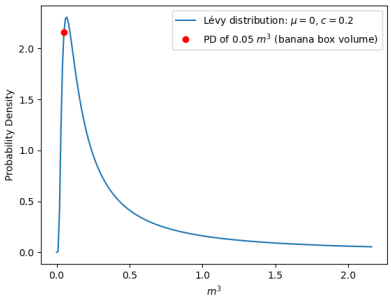


Figure 5.1: (Unscaled) Probability density function of commodity volumes.

m^3. Product weight is calculated by assuming a density uniformly drawn between 50 $\frac{kg}{m^3}$ and 1200 $\frac{kg}{m^3}$.[2].

The average requested product-weight in a demand region is drawn uniformly between the weight of exactly one product k_W and 20.000 kg. Additionally, the drawn demand is lowered by 10% for small regions and increased by 10% for large regions. The average requested number of products $\hat{r}$ is then calculated by dividing the requested product-weight by k_W and rounding to the next integer. A product is requested daily, every two days or weekly. The actual product demand for each request is drawn from a normal distribution with mean $\hat{r}$ and a standard deviation of $0.1 \cdot \hat{r}$.

Each product is in stock in one or multiple warehouses. A product demand is satisfied by the closest warehouse of the associated company. If the product is perishable, its lifetime is set uniformly between the minimum travel time t_{min} of a truck to the most distant associated demand node and $t_{min} + 4$.

Warehouse-stock is generated in the time-period such that a truck can fulfil the requested demand on time. This means, if the demand is at time t the stock is generated for time period $t - t_{min}$. To determine the generated stock size for a commodity its average request size for a particular demand region $\hat{r}$ is summed up over all demand regions it has to service yielding $\hat{r}_{sum}$ and adding 20% interpretable as safety stock.

The maximal capacity of a warehouse is set in such a way that the maximal generated stock for the warehouse is assumed to occupy between 50% and 80% of its capacity (again drawn uniformly). If a warehouse carries no stock, its capacity is set to half of the average capacity found in its region size.

2 Minimum weight-density corresponds to the density of styrofoam [58].

5.1.2.3 *Handling Capacities*

Loosely and tightly capacitated version of each instance are generated. In a loosely capacitated version only warehouses are capacitated in the way explained above. In the tightly capacitated instance each transshipment point has an incoming and outgoing handling capacity of 30% of the average demand in the associated region (inspired by Wolfinger et al. [56])) and a total handling capacity of 60%. If the associated region has no demand, handling capacity is calculated based on the closest region having non-zero demand.

5.1.2.4 *Tariffs*

Tariffs with three levels of volume discounts are employed. Piecewise-linear cost levels are used for lorry connections and constant cost levels for rail and ship connections. On these connections TUs always correspond to 40-feet ISO containers. Handling costs are integrated into the arc costs. Prices are set based on information from an industry partner.

5.2 EXPERIMENTS

This section presents detailed results of and empirical comparisons between the different devised solution algorithms. Furthermore, the statistical significance of the results are examined using the methodology described in Section 5.2.1. Section 5.2.2 examines the performance of applying a state-of-the-art MIP-solver (CPLEX 12.10) on the MIP formulation presented in Chapter 3 and compares the effects of including different strengthening constraints. Section 5.2.3 presents the results of applying the slope scaling heuristic (SSC, see Section 4.1) using different cost-approximation strategies. Then, Section 5.2.4 proceeds with presenting detailed results of applying different local search variants on solutions obtained by slope scaling. Finally, Section 5.2.5 presents a comparisons between the best performing MIP, SSC and local search configurations and hence, summarizes and concludes the algorithmic findings. A reader interested only in the results of the best performing algorithm can skip the individual in-depth result-analyses and jump directly to Section 5.2.5.

Solutions calculated on our test instances are evaluated with respect to a *direct delivery solution*. In a direct delivery solution demand for a product is satisfied by lorry shipments without the possibility of cooperation. This means consolidation is only possible across warehouses associated to the same company. Direct delivery instances are generated by removing transshipment nodes and arcs connecting

warehouses from different companies from the original input instances. Next, a MIP-solver is applied on the adapted instances for one hour. The thereby calculated direct delivery solutions are either optimal or proven to be within a 1% optimality gap.

If not otherwise specified reported numbers when optimizing for emissions refer to the percentual emission reduction compared to the emissions of the direct delivery solution. When optimizing for costs the results refer to the percentual reduction in costs compared to the direct delivery solution. However, when jointly optimizing for both, the reported results refer to the percentual reduction on the sum of emissions and costs by pricing one tonne of CO_2e with 100€ [59] (using the avoidance cost approach [14]). Calculated standard deviations given for instance groups and usually refer to inner-group deviations if not otherwise specified.

All experiments were written in C++ using CPLEX 12.10, had a time limit of one hour, and were performed on an Intel(R) Xeon(R) CPU E5-2640 v3 @ 2.60GHz CPU with 32 GB RAM.

5.2.1 *Statistical Significance of Results*

It is not always clear if a method could beat another method due to being superior or if it just happened by chance. Therefore, when applicable, the statistical significance of the reported results is examined. For this statistical analysis the Wilcoxon signed rank test [60] is employed. It is a paired difference test meaning it assumes matched sample-pairs (X_i, Y_i) (in our context the results of two competing methods on the same instance i) and tests whether their means match. The null hypothesis H_0 of the Wilcoxon signed rank test states that the pair-wise differences $\delta_i = |X_i - Y_i|$ follow a symmetric distribution around zero. This is in contrast to other paired difference tests often assuming δ_i follows a certain parametric distribution. Exemplary, the paired T-test [61] assumes δ_i follows a normal distribution.

In the following δ_i is calculated once for deterministic methods or due to computational limitations from the mean of ten independent runs for stochastic ones. This small sample size makes it unclear if an averaged δ_i can be assumed to be normally distributed [32].

After computing δ_i, the p-value, which is the probability of seeing an event at least as extreme as the current data under the assumption of H_0 is calculate. The p-value can be identified with the probability of falsely rejecting H_0 (i.e. committing a type I error) [61]. As a result, H_0 is discarded with a significance level α if

$$p\text{-value} < \alpha$$

For the analyses in the forthcoming sections X_i corresponds to result of method X on instance i or the mean result if the method has random components. Furthermore, α is chosen to be 0.05.

If more than two methods have to be compared the significance test is performed for each methods pair separately. However, to then correct for the increased chance of type 1 errors due to multiple comparisons Bonferroni correction [62] is used in which the target significance level α is divided by the number of comparisons K. As an example each local search variant is tested against five others, hence, to achieve a target level of $\alpha = 0.05$ for each variant individual tests are performed with $\alpha = 0.01$.

5.2.2 *MIP*

Table 5.1 gives an overview of the MIP formulations compared in this section. Especially, `MIP0` refers to the original model presented in Section 3.2. Other numberings refer to different combinations of strengthening constraints.

Table 5.1: Table of MIP formulations explored. For a detailed explanation of the different types of strong valid inequalities see Section 3.2.5.

Name	Strengthening Constraints
`MIP0`	None
`MIP1`	Capacity
`MIP2`	Capacity + Intra-Tariff
`MIP3`	Capacity + Inner-Tariff
`MIP4`	Capacity + Intra-Tariff + Inner-Tariff

To support CPLEX in finding solutions within the time limit of one hour each run was initialized with its corresponding (feasible) direct delivery solution. Therefore, only improved solutions to the state of the art can be found. However, for all but the smallest instances CPLEX struggled with finding other feasible solutions and needed excessive time in the root node solution process. Therefore, the `MIP emphasis switch` was set to one indicated emphasizing feasibility over optimality [29] alleviating the problem.

Table 5.2 shows the results of applying the different MIP formulations to the problem instances. Good results are achieved when optimizing for emissions especially on small instances. High in-group variance is observed as planning over 7, 14 or 30 days has a significant effect on found emission reductions (see Appendix C.2).

The found average emission reductions reduce when the instance sizes increase. This is not due to fundamentally less emission reduction potential (compare this to the results of the local search in Section 5.2.4) but due to the MIP failing to find better solutions for some larger instances in the group (see Appendix C.2). Especially, the MIP

fails to find improving solutions when handling constraints are employed indicating that handling constraints increase the hardness of the resulting problem. Having handling constraints more often results in the MIP sticking with the direct delivery solution used for initialization and hence, reducing the average emission reductions of groups with larger instances. This is also indicated by the increase of in-group standard deviation. However, for the group with the largest instances `R10_T*_*` the in-group standard deviation decreases again as a lot of instances are including which yield no improving solution, i.e. contribute a 0.0% weighting.

Table 5.2: Average results including instance group standard deviations achieved by the MIP formulations presented in Table 5.1 on different instance groups compared to the direct delivery solution. Note that strengthening constraints are only added when optimizing for costs or both costs and emissions. The MIP mostly fails to solve the optimization problem when a cost term is present.

	MIP0 [%]	MIP1 [%]	MIP2 [%]	MIP3 [%]	MIP4 [%]
Optimize for Emissions					
R2_T*_*	27.3±13.8	-	-	-	-
R5_T*_*	15.3±18.6	-	-	-	-
R10_T*_*	9.2±15.1	-	-	-	-
R*_T*_*	19.4±17.4	-	-	-	-
Optimize for Costs					
R2_T*_*	2.0±4.0	**3.8±5.9**	**3.8±5.8**	**3.8±5.7**	3.6±5.8
R5_T*_*	**-0.0±0.0**	**-0.0±0.0**	**-0.0±0.0**	**-0.0±0.0**	**-0.0±0.0**
R10_T*_*	**0.0±0.0**	-	-	-	-
R*_T*_*	1.5±3.5	**3.0±5.4**	**3.0±5.3**	**3.0±5.3**	2.8±5.3
Joint Optimization for Costs and Emissions					
R2_T*_*	2.1±4.1	3.7±6.0	**3.9±6.2**	3.6±6.2	3.6±6.3
R5_T*_*	**-0.0±0.0**	**-0.0±0.0**	**-0.0±0.0**	**-0.0±0.0**	**-0.0±0.0**
R10_T*_*	**-0.0±0.0**	-	-	-	-
R*_T*_*	1.4±3.5	2.9±5.5	**3.0±5.6**	2.8±5.7	2.9±5.8

Note that when the MIP finds improving solutions while optimizing for emissions it usually proofs closeness towards an optimal solution as can be seen in Table 5.3. For the calculations Table 5.3 instances have been excluded for which the MIP does not find improving solutions (using an improvement threshold of at least 5%). Therefore, it shows that for small instances or medium sized ones without handling constraints the MIP finds provable good solutions and can also serve as a benchmark for the solutions found by the heuristics.

Table 5.3: Average optimality gap with its in-group standard deviation achieved by a MIP solver when successfully optimizing for emissions only. No sensible optimality gaps can be calculated when optimizing for costs.

	Optimality Gap [%]
Optimize for Emissions	
R2_T*_*	2.9±3.3
R5_T*_*	7.6±4.5
R10_T*_*	12.8±6.3
R*_T*_*	5.1±5.0

If the MIP has to optimize either for costs only or for a combined objective it can't operate on the time-expanded network but needs to introduce graph-gadgets to linearize non-linear tariff structures (see Section 3.2). As a result, the model size increases significantly. This model blow-up leads to the MIP either not finding an improving solution or not fitting into the available memory at all. As a consequence, no sensible optimality gaps can be computed. This effect can be seen in Table 5.2, which shows that when optimizing for an objective including costs improving solutions can only be found for the smallest (and hence, unrealistic) test instances (see also Table C.1 and C.3). Additionally, this effect is visualized in Table 5.4. It shows the number of instances fitting into memory. While most of the instances can be loaded into memory when optimizing for emissions this picture reverses when graph-gadgets have to be used. The number of instances fitting into memory is additionally reduced when strengthening constraints are added, which too increase the size of the model formulation. Therefore, one can conclude that if one wants to optimize for costs on realistic large-scale instances incorporating realistic tariff scenarios one has to turn to different methods than exact solution approaches. These results motivate the study of heuristics for the general tactical transportation problem introduced in this work.

5.2.2.1 *A Closer Look at Strengthening the MIP Formulation*

Even though the MIP fails to solve large-scale GTTP instances when the optimization includes a cost term the strengthening constraints have measurable impact. Table 5.2 shows that when the MIP formulation is strengthened in general higher savings can be calculated. These results are statistically significant as shown in Table 5.5. On the one hand this analysis reveals that strengthening the formulation results in statistically significant improvements on the objective function, but on the other hand it also reveals that the strengthened formulations

Table 5.4: Number of instances for which the MIP fits into memory. In the first column the total number of instances in each group are shown.

	MIP0 [%]	MIP1 [%]	MIP2 [%]	MIP3 [%]	MIP4 [%]
Optimize for Emissions					
R2_T*_*	24/24	-	-	-	-
R5_T*_*	22/24	-	-	-	-
R10_T*_*	10/12	-	-	-	-
R*_T*_*	56/60	-	-	-	-
Optimize for Costs					
R2_T*_*	**22/24**	**22**	**22**	**22**	**22**
R5_T*_*	**7/24**	6	6	6	6
R10_T*_*	**1/12**	0	0	0	0
R*_T*_*	**30/60**	28	28	28	28
Joint Optimization for Costs and Emissions					
R2_T*_*	**22/24**	**22**	**22**	**22**	**22**
R5_T*_*	**9/24**	6	6	6	6
R10_T*_*	**1/12**	0	0	0	0
R*_T*_*	**32/60**	28	28	28	28

compared between themselves don't result in significant improvements. Therefore, one can conclude that for practical performance only adding capacity-strengthening constraints is sufficient.

Table 5.5: Statistical significance of different results obtained when choosing one MIP formulation compared to the others.

	MIP0	MIP1	MIP2	MIP3	MIP4
MIP0	-	< 0.01	< 0.01	< 0.01	< 0.01
MIP1	< 0.01	-	0.44	0.69	0.98
MIP2	< 0.01	0.44	-	0.61	0.62
MIP3	< 0.01	0.69	0.61	-	0.38
MIP4	< 0.01	0.98	0.62	0.38	-

Table 5.6 highlights the strengthening effects of the strengthening constraints on the polyhedron induced by the MIP formulation by summing over the objective value of the best found LP solutions on instances solved using all five different formulations `MIP0` - `MIP4`.

