



Università degli Studi di Cagliari

Dipartimento di Matematica e Informatica
Dottorato di Ricerca in Matematica e Informatica
Ciclo XXIX

Ph.D. Thesis

Algebraic structural analysis of a vehicle routing problem of heterogeneous trucks. Identification of the properties allowing an exact approach.

S.S.D. MAT/09

Candidate
Silvia Schirra

Supervisor
Prof. A.Loi

PhD Coordinator
Prof. G.Rodriguez

Final examination academic year 2015/2016, April 20, 2017

Abstract

Although integer linear programming problems are typically difficult to solve, there exist some easier problems, where the linear programming relaxation is integer. This thesis sheds light on a drayage problem which is supposed to have this nice feature, after extensive computational experiments. This thesis aims to provide a theoretical understanding of these results by the analysis of the algebraic structures of the mathematical formulation. Three reformulations are presented to prove if the constraint matrix is totally unimodular. We will show which experimental conditions are necessary and sufficient (or only sufficient or only necessary) for total unimodularity.

Declaration

I declare that to the best of my knowledge the contents of this thesis are original and my work except where indicated otherwise.

Aknowledgments

First of all, I would like to thank my Supervisor Andrea Loi, jointly Paola Zuddas and Massimo Di Francesco for giving me the opportunity to start this project.
I express a sincere gratitude to Claudio Gentile and Giuseppe Stecca for the possibility to collaborate with them at IASI - CNR in Rome.
I would like to thank the referees Antonio Frangioni and Teodor Crainic for their suggestion and remarks, which helped me to improve this work.
I would also like to thank all the members of the Department of Mathematics of University of Cagliari and in particular my collegues Silvia and Gianfranco.
Finally, I would specially thank my parents, my brother, my sister and my boyfriend Andrea for having always been close to me and also supported me in the most difficult times.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Integer linear programming (ILP) problems concern the maximization or minimization of a linear function of variables, which are required to obey inequality and equality linear constraints and integrality restrictions on the variables. A large number of real problems can be represented by integer linear programs, such as facility location, network design, freight distribution, lot sizing and Vehicle Routing Problems. Unfortunately, no general polynomial algorithm has been discovered for these problems. Most of them are $\mathscr{NP}$-complete and, thus, there is a little hope of find efficient exact procedures for them. Solving these ILP-problems is generally very time-consuming and even finding the first feasible solution may be a difficult task.

However, there is a desired property of constraint matrices that makes these problems much easier to solve: the total unimodularity (TU). In fact, when the constraint matrix is TU, linear program relaxations admit integral optimal solutions. As a consequence, one can solve integer problem as a linear problem, by polynomial algorithms embedded in linear programming .

This thesis is motivated by a conjecture on the total unimodularity of the constraint matrix in a formulation of a drayage problem, proposed by [24]. In fact, extensive computational experiments show that the linear relaxation is always integer, when all vehicles are supposed to carry one container and have the same costs. This thesis aims to shed light on the theoretical aspects that may explain these experimental results by the analysis of the algebraic properties of this mathematical model. We reformulate this model to determine necessary and sufficient conditions for TU.

In Chapters 2 we briefly recall the basic concepts of integer and linear programming. Chapter 3 reviews Vehicles Routing Problem and Drayage problems, particularly.
The contributions of the thesis are clustered in Chapter 4, where the following questions are made:

- Is the assumption one-container pr truck necessary for TU?

- If so, is it also sufficient?
- Is the assumption of identical truck costs fundamental for TU?

This thesis aims to answer these questions.

Chapter 2

Integer Linear Programming and Continuous Relaxation

An integer programming problem consists of maximizing or minimizing a real function of many variables, subject to inequality and equality constraints and integrality restrictions on some or all of the variables. If the function that must be maximized or minimized and inequality and equality constraints are linear, the problem is called integer linear programming problem (ILP). A great number of real problems can be represented by integer and combinatorial optimization: facility location, transportation network design, distribution of goods, production scheduling and in general Vehicle Routing Problems, that will be analyzed in detail in Chapter 3.

2.1 Integer Linear Programming

We can write a general linear mixed-integer programming (MIP) problem as follows:

$$\max\{cx + hy : Ax + Gy \leq b, x \in \mathbb{Z}^n_+, y \in \mathbb{R}^p_+\}$$

where $x = (x_1, \ldots, x_n)$, $y = (y_1, \ldots, y_p)$ are the variables, c is a n-vector, h a p-vector, A an $m \times n$ matrix, G an $m \times p$ matrix and b an m-vector. An instance of the problem is a set of data (c, h, A, G, b). Because of the presence of both integer and continuous (real) variables, this problem is called mixed. Moreover, it can be observed that minimizing a function is equivalent to maximizing the negative of the same function and that an equality constraint can be represented by two inequalities.
The set $S = \{x \in \mathbb{Z}^n_+, y \in \mathbb{R}^p_+, Ax + Gy \leq b\}$ is called the feasible region, and an $(x, y) \in S$ is called a feasible solution. An instance is said to be feasible if $S \neq \emptyset$. The function $z = cx + hy$ is called the objective function. A feasible point (x_0, y_0) for which the objective function has the maximum value, that is, $cx_0 + hy_0 \geq cx + hy \quad \forall (x, y) \in S$, is called an optimal solution. If (x_0, y_0) is an optimal solution, $cx_0 + hy_0$ is called the optimal value or weight of the solution.

A feasible instance of MIP may not have an optimal solution. It can be said that an instance is unbounded if there is an $(x, y) \in S$ such that $cx + hy \geq \omega$, for any $\omega \in \mathbb{R}^1$. In this case we use the notation $z = \infty$. If one solves an instance of MIP it is possible to obtain an optimal solution or show that it is either unbounded or infeasible.

When there are no continuous variables we have a special case of MIP called linear (pure) integer programming problem (ILP):

$$\max\{cx : Ax \leq b, x \in \mathbb{Z}_+^n\}$$

On the other hand, when there are no integer variables we obtain a linear programming problem (LP)

$$\max\{hy : Gy \leq b, y \in \mathbb{R}_+^p\}$$

We have an other important frequent case, when the integer variables are used to represent logical relationships. Consequently they can only be equal to 0 or 1. Thus we obtain the 0-1 MIP (respectively 0 -1 IP) in which $x \in \mathbb{Z}_+^n$ is replaced by $x \in B^n$, where B^n is the set of n-dimensional binary vectors.
For example, we use of 0-1 variables to represent binary choice, if we have to choose between two possibilities, as an event that can or cannot occur. In the model of this problem it is introduced a binary variable x, that assume the value 1 if the event occurs, and 0 otherwise.

The study of theory and algorithms of linear programming is fundamental to understand integer programming. It is well known that solving an integer programming problem is much more difficult than a linear programming problem, since the theory and the computational aspects of integer programming are less developed than the ones of linear programming. For this reason the theory of linear programming represents a guide for developing results for integer programming. Moreover, linear programming algorithms are very often used as a subroutine in integer programming algorithms to obtain upper bounds on the value of the integer program. Let

$$z_{IP} = \max\{cx : Ax \leq b, x \in \mathbb{Z}_+^n\}$$

and note that $z_{LP} \geq z_{IP}$ since $\mathbb{Z}_+^n \subset \mathbb{R}_+^n$. The upper bound z_{LP} can be used to prove optimality for IP; that is, if x_0 is a feasible solution to IP and $cx_0 = z_{LP}$, then x_0 is an optimal solution to IP. (see [39] for an exhaustive description of the topic)

2.2 Continuous Relaxation

Let $z_{IP} = \max\{cx : Ax \leq b, x \in \mathbb{Z}_+^n\}$ be an integer problem. If we remove integrality restrictions we obtain the following problem:

$$z_{LP} = \max\{cx : Ax \leq b, x \in \mathbb{R}_+^n\},$$

called continuous relaxation of the original problem. Generally, the continuous relaxation problem is easier to solve and it has a lower execution time than the integer problem associated.

Let S and S^* be feasible regions of z_{IP} and z_{LP}, respectively. Then $S = S^* \cap \mathbb{Z}^n$. Consequently:

- S may be the empty set, though S^* is different to empty set,
- if S^* is bounded, then S is finite.

It follows that the optimal solution could be found calculating the value that the objective function f assumes in every point $(x, y) \in S$ and choosing the maximum value of f. Obviously, this method is allowed only if the cardinality of S is very small. One might remove integrality constraints and approximate the optimal solution of the continuous relaxation. However, this approach is useless for two reasons:

- the approximate solution may be infeasible;
- the approximate solution may be feasible, but very far from the optimal solution (when variables assume very small optimum values, for example if we have binary variables).

Both cases are shown in Fig. 2.1.

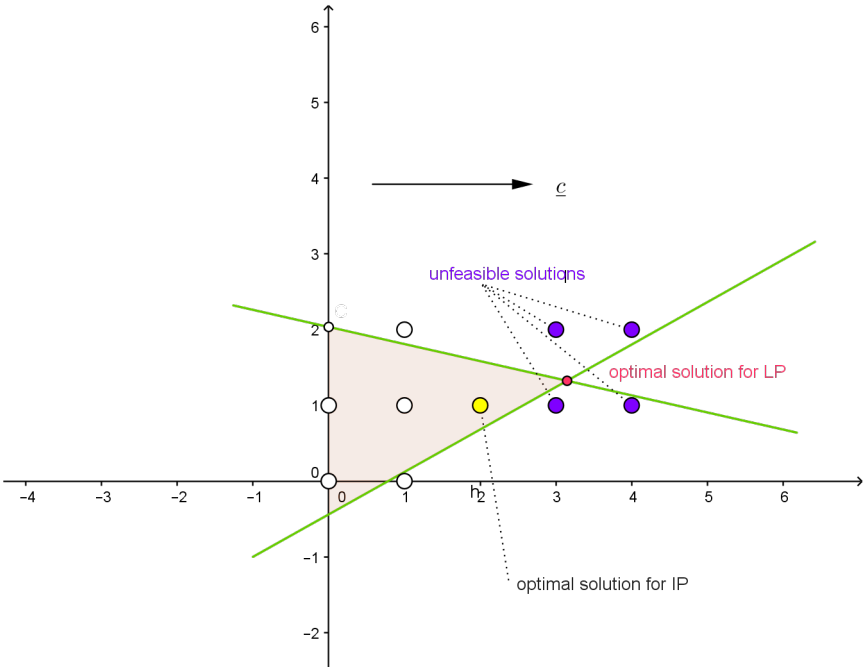

Figure 2.1: Continuous relaxation

Usually, we need an alternative approach using the continuous relaxation, that can be solved with the simplex method. In the paragraph 2.4 we analyze two of these methods: branch-and-bound and cutting-plane.