The results of Table 5.6 empirically proof the fact that certain different types of strengthening constraints indeed are strengthening the

Table 5.6: Effects of different MIP formulations on the best found LP solution. Only the smallest instance group is shown as it is the only instance group where each formulation could solve the LP relaxation in the given time limit of one hour.

	MIP0 [%]	MIP1 [%]	MIP2 [%]	MIP3 [%]	MIP4 [%]
Optimize for Costs					
R2_T*_*	584095	1143872	1143872	**1217610**	**1217610**
Joint Optimization for Costs and Emissions					
R2_T*_*	481982	696292	696292	**895098**	**895098**

formulation. Denote with `MIP0` - `MIP4` the polyhedra of the different formulations. By definition the following relations hold:

- $MIP4 \subseteq MIP3$ and $MIP3 \not\subset MIP4$
- $MIP4 \subseteq MIP2$ and $MIP2 \not\subset MIP4$
- $MIP3 \subseteq MIP1$ and $MIP1 \not\subset MIP3$
- $MIP2 \subseteq MIP1$ and $MIP1 \not\subset MIP2$
- $MIP1 \subseteq MIP0$ and $MIP0 \not\subset MIP1$

These are due to the following fact: Look at the left $\subseteq$ relations. Here, each formulation on the right hand side (RHS) employs constraints, which are a true subset of the employed constraints of the formulation on the left hand side (LHS) (see Table 5.1). Hence, the LHS formulations include at most all the solutions included in the RHS formulations proofing $RHS \not\subset LHS$ or additionally, cut away solutions found in the RHS polyhedron resulting in $LHS \subset RHS$.

Table 5.6 proofs that certain LHS formulations cut away solutions from the RHS polyhedron. Especially, it proofs:

- $MIP4 \subset MIP2$
- $MIP3 \subset MIP1$
- $MIP1 \subset MIP0$

and by transitivity

- $MIP4 \subset MIP1$ and $MIP4 \subset MIP0$
- $MIP3 \subset MIP0$
- $MIP2 \subset MIP0$

Section 3.2.5 shows that $MIP2$ and $MIP3$ are incomparable, i.e. $MIP2 \not\subseteq MIP3$ and $MIP3 \not\subseteq MIP2$, hence $MIP4 = MIP2 \cap MIP3 \subset MIP3$.

Furthermore, Section 3.2.5 shows $MIP2 \subset MIP1$ completing the polyhedral hierarchy investigation.

5.2.3 *Slope Scaling*

This section compares the results of applying different slope scaling mechanism to the problem data. Especially, using a cost-matching objective-coefficients updating scheme or a monotonic one (see Section 4.1) are explored.

Table 5.7 shows that the monotonic update scheme outperforms the objective-costs matching one. The reported variances are mainly due to the different saving potentials depending on the time horizon (see Tables C.4, C.5 and C.6). Table 5.8 proofs the statistical significance of the monotonic update scheme outperforming the cost-matching one.

This indicates that the prevalent design paradigm behind slope scaling mechanisms - matching objective costs between the original and approximate problem [17] - is not necessarily key to its success. It can be outperformed by a monotonic updating scheme trading cost-matching for smoothness.

Table 5.7: Relative improvement over the direct delivery solution with instance-group variance when applying slope scaling for one iteration or with different objective-coefficient updating schemes.

	1 Iteration [%]	Monotonic [%]	Cost-Matching [%]
Optimize for Emissions			
R2_T*_*	29.5±10.1	**29.8±10.0**	29.7±10.2
R5_T*_*	28.2±11.5	**29.3±11.3**	29.0±11.4
R10_T*_*	27.1±10.2	**28.5±10.2**	28.4±10.0
R*_T*_*	28.6±10.6	**29.4±10.4**	29.2±10.5
Optimize for Costs			
R2_T*_*	-0.7±4.5	**0.9±3.4**	-0.1±4.0
R5_T*_*	6.1±5.6	**12.3±5.6**	9.8±4.5
R10_T*_*	8.1±5.1	**16.5±5.0**	14.1±4.5
R*_T*_*	3.6±6.2	**8.3±7.9**	6.5±7.1
Joint Optimization for Costs and Emissions			
R2_T*_*	0.2±5.4	**2.1±4.0**	1.5±4.2
R5_T*_*	10.4±6.7	**14.7±6.4**	12.2±5.6
R10_T*_*	11.7±5.8	**17.9±5.4**	15.8±4.8
R*_T*_*	6.4±8.0	**10.0±8.6**	8.4±7.7

Section 4.1 shows that the original slope scaling updating scheme for binary variable is also monotonic. Therefore, these results raise the question if monotonicity of the updating scheme is a more funda-

mental property for a successful slope scaling algorithm than cost-matching.

Theses insight can be used to devise novel problem specific update rules diverting from pure objective-costs matching. Especially, when devising a slope scaling mechanism for a given non-binary MIP one should examine if cost-matching is appropriate to the given problem structure. As a practical example, the alternative updating schemes based on monotonicity used in this work successfully exploits the problem specific property of economics of scale through always encouraging sending more flow over an already established arc (see Figure 4.1 in Section 4.1).

Table 5.7 additionally shows, that using only one iteration of slope scaling results in good solutions close to the converged ones. This property is exploited by the local search which shows superior performance when applied on the first found solution (see the discussion in Section 5.2.4). Note that for convergence slope scaling often needs hundreds of iterations with most iterations not yielding an improving solution. For some large-scale instances slope scaling does not converge at all.

Table 5.8: Statistical significance of the results of SSC with monotonic updating rule compared to other SSC algorithms. Due to all p-values being smaller than 0.025 (0.05 divided by two for two comparisons, see Bonferroni correction [62]) the hypothesis that the slope scaling update rule or number of iterations result in the same mean objective value can be rejected.

	1 Iteration [\%]	Cost-Matching [\%]
Optimize for Emissions	< 0.025	< 0.025
Optimize for Costs	< 0.025	< 0.025
Joint Optimization	< 0.025	< 0.025

5.2.3.1 *A Closer Look at Exploiting Instance Structure*

Handling constraints capacitate the flow-sum over multiple arcs and hence, destroy the classic minimum-cost network flow structure.

Assume that the linear programm SSC solves at iteration t is called $\mathcal{LP}(t)$. An exact solution to $\mathcal{LP}(t)$ can of course be calculated efficiently by linear programming algorithms. However, if an instance has no capacitating handling constraints $\mathcal{LP}(t)$ takes the form of a multi-commodity minimum-cost flow problem [46]. In this case the results in Table 5.9 show that using the specialized *network simplex* algorithm [2] provided by CPLEX outperforms the more general simplex algorithms with respect to solution speed.

For instances with handling constraints the primal simplex outperforms the network simplex algorithm by an order of magnitude. Fur-

thermore, it also outperforms the dual simplex. Therefore, and as the dual basis information is not needed for the heuristics, in all reported SSC or hybrid heuristic results instances without handling constraints are always solved using the network simplex and instances with handling constraints are always solved using the primal simplex algorithm.

Table 5.9: Solving time of different simplex algorithms on the slope scaling linear programs. Results refer to the average solution time needed to perform the first iteration of slope scaling (i.e. solve $\mathcal{LP}(t = 0)$). Therefore, one does not need to distinguish updating schemes as they share the same initialization scheme. Averages are taken only over instances each method could solve in less then one hour time.

	Primal Simplex [s]	Dual Simplex [s]	Network Simplex [s]
Optimize for Emissions			
R*_T*_l	18.9	27.8	**12.5**
R*_T*_t	**98.7**	192.2	600.8
Optimize for Costs			
R*_T*_l	20.8	29.8	**12.6**
R*_T*_t	**46.6**	53.1	48.5
Joint Optimization for Costs and Emissions			
R*_T*_l	19.5	25.9	**12.7**
R*_T*_t	**46.7**	84.5	202.6

5.2.4 *Local Search*

An analysis of the local search heuristic introduced in Section 4.2 has multiple facets. In Section 5.2.4.1 the results of using different slope scaling variants as construction heuristics are presented. This yields the basis for comparing the use of different neighbourhood structures shown in Section 5.2.4.2. Section 5.2.4.3 examines the robustness of the local search towards small perturbations of the input data. Lastly, Section 5.2.4.4 examines the effectiveness of local search and different rerouting schemes in increasing the utilization rate of transport units.

5.2.4.1 *Comparing Different Local Search Initialization Schemes*

To compare the effect of different initialization schemes the local search variant `LS5` combining two types of path decomposition and rerouting schemes (see Table 5.12) is used. Table 5.10 shows the results of applying `LS5` on top of a solution provided by three different slope scaling variants. It reveals that using only one iteration of slope scaling

as a construction heuristic outperforms the other initialization schemes in a large majority of cases. Table 5.11 shows that the differences in performance are statistically significant.

Table 5.10: Relative improvement over the direct delivery solution when using different local search initialization schemes. Standard deviations of the mean are calculated over 10 independent runs. For the monotonic and cost-matching updating scheme slope scaling is run until convergence (or the time limit of one hour).

	1 Iteration [%]	Monotonic [%]	Cost-Matching [%]
Optimize for Emissions			
R2_T*_*	30.7±0.1	**30.9±0.1**	30.5±0.1
R5_T*_*	**31.9±0.2**	31.2±0.1	30.8±0.1
R10_T*_*	**30.9±0.3**	29.9±0.2	29.2±0.3
R*_T*_*	**31.2±0.1**	30.8±0.1	30.4±0.1
Optimize for Costs			
R2_T*_*	**6.0±0.2**	5.8±0.2	5.9±0.2
R5_T*_*	**20.1±0.3**	19.3±0.2	16.0±0.4
R10_T*_*	**23.6±0.5**	19.9±0.2	17.1±0.2
R*_T*_*	**14.9±0.3**	13.8±0.2	12.0±0.3
Joint Optimization for Costs and Emissions			
R2_T*_*	**8.0±0.2**	7.7±0.2	7.8±0.2
R5_T*_*	**21.5±0.3**	20.8±0.1	17.3±0.3
R10_T*_*	**24.3±0.4**	21.8±0.2	18.7±0.2
R*_T*_*	**16.4±0.3**	15.6±0.2	13.6±0.2

The superiority of using only one iteration of slope scaling can be explained due to two separate reasons:

- *Fast initial solution construction:* One iteration of SSC is completed very fast (see Table 5.9) giving the local search more time to improve the solution before the time limit of one hour is reached. This is a major factor for medium to large-scale instances as for these SSC sometimes needs hundreds of iterations to converge or does not converge at all. Consequently, the LS does not converge to a local optimum in the given time limit.
- *Diverse, suboptimal initial solution:* A solution obtained after one iteration of SSC is already quite good (see Table 5.7) but often uses a more diverse set of transport connections than a converged solution, which already cut away a lot of suboptimal routes. However, having a diverse set of opened routes seems to help the LS to explore different regions of the search space as by construction the LS favours using already established connections

than opening new ones. If the provided initial solution includes only a very restricted set of opened transport connections as in a converged SSC solution one can observe that the LS usually converges faster towards a suboptimal local minimum. In the extreme case of using the direct delivery solution as an initial solution, which only has direct lorry deliveries as established routes LS can be observed to sometimes fail to find any improvements at all. Therefore, it seems critical that the construction heuristic provides a solution with a diverse set of transport connections in line with findings for a similar problem reported in Harks et al. [21]. One iteration of slope scaling provides such an initial solution and the preliminary experiments with initializing the LS with the direct delivery solution were discontinued.

Table 5.11: Statistical significance of using one SSC iteration to initialize the local search compared to other SSC initialization schemes. Due to all p-values being smaller than 0.025 (0.05 divided by two for two comparisons, see Bonferroni correction [62] and Section 5.2.1) the hypothesis that the mean results of the local search are independent of the slope scaling variant can be rejected.

	Monotonic [\%]	Cost-Matching [\%]
Optimize for Emissions	< 0.025	< 0.025
Optimize for Costs	< 0.025	< 0.025
Joint Optimization	< 0.025	< 0.025

5.2.4.2 *A Closer Look at Using Different Local Search Neighbourhoods*

This section compares using different combinations of neighbourhoods introduced in Section 4.2. Table 5.12 gives on over of the local search variants explored. Each variant uses as a construction heuristic one iteration of slope scaling (see previous Section 5.2.4.1).

Table 5.13 compares the results of the different LS variants introduced in Table 5.12. Interestingly, it shows that not `LS5`, which uses the most different neighbourhoods but `LS2` provides the best solutions. Also, it can be observed that LS variants making use of the cheapest rerouting scheme in general outperform LS variants only making use of heaviest rerouting (`LS1` and `LS4`). This can be explained by the fact that cheapest relative rerouting searches - per definition - for the route which best uses a new or established transport connection and in this making very effective use of consolidation potentials.

However, the differences between the achieved objective values in Table 5.13 are small making the question of statistical significance an interesting one. When optimizing for emissions Table 5.14 shows

Table 5.12: Table of local search variants explored. Max. weight refers to a path decomposition calculated using an unidirectional DFS based on maximal associable weight and max. savings refers to additionally using a path decomposition calculated using a bidirectional DFS based on maximal achievable savings when changing to the local search remove-all move (see Section 4.2). Cheapest revers to employing cheapest-relative-cost rerouting while heaviest refers to heaviest-first rerouting. If combined LS starts with cheapest rerouting (see again Section 4.2).

Name	Path Decomposition	Rerouting Scheme
LS0	Max. Weight	Cheapest
LS1	Max. Weight	Heaviest
LS2	Max. Weight	Cheapest + Heaviest
LS3	Max. Weight + Max. Savings	Cheapest
LS4	Max. Weight + Max. Savings	Heaviest
LS5	Max. Weight + Max. Savings	Cheapest + Heaviest

Table 5.13: Results of the different local search variants defined in Table 5.12. Variance refers to the standard deviation from the mean over ten independent runs.

	LS0 [%]	LS1 [%]	LS2 [%]	LS3 [%]	LS4 [%]	LS5 [%]
Optimize for Emissions						
R2_T*_*	**30.7±0.1**	30.5±0.1	**30.7±0.1**	**30.7±0.1**	30.5±0.1	**30.7±0.1**
R5_T*_*	31.8±0.1	31.6±0.1	**31.9±0.1**	**31.9±0.1**	31.7±0.1	**31.9±0.2**
R10_T*_*	**30.9±0.1**	30.6±0.1	**30.9±0.1**	**30.9±0.1**	30.7±0.1	**30.9±0.3**
R*_T*_*	**31.2±0.1**	31.0±0.1	**31.2±0.1**	**31.2±0.1**	31.0±0.1	**31.2±0.1**
Optimize for Costs						
R2_T*_*	5.9±0.2	3.3±0.1	**6.0±0.2**	5.9±0.2	3.5±0.1	**6.0±0.2**
R5_T*_*	20.2±0.2	18.9±0.2	**20.3±0.2**	20.2±0.2	19.0±0.2	20.1±0.3
R10_T*_*	23.8±0.3	23.4±0.2	**23.9±0.3**	**23.9±0.3**	23.6±0.2	23.6±0.5
R*_T*_*	14.9±0.2	13.2±0.2	**15.0±0.2**	14.9±0.2	13.4±0.2	14.9±0.3
Joint Optimization for Costs and Emissions						
R2_T*_*	7.9±0.2	6.2±0.1	**8.0±0.2**	7.9±0.2	6.4±0.1	**8.0±0.2**
R5_T*_*	21.5±0.2	20.7±0.2	**21.6±0.2**	21.5±0.2	20.8±0.2	21.5±0.3
R10_T*_*	24.4±0.2	24.1±0.2	**24.6±0.2**	24.5±0.2	24.2±0.2	24.3±0.4
R*_T*_*	16.4±0.2	15.3±0.2	**16.5±0.2**	16.4±0.2	15.4±0.2	16.4±0.3

that LS variants only depending on heaviest rerouting result in statistically significant different results to LS variants also depending on cheapest rerouting. This phenomenon also appears when optimizing

for costs (see Table 5.15) or a joint optimization (see Table 5.16). This establishes the importance of using cheapest rerouting and that as a stand-alone (`LS0` and `LS3`) cheapest rerouting performs superior to heaviest rerouting. In all three significance test tables *LS2*, *LS3*, and *LS5* are statistically indistinguishable. However, comparing to *LS0* sometimes yielding statistically significant results. Therefore, one can argue that either using a second path decomposition or a second rerouting schemes helps the LS to find better local optima, but the presence of both is unnecessary and can even hurt. As a result, one can recommend choosing a second rerouting schemes rather than a second path decomposition as in the case here implementing the second rerouting schemes is easy whereas implementing the second path decomposition is more complex. Note that `LS2` is the only LS variant always yielding significantly better results to `LS0`. However, its results still are only slightly better than `LS0` making the most simple LS variant already a good choice to solve the current problem.

Table 5.14: Statistical significance of different local search variants when optimizing for emissions. To achieve a statistical significance level of 0.05 each individual test is performed with a significance level of 0.01 (see Section 5.2.1)

Optimizing for Emissions

	LS0	LS1	LS2	LS3	LS4	LS5
LS0	-	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01
LS1	< 0.01	-	< 0.01	< 0.01	< 0.01	< 0.01
LS2	< 0.01	< 0.01	-	0.68	< 0.01	0.09
LS3	< 0.01	< 0.01	0.68	-	< 0.01	0.06
LS4	< 0.01	< 0.01	< 0.01	< 0.01	-	< 0.01
LS5	< 0.01	< 0.01	0.09	0.06	< 0.01	-

Table 5.15: Statistical significance of different local search variants when optimizing for costs. To achieve a statistical significance level of 0.05 each individual test is performed with a significance level of 0.01 (see Section 5.2.1)

Optimizing for Costs

	LS0	LS1	LS2	LS3	LS4	LS5
LS0	-	< 0.01	< 0.01	0.06	< 0.01	0.14
LS1	< 0.01	-	< 0.01	< 0.01	< 0.01	< 0.01
LS2	< 0.01	< 0.01	-	0.09	< 0.01	0.48
LS3	0.06	< 0.01	0.09	-	< 0.01	0.51
LS4	< 0.01	< 0.01	< 0.01	< 0.01	-	< 0.01
LS5	0.14	< 0.01	0.48	0.51	< 0.01	-

Table 5.16: Statistical significance of different local search variants when jointly optimizing for costs and emissions. To achieve a statistical significance level of 0.05 each individual test is performed with a significance level of 0.01 (see Section 5.2.1)

Joint Optimization for Costs and Emissions

	LS0	LS1	LS2	LS3	LS4	LS5
LS0	-	< 0.01	< 0.01	< 0.01	< 0.01	0.05
LS1	< 0.01	-	< 0.01	< 0.01	< 0.01	< 0.01
LS2	< 0.01	< 0.01	-	0.06	< 0.01	0.72
LS3	< 0.01	< 0.01	0.06	-	< 0.01	0.28
LS4	< 0.01	< 0.01	< 0.01	< 0.01	-	< 0.01
LS5	0.05	< 0.01	0.72	0.28	< 0.01	-

5.2.4.3 *A Note on the Robustness of Results*

Local search using slope scaling as a construction heuristic proofs the most powerful solution algorithm for the optimization problem introduced in this work (see Section 5.2.5). Therefore, one could ask the question how robust solution of the LS are to slight perturbations of the input instance. This question is examined by taking a medium sized instance and generating the commodity distributions in it using 30 different seeds resulting in a total of 60 slightly different instances - 30 with and 30 without handling constraints.

Table 5.17 shows the result of applying `LS5` on this instance set. The variances refer to the standard deviation of the mean objective value calculation over the 30 instances. As these standard deviations are very low one can conclude that the results of the LS are robust with respect to small variations of the input instance.

Table 5.17: Table showing how much the solutions of the hybrid local search vary when the same instance is generated with 30 different seeds (i.e. different commodity and demand distributions). For this experiment, the following test instance has been chosen: fife regions, two warehouses per region, time horizon of 14, 100 commodities. Results are the mean improvements compared to the direct delivery solution and the standard deviations over the 30 seeds.

Optimize for	No Handling Constraints [%]	Handling Constraints [%]
Emissions	37.9 $\pm$ 1.8	30.7 $\pm$ 1.3
Costs	25.4 $\pm$ 1.6	24.7 $\pm$ 1.8
Both	26.6 $\pm$ 1.5	25.0 $\pm$ 1.6

Table 5.18: Utilization of transport units is calculated based on the maximum (percentual) filling degree with respect to volume and weight. The utilization values for each used transport mode are then averaged to yield a total utilization percentage. Local Search refers to LH2 and Slope Scaling to one iteration of SSC. LS variation refers to the standard deviation around the mean over 10 independent runs.

	Local Search [%]	Slope Scaling [%]	Direct Delivery [%]
Optimizing for Emissions			
R2_T*_*	65.4±0.5	62.6	**69.5**
R5_T*_*	**66.6±0.5**	59.5	62.7
R10_T*_*	**58.9±0.5**	51.8	54.3
R*_T*_*	**64.8±0.5**	59.4	63.7
Optimizing for Costs			
R2_T*_*	**70.8±1.3**	66.2	69.5
R5_T*_*	**71.6±0.8**	59.1	62.7
R10_T*_*	**64.7±0.7**	51.2	54.3
R*_T*_*	**70.1±1.0**	60.7	63.7
Joint Optimization for Costs and Emissions			
R2_T*_*	**70.1±1.1**	59.7	69.5
R5_T*_*	**71.5±0.7**	59.4	62.7
R10_T*_*	**64.3±0.7**	51.7	54.3
R*_T*_*	**69.7±0.9**	58.2	63.7

5.2.4.4 *Utilization of Transport Units*

Table 5.18 shows the degree (in percent) to which transport units are filled in the direct delivery solution, the slope scaling solution after one iteration (as a basis for LS) and after having applied LS. The results show that LS greatly increases the filling degree of used transport units from the SSC solution and significantly exceeding the direct delivery solution. Hence, one can argue that the employed LS is effective in finding solutions making good use of newly arising consolidation potentials. Furthermore, in the direct delivery column only the filling degree of lorry shipments are reported as no other transport modes are used. However, every LS solution uses at least two modes of transport and a lot use three, hence the reported filling degrees are averages over the transport modes and indicate very high utilization across transport modes.

Slope scaling after one iteration yields a suboptimal but very intermodal solution (see discussion in Section 5.2.4.1). Therefore, it shifts

a lot of flow on a diverse set of transport connections, which in the direct delivery solution are only transported via direct lorry shipments explaining its slightly lower utilization ratio.

While these findings have managerial implications as discussed in Section 5.3, in this section an algorithmic point is raised. Table 5.19 shows that in most cases using a cheapest compared to a heaviest rerouting scheme better utilizes existing transportation routes and as a result makes better use of consolidation potentials. This further explains the superiority of cheapest relative routing introduced in this work compared to heaviest rerouting introduced by Harks et al. [21].

Table 5.19: Table showing the utilization of transport units in local search solutions calculated either through cheapest (`LS0`) or heaviest (`LS1`) rerouting. Utilization of transport units is calculated as in Table 5.18. LS variation again refers to the standard deviation around the mean over 10 independent runs.

	LH0 [%]	LH1 [%]
Optimizing for Emissions		
R2_T*_*	**65.3±0.5**	64.8±0.4
R5_T*_*	**66.6±0.5**	66.1±0.5
R10_T*_*	**58.8±0.5**	58.2±0.4
R*_T*_*	**64.7±0.5**	64.2±0.4
Optimizing for Costs		
R2_T*_*	70.8±1.3	**70.9±0.4**
R5_T*_*	**71.5±0.8**	71.1±0.6
R10_T*_*	**64.5±0.8**	64.2±0.7
R*_T*_*	**70.0±1.0**	69.8±0.6
Joint Optimization for Costs and Emissions		
R2_T*_*	**70.1±1.0**	69.0±0.6
R5_T*_*	**71.4±0.7**	70.7±0.6
R10_T*_*	**64.1±0.7**	63.5±0.7
R*_T*_*	**69.6±0.8**	68.7±0.6

5.2.5 *Summary of Solution Approaches*

Table 5.20 compares the best performing algorithmic variant of each solution approach analyzed in the previous Sections 5.2.2, 5.2.3, and 5.2.4. It clearly shows that the hybrid local search outperforms the other solution approaches by large margins and most prominently when a cost term is included in the objective. It also shows that when

optimizing for emissions only slope scaling already results in good solutions. Furthermore, exactly solving the optimization problem fails except for the smallest of instances.

Table 5.20: Contrasting the three different solution approaches developed in this work by picking the best performing algorithmic variant of each approach for comparison. Therefore, local search refers to variant `LH2` (see Table 5.12), slope scaling uses a monotonic updating scheme (see Table 5.7), and MIP is `MIP2` (see Table 5.1). Reported results are percentual improvements over the direct delivery solutions. Standard deviations are calculated for each instance-group and independent of the method or objective are mainly caused by instances having different time horizons of 7, 14 or 30 days. For the local search results the mean values (of the instance-group mean and standard deviation) over 10 independent runs are reported.

	Local Search [%]	Slope Scaling [%]	MIP [%]
Optimize for Emissions			
R2_T*_*	**30.7±10.1**	29.8±10.0	27.3±13.9
R5_T*_*	**31.9±10.8**	29.3±11.3	15.3±18.6
R10_T*_*	**30.9±9.8**	28.5±10.2	9.2±15.1
R*_T*_*	**31.2±10.2**	29.4±10.4	19.4±17.4
Optimize for Costs			
R2_T*_*	**6.0±4.9**	0.9±3.4	3.8±5.8
R5_T*_*	**20.3±5.2**	12.3±5.6	0.0±0.0
R10_T*_*	**23.9±5.7**	16.5±5.0	-
R*_T*_*	**15.0±9.3**	8.3±7.9	3.0±5.3
Joint Optimization for Costs and Emissions			
R2_T*_*	**8.0±5.3**	2.1±4.0	3.9±6.2
R5_T*_*	**21.6±5.8**	14.7±6.4	0.0±0.0
R10_T*_*	**24.6±5.8**	17.9±5.4	-
R*_T*_*	**16.5±9.1**	10.0±8.6	3.0±5.6

The results are statistically significant due to the fact that the paired differences of the mean between `LH2` and `LH0` are already statistically significant but smaller by large margins than the paired differences between the different solution approaches in Table 5.20. Hence, for brevity the statistical significance test results are omitted.

5.3 MANAGERIAL INSIGHTS

The following discussion is based upon the best obtained results on each instance, which are consistently achieved by local search (see Table 5.20). While Table 5.20 shows the value of the objective, which may be costs, emissions or a combination of both, Table 5.21 shows the mean costs and emissions calculated for each instance group relative to the direct delivery solution. Based on these results the managerial implications of optimizing for emission reductions are discussed in Section 5.3.1, the implications of minimizing costs are discussed in Section 5.3.2, and a joint optimization is discussed in Section 5.3.3. A detailed analysis on the effects of collaboration on consolidation and intermodal usage can be found in Section 5.3.4.

Table 5.21: Percentual improvement of the costs and emissions of the best found solutions (calculated by local search variant `LH2`, see Section 5.2.4.2) compared to the respective quantity in the direct delivery solution. The reported standard deviations are calculated for each instance-group. Results are averages over 10 independent runs and found to be robust. Results for individual instances and run-wise standard deviations can be found in Appendix C.3.

	Costs [%]			**Emissions [%]**		
	Average	Min.	Max.	Average	Min.	Max.
Optimize for Emissions						
R2_T*_*	-7.1±5.8	-16.6	5.9	30.7±10.1	13.8	48.6
R5_T*_*	13.5±7.8	-3.0	25.7	31.9±10.8	16.7	53.0
R10_T*_*	17.3±8.7	-1.2	28.1	30.9±9.8	18.7	50.6
R*_T*_*	5.6±13.0	-16.6	28.1	31.2±10.2	13.8	53.0
Optimize for Costs						
R2_T*_*	6.0±4.9	0.4	16.7	8.6±4.8	1.3	18.7
R5_T*_*	20.3±5.2	10.7	30.1	23.6±8.4	11.1	37.2
R10_T*_*	23.9±5.7	16.3	30.8	24.3±7.3	13.0	37.7
R*_T*_*	15.0±9.3	0.4	30.8	17.5±10.1	1.3	37.7
Joint Optimization for Costs and Emissions						
R2_T*_*	4.8±5.6	-1.6	17.3	20.0±8.5	4.7	31.1
R5_T*_*	20.0±5.6	10.2	30.7	28.3±9.9	14.2	45.9
R10_T*_*	23.9±5.9	15.6	31.3	27.8±7.9	16.2	43.7
R*_T*_*	14.3±9.9	-1.6	31.3	24.8±9.8	4.7	45.9

5.3.1 *Concerning Minimizing Emissions*

Table 5.21 shows that optimizing for emissions universally leads to significant CO_2e reduction possibilities averaging 31.2%±10.2% (13.8% - 53.0%) *independent* of the size of the collaboration. However, we find emission reduction potentials are strongly correlated with the planning horizon. A planning horizon of 7 days lead to on average 20.5%±3.7% CO_2e reduction, 14 days to 33.0%±5.2%, and 30 days to 41.1%±7.6%. This is due to the fact that intermodal transport has longer lead times. Therefore, some changeover time from lorry transportation is needed to make effective use of its possibilities.