2.3 Dual Problem

Given a linear programming problem there is another linear programming problem associated, called the dual problem. The dual linear program possesses many important properties relative to the original (primal) linear program. There are two important definitions of duality: the canonical form of duality and the standard form of duality. These two forms are completely equivalent. They arise from the canonical and the standard representation of linear programming problems respectively. We state the primal problem as

$$z_{LP} = \max\{cx : Ax \leq b, x \in \mathbb{R}^n_+\}$$

Its dual is defined as the linear program

$$w_{LP} = \min\{ub : uA \geq c, u \in \mathbb{R}^m_+\}$$

Note that there is exactly one dual variable for each primal constraint and exactly one dual constraint for each primal variable. We can use the following table to compute the dual of a primal problem given.

MINIMIZATION PROBLEM		MAXIMIZATION PROBLEM
Variables ≥ 0	$\longleftrightarrow$	Constraints ≤ 0
Variables ≤ 0	$\longleftrightarrow$	Constraints ≥ 0
Variables Unrestricted	$\longleftrightarrow$	Constraints $= 0$
Constraints ≥ 0	$\longleftrightarrow$	Variables ≥ 0
Constraints ≤ 0	$\longleftrightarrow$	Variables ≤ 0
Constraints $= 0$	$\longleftrightarrow$	Variables Unrestricted

Property:

1. (involutory property of duality) The dual of the dual is the primal.
   This property indicates that the definitions may be applied in reverse. The terms "primal" and "dual" are relative to the frame of reference we choose.

2. (Weak Duality). If x^* is primal feasible and u^* is dual feasible, then

$$cx^* \leq z_{LP} \leq w_{LP} \leq u^*b.$$

3. If problem P has unbounded optimal value, then D is infeasible. This property indicates that unboundedness in one problem implies infeasibility in the other problem. The conversely is not true. This property is not symmetric.

4. (Strong Duality). If z_{LP} or w_{LP} is finite, then both P and D have finite optimal value and $z_{LP} = w_{LP}$.

There are only four possibilities for a dual pair of problems P (primal) and D (dual).

- z_{LP} and w_{LP} are finite and equal.
  If one problem possesses an optimal solution, then both problems possess optimal solutions and the two optimal objective values are equal.

- $z_{LP} = \infty$ and D is infeasible.

- $w_{LP} = -\infty$ and P is infeasible.

- Both P and D are infeasible.

(see [1] for an exhaustive description of the topic)

2.4 Exact Methods for ILP

Linear programming problems and in particular integer linear programming problems are very difficult to solve.
Algorithms for integer programming problems can be divided in three main sets:

- Exact algorithms, as cutting-planes, branch-and-bound, and dynamic programming (for some MIPs), that guarantee to find an optimal solution; however the number of iterations may be often exponential.

- Heuristic algorithms that produce a suboptimal solution, but does not ensure its quality. Even if the running time may not be polynomial, empirical evidence shows that exists (meta)heuristics find good solutions quickly.

- Approximation algorithms that assure in polynomial time a suboptimal solution and a bound on the degree of sub-optimality.

In this section we analyze two exact method: branch-and-bound and cutting plane.

2.4.1 Branch and bound

Branch-and-bound was developed by Land and Doig [25] and by Dakin [10]. In this methods is very important to have an upper bound for the maximum value of ILP, easy to compute and not far from the optimum value. Gomory' s cutting plane method is one method of obtaining an upper bound.
We report a general branch-and-bound algorithm for solving IP. In the description of the algorithm, $\mathscr{L}$ is a collection of integer programs $\{IP_i\}$, each of which is of the form $z_{iIP} = \max\{cx : x \in S_i\}$ where $S_i \subseteq S$. Associated with each problem in $\mathscr{L}$ is an upper bound $\overline{z}_i \geq z_{iIP}$.

General Branch-and-Bound Algorithm:

1. (Initialization): $\mathscr{L} = \{IP\}, S_0 = S, \overline{z}_0 = \infty$, and $\underline{z}_{IP} = -\infty$.
2. (Termination test): If $\mathscr{L} = \emptyset$, then the solution x_0 that yielded $\underline{z}_{IP} = cx_o$ is optimal.
3. (Problem selection and relaxation): Select and delete a problem IP_i from $\mathscr{L}$. Solve its relaxation RP_i. Let z_{iR} be the optimal value of the relaxation and let x_{iR} be an optimal solution if one exists.
4. (Pruning):
   (a) If $z_{iR} \leq \underline{z}_{IP}$, go to Step 2. (Note if the relaxation is solved by a dual algorithm, then the step is applicable as soon as the dual value reaches or falls below $\underline{z}_{IP}$)
   (b) If $x_{iR} \notin S_i$, go to Step 5.
   (c) If $x_{iR} \in S_i$ and $cx_{iR} > \underline{z}_{IP}$, let $\underline{z}_{IP} = cx_{iR}$. Delete from $\mathscr{L}$ all problems with $\overline{z} \leq \underline{z}_{IP}$. If $cx_{iR} = z_{iR}$, go to Step 2; otherwise go to Step 5.
5. (Division): Let $\{S_{ij}\}_{j=1}^{k}$ be a division of S_i. Add problems $\{IP_{ij}\}_{j=1}^{k}$ to $\mathscr{L}$, where $\overline{z}_{ij} = z_{iR}$ for j= 1, ... ,k. Go to Step 2.

Now we give the basic algorithm used by all commercial codes for solving mixed-integer programming problems. For simplicity of notation, we consider only the IP. Let

$$z_{IP} = \max\{cx : x \in S\}, \text{ where } S = \{x \in Z_+^n : Ax \leq b\}$$

be the general integer programming problem. In the initial relaxation, S is replaced by $S_{0LP} = \{x \in R_+^n : Ax \leq b\}$. We also take $z_R(x) = cx$ in each relaxation.

When solving linear programming relaxations we can apply the following pruning criteria of infeasibility, optimality, and value dominance.

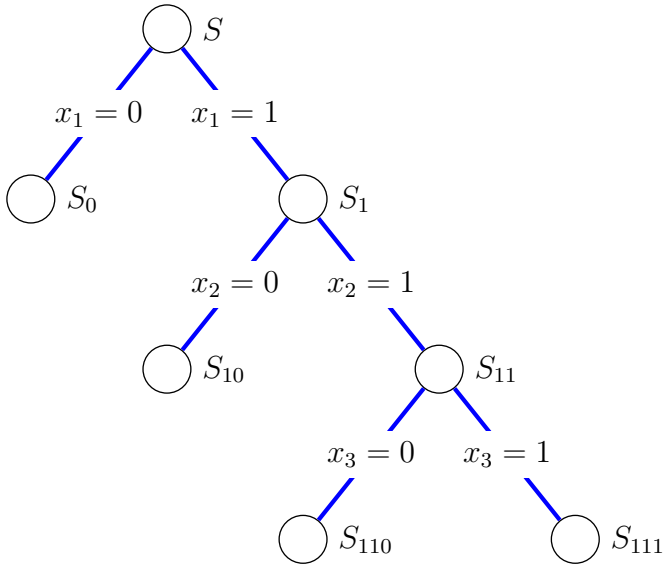

Figure 2.2: Example of branch-and-bound method

Proposition 1. *The enumeration tree can be pruned at the node corresponding to S_i if any of the following three conditions holds.*

- *RP_i is infeasible.*
- *An optimal solution x_{iR} to RP_i satisfies $x_{iR} \in S_i$ and $z_{iR} = cx_{iR}$.*
- *$z_{iR} \leq z_{IP}$ where z_{IP} is the value of some feasible solution of IP.*

Proposition 2. *The enumeration tree can be pruned at the node corresponding to S_i if one of the following two conditions holds.*

- *The objective value of DP_i (dual problem of IP_i) is unbounded from below.*
- *DP_i has a feasible solution of value equal to or less than z_{IP}.*

Suppose the linear programming relaxation at node i of the enumeration tree is

$$z_{iLP} = \max\{cx : x \in S_{iLP}\}, \text{ where } S_{iLP} = \{x \in \mathbb{R}^n_+ : A_i x \leq b_i\}.$$

If LP_i has an optimal solution, we denote the one found by x_i. The pruning conditions are:

1. $S_{iLP} = \emptyset$ (infeasibility);
2. $x_i \in \mathbb{Z}^n_+$ (optimality);

3. $z_{iLP} \leq z_{IP}$ where z_{IP} is the value of a known feasible solution to IP (value dominance). Moreover if LP_i is solved by a dual algorithm, it is possible to prune before an optimal solution to LP_i is found. Also, we might use the weaker condition $z_{iLP} \leq z_{IP} + \epsilon$ for some given tolerance $\epsilon > 0$.

Since we use a linear programming relaxation at each node, the division is obtained by adding linear constraints. An natural way to do this is to consider $S = S_1 \cup S_2$ with $S_1 = S \cap \{x \in \mathbb{R}^n_+ : dx \leq d_0\}$ and $S_2 = S \cap \{x \in \mathbb{R}^n_+ : dx \geq d_0 + 1\}$, where $(d, do) \in \mathbb{Z}^{n+1}$. If x_0 is the solution to the relaxation

$$z_{0LP} = \max\{cx : x \in \mathbb{R}^n_+, Ax \leq b\}$$

we can choose (d, d_0) so that $d_0 < dx_0 < d_0 + 1$. This is expected because it yields $x_0 \notin S_{1LP} \cup S_{2LP}$ and therefore gives the possibility that for $i = 1, 2$ we will obtain $z_{iLP} = \max\{cx : x \in S^i_{LP}\} < z_{0LP}$. In practice, only very special choices of (d, d_0) are used, for example:

- Variable dichotomy. We consider $d = e_j$ for some $j \in \mathbb{N}$. Then x_0 may be infeasible in the relaxations if $x_{0j} \notin \mathbb{Z}^1$ and $d_0 = \lfloor x_{0j} \rfloor$. A very important advantage of this division is that only simple lower- and upper- bound constraints are added to the linear programming relaxation. Therefore the size of the basis does not increase.

- If x_j is bounded ($0 \leq x_j \leq k_j$), we can take each integral value of x_j separately. This approach is not used in commercial integer programming codes.

Note that each of the previous divisions is a partition.

Given a list $\mathscr{L}$ of active subproblems we have to choose which node should be examined in detail next. There are two basic options:

- a priori rules that determine, previously, the order in which the tree will be developed;

- adaptive rules that choose a node using information about the status of the active nodes.

An example of an a priori rule is depth-first search plus backtracking. In depth-first search, if the current node is not pruned, the next node considered is one of its two sons. Backtracking means that when a node is pruned, we go back on the path from this node toward the root until we find the first node that has a son that has not yet been considered. Depth-first search plus backtracking is a completely a priori rule if we fix a rule for choosing branching variables and specify that the left son is considered before the right son. This rule has several advantages:

- The linear programming relaxation for a son is obtained from the linear programming relaxation of its father by adding a simple lower- or upper-bound constraint. Then, given the optimal solution for the father node, it is possible to reoptimize by the dual simplex algorithm.

- Empirical results show that feasible solutions are more likely to be found deep in the tree than at nods near the root. The good result of a branch-and-bound algorithm is very dependent on having a good lower-bound $\underline{z}_{IP}$ for value dominance pruning.

The default option in most commercial codes is depth first when the current node is not pruned. At least one son is considered immediately. Typically, when a node is pruned, the next node is not determined by the backtracking strategy.

There is an other very important example of a priori rule: the breadth-first search. In this case all of the nodes at a given level are considered before any nodes at the next lower level. While this means of node selection is not practical for solving general integer programs using linear programming relaxations, it has some interesting properties, one of which is its use in heuristics. One can choose an active node using the following criteria:

- Choose a node that has to be considered in any case. There is a unique node with the largest upper bound it must be considered. When a node has been pruned, next select from all active nodes one that has the largest upper bound. Hence if $\mathscr{L}$ is the set of active nodes, determine an $i \in \mathscr{L}$ that maximizes $\bar{z}_i$.

- Choose a node that is more likely to contain an optimal solution. The reason for this is that once we have found an optimal solution, even if we are unable to prove immediately that it is optimal, we will have obtained the largest possible value of $\underline{z}_{IP}$. This is very important for subsequent pruning. Suppose $\hat{z}_i \leq \bar{z}_i$ is an estimate of z_{iIP}. The rule best estimate is to choose an $i \in \mathscr{L}$ that maximizes $\hat{z}_i$.

- Try to find quickly a feasible solution $\hat{x}$ such that $c\hat{x} > \underline{z}_{IP}$. The criterion

$$\max_{i \in \mathscr{L}} \frac{\bar{z}_i - \underline{z}_{IP}}{\bar{z}_i - \hat{z}_i}$$

  is called quick improvement. Note that node i with $\hat{z}_i > \underline{z}_{IP}$ will be preferred to node j with $\hat{z}_j \leq \underline{z}_{IP}$. Moreover, preference will be given to nodes for which $\bar{z}_i - \hat{z}_i$ is small. One expects that such nodes will yield a feasible solution quickly. Quick improvement is used in some commercial codes as the default option once a feasible solution is known.

Hypothesize we have chosen an active node i. There is a the linear programming solution x^i associated with it. Secondly, we must select a variable to define the division. We restrict it to the index set $N^i = \{j \in \mathbb{N} : x^i_j \notin \mathbb{Z}^1\}$. Experimental evidence shows that the choice of a $j \in N^i$ can be very important to the running time of the algorithm. Since robust methods for determining such variables have not been established, a common way of choosing a branching variable is by user-specified priorities. This means that an ordering of the variables is specified as part of the input and that branching variables are selected from N^i according to this order.

Other possibilities involve degradations or penalties. Degradation attempts to estimate the decrease in $\bar{z}^i$ that is caused by requiring x_j to be integral. Suppose $x_j = x_j^i = \lfloor x_j^i \rfloor + f_j^i$ and $f_j^i > 0$. Then by branching on x_j, we estimate a decrease of $D_j^{-i} = p_j^{-i}$ for the left son and $D_j^{+i} = p_j^{+i}(1 - f_j^i)$ for the right son. The coefficients $\{p_j^{-i}, p_j^{+i}\}$ can be specified as part of the input or estimated in several different ways.

Given $\{D_j^{-i}, D_j^{+i}\}$ for $j \in N^i$, a common way to choose the branching variable is by the criterion

$$\max_{j \in N^i} \min\{D_j^{-i}, D_j^{+i}\}.$$

The idea is that a variable whose smallest degradation is largest is most important for achieving integrality. When $D_j^{-i} = f_j^i$ and $D_j^{+i} = 1 - f_j^i$, criterion is called maximum integer infeasibility.

Other rules are also used, for example, $\max_{j \in N^i} \max\{D_j^{-i}, D_j^{+i}\}$. Here the idea is that one branch may easily be pruned by value dominance.

2.4.2 Cutting-plane method

Let z_{IP} be a linear integer programming problem, x^* the optimal solution (optimal value z^*), z_{LP} the continuous relaxation of z_{IP}, x_0 the optimal solution of z_{LP} (optimal value z_0). An hyperplane $ax \geq a_0$ is called cutting plane if:

- x_0 is unfeasible ($ax_0 < a_0$)
- is feasible for all optimal integer solution of the original problem ($ax \geq a_0, \forall x$ feasible and integer)

Cutting planes algorithm:

begin

1. solve z_{LP} obtaining x_0
2. if z_{LP} is unbounded or impossible then stop;
3. while x_0 is not integer do
4. determine a cutting plane $ax \geq a_0$ and add it to constraints of P
5. solve z_{LP} obtaining x_0
6. if z_{LP} is impossible then stop;
7. end while

end (you have $x^* = x_0$)

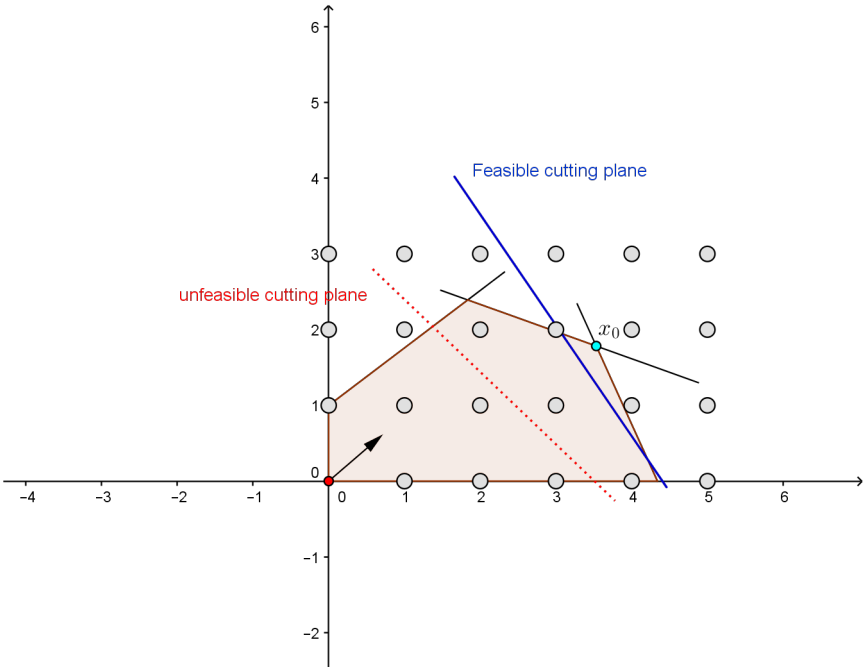

Figure 2.3: Example of feasible and unfeasible cutting plane

This method has some disadvantages: first of all, its computational complexity is not polynomial. The feasible region of an ILP problem may be determined by constraints that can be more or less stringent. In these cases the formulation of ILP are equivalent, but if you remove integrality constraints, generally, you obtain different optimal solutions, as shown in Fig. 2.4. To understand which is the ideal

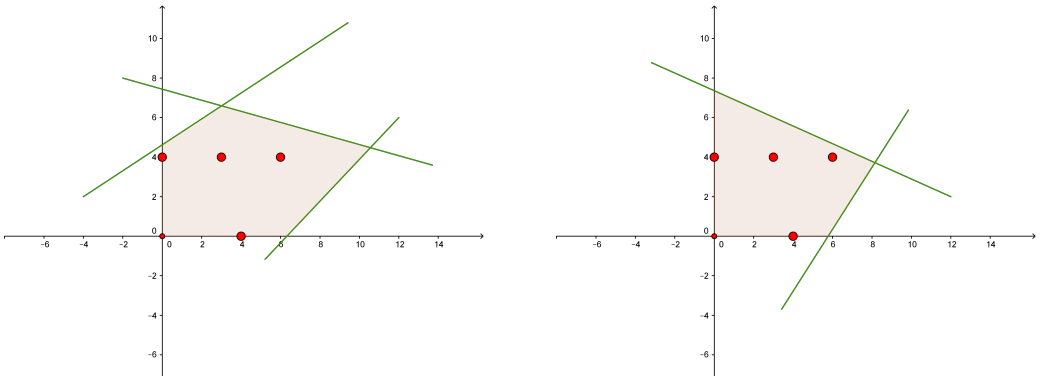

Figure 2.4: Example of different feasible regions of the same ILP

formulation we need the following definition:
Given a set $S \subseteq \mathbb{R}^n$, a point $x \in \mathbb{R}^n$ is a convex combination of points of S if there exists a finite set of points $\{x_i\}_i^t = l \in S$ and a $\lambda \in \mathbb{R}_+^t$ with $\sum_{i=l}^t \lambda_i = 1$ and $x = \sum_{i=l}^t \lambda_i x_i$.