An unexptected result is that the best solutions for small instances with collaborating companies between two regions result on average in slightly increased costs (7.1%±5.8%). Interestingly, the calculated small cost increases together with the average emission reductions of roughly 30% are consistent with a real life case study conducted in the EU in which four companies (shippers) from two regions collaborated [63]. Hence, the results showcase the existence of a minimum necessary size of the collaboration such that decisions aiming at emission reduction also result in cost savings.

In general, one can conclude that larger collaborations correlate with larger cost saving potentials and using more than two collaborating regions leads to a direct net profit for collaborating companies. An open problem is how to best distribute the costs between the collaborating enterprises. This question of fairness is not addressed in this work, however Cruijssen [64] shows how it can be approached in practice based on concepts from cooperative game theory.

Note that there are three instances, two in instance group `R5_T*_*` and one in group `R10_T*_*`, which resulted in a cost increase when optimizing for emissions. However, these results are very specific outliers produced by the largest and computationally most demanding instances in these groups (see Appendix C.3). For these three instances the construction heuristic for the local search - one iteration of slope scaling - needed up one hour of time and the solution could not be improved by local search moves. Therefore, these results can be seen as artefacts due to stopping the optimization prematurely. All three instances are instances with handling constraints. In general, results on the same instance with and without handling constraints active show very similar results up to a few percentage points (see again Appendix C.3). Hence, it can be concluded that the results from their no handling constraints counterpart instances are a good predictor of the results obtained when the optimization could have proceeded. All three instances counterparts achieve double-digit cost savings strongly supporting the assumption that these three instances would achieve high cost savings given longer computation time.

5.3.2 *Concerning Minimizing Costs*

When optimizing for transportation costs contrary to emission minimization Table 5.21 showcases as expected a clear correlation of the objective with the size of the collaboration. This is due to the fact that larger collaborations increasingly enable cost-efficient consolidation and intermodality. Significant cost reductions are possible on all looked upon instances averaging 15.0%±9.3% (0.4% - 30.8%). Similar to optimizing for emissions a dependence on the planning horizon is found. Concretely, there is less saving potential for a horizon of 7 days (12.0%±8.9%) than 14 days (15.9%±9.2%) than 30 days (17.0%±9.1%). This can be explained by increased consolidation possibilities when more time periods are considered (see discussion concerning Table 5.22).

Additionally, Table 5.21 shows that as expected optimizing for transportation costs simultaneously leads to emission reductions which are more pronounced the larger the size of the collaboration and the more time periods considered. However, emission reductions are not as large as when optimizing for them only (see Section 5.3.1). Furthermore, there is no threshold on the size of the collaboration such that minimizing costs would also results in reduced emissions. Therefore, collaboration whatever the size always is a climate positive action.

5.3.3 *Concerning Jointly Minimizing Costs and Emissions*

Table 5.21 shows that internalizing CO_2e emissions results in a good trade-off between optimizing for costs or emissions only. It shows the same qualitative behaviour as optimizing for costs (see Section 5.3.2). On average it leads to only slightly less or equal cost saving solutions while significantly improving on the CO_2 footprint.

That a joint optimization can result in a few outliers with slightly better cost savings than only optimizing for costs can be seen in the maximum found cost saving column in Table 5.21. The increased cost savings are small and can be explained by the fact that including an emission term in the optimization can aid finding cost-effective solution when both objectives point towards the same regions in the solution space.

As explained in Section 5.2 for the current optimization one tonne of CO_2e is priced with 100€ [59] (using the avoidance cost approach [14]). Increasing the price tag on carbon emissions will most likely further encourage the use of greener intermodal transport.

Table 5.22: In-depth analysis of consolidation and intermodality. The **first column** gives a measure on the relative intermodal consolidation. It measures in percent how much less outgoing rail, ship and cross-dock to cross-dock transport units are ordered relative to the incoming truckloads (lorry TUs). The **second column** shows the number of ordered ship and rail TUs relative to all ordered TUs. The **last column** shows the percentual increase of the average filling rate calculated over all ordered TUs compared to the direct delivery solution. The non-relative total utilization ratio can be found in Table 5.18. Again, reported results are averages over 10 independent runs and found to be robust.

	Consolidation [%]	Rail & Ship [%]	Filling Rate [%]
Optimize for Emissions			
R2_T*_*	25.4±16.7	6.3±2.7	-5.6±9.1
R5_T*_*	13.1±8.2	7.2±4.4	7.4±10.2
R10_T*_*	20.2±10.6	7.9±5.6	13.8±13.5
R*_T*_*	19.4±13.7	6.9±4.0	3.1±12.8
Optimize for Costs			
R2_T*_*	10.9±9.0	3.2±1.6	3.0±5.9
R5_T*_*	15.6±7.5	2.8±0.5	15.1±8.3
R10_T*_*	19.5±7.7	2.9±0.6	24.8±13.8
R*_T*_*	14.3±8.7	3.0±1.1	11.8±11.8
Joint Optimization for Costs and Emissions			
R2_T*_*	10.2±8.9	4.3±1.4	1.8±5.2
R5_T*_*	13.9±7.7	4.0±1.2	15.0±8.4
R10_T*_*	18.2±7.1	4.1±1.5	23.9±13.4
R*_T*_*	13.1±8.5	4.1±1.4	11.1±11.8

5.3.4 *A Closer Look at Consolidation and Intermodal Usage*

Table 5.22 highlights intermodal consolidation, the use of greener transport modes such as ship and rail, and the general filling rates of ordered transport units. It shows that when optimizing for costs or jointly for costs and emissions consolidation effects are more pronounced and transport units better utilized when the size of the collaboration increases. Furthermore, internalizing emissions leads to higher usage of rail and ship connections. The small decrease in the measured consolidation effect and utilization rate when optimizing for both objectives compared to costs only can be explained due to the fact that greener intermodal opportunities start to be viable with less transport volume. This viability differences can be explained by emission reductions compensating higher intermodal consolidation

requirements, which would otherwise be necessary so that the chosen intermodal route is cost-efficient. When optimizing for emission the results show that there are a lot of consolidation and intermodal routing opportunities which could drastically cut emissions but are not yet viable from a cost perspective. An emission minimizing solution results in more than double the use of intermodal containers than optimizing for costs only.

Another type of consolidation not shown in Table 5.22 involves shipments between warehouses. These can happen either to send out consolidated shipments or to bring in a large volume of products into a region to satisfy its demands over multiple time periods using one consolidated shipment. Such a large shipment of products then requires storage in a regional warehouse over one or multiple time periods. We find when optimizing for emissions TUs on warehouse to warehouse connections make up on average 3.7%±4.2% (0% - 14.9%) of the total number of ordered TUs. When optimizing for costs they make up 2.6%±2.5%(0% - 8.4%) and when optimizing for both they make up 2.4%±2.5%(0% - 8.5%). This small decrease when optimizing for both objectives can be explained by the increased intermodal usage. Without horizontal collaboration (in the direct delivery solutions) this quantity is effectively zero and hence, these results point towards an active use of warehouse sharing.

Lastly, the average delivery time of a shipment on the last arc to a demand node drops on average to 0.3±0.2 days when optimizing for emissions compared to on average 1.2±0.2 days in the direct delivery solutions. Optimizing for costs results in 0.7±0.1 days and optimizing for both objectives results in 0.6±0.2 days. This indicates that products are very often shipped into a demand region by means different to direct delivery by lorry.

6

CONCLUSION

The conclusion will appear in partly shortened form in Gosch et al. [1].

This work introduced a new mathematical model for tactical transport planning in a horizontal collaboration. The model incorporates realistic tariff structures, intermodal transport, handling capacities and storage possibilities. Furthermore, it can be applied to plan transportation with a diverse set of products among others supporting perishability or special transport needs such as cooling. The model allows for sustainable planning through internalizing CO_2e costs. Hence, it can be used to minimize transportation costs, emissions or both together. Subsequently, graph-structures to linearize the non-linear cost-functions induced by the tariffs are developed and a mixed-integer formulation of the model is derived. To aid the exact solution process, multiple valid inequalities have been developed and proven to strengthen the model formulation.

Furthermore, a memory efficient hybrid heuristic to solve large-scale instances has been developed. The hybrid heuristic is composed of two parts, a matheuristic called slope scaling, which is generalized to non-negative integer variables and a local search. In generalizing slope scaling, a new design principle based upon monotonicity has been found to outperform the traditional cost-matching one.

These solution techniques were successfully applied to generated problem instances based on the real transportation infrastructure - including railway and shipping - in the Danube Region. This revealed significant saving potentials in both costs and emissions. As expected, optimizing for transportation costs automatically leads to a reduced carbon footprint. However, optimizing for the carbon footprint only necessitates a minimum size of the collaboration for transportation costs to simultaneously decrease. Therefore, these results support the assumption that horizontal collaboration in warehouse-sharing and unlocking intermodal freight transport opportunities positively impact emissions and climate-neutral actions. As a result, further efforts need to be made to promote collaboration in real-world settings.

Interesting future work perspectives are to incorporate operational or strategic decisions into the model formulation. From an algorithmic perspective, it would be interesting to incorporate stochastic elements into the slope scaling mechanism and investigate if this allows to visit larger regions of the search space and avoid being trapped in a solution loop. Furthermore, it would be interesting to develop non-model based heuristics to be able to scale to even larger instances than generated

in this work. Large fixed-charge network design problems (MCND) are still challenging for exact and heuristic solution approaches [19, 48]. However, the standard benchmark dataset, modern heuristics for the MCND are evaluated on [19, 20], was introduced nearly 20 years ago by Crainic et al. [65] and its instances are comparatively small to the instances generated in this work. Exemplarely, instances in the standard benchmark dataset have at most 100 nodes and 700 arcs. However, instances in this work can have thousands of nodes and up to hundreds of thousands of arcs. Therefore, it would be a worthwhile endeavour to generate a new larger benchmark dataset and investigate state-of-the-art heuristics on how well they scale. Such an undertaking would greatly benefit other researcher and practitioners in choosing the correct algorithmic solution approach when confronted with possibly real-world instances in size not represented through the standard benchmark dataset.

Wahrlich:
Erkennst du das Da-Sein als einen Gewinn,
Erkenne: Das Nicht-Sein macht brauchbar.

– Laozi

BIBLIOGRAPHY

[1] Lukas Gosch, Matthias Prandtstetter, and Karl F. Doerner. "On Modelling and Solving Green Collaborative Transportation Planning." In: *Proceedings of the 8th International Physical Internet Conference IPIC 2021* (2021).

[2] Dimitris Bertsimas and John Tsitsiklis. *Introduction to Linear Optimization*. 1st. Athena Scientific, 1997.

[3] Laurence A. Wolsey. *Integer Programming*. John Wiley & Sons, Inc., 1998.

[4] Ralph Sims et al. "Transport. In: Climate Change 2014: Mitigation of Climate Change. Contribution of Working Group III to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change." In: Sept. 2014.

[5] OECD. *Delivering the Goods*. 2003. URL: https://www.oecd-ilibrary.org/content/publication/9789264102828-en.

[6] International Transport Forum. *ITF Transport Outlook 2019*. 2019. URL: https://www.oecd-ilibrary.org/content/publication/transp_outlook-en-2019-en.

[7] Sean Doherty and Seb Hoyle. "Supply Chain Decarbonization." In: *World Economic Forum, Geneva* (2009).

[8] European Union. "Commission notice: guidelines on the applicability of Article 81 of the EC Treaty to horizontal cooperation agreements." In: *2001/C3/02* (2001).

[9] ALICE. *Global supply network coordination and collaboration: research and innovation roadmap*. 2014. URL: https://www.etp-logistics.eu/wp-content/uploads/2015/08/W46mayo-kopie.pdf.

[10] ALICE. *A framework and process for the development of a ROADMAP TOWARDS ZERO EMISSIONS LOGISTICS 2050*. 2019. URL: http://www.etp-logistics.eu/wp-content/uploads/2019/12/Alice-Zero-Emissions-Logistics-2050-Roadmap-WEB.pdf.

[11] Benoit Montreuil. "Toward a Physical Internet: meeting the global logistics sustainability grand challenge." en. In: *Logistics Research* 3.2-3 (May 2011), pp. 71–87.

[12] Teodor Gabriel Crainic. "Service network design in freight transportation." In: *European Journal of Operational Research* 122.2 (2000), pp. 272 –288.

[13] Gianpaolo Ghiani, Gilbert Laporte, and Roberto Musmanno. "Introducing Logistics." In: *Introduction to Logistics Systems Management*. John Wiley & Sons, Ltd, 2013. Chap. 1, pp. 1–43.

[14] Alan McKinnon, Michael Brown, Maja Piecyk, and Anthony Whiteing. *Green Logistics : improving the environmental sustainability of logistics*. 3th. Kogan Page Limited, 2015.

[15] Charles L. Munson and Jonathan Jackson. "Quantity Discounts: An Overview and Practical Guide for Buyers and Sellers." English. In: *Foundations and Trends® in Technology, Information and Operations Management* 8.1–2 (June 2015). Publisher: Now Publishers, Inc., pp. 1–130.

[16] T. L. Magnanti and R. T. Wong. "Network Design and Transportation Planning: Models and Algorithms." In: *Transportation Science* 18.1 (1984), pp. 1–55.

[17] Dukwon Kim and Panos M. Pardalos. "A solution approach to the fixed charge network flow problem using a dynamic slope scaling procedure." In: *Operations Research Letters* 24.4 (1999), pp. 195–203.

[18] Teodor Gabriel Crainic, Bernard Gendron, and Geneviève Hernu. "A Slope Scaling/Lagrangean Perturbation Heuristic with Long-Term Memory for Multicommodity Capacitated Fixed-Charge Network Design." In: *J. Heuristics* 10 (Sept. 2004), pp. 525–545.