The convex hull of S, denoted by $conv(S)$, is the set of all points that are convex combinations of points in S. If $S \subseteq \mathbb{Z}^n$, $conv(S)$ represents a polytope P' with every corner is an integer point. Given S it is possible to find A', d', subject to $P' = \{x \in \mathbb{R}^n : A'x \geq d', x \geq 0\} = conv(S)$ and it means that $\min\{c^T x : x \in S\} = \min\{c^T x : A'x \geq d, x \geq 0\}$
In this case you can solve the ILP through simplex method.
Unfortunately, to determine $conv(S)$ is very difficult, because in general, the system $A'x \geq d'$ has a large number of constraints. However, there is a case in which the natural formulation of ILP is equivalent to the ideal formulation: when the constraint matrix fulfills an important property, i.e., if it is totally unimodular. We analyze this property in detail in section 2.5.

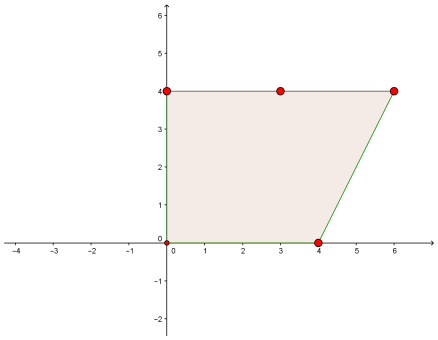

Figure 2.5: Convex hull

2.5 Total Unimodularity

In this paragraph an important property of constraint matrix will be analyzed.

A matrix $A \in \mathbb{Z}^{mxn}$ is totally unimodular (TU) if the determinant of every square submatrix is { 0,1,-1 }.

The following statements are equivalent.

1. A is TU.
2. The transpose of A is TU.
3. (A, l) is TU.
4. A matrix obtained by deleting a unit row (column) of A is TU.
5. A matrix obtained by multiplying a row (column) of A by -1 is TU.
6. A matrix obtained by interchanging two rows (columns) of A is TU.
7. A matrix obtained by duplicating columns (rows) of A is TU.

8. A matrix obtained by a pivot operation on A is TU.

Properties:

- Let A be totally unimodular. If $b \in \mathbb{Z}^m$, then $P := \{x \in \mathbb{R}^n | Ax \leq b\}$ is an integral polyhedron.
- The linear program $\max\{cx : Ax \leq b, x \in \mathbb{R}^n_+\}$ has an integral optimal solution for all integer vectors b for which it has a finite optimal value if and only if A is totally unimodular.
- If A is totally unimodular and b and w are integer vectors, then both sides of the LP-duality equation
  $\max\{wx | Ax \leq b\} = \min\{yb | y \geq 0, yA = w\}$
  have integer optimum solutions[17].
- Let A be a matrix of full row rank. Then the following are equivalent :
  - for each basis B of A, the matrix $B^{-1}A$ is integral;
  - for each basis B of A, the matrix $B^{-1}A$ is totally unimodular;
  - there exists a basis B of A for which $B^{-1}A$ is totally unimodular.
- The following characterization is due to Chandrasekaran [6]: a matrix A is totally unimodular if and only if for each nonsingular submatrix B of A and for each nonzero (0, ± 1)-vector y, the g.c.d. of the entries in yB is 1.

We now report a result by Ghouila- Houri [15] to prove that the continuous relaxation of the model (4.32)-(4.35) always admits an integer optimal solution.

Theorem 1. *Let $A \in \mathbb{Z}^{mxn}$. A is totally unimodular if and only if the following property holds:*
Let $C \subseteq N = \{1, ..., n\}$. Then $\exists$ a partition of C into C_1 and C_2 subject to

$$|\sum_{c_1 \in C_1} a_{ic_1} - \sum_{c_2 \in C_2} a_{ic_2}| \leq 1 \qquad \forall i.$$

Proof. $\Longrightarrow$ Let C be an arbitrary subset of $\mathbb{N}$. Define z by $z_c = 1$ if $c \in C$, $z_c = 0$ otherwise. Also let $d' = 0, d = z, g = Az, b'_i = b_i = -1/2g_i$ if g_i is even, and $b'_i = -1/2(gi - 1), b_i = b'_i + 1$ if g_i is odd. Now consider

$$P(b, b', d, d') = \{x \in \mathbb{R}^n_+ : b' \leq Ax \leq b, d' \leq x \leq d\}.$$

Note that $x = z/2 \in P(b, b', d, d')$. Since A is TU, we have $b', b \in \mathbb{Z}^m, d', d \in \mathbb{Z}^n$ and $P \neq \emptyset$. Then P is integral. Thus there exists $x^0 \in P \cap B_n$ with $x^0_c = 0$ for

$c \in N\ C$ and $x_c^0 \in \{0,1\}$ for $c \in C$. Note that $z_c - 2x_c = \pm 1$ for $c \in C$. Let $C_1 = \{c \in C : z_c - 2x_c^0 = 1\}$ and $C_2 = \{c \in C : z_c - 2x_c^0 = -1\}$. We have

$$\sum_{c_1 \in C_1} a_{ic_1} - \sum_{c_2 \in C_2} a_{ic_2} = \sum_{c \in C} a_{ic}(z_c - 2x_c^0) = \begin{cases} g_i - g_i = 0 & if g_i is even \\ g_i - (g_i \pm 1) = \pm 1 & if g_i is odd \end{cases}$$

Thus

$$\sum_{c_1 \in C_1} a_{ic_1} - \sum_{c_2 \in C_2} a_{ic_2} \leq 1 \quad \forall i$$

$\Longleftarrow$ If $|C| = 1$ we have $a_{ic} \in \{0, \pm 1\} \quad \forall i, \forall c$. The proof is by induction on the size of the nonsingular submatrices of A using the hypothesis that the determinant of every $(k-1) \times (k-1)$ submatrix of A is equal to $0, \pm 1$. Let B be a $k \times k$ nonsingular submatrix of A, and let $r = |detB|$. Our goal is to prove that $r = 1$. By the induction hypothesis and Cramer's rule, we have $B^{-1} = B^*/r$, where $b_{ic}^* = \{0, \pm 1\}$. By the definition of B^* we have $Bb_i^* = re_1$ where b_1^* is the first column of $B*$ Let $C = \{i : b_{i1}^* \neq 0\}$ and $C_1' = \{i \in C : b_{i1}^* = 1\}$. Hence for $i = 2, \ldots, k$, we have

$$(Bb_1^*)_i = \sum_{c \in C_1'} b_{ic} - \sum_{c \in C - C_1'} b_{ic} = 0.$$

Thus $|\{i \in C : b_{ic} \neq 0\}|$ is even; so for any partition $(\tilde{C}_1, \tilde{C}_2)$ of C, it follows that $\sum_{c \in \tilde{C}_1} b_{ic} - \sum_{c \in \tilde{C}_2} b_{ic}$ is even for $i = 2, ..., k$. Now by hypothesis, there is a partition (C_1, C_2) of C such that $|\sum_{c \in C_1} b_{ic} - \sum_{c \in C_2} b_{ic}| \leq 1$. Then

$$\sum_{c \in C_1} b_{ic} - \sum_{c \in C_2} b_{ic} = 0, \quad i = 2, ..., k.$$

Now consider the value of $\alpha_1 = |\sum_{c \in C_1} b_{1c} - \sum_{c \in C_2} b_{1c}|$. If $\alpha_1 l = 0$, define $y \in R^k$ by $y_i = 1$ for $i \in C_1, y_i = -1$ for $i \in C_2$, and $y_i = 0$ otherwise. Since $By = 0$ and B is nonsingular, we have $y = 0$, which contradicts $C \neq \emptyset$. Hence by hypothesis we have $\alpha_1 = 1$ and $By = \pm e_1$. However, $Bb_i^* = re_1$. Since y and b_1^* are $(0, \pm 1)$ vectors, it follows that $b_i^* = \pm y$ and $|r| = 1$. □

Since A is TU if and only if its transpose is TU, Theorem 1 holds also if we consider a partition of rows of A.

We here itemize some basic examples of totally unimodular matrices, in particular network matrices:

1. Bipartite graphs. Let $G = (V, E)$ be an undirected graph, and let M be the $V \times E$ -incidence matrix of G (i.e. M is the $(0,1)$-matrix with rows and columns indexed by the vertices and edges of G, respectively, where $M_{v,e} = 1$ if and only if $v \in e$). Then:
   M is totally unimodular if and only if G is bipartite.
   So M is totally unimodular if and only if the rows of M can be split into two classes so that each column contains a 1 in each of these classes. Assertion easily follows from Ghouila-Houris characterization 1.

2. Directed graphs. Let $D = (V, A)$ be a directed graph, and let M be the $V \times A$-incidence matrix of D (i.e. M is a $(0, \pm 1)$-matrix with rows and columns indexed by the vertices and arcs of D, respectively, where $M_{u,a} = +1(= -1)$ if and only if a enters (leaves) u). Then M is totally unimodular. So a $(0, \pm 1)$-matrix with in each column exactly one + 1 and exactly one - 1 is totally unimodular.
   Again with Hoffman and Kruskals theorem, the total unimodularity of incidence matrices of digraphs implies several graph-theoretical results, like Mengers theorem, the max-flow min-cut theorem, Hoffmans circulation theorem, and Dilworths theorem.

3. Network matrices. Let $D = (V, A)$ be a directed graph and let $T = (V, A)$ be a directed tree on V. Let M be the $A_0 \times A$-matrix defined by, for $a = (v, w) \in A$ and $a \in A_0$: M'_a, a

   - + 1 if the unique v-w-path in T passes through a forwardly;
   - - 1 if the unique v-w-path in T passes through a backwardly;
   - 0 if the unique v-w-path in T does not pass through a.

   Matrices arising in this way are called network matrices. If M, T, and D are as above, we say that T and D represent M. Note that the class of network matrices is closed under taking submatrices: deleting a column corresponds to deleting an arc of D, while deleting a row corresponds to contracting an arc of T.
   Network matrices are totally unimodular.

4. Minimum-cost network flow.

   The constraint matrix A of a minimum-cost flow problem is totally unimodular. We consider the node-arc incidence matrix A; since all entries are ± 1 or 0, every $|x|$ submatrix has determinant ± 1 or 0. Hence, by induction, suppose that this property is true for every square submatrix of size $(k-1) \times (k-1)$, and let A_k be any $k \times k$ submatrix of A, where $k > 2$. We must show that det $A_k = \pm 1$ or 0. Note that each column of A_k has either all zeros or only a single nonzero entry that is $+1$ or -1 , or it has exactly two nonzero entries, namely a $+1$ and a -1. If any column of A_k is zero, then det $A_k = 0$. If any column of A_k has a single nonzero entry, then expanding the determinant of A_k by the minors of that column, we get det $A_k = \pm detA_{k-1}$, where A_{k-1} is a square submatrix of size $(k-1) \times (k-1)$. By the induction hypothesis, det $A_{k-1} = \pm 1$ or 0, and hence det $A_k = \pm 1$ or 0. Otherwise, every column of A_k must have a $+1$ and a -1 . In this case, since the rows of A_k add up to the zero vector, we have that det $A_k = 0$. Then, A is totally unimodular.

(see [32] for an exhaustive description of the topic)

2.6 Computational Complexity

There are several ways of defining a concept of running time, to indicate the number of elementary bit operations in the execution of an algorithm by a computer or computer model. In practice, this time will depend on the eventual implementation of the algorithm. An algorithm is called polynomial if its running time function is polynomially bounded. A problem is said to be solvable in polynomial time or polynomially solvable if the problem can be solved by a polynomial algorithm. The class of decision problems solvable in polynomial time is denoted by $\mathscr{P}$. Another, possibly larger, complexity class is the class $\mathscr{NP}$, that means solvable by a Non-deterministic Turing machine in Polynomial time, and not for nonpolynomial. Informally, the class $\mathscr{NP}$ can be described as the class of those decision problems $\mathscr{L}$ satisfying: for any $z \in \mathscr{L}$, the fact that z is in $\mathscr{L}$ has a proof of length polynomially bounded by the size of z. It can be shown that certain problems in the class $\mathscr{NP}$ are the hardest among all problems in $\mathscr{NP}$, under a certain ordering of problems by difficulty, like the integer linear programming problem. These problems are called $\mathscr{NP}$-complete.
The simplex method was designed by Dantzig [11], and it is the main method used to solve linear programming. Although some artificial examples show exponential running time, in practice and on the average the method is very efficient. It is not a polynomial-time method, because in the worst case its computational complexity is exponential. However, in practice the simplex method is very fast and experience suggests that the number of pivot steps is about linear in the problem dimensions.

Chapter 3

Vehicle Routing Problems

Vehicle Routing Problems (VRP) or Vehicle Scheduling Problems are problems in which some goods are distributed between a set of depots and a set of customers by a set of vehicles, operated by a set of drivers who can move on a given road network. There are a large number of different real applications: solid waste collection, street cleaning, delivery and collection of goods, school bus routing, dial-a-ride systems, transportation of handicapped persons, routing of salespeople. The VRP generalizes one of the most famous and important combinatorial optimization problems: the traveling salesman problem (TSP)
The objective is to determine a set of routes, one for each vehicle that starts and ends at its own depot, such that all customers' requirements and operational constraints are satisfied and global transportations costs are minimized. The road network, used for the delivery or collection of goods, can be described through a graph where the arcs are roads and nodes are junctions between them and correspond with the depot and customer locations. Each arc has an associated cost which is generally its length or travel time. The arcs may be directed or undirected if they can be traversed in only one direction or in both direction respectively. For example, we have a directed arc if it correspond to a one way street or if the cost are different in each direction.

A customers is characterized by:

- his location, corresponding to vertex position of the graph
- the demand, quantity of freights that must be delivered or collected
- time windows, period of the day in which customer must be serviced
- unloading or loading times, if goods must be delivered or collected respectively
- subset of vehicles that can be used to serve the customer

A vehicle is characterized by:

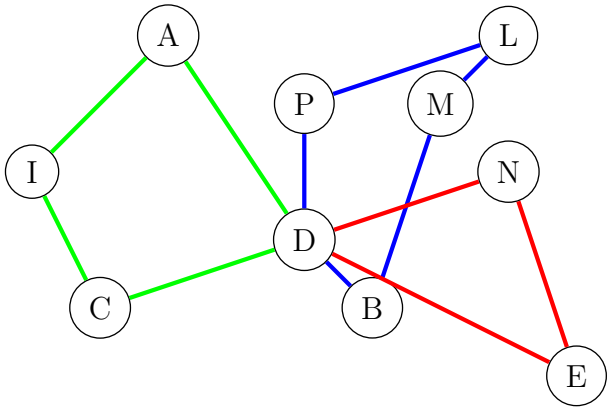

Figure 3.1: A VRP graph

- home depot of the vehicle, and the possibility to end service at a depot other than the home one;
- capacity of the vehicle, expressed as the maximum weight, or volume, or number of pallets, the vehicle can load;
- possible subdivision of the vehicle into compartments, each characterized by its capacity and by the types of goods that can be carried;
- devices available for the loading and unloading operations;
- subset of arcs of the road graph which can be traversed by the vehicle; and
- costs associated with utilization of the vehicle (per distance unit, per time unit, per route, etc.).

In a VRP problem, as previously mentioned, we can have different depots and each of these is characterized by the number and the types of vehicles. In real applications, a set of customers can be assigned to only one depot. In this way, we obtain a decomposition of the problem into several independent VRP problems, one for each different depot in the road network. Drivers must comply various constraints laid down by union contracts and company regulations, as working periods during the day, number and duration of breaks during service, maximum duration of driving periods, overtime.

Routes must satisfy several operational constraints, as the following:

- along each route, the current load of the associated vehicle must be lower than the vehicle capacity;
- the customers served in a route can require only the delivery or the collection of goods, or both possibilities;

- customers can be served only within pre established time windows;
- the duration of any route must be lower than a work shift duration.

Sometimes it is impossible to fully satisfy the demand of each customer. May be necessary to reduce the amount of goods or do not satisfy all customers. In this case we can associate different priority to customers.

Some of the most important basic vehicle routing problems are the following:

- Capacitated VRP (CVRP). The basic version of VRP is the Capacitated VRP. In the CVRP, all the customers correspond to deliveries and the demands are deterministic, known in advance, and may not be split. The vehicles are identical and based at a single central depot, and only the capacity restrictions for the vehicles are imposed. The objective is to minimize the total cost (i.e., a weighted function of the number of routes and their length or travel time) to serve all the customers. The CVRP may be described as the following graph theoretic problem. Let $G = (V, A)$ be a complete graph, where $V = \{0, \ldots, n\}$ is the vertex set and A is the arc set. Vertices $i = 1, \ldots, n$ correspond to the customers, whereas vertex 0 corresponds to the depot. Sometimes the depot is associated with vertex $n + 1$. A nonnegative cost, c_{tj}, is associated with each arc(i, j) e A and represents the travel cost spent to go from vertex i to vertex j. Generally, the use of the loop arcs, (i, i),is not allowed and this is imposed by defining $c_{ii} = \infty \quad \forall i \in V$. If G is a directed graph, the cost matrix c is asymmetric, and the corresponding problem is called asymmetric CVRP (ACVRP). Otherwise, we have $c_{ij} = c_{ji} \quad \forall (i, j) \in A$, the problem is called symmetric CVRP (SCVRP), and the arc set A is generally replaced by a set of undirected edges, E.

  In several practical cases, the cost matrix satisfies the triangle inequality,

  $$c_{ik} + c_{kj} \geq c_{ij} \quad \forall i, j, k \in V.$$

  In other words, it is not convenient to deviate from the direct link between two vertices i and j. The presence of the triangle inequality is sometimes required by the algorithms for CVRP.
  The CVRP consists of finding a collection of exactly K simple circuits (each corresponding to a vehicle route) with minimum cost, defined as the sum of the costs of the arcs belonging to the circuits, and such that

  - each circuit visits the depot vertex;
  - each customer vertex is visited by exactly one circuit; and
  - the sum of the demands of the vertices visited by a circuit does not exceed the vehicle capacity, C.