[19] Bernard Gendron, Saïd Hanafi, and Raca Todosijević. "Matheuristics based on iterative linear programming and slope scaling for multicommodity capacitated fixed charge network design." In: *European Journal of Operational Research* 268.1 (2018), pp. 70 –81.

[20] Mohammad Rahim Akhavan Kazemzadeh, Tolga Bektaş, Teodor Gabriel Crainic, Antonio Frangioni, Bernard Gendron, and Enrico Gorgone. "Node-based Lagrangian relaxations for multicommodity capacitated fixed-charge network design." In: *Discrete Applied Mathematics* (2021).

[21] Tobias Harks, Felix König, Jannik Matuschke, Alexander Richter, and Jens Schulz. "An Integrated Approach to Tactical Transportation Planning in Logistics Networks." In: *Transportation Science* 50(2) (2016), pp. 439–460.

[22] George L. Nemhauser and Laurence A. Wolsey. *Integer and Combinatorial Optimization*. USA: Wiley-Interscience, 1988.

[23] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. second. New York, NY, USA: Springer, 2006.

[24] Markus Grasmair. *Discrete Optimisation, Lecture Notes Winter 2010/11*. Computational Science Center, University of Vienna.

[25] GEORGE B. DANTZIG. *Linear Programming and Extensions*. Princeton University Press, 1991.

[26] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. 3rd. The MIT Press, 2009.

[27] David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. 1st. USA: Cambridge University Press, 2011. ISBN: 0521195276.

[28] Michel Gendreau and Jean-Yves Potvin. *Handbook of Metaheuristics*. 2nd. Springer Publishing Company, Incorporated, 2010.

[29] IBM. *IBM ILOG CPLEX 12.10 User's Manual*. Incline Village, NV: IBM ILOG CPLEX Division, 2019.

[30] LLC Gurobi Optimization. *Gurobi Optimizer Reference Manual*. 2021. URL: http://www.gurobi.com.

[31] Kenneth Sörensen. "Metaheuristics—the metaphor exposed." In: *International Transactions in Operational Research* 22.1 (2015), pp. 3–18.

[32] El-Ghazali Talbi. *Metaheuristics: From Design to Implementation*. Wiley Publishing, 2009.

[33] John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. Cambridge, MA, USA: MIT Press, 1992.

[34] Augustin Cauchy. "Methode generale pour la resolution des systemes d equations simultanees." In: *Compte Rendu a l Academie des Sciences* 25 (1847), pp. 536–538.

[35] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. URL: http://www.deeplearningbook.org.

[36] D.H. Wolpert and W.G. Macready. "No free lunch theorems for optimization." In: *IEEE Transactions on Evolutionary Computation* 1.1 (1997), pp. 67–82.

[37] Günther Raidl, Jakob Puchinger, and Christian Blum. "Metaheuristic Hybrids." In: Sept. 2010, pp. 469–496.

[38] Christian Blum, Jakob Puchinger, Günther R. Raidl, and Andrea Roli. "Hybrid metaheuristics in combinatorial optimization: A survey." In: *Applied Soft Computing* 11.6 (2011), pp. 4135–4151.

[39] Matteo Fischetti and Andrea Lodi. "Local branching." In: *Mathematical Programming* 98 (Sept. 2003), pp. 23–47.

[40] Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. "Machine learning for combinatorial optimization: A methodological tour d'horizon." In: *European Journal of Operational Research* 290.2 (2021), pp. 405–421.

[41] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. "Neural Message Passing for Quantum Chemistry." In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 1263–1272.

[42] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. "The Graph Neural Network Model." In: *IEEE Transactions on Neural Networks* 20.1 (2009), pp. 61–80.

[43] Quentin Cappart, Didier Chételat, Elias B. Khalil, Andrea Lodi, Christopher Morris, and Petar Velickovic. "Combinatorial optimization and reasoning with graph neural networks." In: *CoRR* abs/2102.09544 (2021). arXiv: 2102.09544. URL: https://arxiv.org/abs/2102.09544.

[44] Sunil Chopra. *Supply Chain Management: Strategy, Planning, and Operation*. English (US). 7th. Pearson Education, 2019.

[45] Teodor Gabriel Crainic and Gilbert Laporte. "Planning models for freight transportation." In: *European Journal of Operational Research* 97.3 (1997), pp. 409–438.

[46] Ravindra K Ahuja, Thomas L Magnanti, and James B Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice hall, 1993.

[47] Bernard Gendron, Teodor Gabriel Crainic, and Antonio Frangioni. "Multicommodity Capacitated Network Design." In: *Telecommunications Network Planning*. Ed. by Brunilde Sansò and Patrick Soriano. Boston, MA: Springer US, 1999, pp. 1–19.

[48] Mervat Chouman, Teodor Gabriel Crainic, and Bernard Gendron. "Commodity Representations and Cut-Set-Based Inequalities for Multicommodity Capacitated Fixed-Charge Network Design." In: *Transportation Science* 51 (July 2016).

[49] Bernard Gendron and Mathieu Larose. "Branch-and-price-and-cut for large-scale multicommodity capacitated fixed-charge network design." In: *EURO Journal on Computational Optimization* 2 (June 2014), pp. 55–75.

[50] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. USA: W. H. Freeman & Co., 1990. ISBN: 0716710455.

[51] Jens Kempkes and Achim Koberstein. "A Resource Based Mixed Integer Modelling Approach for Integrated Operational Logistics Planning." In: *Lecture Notes in Business Information Processing* 46 (Jan. 2010), pp. 281–294.

[52] Tolga Bektaş, Jan Fabian Ehmke, Harilaos N. Psaraftis, and Jakob Puchinger. "The role of operational research in green freight transportation." In: *European Journal of Operational Research* 274.3 (2019), pp. 807–823.

[53] Christophe Wilbaut and Said Hanafi. "New convergent heuristics for 0–1 mixed integer programming." In: *European Journal of Operational Research* 195.1 (2009), pp. 62 –74.

[54] Saïd Hanafi and Christophe Wilbaut. "Improved convergent heuristics for the 0-1 multidimensional knapsack problem." en. In: *Annals of Operations Research* 183.1 (Mar. 2011), pp. 125–142.

[55] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer, Berlin, Germany, 2004.

[56] David Wolfinger, Fabien Tricoire, and Karl F. Doerner. "A matheuristic for a multimodal long haul routing problem." In: *EURO Journal on Transportation and Logistics* 8.4 (2019), pp. 397–433.

[57] Matthias Prandtstetter, Markus Straub, and Jakob Puchinger. "On the way to a multi-modal energy-efficient route." In: *IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society*. 2013, pp. 4779–4784.

[58] URL: https://www.aqua-calc.com/page/density-table/substance/styrofoam (visited on 05/29/2021).

[59] CE Delft. *Handbook on the external costs of transport*. Publications Office of the European Union, Luxembourg, 2019.

[60] Frank Wilcoxon. "Individual Comparisons by Ranking Methods." In: *Biometrics Bulletin* 1.6 (1945), pp. 80–83.

[61] Rudolf Frühwirth. *Wahrscheinlichkeitsrechnung und Statistik für Studierende der Physik*. Institut für Hochenergiephysik der Österreichischen Akademie der Wissenschaften (HEPHY), 2014.

[62] Winston Haynes. "Bonferroni Correction." In: *Encyclopedia of Systems Biology*. Ed. by Werner Dubitzky, Olaf Wolkenhauer, Kwang-Hyun Cho, and Hiroki Yokota. New York, NY: Springer New York, 2013, pp. 154–154. URL: https://doi.org/10.1007/978-1-4419-9863-7_1213.

[63] Kurt Jacobs, Steven Vercammen, and Sven Verstrepen. *CO^3 Test Project Report: Creation of an Orchestrated Intermodal Partnership Between Multiple Shippers*. 2013. URL: http://www.co3-project.eu/wo3/wp-content/uploads/2011/12/CO3-D4-2-exec-summary.pdf.

[64] Frans Cruijssen. *CO^3 Position Paper: Framework for Collaboration*. 2011. URL: http://www.co3-project.eu/wo3/wp-content/uploads/2011/12/CO3-D-2-1-Framework-for-collaboration-full-report-2.pdf.

[65] Teodor Gabriel Crainic, Antonio Frangioni, and Bernard Gendron. "Bundle-based relaxation methods for multicommodity capacitated fixed charge network design." In: *Discrete Applied Mathematics* 112.1 (2001). Combinatorial Optimization Symposium, Selected Papers, pp. 73–99.

A

IMPLEMENTATION NOTES

A.1 STRENGTHENED CAPACITY CONSTRAINTS

The instances used in this work separate two types of commodities: those that need electricity (σ_1) and those who don't (σ_2). Products of type σ_1 are always perishable, products of type σ_2 are always non-perishable.

An upper bound for the product-extent of type σ_1 for an arc starting at a node v in time period t and ending in time period t' is calculated as follows: sum over the extent of all σ_1-products produced up to time period $t-1$ and, which won't be expired in time period t'. Then, add the extent of σ_1-products sourced by node v at time period t.

An upper bound for products of type σ_2 on the same arc is calculated by summing over the extent of all σ_2-products supplied up to time period $t-1$ and subtracting the extent of all σ_2-products demanded up to time period t. Furthermore, add all σ_2-products supplied by node v at time t.

A.2 PATH DECOMPOSITION PSEUDOCODE

Algorithm 9 presents the pseudocode for how to calculate a bin and demand path decomposition based on a unidirectional depth-first search (see Section 4.2.1.1). It operates on a time-expanded graph $\mathcal{G}_\mathcal{T} = (\mathcal{V}_\mathcal{T}, \mathcal{A}_\mathcal{T})$ assumed to be globally accessable. Furthermore the existence of a global unassigned flow variable x_a for each arc $a \in \mathcal{A}_\mathcal{T}$ is assumed that initially corresponds to the flow on arc a in a calculated initial solution.

Algorithm 9 can be adapted to the bidirectional case (see Section 4.2.1.2) by implemented a backward direction in `dfs_construct` and calling it before line 11. Additionally, instead of lines 4 to 8 do iterate over all product types σ, sort arcs based on highest savings potential and always choose the arc with the current highest savings potential for the inner loop part.

Algorithm 9: Calculate Unidirectional Path Decomposition

```
1  Function calc_path_decomposition():
2      bin_path_decomp ← empty list
3      demand_path_decomp ← empty list
4      Select subset S of source nodes in V_T
5      Sort all s ∈ S chronologically
6      foreach s ∈ S in chronological order do
7          foreach product type σ do
8              while there is unassinged outgoing flow of type σ do
9                  P ← new empty path
10                 P.σ ← σ
11                 dfs_construct(P, s)
12                 if destination node of P is of type bin then
13                     Add P to bin_path_decomp
14                 else
15                     Add P to demand_path_decomp
16                 foreach arc a in P do
17                     x_a ← x_a − P.x
18 Function dfs_construct(P, n):
19     a_best ← None
20     x_sum_best ← −1
21     x_best ← None
22     foreach outgoing arc a from node n do
23         x ← Get assignable flow of type P.σ to path P when
             choosing arc a
24         x_sum ← Calculate aggregated flow attribute (such as
             total weight) for x
25         if x_sum > x_sum_best and x not empty then
26             a_best ← a
27             x_sum_best ← x_sum
28             x_best ← x
29     if a_best is None then
30         return
31     else
32         Add arc a_best to path P
33         P.x ← x_best
34         n ← destination node of a_best
35         dfs_construct(P, n)
```

B

INSTANCES TABLE

Table B.1 lists all generated instances and their properties. Note that each row in Table B.1 corresponds to two instances one with and one without handling capacity constraints.

Table B.1: Lists the properties of all 60 generated instances. The properties in the table are given in the following order: number of selected regions, number of collaborating companies, number of total warehouses, number of included transshipment nodes (cross-docks, train stations and ports), number of commodities (half of which are perishable), and time horizon T. Each row corresponds to two instances one with and one without handling capacity constraints.

Regions	Companies	Warehouses	Transshipment	Commodities	T
2	2	2	6	20	7
2	2	2	6	20	14
2	2	2	6	20	30
2	2	2	6	100	7
2	2	2	6	100	14
2	2	2	6	100	30
2	8	8	6	100	7
2	8	8	6	100	14
2	8	8	6	100	30
2	8	8	6	500	7
2	8	8	6	500	14
2	8	8	6	500	30
5	10	10	14	100	7
5	10	10	14	100	14
5	10	10	14	100	30
5	10	10	14	500	7
5	10	10	14	500	14
5	10	10	14	500	30
5	15	20	14	100	7
5	15	20	14	100	14
5	15	20	14	100	30
5	15	20	14	500	7
5	15	20	14	500	14
5	15	20	14	500	30
10	20	30	22	100	7
10	20	30	22	100	14
10	20	30	22	100	30
10	20	30	22	500	7
10	20	30	22	500	14
10	20	30	22	500	30

C

DETAILED RESULT TABLES

This chapter shows the results of the applied solution procedures on every generated instance. Individual instances are identified using *R<#>_W<#>_C<#>_K<#>_T<#>_t/l*. *W<#>* indicates total number of warehouses and *C<#>* the number of collaborating companies. *K<#>* indicates the number of commodities in the instance.

The results of applying a MIP solver on the different MIP formulation are displayed in Section C.1. The results obtained by slope scaling can be found in Section C.2. Section C.3 shows not the objective but the costs and emissions of each instance when applying local search.

When not otherwise specified results are percentual improvements with 0.00 referring to 0% and 1.00 referring to 100%.

C.1 MIXED-INTEGER PROGRAMMING

Most instances have no relative improvement as the MIP is initialized with the direct delivery solution but fails to find an improved solution. NaN indicates that the MIP did not fit into the available memory. Only instances where at least one MIP fitted into memory are shown.

Table C.1: **Optimize for Costs:** Relative improvement on the objective function compared to the direct delivery solution obtained when using the different MIP formulation with CPLEX 12.10. and 1h computing time.