- VRP with Backhauls.

  The VRP with Backhauls (VRPB) is the extension of the CVRP in which the customer set $V \setminus 0$ is partitioned into two subsets. The first subset, L, contains n Linehaul customers, each requiring a given quantity of product to be delivered. The second subset, B, contains m Backhaul customers, where a given quantity of inbound product must be picked up. Customers are numbered so that $L = \{1, \ldots, n\}$ and $B = \{n+1, \ldots, n+m\}$. In the VRPB, a precedence constraint between linehaul and backhaul customers exists: whenever a route serves both types of customer, all the linehaul customers must be served before any backhaul customer may be served. A nonnegative demand, d_i, to be delivered or collected depending on its type, is associated with each customer i, and the depot is associated with a fictitious demand $d_0 = 0$

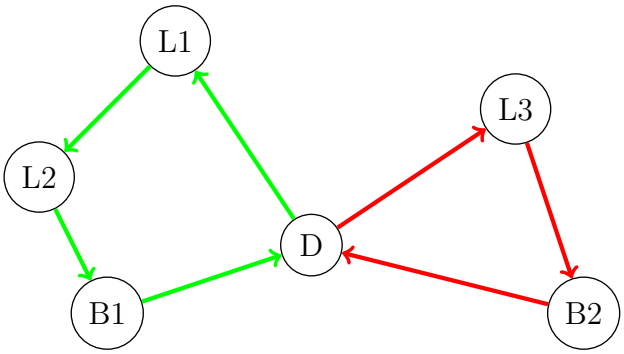

Figure 3.2: A VRP with Backhauls

- VRP with Pickup and Delivery The VRP with Pickup and Delivery consists of finding a collection of exactly K simple circuits with minimum cost, and such that:
  - each circuit visits the depot vertex;
  - each customer vertex is visited by exactly one circuit;
  - the current load of the vehicle along the circuit must be nonnegative and may never exceed the vehicle capacity C;
  - for each customer i, the customer O_i, when different from the depot, must be served in the same circuit and before customer i;
  - for each customer i, the customer D_i, when different from the depot, must be served in the same circuit and after customer i.
- VRP with Time Windows The VRP with Time Windows (VRPTW) is the extension of the CVRP in which capacity constraints are imposed and each customer i is associated with a time interval $[a_i, b_i]$, called a time window.

(see [34] for an exhaustive description of the topic)

3.1 Drayage & VRP with Full Container Loads

In the field of intermodal freight transportation, the distribution of containers by trucks between importers, exporters and intermodal terminals is called *drayage*. Drayage is a critical part of the service provided by liner shipping companies. While their vessels carry thousands of containers in maritime networks, the supply of door-to-door services between intermodal terminals and customers must be performed by trucks carrying one or two containers. Therefore, drayage problems are much more resource-intensive and energy-consuming than the maritime counterpart [29].

In container drayage, mostly 20-foot and 40-foot standard containers are transported, and usually only up to two 20-foot containers or one single 40-foot container can be carried by a truck [35]. In recent years the literature on drayage has increased significantly, and many problem settings have been investigated.

Drayage problems were often faced by optimization methods in a variant of VRP, called *Vehicle Routing Problem with Full Container Loads*(VRPFC), in order to plan how to serve the requests of import and export customers, who must receive and ship container loads, respectively. In its basic setting, the VRPCF can be defined as follows: given a homogeneous container type and a fleet of trucks with possibly different hauling costs, the objective is to find the optimal assignment of vehicles to a set of delivery and pickup point pairs (hereafter referred to as DP pairs), in order to minimize the total distribution cost.

The VRPCF belongs to the field of pickup and delivery problems, because there are two types of customers needing to ship or receive container loads ([30]). When all deliveries must be performed before all pickups in each route, this VRP variant belongs to the class of *Vehicle Routing Problems with Clustered Backhauls*, according to the problem classification in [28] or *One-to-many-to-one pickup and delivery problems with Single Demands and Backhauls* in accordance with [2].

When customers can be visited in any order the drayage problem is classified as *Vehicle Routing Problem with Mixed Linehauls and Backhauls* (VRPMB) in [28] or *One-to-many-to-one pickup and delivery problems with Single Demands and Mixed Solutions* in[2].

In the vast literature on VRPFC, two types of transportation requests are investigated: *drop&pick* and *stay-with*. In the first case, while containers are left at customer locations, drivers can move to other customers, thus bypassing packing and unpacking operations [38, 21, 43, 46, 3, 27]. Conversely, in *stay-with* operations drivers wait for containers and trucks carry the same container(s) throughout their routes [29, 19, 5].

In what follows we summarize the main features of the most important papers in the field of VRPFC.

Wang and Regan (2002) [38] considered a local truckload pickup and delivery problems with a fixed number of vehicles; they developed a solution method based on windows partition.

Gronalt et al. (2003) [16] dealt with the pickup and delivery of full truckloads be-

tween distribution centers with time window constraints. The objective function is to minimize empty vehicle movements, as these use resources without directly adding value to the products transported. They proposed an exact formulation of the problem and a relaxed problem formulation based on network flows, which can be used to calculate a lower bound to the solution value. Finally, they used four different savings based heuristics for the problem.
Jula et al. (2005) [21] presented a drayage problem as an asymmetric multi-traveling salesman problem with time windows constraints for container movements. The containers (loaded and empty) move from an intermodal facility to a customer and vice versa. The problem is solved using a hybrid method of dynamic programming and genetic algorithm.
Ileri et al. (2006) [18] presented a daily drayage problem, which is a complex real-word problem, where loaded or empty equipments move between customer locations and rail ramps. They solved the problem by an exact approach, through a formulation consists on a set partitioning model whose columns represent routes.
Chung et al. (2007) [8] dealt with the workflow in container transportation and then developed mathematical models integrating the various operating and design characteristics for the container transportation problem in Korean trucking industries. They presented several formulations that utilize the standard formulations of Traveling Salesman Problem and VRP, and also heuristic algorithms to solve these models.
Francis et al. (2007) [13] proposed modeling and solution method improvements for the Multi-Resource Routing Problem. They presented a heuristic approach that uses randomized route generation with flexible tasks.
Imai et al. (2007) [19] investigated a VRP with pickups and deliveries, more specifically a VRPFC with own and chartered vehicles incident to an intermodal terminal and time lengths constraints. They solved the problem using a heuristic based on a Lagrangian relaxation which allows to identify a near optimal solution. The heuristic consists of some sub-problems, which are the classical and the generalized assignment problem. This procedure is tested on a great number of problem examples and results demonstrate that it can be used to solve a large number of instances.
Namboothiri and Erera (2008) [26] considered the management of vehicles providing container pickup and delivery service to a port with an appointment-based access control system. They proposed a heuristic approach determining pickup and delivery sequences for daily drayage operations with minimum transportation cost.
Caris and Janssens (2009) [5] investigated a drayage problem of containers in the service area of an intermodal terminal, that is modelled as a full truckload pickup and delivery problem with time windows. They proposed a two-phase insertion heuristic to construct an initial solution, that is improved with a local search heuristic based on three neighbourhoods.
Zhang et al. (2009) [45] considered a truck scheduling problem in which empty containers are considered as a transportation resource. A cluster method and a reactive tabu search algorithm have been developed to solve the problem.

Zhang et al. (2010) [43] investigated a problem in which containers move between multiple depots and multiple terminals. They modeled the problem as an extension of the multiple traveling salesman problem with double-side time windows of loads at both the origin and the destination.
Vidovic et al. (2011) [36] studied a problem that is closely related to the vehicle routing problem with backhauls (VRPB), in oder to determine an optimal set of orders (or routes) visiting deliveries (linehauls) and pickups (backhauls). Containers of different sizes, but mostly 20ft, and 40ft empty and/or loaded should be delivered to, or collected from the customers.
Zhang et al. (2011) [46] dealt with a multiple traveling salesman problem with time windows with resource constraints. They developed a heuristic approach based on a reactive tabu search.
Braekers et al. (2013) [3] studied a VRP for transporting loaded and empty containers in drayage operations. They formulated the problem as an asymmetric multiple vehicle traveling salesman problem with time windows and proposed two solution approaches.
Sterzik and Kopfer (2013) [33] analyzed the movement of full and empty containers among a number of terminals, depots and customers. A trucking company with a homogeneous fleet of trucks has to serve customers which either receive goods by inbound containers or ship goods by outbound containers. They solved vehicle routing and scheduling problem in container drayage operations with tabu search metaheuristics.
Wang and Yun (2013) [37] investigated a container transportation problem by truck and train with time windows constraints. Containers are classified into four types according to the direction (inbound containers and outbound containers) and container state (full containers or empty containers). The authors developed a mathematical model by a graph model and proposed a hybrid tabu search.
Nossack and Pesch (2013) [27] dealt with a truck scheduling problem that arises in intermodal container transportation, where containers move between customers and container terminals and vice versa. The problem is formulated as Full-Truckload Pickup and Delivery Problem with time windows and is solved by a 2-stage heuristic solution approach.
Braekers et al. (2014) [4] investigated a full truckload vehicle routing problem in drayage operations around intermodal container. The problem is formulated as an asymmetric multiple vehicle Traveling Salesman Problem with time windows. The authors proposed a two-phase hybrid deterministic annealing and tabu search algorithm.
Xue et al. (2014) [41] examined a Local Container Drayage Problem (LCDP) under an operation mode in which a tractor can be detached from its companion trailer and assigned to a new task. They solved the problem with a tabu search algorithm.
Zhang et al. (2014) [42] studied a container drayage problem with flexible orders defined by requiring and releasing attributes as a unified formulation of various order types. A determined-activities-on-vertex graph introduces a temporary vertex set