	MIP1	MIP2	MIP3	MIP4	MIP5
R2_W2_C2_K100_T14_l	-0.00	-0.00	-0.00	-0.00	-0.00
R2_W2_C2_K100_T14_t	0.00	-0.00	-0.00	-0.00	-0.00
R2_W2_C2_K100_T30_l	-0.00	-0.00	-0.00	-0.00	-0.00
R2_W2_C2_K100_T30_t	-0.00	-0.00	-0.00	-0.00	-0.00
R2_W2_C2_K100_T7_l	-0.00	-0.00	-0.00	-0.00	-0.00
R2_W2_C2_K100_T7_t	-0.00	-0.00	-0.00	-0.00	0.00
R2_W2_C2_K20_T14_l	0.11	0.14	0.14	0.13	0.14
R2_W2_C2_K20_T14_t	0.12	0.13	0.13	0.14	0.14
R2_W2_C2_K20_T30_l	0.06	0.17	0.16	0.15	0.15

Continued on next page

Table C.1: **Optimize for Costs:** Relative improvement on the objective function compared to the direct delivery solution obtained when using the different MIP formulation with CPLEX 12.10. and 1h computing time.

	MIP1	MIP2	MIP3	MIP4	MIP5
R2_W2_C2_K20_T30_t	-0.00	0.15	0.15	0.17	0.15
R2_W2_C2_K20_T7_l	0.07	0.07	0.07	0.07	0.07
R2_W2_C2_K20_T7_t	0.07	0.07	0.07	0.07	0.07
R2_W8_C8_K100_T14_l	-0.00	-0.00	-0.00	-0.00	-0.00
R2_W8_C8_K100_T14_t	-0.00	-0.00	-0.00	-0.00	-0.00
R2_W8_C8_K100_T30_l	-0.00	-0.00	-0.00	-0.00	-0.00
R2_W8_C8_K100_T30_t	-0.00	-0.00	-0.00	-0.00	-0.00
R2_W8_C8_K100_T7_l	-0.00	0.05	0.06	0.06	0.05
R2_W8_C8_K100_T7_t	-0.00	0.04	0.05	0.05	-0.00
R2_W8_C8_K500_T14_l	-0.00	-0.00	-0.00	-0.00	-0.00
R2_W8_C8_K500_T14_t	-0.00	-0.00	-0.00	-0.00	-0.00
R2_W8_C8_K500_T7_l	-0.00	-0.00	-0.00	-0.00	-0.00
R2_W8_C8_K500_T7_t	-0.00	-0.00	-0.00	-0.00	-0.00
R5_W10_C10_K100_T14_l	-0.00	-0.00	-0.00	-0.00	-0.00
R5_W10_C10_K100_T14_t	-0.00	-0.00	-0.00	-0.00	-0.00
R5_W10_C10_K100_T7_l	-0.00	-0.00	-0.00	-0.00	-0.00
R5_W10_C10_K100_T7_t	-0.00	-0.00	-0.00	-0.00	-0.00
R5_W20_C15_K100_T14_l	-0.00	NaN	NaN	NaN	NaN
R5_W20_C15_K100_T7_l	-0.00	-0.00	-0.00	-0.00	-0.00
R5_W20_C15_K100_T7_t	-0.00	-0.00	-0.00	-0.00	-0.00
R10_W30_C20_K100_T7_l	-0.00	NaN	NaN	NaN	NaN

Table C.2: **Optimize for Emissions:** Relative improvement on the objective function compared to the direct delivery solution obtained when using the different MIP formulation with CPLEX 12.10. and 1h computing time.

	MIP1	MIP2	MIP3	MIP4	MIP5
R2_W2_C2_K100_T14_l	0.39	0.39	0.39	0.39	0.39
R2_W2_C2_K100_T14_t	0.34	0.34	0.34	0.34	0.34
R2_W2_C2_K100_T30_l	0.49	0.49	0.49	0.49	0.49
R2_W2_C2_K100_T30_t	0.41	0.41	0.41	0.41	0.41

Continued on next page

Table C.2: **Optimize for Emissions:** Relative improvement on the objective function compared to the direct delivery solution obtained when using the different MIP formulation with CPLEX 12.10. and 1h computing time.

	MIP1	MIP2	MIP3	MIP4	MIP5
R2_W2_C2_K100_T7_l	0.24	0.24	0.24	0.24	0.24
R2_W2_C2_K100_T7_t	0.21	0.21	0.21	0.21	0.21
R2_W2_C2_K20_T14_l	0.29	0.29	0.29	0.29	0.29
R2_W2_C2_K20_T14_t	0.28	0.28	0.28	0.28	0.28
R2_W2_C2_K20_T30_l	0.39	0.39	0.39	0.39	0.39
R2_W2_C2_K20_T30_t	0.39	0.39	0.39	0.39	0.39
R2_W2_C2_K20_T7_l	0.17	0.17	0.17	0.17	0.17
R2_W2_C2_K20_T7_t	0.16	0.16	0.16	0.16	0.16
R2_W8_C8_K100_T14_l	0.35	0.35	0.35	0.35	0.35
R2_W8_C8_K100_T14_t	0.30	0.30	0.30	0.30	0.30
R2_W8_C8_K100_T30_l	0.47	0.47	0.47	0.47	0.47
R2_W8_C8_K100_T30_t	0.01	0.01	0.01	0.01	0.01
R2_W8_C8_K100_T7_l	0.23	0.23	0.23	0.23	0.23
R2_W8_C8_K100_T7_t	0.20	0.20	0.20	0.20	0.20
R2_W8_C8_K500_T14_l	0.35	0.35	0.35	0.35	0.35
R2_W8_C8_K500_T14_t	0.04	0.03	0.04	0.05	0.04
R2_W8_C8_K500_T30_l	0.45	0.45	0.45	0.45	0.45
R2_W8_C8_K500_T30_t	0.00	0.00	0.00	0.00	0.00
R2_W8_C8_K500_T7_l	0.22	0.22	0.22	0.22	0.22
R2_W8_C8_K500_T7_t	0.19	0.19	0.20	0.19	0.19
R5_W10_C10_K100_T14_l	0.42	0.42	0.42	0.43	0.41
R5_W10_C10_K100_T14_t	-0.00	-0.00	-0.00	-0.00	-0.00
R5_W10_C10_K100_T30_l	0.51	0.51	0.51	0.51	0.51
R5_W10_C10_K100_T30_t	0.00	0.00	0.00	0.00	0.00
R5_W10_C10_K100_T7_l	0.25	0.25	0.25	0.25	0.25
R5_W10_C10_K100_T7_t	0.21	0.05	0.21	0.21	0.06
R5_W10_C10_K500_T14_l	0.35	0.34	0.35	0.35	0.34
R5_W10_C10_K500_T14_t	0.00	0.00	0.00	0.00	0.00
R5_W10_C10_K500_T30_l	0.00	0.00	0.00	0.00	0.00
R5_W10_C10_K500_T30_t	0.00	0.00	0.00	0.00	0.00
R5_W10_C10_K500_T7_l	0.22	0.22	0.22	0.22	0.22

Continued on next page

Table C.2: **Optimize for Emissions:** Relative improvement on the objective function compared to the direct delivery solution obtained when using the different MIP formulation with CPLEX 12.10. and 1h computing time.

	MIP1	MIP2	MIP3	MIP4	MIP5
R5_W10_C10_K500_T7_t	-0.00	-0.00	-0.00	-0.00	-0.00
R5_W20_C15_K100_T14_l	0.38	0.39	0.38	0.39	0.38
R5_W20_C15_K100_T14_t	0.00	0.00	0.00	0.00	0.00
R5_W20_C15_K100_T30_l	0.51	0.51	0.51	0.51	0.51
R5_W20_C15_K100_T30_t	-0.00	-0.00	-0.00	-0.00	-0.00
R5_W20_C15_K100_T7_l	0.26	0.26	0.26	0.26	0.26
R5_W20_C15_K100_T7_t	0.03	-0.00	0.03	-0.00	-0.00
R5_W20_C15_K500_T14_l	0.00	0.35	0.35	0.00	0.35
R5_W20_C15_K500_T14_t	0.00	0.00	0.00	0.00	0.00
R5_W20_C15_K500_T7_l	0.22	0.22	0.22	0.22	0.22
R5_W20_C15_K500_T7_t	0.00	0.00	0.00	0.00	0.00
R10_W30_C20_K100_T14_l	0.37	0.37	0.37	0.37	0.37
R10_W30_C20_K100_T14_t	-0.00	-0.00	-0.00	-0.00	-0.00
R10_W30_C20_K100_T30_l	0.00	0.00	0.00	0.00	0.00
R10_W30_C20_K100_T30_t	0.00	0.00	0.00	0.00	0.00
R10_W30_C20_K100_T7_l	0.30	0.30	0.30	0.30	0.30
R10_W30_C20_K100_T7_t	0.00	0.00	0.00	0.00	0.00
R10_W30_C20_K500_T14_l	-0.00	-0.00	-0.00	-0.00	-0.00
R10_W30_C20_K500_T14_t	-0.00	-0.00	-0.00	-0.00	-0.00
R10_W30_C20_K500_T7_l	0.25	0.25	0.25	0.25	0.25
R10_W30_C20_K500_T7_t	0.00	0.00	0.00	0.00	0.00

Table C.3: **Joint Optimization for Costs and Emissions:** Relative improvement on the objective function compared to the direct delivery solution obtained when using the different MIP formulation with CPLEX 12.10. and 1h computing time.

	MIP1	MIP2	MIP3	MIP4	MIP5
R2_W2_C2_K100_T14_l	-0.00	-0.00	-0.00	-0.00	-0.00
R2_W2_C2_K100_T14_t	-0.00	-0.00	-0.00	-0.00	-0.00
R2_W2_C2_K100_T30_l	-0.00	-0.00	-0.00	-0.00	-0.00
R2_W2_C2_K100_T30_t	-0.00	-0.00	-0.00	-0.00	-0.00

Continued on next page

Table C.3: **Joint Optimization for Costs and Emissions:** Relative improvement on the objective function compared to the direct delivery solution obtained when using the different MIP formulation with CPLEX 12.10. and 1h computing time.

	MIP1	MIP2	MIP3	MIP4	MIP5
R2_W2_C2_K100_T7_l	-0.00	-0.00	-0.00	-0.00	-0.00
R2_W2_C2_K100_T7_t	-0.00	-0.00	-0.00	0.01	0.02
R2_W2_C2_K20_T14_l	0.12	0.14	0.13	0.14	0.15
R2_W2_C2_K20_T14_t	0.12	0.14	0.14	0.14	0.15
R2_W2_C2_K20_T30_l	0.07	0.16	0.15	0.17	0.17
R2_W2_C2_K20_T30_t	-0.00	0.17	0.18	0.17	0.18
R2_W2_C2_K20_T7_l	0.07	0.07	0.07	0.07	0.07
R2_W2_C2_K20_T7_t	0.07	0.07	0.07	0.07	0.07
R2_W8_C8_K100_T14_l	-0.00	-0.00	-0.00	-0.00	-0.00
R2_W8_C8_K100_T14_t	-0.00	-0.00	-0.00	-0.00	-0.00
R2_W8_C8_K100_T30_l	-0.00	-0.00	-0.00	-0.00	-0.00
R2_W8_C8_K100_T30_t	-0.00	-0.00	-0.00	-0.00	-0.00
R2_W8_C8_K100_T7_l	-0.00	0.04	0.03	-0.00	-0.00
R2_W8_C8_K100_T7_t	0.00	0.03	-0.00	-0.00	-0.00
R2_W8_C8_K500_T14_l	-0.00	-0.00	NaN	-0.00	-0.00
R2_W8_C8_K500_T14_t	-0.00	-0.00	NaN	-0.00	-0.00
R2_W8_C8_K500_T7_l	-0.00	-0.00	-0.00	-0.00	-0.00
R2_W8_C8_K500_T7_t	-0.00	-0.00	-0.00	-0.00	-0.00
R5_W10_C10_K100_T14_l	-0.00	-0.00	-0.00	-0.00	-0.00
R5_W10_C10_K100_T14_t	-0.00	-0.00	-0.00	-0.00	-0.00
R5_W10_C10_K100_T7_l	-0.00	-0.00	-0.00	-0.00	-0.00
R5_W10_C10_K100_T7_t	-0.00	-0.00	-0.00	-0.00	-0.00
R5_W10_C10_K500_T7_l	-0.00	NaN	NaN	NaN	NaN
R5_W10_C10_K500_T7_t	-0.00	NaN	NaN	NaN	NaN
R5_W20_C15_K100_T14_l	0.00	NaN	NaN	NaN	NaN
R5_W20_C15_K100_T7_l	-0.00	-0.00	-0.00	-0.00	-0.00
R5_W20_C15_K100_T7_t	-0.00	-0.00	-0.00	-0.00	-0.00
R10_W30_C20_K100_T7_l	-0.00	NaN	NaN	NaN	NaN

Table C.4: **Optimize for Costs:** Relative improvement on the objective function compared to the direct delivery solution obtained when applying slope scaling and stopping after the first iteration or until it converged using the monotonic or cost-matching update scheme.

	1 Iteration	Monotonic	Cost-Matching
R2_W2_C2_K100_T14_l	-0.02	-0.00	-0.02
R2_W2_C2_K100_T14_t	-0.01	0.01	-0.01
R2_W2_C2_K100_T30_l	0.00	0.01	0.00
R2_W2_C2_K100_T30_t	-0.01	0.01	-0.01
R2_W2_C2_K100_T7_l	-0.04	-0.00	-0.02
R2_W2_C2_K100_T7_t	-0.03	-0.00	-0.02
R2_W2_C2_K20_T14_l	-0.06	-0.03	-0.05
R2_W2_C2_K20_T14_t	-0.04	-0.03	-0.01
R2_W2_C2_K20_T30_l	-0.00	0.03	0.01
R2_W2_C2_K20_T30_t	0.04	0.04	0.04
R2_W2_C2_K20_T7_l	-0.11	-0.05	-0.08
R2_W2_C2_K20_T7_t	-0.08	-0.04	-0.08
R2_W8_C8_K100_T14_l	0.03	0.03	0.03
R2_W8_C8_K100_T14_t	0.03	0.03	0.03
R2_W8_C8_K100_T30_l	0.10	0.10	0.10
R2_W8_C8_K100_T30_t	0.08	0.08	0.08
R2_W8_C8_K100_T7_l	-0.01	-0.00	-0.01
R2_W8_C8_K100_T7_t	-0.00	-0.00	-0.00
R2_W8_C8_K500_T14_l	-0.02	-0.00	-0.01
R2_W8_C8_K500_T14_t	-0.02	-0.00	-0.01
R2_W8_C8_K500_T30_l	-0.01	0.02	0.01
R2_W8_C8_K500_T30_t	-0.01	0.02	0.00
R2_W8_C8_K500_T7_l	-0.01	0.00	-0.00
R2_W8_C8_K500_T7_t	-0.01	0.00	-0.00
R5_W10_C10_K100_T14_l	0.06	0.15	0.12
R5_W10_C10_K100_T14_t	0.08	0.14	0.11
R5_W10_C10_K100_T30_l	0.14	0.21	0.18
R5_W10_C10_K100_T30_t	0.14	0.20	0.14
R5_W10_C10_K100_T7_l	-0.02	0.07	0.05
R5_W10_C10_K100_T7_t	-0.01	0.06	0.04
R5_W10_C10_K500_T14_l	0.05	0.10	0.08

Continued on next page

Table C.4: **Optimize for Costs:** Relative improvement on the objective function compared to the direct delivery solution obtained when applying slope scaling and stopping after the first iteration or until it converged using the monotonic or cost-matching update scheme.

	1 Iteration	Monotonic	Cost-Matching
R5_W10_C10_K500_T14_t	0.05	0.09	0.07
R5_W10_C10_K500_T30_l	0.10	0.14	0.12
R5_W10_C10_K500_T30_t	0.10	0.14	0.10
R5_W10_C10_K500_T7_l	0.01	0.04	0.05
R5_W10_C10_K500_T7_t	0.01	0.03	0.03
R5_W20_C15_K100_T14_l	0.06	0.15	0.11
R5_W20_C15_K100_T14_t	0.07	0.14	0.11
R5_W20_C15_K100_T30_l	0.13	0.23	0.19
R5_W20_C15_K100_T30_t	0.15	0.19	0.18
R5_W20_C15_K100_T7_l	-0.03	0.07	0.05
R5_W20_C15_K100_T7_t	-0.02	0.07	0.05
R5_W20_C15_K500_T14_l	0.06	0.13	0.12
R5_W20_C15_K500_T14_t	0.07	0.13	0.11
R5_W20_C15_K500_T30_l	0.12	0.18	0.12
R5_W20_C15_K500_T30_t	0.12	0.17	0.12
R5_W20_C15_K500_T7_l	0.01	0.07	0.06
R5_W20_C15_K500_T7_t	0.01	0.06	0.06
R10_W30_C20_K100_T14_l	0.07	0.20	0.19
R10_W30_C20_K100_T14_t	0.10	0.19	0.19
R10_W30_C20_K100_T30_l	0.15	0.26	0.18
R10_W30_C20_K100_T30_t	0.18	0.18	0.18
R10_W30_C20_K100_T7_l	0.03	0.16	0.14
R10_W30_C20_K100_T7_t	0.06	0.16	0.14
R10_W30_C20_K500_T14_l	0.07	0.17	0.14
R10_W30_C20_K500_T14_t	0.08	0.14	0.08
R10_W30_C20_K500_T7_l	0.03	0.10	0.09
R10_W30_C20_K500_T7_t	0.03	0.09	0.08

Table C.5: **Optimize for Emissions:** Relative improvement on the objective function compared to the direct delivery solution obtained when applying slope scaling and stopping after the first iteration or until it converged using the monotonic or cost-matching update scheme.

	1 Iteration	Monotonic	Cost-Matching
R2_W2_C2_K100_T14_l	0.39	0.39	0.39
R2_W2_C2_K100_T14_t	0.30	0.30	0.30
R2_W2_C2_K100_T30_l	0.48	0.48	0.48
R2_W2_C2_K100_T30_t	0.36	0.36	0.36
R2_W2_C2_K100_T7_l	0.23	0.23	0.23
R2_W2_C2_K100_T7_t	0.17	0.18	0.17
R2_W2_C2_K20_T14_l	0.27	0.27	0.27
R2_W2_C2_K20_T14_t	0.23	0.24	0.23
R2_W2_C2_K20_T30_l	0.37	0.37	0.37
R2_W2_C2_K20_T30_t	0.32	0.33	0.34
R2_W2_C2_K20_T7_l	0.16	0.16	0.16
R2_W2_C2_K20_T7_t	0.13	0.13	0.13
R2_W8_C8_K100_T14_l	0.35	0.35	0.35
R2_W8_C8_K100_T14_t	0.26	0.27	0.27
R2_W8_C8_K100_T30_l	0.47	0.47	0.47
R2_W8_C8_K100_T30_t	0.35	0.36	0.36
R2_W8_C8_K100_T7_l	0.22	0.22	0.22
R2_W8_C8_K100_T7_t	0.15	0.16	0.15
R2_W8_C8_K500_T14_l	0.35	0.35	0.35
R2_W8_C8_K500_T14_t	0.30	0.30	0.30
R2_W8_C8_K500_T30_l	0.44	0.44	0.44
R2_W8_C8_K500_T30_t	0.37	0.37	0.37
R2_W8_C8_K500_T7_l	0.22	0.22	0.22
R2_W8_C8_K500_T7_t	0.18	0.18	0.18
R5_W10_C10_K100_T14_l	0.37	0.38	0.38
R5_W10_C10_K100_T14_t	0.24	0.26	0.25
R5_W10_C10_K100_T30_l	0.50	0.51	0.51
R5_W10_C10_K100_T30_t	0.33	0.34	0.33
R5_W10_C10_K100_T7_l	0.17	0.19	0.18
R5_W10_C10_K100_T7_t	0.12	0.14	0.12

Continued on next page

Table C.5: **Optimize for Emissions:** Relative improvement on the objective function compared to the direct delivery solution obtained when applying slope scaling and stopping after the first iteration or until it converged using the monotonic or cost-matching update scheme.

	1 Iteration	Monotonic	Cost-Matching
R5_W10_C10_K500_T14_l	0.33	0.33	0.33
R5_W10_C10_K500_T14_t	0.27	0.27	0.27
R5_W10_C10_K500_T30_l	0.44	0.45	0.45
R5_W10_C10_K500_T30_t	0.30	0.30	0.30
R5_W10_C10_K500_T7_l	0.20	0.20	0.20
R5_W10_C10_K500_T7_t	0.15	0.15	0.15
R5_W20_C15_K100_T14_l	0.36	0.38	0.37
R5_W20_C15_K100_T14_t	0.25	0.26	0.27
R5_W20_C15_K100_T30_l	0.49	0.51	0.50
R5_W20_C15_K100_T30_t	0.33	0.34	0.33
R5_W20_C15_K100_T7_l	0.17	0.19	0.19
R5_W20_C15_K100_T7_t	0.12	0.15	0.15
R5_W20_C15_K500_T14_l	0.33	0.35	0.34
R5_W20_C15_K500_T14_t	0.26	0.27	0.26
R5_W20_C15_K500_T30_l	0.45	0.46	0.45
R5_W20_C15_K500_T30_t	0.27	0.27	0.27
R5_W20_C15_K500_T7_l	0.19	0.20	0.20
R5_W20_C15_K500_T7_t	0.14	0.15	0.14
R10_W30_C20_K100_T14_l	0.35	0.37	0.37
R10_W30_C20_K100_T14_t	0.25	0.25	0.25
R10_W30_C20_K100_T30_l	0.48	0.49	0.48
R10_W30_C20_K100_T30_t	0.31	0.31	0.31
R10_W30_C20_K100_T7_l	0.24	0.27	0.27
R10_W30_C20_K100_T7_t	0.16	0.19	0.20
R10_W30_C20_K500_T14_l	0.35	0.37	0.36
R10_W30_C20_K500_T14_t	0.21	0.21	0.21
R10_W30_C20_K500_T7_l	0.22	0.23	0.23
R10_W30_C20_K500_T7_t	0.15	0.16	0.15

Table C.6: **Joint Optimization for Costs and Emissions:** Relative improvement on the objective function compared to the direct delivery solution obtained when applying slope scaling and stopping after the first iteration or until it converged using the monotonic or cost-matching update scheme.

	1 Iteration	Monotonic	Cost-Matching
R2_W2_C2_K100_T14_l	0.01	0.01	0.01
R2_W2_C2_K100_T14_t	0.01	0.01	0.01
R2_W2_C2_K100_T30_l	0.03	0.04	0.03
R2_W2_C2_K100_T30_t	0.02	0.02	0.02
R2_W2_C2_K100_T7_l	-0.03	0.00	-0.02
R2_W2_C2_K100_T7_t	-0.03	0.00	-0.02
R2_W2_C2_K20_T14_l	-0.07	-0.01	-0.02
R2_W2_C2_K20_T14_t	-0.05	-0.01	-0.00
R2_W2_C2_K20_T30_l	-0.01	0.03	0.03
R2_W2_C2_K20_T30_t	0.01	0.04	0.05
R2_W2_C2_K20_T7_l	-0.11	-0.06	-0.08
R2_W2_C2_K20_T7_t	-0.11	-0.04	-0.05
R2_W8_C8_K100_T14_l	0.02	0.03	0.02
R2_W8_C8_K100_T14_t	0.03	0.04	0.03
R2_W8_C8_K100_T30_l	0.12	0.12	0.12
R2_W8_C8_K100_T30_t	0.11	0.12	0.11
R2_W8_C8_K100_T7_l	0.00	0.01	0.00
R2_W8_C8_K100_T7_t	0.01	0.01	0.01
R2_W8_C8_K500_T14_l	0.01	0.02	0.02
R2_W8_C8_K500_T14_t	0.01	0.02	0.02
R2_W8_C8_K500_T30_l	0.03	0.05	0.04
R2_W8_C8_K500_T30_t	0.03	0.04	0.04
R2_W8_C8_K500_T7_l	0.00	0.00	0.00
R2_W8_C8_K500_T7_t	0.00	0.00	0.01
R5_W10_C10_K100_T14_l	0.12	0.18	0.15
R5_W10_C10_K100_T14_t	0.10	0.15	0.11
R5_W10_C10_K100_T30_l	0.20	0.25	0.23
R5_W10_C10_K100_T30_t	0.18	0.21	0.18
R5_W10_C10_K100_T7_l	0.00	0.08	0.05
R5_W10_C10_K100_T7_t	0.03	0.07	0.04

Continued on next page

Table C.6: **Joint Optimization for Costs and Emissions:** Relative improvement on the objective function compared to the direct delivery solution obtained when applying slope scaling and stopping after the first iteration or until it converged using the monotonic or cost-matching update scheme.

	1 Iteration	Monotonic	Cost-Matching
R5_W10_C10_K500_T14_l	0.10	0.14	0.12
R5_W10_C10_K500_T14_t	0.09	0.13	0.09
R5_W10_C10_K500_T30_l	0.16	0.19	0.17
R5_W10_C10_K500_T30_t	0.15	0.17	0.15
R5_W10_C10_K500_T7_l	0.05	0.07	0.07
R5_W10_C10_K500_T7_t	0.04	0.05	0.05
R5_W20_C15_K100_T14_l	0.11	0.17	0.13
R5_W20_C15_K100_T14_t	0.11	0.16	0.12
R5_W20_C15_K100_T30_l	0.20	0.27	0.21
R5_W20_C15_K100_T30_t	0.20	0.21	0.20
R5_W20_C15_K100_T7_l	-0.01	0.09	0.07
R5_W20_C15_K100_T7_t	0.01	0.08	0.05
R5_W20_C15_K500_T14_l	0.12	0.16	0.15
R5_W20_C15_K500_T14_t	0.10	0.15	0.11
R5_W20_C15_K500_T30_l	0.19	0.22	0.19
R5_W20_C15_K500_T30_t	0.16	0.20	0.16
R5_W20_C15_K500_T7_l	0.05	0.08	0.08
R5_W20_C15_K500_T7_t	0.04	0.07	0.06
R10_W30_C20_K100_T14_l	0.12	0.22	0.20
R10_W30_C20_K100_T14_t	0.13	0.19	0.17
R10_W30_C20_K100_T30_l	0.21	0.28	0.23
R10_W30_C20_K100_T30_t	0.22	0.22	0.22
R10_W30_C20_K100_T7_l	0.06	0.17	0.14
R10_W30_C20_K100_T7_t	0.07	0.15	0.14
R10_W30_C20_K500_T14_l	0.13	0.19	0.17
R10_W30_C20_K500_T14_t	0.12	0.15	0.12
R10_W30_C20_K500_T7_l	0.06	0.12	0.10
R10_W30_C20_K500_T7_t	0.06	0.10	0.09

C.3 LOCAL SEARCH

For brevity only result tables useful for the discussions in Chapter 5 are given.