to formulate different truck statuses. The problem is formulated as a mixed-integer nonlinear programming model based on the graph.
Lai et al. (2015) [22] proposed an Adaptive Guidance metaheuristic to investigate the use of a homogeneous fleet of large vehicles, which can be adopted to face the problem presented by[24].
Xue et al. (2015) [40] studied a Local Container Drayage Problem under a special operation mode in which tractors and trailers can be separated; tractors can be assigned to a new task at another location while trailers are waiting for packing or unpacking. The containers for pickup and delivery customers are owned by different shippers. The problem is formulated as a vehicle routing and scheduling problem with temporal constraints.
Zhang et al. (2015) [44] presented a multi-size container truck transportation problem in which a truck can carry one 40 ft or two 20 ft containers. This problem considers both fixed and flexible drayage orders. The problem is modeled as a sequence-dependent multiple-traveling salesman problem with social constraints in which the distances between cities depend on the sequence of cities visited before.
Funke and Kopfer (2016) [14] investigated a routing problem in which trucks can transport up to two 20-foot or one 40-foot container at a time along routes with various pickup and delivery locations. A mixed-integer linear program for this problem is presented using two alternative objective functions: minimization of the total travel distance and minimization of the total operation time of the trucks.
Reinhardt et al. (2016) [29] proposed an intermodal transportation problem arising in the liner shipping industry. The authors proposed several different mathematical models for optimizing a one-day schedule covering all import and export orders.
Vidovic et al. (2016) [35] studied a problem of vehicle routing in drayage operations, where vehicles can carry containers of different sizes. This multisize container drayage problem with time windows is modeled as a multiple matching problem and formulated as a mixed integer linear program model. They solved larger sized problems using a variable neighborhood search heuristic.
To summarize, all papers in the field of VRPFC aimed at solving specific variants of this problem. The objective of this thesis is different: it aims to investigates why the continuous relaxations of several formulations of VRPFC admit an integer optimal solution.

Chapter 4

The analysis of algebraic structures in a VRPFC

4.1 Problem Description

Consider a fleet of trucks and containers based at the port. Trucks can carry up one or two containers to serve two types of customers: importers and exporters. In the first case the transportation request consists in the delivery of container loads from the port to the importers; in the second case it consists of the shipment of container loads from the exporters to the port. Typically, each customer needs to ship or receive more-than-one container load. Therefore, usually each customer must be visited by more than one truck.
It is important to note that in this problem containers are never unloaded or reloaded from the chassis of the truck along a route. They are brought to the customers, where they are packed or unpacked and moved away by the same trucks. Therefore, while containers are emptied at importer locations, drivers wait for empty containers to be returned. Similarly, trucks move empty containers to export customers and drivers wait for loaded containers to be returned, while packing operations are performed. Trucks and containers are coupled in the sense that the truck carries the same set of containers throughout the route.
Although customers cannot be provided with large time windows for loading or unloading, this practice is perceived as a high quality service, because the integrity and the content of the cargo can be immediately verified by drivers. More precisely, they supervise the unloading operations making promptly sure that the container loads are the correct ones, in the right quantity and there are not damages. The number of claims and cargo returns is therefore limited. From the carrier's point of view, this policy improves container safety and integrity, because containers are never left unsupervised at customer locations.
The carrier is aware of the fact that leaving containers at customer locations would save drivers the time to supervise loading and unloading operations and they could

move to other customers in the meanwhile ([7]). However, possible cargo problems may be noticed only after truck departures and corrective actions may not be taken rapidly. Moreover, when containers are left at customer locations without chassis, special equipment is required for decoupling and coupling the container to the chassis, whereas in this case study the carrier's policy demands customers to be equipped only with forklift trucks for packing and unpacking operations.
As expected from carrier's policy, importers must be service before exporters. The use of containers emptied at importers to collect cargoes from the exporters ([20], [12]) is also called *street-turn*. Since the number of container loads to be delivered to importers and picked from exporters is typically different, street-turns are typically insufficient to meet all customer requests and additional empty containers must be picked up or returned to the port with loaded containers. More precisely, when the number of container loads to be delivered is larger than the number of container loads to be picked up, trucks return empty containers back to the port. Conversely, trucks leave the port carrying empty containers when the container loads of exporters are larger than those of importers. We assume there is always a number of containers at the port sufficient to serve all import and export requests.

If each truck carries one container, it can service two customers (one importer and one exporter) at most. Three possible routes are allowed in this case:

- moving a loaded container from the port to an importer and the empty container from the importer to the port;
- moving a loaded container from the port to an importer, the empty container from the importer to an exporter and, finally, the loaded container from the exporter to the port;
- moving an empty container from the port to an exporter and the loaded container from the exporter to the port.

There are costs generated by the movement of trucks: routing costs and handling costs. Routing costs depend on the distance and on the trucks. In this problem trucks with capacity two have higher costs then trucks with capacity one. Handling costs are paid to put containers on trucks at the port. The objective is to determine routes minimizing routing and handling costs.

4.2 Initial formulation of the problem

We recall the general model proposed in [24]. We consider a port p, a set of Importers I, a set of Exporters E, a set of trucks K.

Given a graph G(N, A), where $N = p \cup I \cup E$ and $A = A_1 \cup A_2, A_1 = \{(i,j)|i \in I \cup p, j \in N, i \neq j\}, A_2 = \{(i,j)|i \in E, j \in E \cup p, i \neq j\}$, the following decision variables are defined:
x_{ij}^k is the routing selection variable, which is equal to 1 if arc $(i,j) \in A$ is traversed

by truck $k \in K$, 0 otherwise;
y_{ij}^k represents the number of loaded containers moved along arc $(i, j) \in A$ by truck $k \in K$;
z_{ij}^k represents the number of empty containers moved along arc $(i, j) \in A$ by truck $k \in K$;
Moreover, let c_{ij}^k be the routing cost of truck $k \in K$ moving along arc $(i, j) \in A$, h_{pi}^k the handling cost of a container put on truck $k \in K$ at the port p to serve customer $i \in I \cup E$, $d_i \geq 0$ the number of containers used to serve customer $i \in I \cup E$, and u_k the maximum number of containers carried by truck $k \in K$. In container drayage one or two containers are typically carried by conventional trucks (i.e. $u_k = 1 \vee u_k = 2$). A large number of containers can be carried by specialized vehicles, if their circulation is allowed [23]. The optimization model is reported hereafter:

$$\min \sum_{k\in K}\left[\sum_{(i,j)\in A} c_{ij}^k x_{ij}^k + \sum_{i\in N} h_{pi}^k (y_{pi}^k + z_{pi}^k)\right] \tag{4.1}$$

$$\sum_{k\in K}\sum_{l\in N} y_{il}^k = \sum_{k\in K}\sum_{j\in p\cup I} y_{ji}^k - d_i \qquad \forall i \in I \tag{4.2}$$

$$\sum_{k\in K}\sum_{l\in N} z_{il}^k = \sum_{k\in K}\sum_{j\in p\cup I} z_{ji}^k + d_i \qquad \forall i \in I \tag{4.3}$$

$$\sum_{l\in N} y_{il}^k \le \sum_{j\in p\cup I} y_{ji}^k \qquad \forall i \in I, \forall k \in K \tag{4.4}$$

$$\sum_{l\in N} z_{il}^k \ge \sum_{j\in p\cup I} z_{ji}^k \qquad \forall i \in I, \forall k \in K \tag{4.5}$$

$$\sum_{k\in K}\sum_{l\in p\cup E} y_{il}^k = \sum_{k\in K}\sum_{j\in N} y_{ji}^k + d_i \qquad \forall i \in E \tag{4.6}$$

$$\sum_{k\in K}\sum_{l\in p\cup E} z_{il}^k = \sum_{k\in K}\sum_{j\in N} z_{ji}^k - d_i \qquad \forall i \in E \tag{4.7}$$

$$\sum_{l\in p\cup E} y_{il}^k \ge \sum_{j\in N} y_{ji}^k \qquad \forall i \in E, \forall k \in K \tag{4.8}$$

$$\sum_{l\in p\cup E} z_{il}^k \le \sum_{j\in N} z_{ji}^k \qquad \forall i \in E, \forall k \in K \tag{4.9}$$

$$\sum_{(ji)\in A} (y_{ji}^k + z_{ji}^k) = \sum_{(il)\in A} (y_{il}^k + z_{il}^k) \qquad \forall i \in I \cup E, \forall k \in K \tag{4.10}$$

$$y_{ij}^k + z_{ij}^k \le u_k x_{ij}^k \qquad \forall (i,j) \in A, \forall k \in K \tag{4.11}$$

$$\sum_{j\in N} x_{ji}^k - \sum_{l\in N} x_{il}^k = 0 \qquad \forall i \in N, \forall k \in K \tag{4.12}$$

$$\sum_{j\in N} x_{pj}^k \le 1 \qquad \forall k \in K \tag{4.13}$$

$$\sum_{k\in K}\sum_{i\in I\cup E} z_{ip}^k - \sum_{k\in K}\sum_{i\in I\cup E} z_{pi}^k = \sum_{i\in I} d_i - \sum_{i\in E} d_i \tag{4.14}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall (i,j) \in A, \forall k \in K \tag{4.15}$$

$$y_{ij}^k \in Z^+ \qquad \forall (i,j) \in A, \forall k \in K \tag{4.16}$$

$$z_{ij}^k \in Z^+ \qquad \forall (i,j) \in A, \forall k \in K \tag{4.17}$$

Constraints (4.2) and (4.3) are the flow conservation constraints of loaded and empty containers at each importer, respectively. Constraints (4.4) and (4.5) guarantee that the number of loaded containers cannot be increased after a service at each importer and the number of empty containers cannot be reduced. Constraints (4.6) and

(4.7) are the flow conservation constraints of loaded and empty containers at each exporter, respectively. Constraints (4.8) and constraints (4.9) guarantee that the number of loaded containers cannot be reduced after a service at each exporter and the number of empty containers cannot be increased. Constraints (4.10) guarantee that the number of containers carried by each truck is not changed throughout its route. Constraints (4.11) impose that the number of containers carried by each truck is not larger than its transportation capacity u_k. Constraints (4.12) represent flow conservation constraints for trucks at each node; constraints (4.13) impose that trucks are not used more than once; constraint (4.14) is the flow conservation of empty containers at port p. Finally, (4.15), (4.16) and (4.17) describe the domain of the decision variables.