Costs and Emissions of Local Search Results

Table C.7: **Optimize for Costs:** Relative improvement on costs and emissions compared to the direct delivery solution obtained when applying the most successful heuristic (local search variant LS2)

	Costs [%]	Std [%]	Emissions [%]	Std [%]
R2_W2_C2_K100_T14_l	2.6	0.1	10.3	0.3
R2_W2_C2_K100_T14_t	1.9	0.2	7.8	0.4
R2_W2_C2_K100_T30_l	3.3	0.1	14.7	0.7
R2_W2_C2_K100_T30_t	2.0	0.2	10.2	0.4
R2_W2_C2_K100_T7_l	0.5	0.2	1.5	0.9
R2_W2_C2_K100_T7_t	0.4	0.2	1.3	1.1
R2_W2_C2_K20_T14_l	8.0	0.5	7.1	0.2
R2_W2_C2_K20_T14_t	7.3	0.6	7.7	0.4
R2_W2_C2_K20_T30_l	12.6	0.4	12.3	0.4
R2_W2_C2_K20_T30_t	12.3	0.3	12.7	0.2
R2_W2_C2_K20_T7_l	1.2	0.0	2.7	0.0
R2_W2_C2_K20_T7_t	1.3	0.0	4.3	0.3
R2_W8_C8_K100_T14_l	11.2	0.3	11.3	0.3
R2_W8_C8_K100_T14_t	11.1	0.4	11.3	0.4
R2_W8_C8_K100_T30_l	16.7	0.2	18.7	0.1
R2_W8_C8_K100_T30_t	15.8	0.2	17.8	0.3
R2_W8_C8_K100_T7_l	6.5	0.5	6.3	0.3
R2_W8_C8_K100_T7_t	6.6	0.3	6.4	0.3
R2_W8_C8_K500_T14_l	3.5	0.2	6.8	0.5
R2_W8_C8_K500_T14_t	3.6	0.2	6.8	0.6
R2_W8_C8_K500_T30_l	4.9	0.1	11.2	0.3
R2_W8_C8_K500_T30_t	4.8	0.1	10.6	0.3
R2_W8_C8_K500_T7_l	2.4	0.1	3.2	0.5
R2_W8_C8_K500_T7_t	2.4	0.1	3.3	0.3
R5_W10_C10_K100_T14_l	25.1	0.3	28.1	0.4

Continued on next page

Table C.7: **Optimize for Costs:** Relative improvement on costs and emissions compared to the direct delivery solution obtained when applying the most successful heuristic (local search variant LS2)

	Costs [%]	Std [%]	Emissions [%]	Std [%]
R5_W10_C10_K100_T14_t	24.0	0.3	24.9	0.5
R5_W10_C10_K100_T30_l	29.2	0.1	37.2	0.4
R5_W10_C10_K100_T30_t	25.4	0.3	32.7	0.3
R5_W10_C10_K100_T7_l	19.5	0.2	14.2	0.5
R5_W10_C10_K100_T7_t	19.4	0.4	13.6	0.6
R5_W10_C10_K500_T14_l	17.2	0.2	23.8	0.4
R5_W10_C10_K500_T14_t	16.6	0.1	22.9	0.3
R5_W10_C10_K500_T30_l	19.2	0.2	31.2	0.2
R5_W10_C10_K500_T30_t	16.7	0.1	30.4	0.3
R5_W10_C10_K500_T7_l	11.1	0.1	12.9	0.4
R5_W10_C10_K500_T7_t	10.7	0.1	11.1	0.3
R5_W20_C15_K100_T14_l	26.0	0.3	27.3	0.5
R5_W20_C15_K100_T14_t	25.4	0.3	25.0	0.2
R5_W20_C15_K100_T30_l	30.1	0.3	35.0	0.3
R5_W20_C15_K100_T30_t	26.5	0.3	31.1	0.2
R5_W20_C15_K100_T7_l	19.4	0.4	15.3	0.5
R5_W20_C15_K100_T7_t	19.4	0.3	14.8	0.3
R5_W20_C15_K500_T14_l	20.2	0.2	24.5	0.4
R5_W20_C15_K500_T14_t	19.2	0.2	22.8	0.3
R5_W20_C15_K500_T30_l	20.4	0.3	32.0	0.1
R5_W20_C15_K500_T30_t	17.4	0.2	31.0	0.2
R5_W20_C15_K500_T7_l	14.9	0.1	12.5	0.2
R5_W20_C15_K500_T7_t	14.4	0.1	11.2	0.4
R10_W30_C20_K100_T14_l	30.4	0.2	29.4	0.2
R10_W30_C20_K100_T14_t	28.1	0.3	25.4	0.3
R10_W30_C20_K100_T30_l	30.8	0.3	37.7	0.2
R10_W30_C20_K100_T30_t	23.2	0.3	30.8	0.2
R10_W30_C20_K100_T7_l	27.6	0.4	22.2	0.3
R10_W30_C20_K100_T7_t	27.3	0.4	20.3	0.4
R10_W30_C20_K500_T14_l	21.6	0.2	26.1	0.2
R10_W30_C20_K500_T14_t	16.7	0.2	22.8	0.2
R10_W30_C20_K500_T7_l	17.2	0.1	15.5	0.1

Continued on next page

Table C.7: **Optimize for Costs:** Relative improvement on costs and emissions compared to the direct delivery solution obtained when applying the most successful heuristic (local search variant LS2)

	Costs [%]	Std [%]	Emissions [%]	Std [%]
R10_W30_C20_K500_T7_t	16.3	0.2	13.0	0.3

Table C.8: **Optimize for Emissions:** Relative improvement on costs and emissions compared to the direct delivery solution obtained when applying the most successful heuristic (local search variant LS2).

	Costs [%]	Std [%]	Emissions [%]	Std [%]
R2_W2_C2_K100_T14_l	-10.7	0.0	38.9	0.0
R2_W2_C2_K100_T14_t	-11.3	0.3	32.1	0.1
R2_W2_C2_K100_T30_l	-10.0	0.1	48.6	0.0
R2_W2_C2_K100_T30_t	-12.1	0.2	38.2	0.2
R2_W2_C2_K100_T7_l	-14.4	0.0	23.4	0.0
R2_W2_C2_K100_T7_t	-12.2	0.6	18.9	0.2
R2_W2_C2_K20_T14_l	-7.1	0.0	28.1	0.0
R2_W2_C2_K20_T14_t	-2.0	0.5	26.2	0.1
R2_W2_C2_K20_T30_l	-1.5	0.4	38.4	0.0
R2_W2_C2_K20_T30_t	1.4	0.2	36.1	0.3
R2_W2_C2_K20_T7_l	-16.6	0.0	16.5	0.0
R2_W2_C2_K20_T7_t	-11.7	0.0	13.8	0.0
R2_W8_C8_K100_T14_l	-3.7	0.1	35.0	0.0
R2_W8_C8_K100_T14_t	-1.0	0.3	28.4	0.1
R2_W8_C8_K100_T30_l	3.9	0.1	46.9	0.0
R2_W8_C8_K100_T30_t	5.9	0.1	38.5	0.1
R2_W8_C8_K100_T7_l	-6.4	0.0	22.5	0.0
R2_W8_C8_K100_T7_t	-3.2	0.4	16.4	0.1
R2_W8_C8_K500_T14_l	-10.8	0.0	35.2	0.0
R2_W8_C8_K500_T14_t	-9.4	0.2	30.9	0.1
R2_W8_C8_K500_T30_l	-10.2	0.0	44.5	0.0
R2_W8_C8_K500_T30_t	-8.4	0.1	38.1	0.0
R2_W8_C8_K500_T7_l	-10.5	0.0	22.4	0.0
R2_W8_C8_K500_T7_t	-7.4	0.2	18.8	0.1
R5_W10_C10_K100_T14_l	20.3	0.2	40.9	0.1

Continued on next page

Table C.8: **Optimize for Emissions:** Relative improvement on costs and emissions compared to the direct delivery solution obtained when applying the most successful heuristic (local search variant LS2).

	Costs [%]	Std [%]	Emissions [%]	Std [%]
R5_W10_C10_K100_T14_t	19.6	0.2	31.1	0.1
R5_W10_C10_K100_T30_l	25.7	0.1	53.0	0.1
R5_W10_C10_K100_T30_t	22.3	0.3	38.6	0.2
R5_W10_C10_K100_T7_l	12.2	0.2	22.2	0.1
R5_W10_C10_K100_T7_t	13.3	0.3	18.2	0.2
R5_W10_C10_K500_T14_l	11.5	0.1	34.9	0.0
R5_W10_C10_K500_T14_t	9.8	0.2	30.1	0.1
R5_W10_C10_K500_T30_l	14.8	0.1	45.9	0.0
R5_W10_C10_K500_T30_t	-3.0	0.1	29.9	0.1
R5_W10_C10_K500_T7_l	5.3	0.1	21.6	0.0
R5_W10_C10_K500_T7_t	4.0	0.2	16.7	0.1
R5_W20_C15_K100_T14_l	18.2	0.1	39.8	0.0
R5_W20_C15_K100_T14_t	20.4	0.3	31.8	0.1
R5_W20_C15_K100_T30_l	25.0	0.1	51.7	0.1
R5_W20_C15_K100_T30_t	22.2	0.3	37.8	0.2
R5_W20_C15_K100_T7_l	12.5	0.2	23.3	0.1
R5_W20_C15_K100_T7_t	13.6	0.3	19.0	0.2
R5_W20_C15_K500_T14_l	14.5	0.1	36.2	0.0
R5_W20_C15_K500_T14_t	12.6	0.1	29.7	0.1
R5_W20_C15_K500_T30_l	16.8	0.2	46.4	0.1
R5_W20_C15_K500_T30_t	-2.9	0.2	27.5	0.1
R5_W20_C15_K500_T7_l	7.5	0.1	21.7	0.0
R5_W20_C15_K500_T7_t	6.7	0.2	17.2	0.0
R10_W30_C20_K100_T14_l	23.7	0.2	39.7	0.1
R10_W30_C20_K100_T14_t	24.2	0.2	31.9	0.1
R10_W30_C20_K100_T30_l	28.1	0.4	50.6	0.2
R10_W30_C20_K100_T30_t	18.0	0.2	31.3	0.1
R10_W30_C20_K100_T7_l	21.2	0.2	29.9	0.1
R10_W30_C20_K100_T7_t	22.1	0.3	23.5	0.2
R10_W30_C20_K500_T14_l	16.8	0.1	38.0	0.0
R10_W30_C20_K500_T14_t	-1.2	0.2	21.0	0.1
R10_W30_C20_K500_T7_l	10.3	0.1	24.8	0.0

Continued on next page

Table C.8: **Optimize for Emissions:** Relative improvement on costs and emissions compared to the direct delivery solution obtained when applying the most successful heuristic (local search variant LS2).

	Costs [%]	Std [%]	Emissions [%]	Std [%]
R10_W30_C20_K500_T7_t	9.9	0.2	18.7	0.1

Table C.9: **Joint Optimization for Costs and Emissions:** Relative improvement on costs and emissions compared to the direct delivery solution obtained when applying the most successful heuristic (local search variant LS2)

	Costs [%]	Std [%]	Emissions [%]	Std [%]
R2_W2_C2_K100_T14_l	-0.2	0.4	23.9	1.1
R2_W2_C2_K100_T14_t	-0.5	0.4	23.4	1.2
R2_W2_C2_K100_T30_l	1.4	0.3	29.3	0.7
R2_W2_C2_K100_T30_t	0.2	0.3	27.9	0.6
R2_W2_C2_K100_T7_l	-1.4	0.2	9.8	0.3
R2_W2_C2_K100_T7_t	-1.6	0.5	10.3	1.3
R2_W2_C2_K20_T14_l	7.2	0.5	22.1	0.4
R2_W2_C2_K20_T14_t	7.1	0.6	22.1	0.4
R2_W2_C2_K20_T30_l	10.9	0.5	29.8	0.3
R2_W2_C2_K20_T30_t	10.9	0.5	29.5	0.6
R2_W2_C2_K20_T7_l	1.7	0.6	4.7	1.3
R2_W2_C2_K20_T7_t	1.7	0.5	5.1	1.6
R2_W8_C8_K100_T14_l	11.2	0.2	22.2	0.3
R2_W8_C8_K100_T14_t	11.0	0.3	21.9	0.4
R2_W8_C8_K100_T30_l	17.3	0.2	30.2	0.2
R2_W8_C8_K100_T30_t	16.3	0.1	31.1	0.2
R2_W8_C8_K100_T7_l	6.4	0.3	12.5	0.4
R2_W8_C8_K100_T7_t	6.2	0.3	11.8	0.2
R2_W8_C8_K500_T14_l	0.8	0.2	20.9	0.5
R2_W8_C8_K500_T14_t	0.8	0.3	20.7	0.7
R2_W8_C8_K500_T30_l	2.6	0.1	25.4	0.3
R2_W8_C8_K500_T30_t	2.2	0.2	25.9	0.3
R2_W8_C8_K500_T7_l	0.9	0.4	10.4	0.9
R2_W8_C8_K500_T7_t	1.2	0.2	9.1	0.6

Continued on next page

Table C.9: **Joint Optimization for Costs and Emissions:** Relative improvement on costs and emissions compared to the direct delivery solution obtained when applying the most successful heuristic (local search variant LS2)

	Costs [%]	Std [%]	Emissions [%]	Std [%]
R5_W10_C10_K100_T14_l	25.2	0.2	35.0	0.4
R5_W10_C10_K100_T14_t	23.3	0.3	27.8	0.3
R5_W10_C10_K100_T30_l	29.7	0.1	45.9	0.2
R5_W10_C10_K100_T30_t	25.2	0.2	35.4	0.2
R5_W10_C10_K100_T7_l	19.5	0.4	17.6	0.4
R5_W10_C10_K100_T7_t	18.9	0.6	15.3	0.3
R5_W10_C10_K500_T14_l	16.9	0.1	30.4	0.2
R5_W10_C10_K500_T14_t	15.8	0.1	27.4	0.2
R5_W10_C10_K500_T30_l	18.9	0.2	40.4	0.2
R5_W10_C10_K500_T30_t	15.1	0.1	35.3	0.1
R5_W10_C10_K500_T7_l	10.8	0.1	17.5	0.1
R5_W10_C10_K500_T7_t	10.2	0.2	14.3	0.4
R5_W20_C15_K100_T14_l	26.4	0.4	32.6	0.6
R5_W20_C15_K100_T14_t	25.8	0.5	28.0	0.4
R5_W20_C15_K100_T30_l	30.7	0.2	43.0	0.3
R5_W20_C15_K100_T30_t	26.9	0.4	34.3	0.3
R5_W20_C15_K100_T7_l	18.8	0.4	17.7	0.5
R5_W20_C15_K100_T7_t	18.3	0.5	15.4	0.3
R5_W20_C15_K500_T14_l	19.9	0.1	31.4	0.2
R5_W20_C15_K500_T14_t	18.6	0.1	26.9	0.2
R5_W20_C15_K500_T30_l	20.1	0.2	40.6	0.1
R5_W20_C15_K500_T30_t	16.0	0.4	34.5	0.2
R5_W20_C15_K500_T7_l	14.2	0.2	17.5	0.2
R5_W20_C15_K500_T7_t	14.0	0.2	14.2	0.2
R10_W30_C20_K100_T14_l	30.4	0.2	33.0	0.2
R10_W30_C20_K100_T14_t	28.1	0.2	27.9	0.2
R10_W30_C20_K100_T30_l	31.3	0.3	43.7	0.2
R10_W30_C20_K100_T30_t	23.9	0.4	32.6	0.2
R10_W30_C20_K100_T7_l	27.4	0.5	24.8	0.5
R10_W30_C20_K100_T7_t	26.8	0.4	21.4	0.3
R10_W30_C20_K500_T14_l	22.0	0.2	31.9	0.2

Continued on next page

Table C.9: **Joint Optimization for Costs and Emissions:** Relative improvement on costs and emissions compared to the direct delivery solution obtained when applying the most successful heuristic (local search variant LS2)

	Costs [%]	Std [%]	Emissions [%]	Std [%]
R10_W30_C20_K500_T14_t	16.4	0.2	25.7	0.2
R10_W30_C20_K500_T7_l	16.8	0.2	20.4	0.1
R10_W30_C20_K500_T7_t	15.6	0.1	16.2	0.1