A feasible solution was built by an adaptation of the Clarke and Wright (1964) algorithm [9] and was improved using two neighborhoods with a best improvement strategy. In the first phase a feasible solution is determined by a variant of the Clarke and Wright method, in which routes are merged and assigned to trucks. In the second phase,, this solution is improved by several local search phases. The search space of the local search phases is the set of truck assignments to routes satisfying all constraints. Two neighborhoods are used: in the first, a node is moved from its current route and inserted into another route by the best-insertion method and trucks are reassigned to routes involved in this local move. In the second, two nodes are swapped between two different routes and trucks are reassigned to routes involved in this local move.

4.3 Experimentation and a conjecture on TU

In this Section we solve the previous model in the case of one-container per truck and identical routing costs for each vehicle. The reported experimentation is carried out on 35 artificial instances, in which the coordinates of customers and the number of containers to be shipped or received are taken from [24]. These instances are divided into five classes:

- 3 instances with 10 customers, who must be served by 28 containers;
- 5 instances with 20 customers, who must be served by 61 containers;
- 7 instances with 30 customers, who must be served by 88 containers;
- 9 instances with 40 customers, who must be served by 125 containers;

The model and its continuous relaxation have been solved by Cplex 12.5 on a Workstation with Windows 8 64 bit, Intel i7-4700 MQ 2.40 Ghz processor, 8 GB of RAM, default parameter settings and maximum running time of 1 hour1. The outcomes are reported in the following table:

|I|	|E|	|K|	opt-sol IP	opt-sol LP	GAP IP/ LP	ex-time IP	ex-time LP
2	8	19	29444.821	29444.821	0	1.718	0.219
5	5	16	30012.495	30012.495	0	1.296	0.766
8	2	23	30067.104	30067.104	0	2.031	1.047
2	18	52	62406.297	62406.297	0	17.553	6.765
5	15	45	36048.282	34590.210	0	601.508	3.047
10	10	33	48782.049	48782.049	0	11.974	3.598
15	5	45	56692.138	56692.138	0	15.539	5.551
18	2	53	65765.831	65765.831	0	16.393	7.016
2	28	79	97366.504	97366.504	0	74.977	25.715
5	25	72	90911.344	90911.344	0	63.678	22.14
10	20	60	79011.592	79011.592	0	44.992	17.426
15	15	45	74324.637	74324.637	0	36.603	11.203
2	28	79	80744.698	80744.698	0	53.882	16.139
25	5	75	93938.997	93938.997	0	62.359	23.509
28	2	84	102584.528	102584.528	0	79.991	25.824
2	38	118	146631.032	146631.032	0	143.077	63.944
5	35	108	135257.555	135257.555	0	161.381	57.963
10	30	89	114334.017	114334.017	0	130.588	41.536
15	25	74	102522.340	102522.340	0	95.578	33.283
20	20	70	105772.860	105772.860	0	108.078	32.351
25	15	86	115210.468	115210.468	0	123.025	39.325
30	10	102	132274.337	132274.337	0	149.433	47.878
35	5	115	146420.060	146420.060	0	167.167	52.319
38	2	121	152394.732	152394.732	0	164.022	58.271

Table 4.1: First experimentation

Table 4.1 shows that the continuous relaxation is always integer. This outcome was also observed in extensive computational experiments which are not reported in this thesis. Therefore, one can assume that the constraint matrix of the previous model is TU. It is worth noting that the condition one container per truck ($u_k = 1$, $\forall k \in K$) is necessary for TU. In fact, in (4.11), if $u_k \neq 1$ the constraint matrix has some entries different to 1, 0 or -1, then it is not totally unimodular. In sections 4.4 and 4.5 we discuss if the condition of one-container per truck is sufficient to prove this property.

4.4 Alternative formulations of the model to verify the conjecture

As mentioned previously, in this section we reformulate the model (4.1)-(4.17) proposed by [24], in order to investigate the properties of constraint matrix and verify if it is TU. First of all, in paragraph 4.4.1 we reformulate the problem in the case $u_k = 1$. In paragraph 4.4.2 we propose an other reformulation, obtained by the model (4.26)-(4.30) adding the condition that all vehicles have the same costs.

4.4.1 Case with one-container per truck and heterogeneous vehicles

If all vehicles carry one container, the model (4.1)-(4.17) can be reformulated as shown in this section. The variables on loaded and empty containers can be removed, because each vehicle has a one-to-one relationship with a container. Moreover, the flows of loaded or empty containers between any pair of nodes can be recognized by the type of nodes. To clarify:

- A truck moving from a port to any importer can carry a loaded container only.
- A truck moving from an importer to the port can carry an empty container only.
- A truck moving from a port to any exporter can carry an empty container only.
- A truck moving from any exporter to the port can carry a loaded container only.
- A truck moving from an importer to any exporter can carry an empty container only.

When the capacity of the trucks is one, the model can be simplified. The number of variables can be reduced and associated with one-to-one correspondence to possible routes.

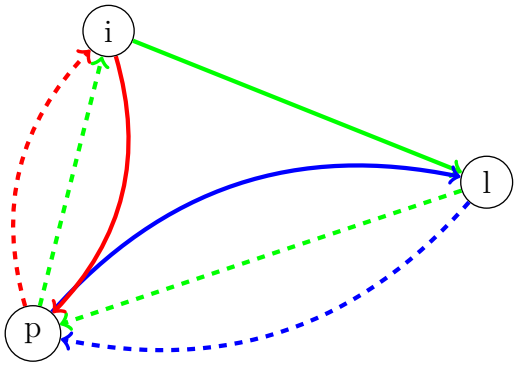

Figure 4.1: Possible routes

In Fig. 4.1 we show the allowed routes: red lines represent the route starting from the port to an importer and from the importer to the port ; green lines show the route starting from the port to an importer, from the importer to an exporter, from the exporter to the port ; finally, blue lines indicate the route starting from

the port to an exporter and from the exporter to the port (blue lines); dashed and solid lines represent loaded containers and empty containers respectively.

The variable y_{ij}^k is equal to x_{ij}^k if $i = p, j \in I$, or $i \in E, j = p, 0$ otherwise; z_{ij}^k is equal to x_{ij}^k if $i \in \{p\} \cup I, j \in \{p\} \cup E, 0$ otherwise.
Then, the previous model can be rewritten as follows:

$$\min \sum_{k \in K} (\sum_{i \in I; l \in E} (c_{pi}^k + c_{il}^k + c_{lp}^k + h_{pi}^k) x_{il}^k + \sum_{i \in I} (c_{pi}^k + c_{ip}^k + h_{pi}^k) x_{ip}^k + \sum_{l \in E} (c_{pl}^k + c_{lp}^k + h_{pl}^k) x_{pl}^k)) \tag{4.18}$$

$$x_{pi}^k - \sum_{l \in \{p\} \cup E} x_{il}^k = 0 \qquad \forall i \in I, \forall k \in K \tag{4.19}$$

$$\sum_{j \in \{p\} \cup I} x_{jl}^k - x_{lp}^k = 0 \qquad \forall l \in E, \forall k \in K \tag{4.20}$$

$$\sum_{j \in I \cup E} x_{jp}^k - \sum_{i \in I \cup E} x_{pi}^k = 0 \qquad \forall k \in K \tag{4.21}$$

$$\sum_{i \in I \cup E} x_{pi}^k \leq 1 \qquad \forall k \in K \tag{4.22}$$

$$\sum_{k \in K} x_{pi}^k = d_i \qquad \forall i \in I \tag{4.23}$$

$$\sum_{k \in K} x_{lp}^k = d_l \qquad \forall l \in E \tag{4.24}$$

$$x_{pj}^k, x_{ip}^k, x_{ij}^k \in \{0, 1\} \qquad \forall i \in I, \forall j \in E, \forall k \in K \tag{4.25}$$

It is worth noting that (4.18) has a lower number of variables as opposed to (4.1). Generally speaking, there are three types of routes:

- routes with one importer and one exporter, which are characterized by the connections from the importer to the exporter, i.e. by variable x_{il}^k, with $i \in I, l \in E$
- routes with one importer, which are characterized by the connections from the importer to the port, i.e. by variable x_{ip}^k, with $i \in I$
- routes with one exporter, which are characterized by the connections from the port to the exporter, i.e. by variable x_{pl}^k, with $l \in E$

Constraints (4.19) and (4.20) enforce the flow conservation of vehicles at each importer and exporter, respectively. Constraints (4.21) are flow conservation of trucks

in the port, constraints (4.22) guarantee that trucks are not used more than once. Constraints (4.23) and (4.24) guarantee to service the requested number of containers for each importer and exporter, respectively. Constraints (4.25) describe the domain of decision variables.

Considering the constraints (4.19) and (4.20) we obtain:

$$x^k_{pi} = \sum_{l \in \{p\} \cup E} x^k_{il} \qquad \forall i \in I, \forall k \in K$$

$$x^k_{lp} = \sum_{j \in \{p\} \cup I} x^k_{jl} \qquad \forall l \in E, \forall k \in K$$

Now, we replace variables x^k_{pi} and x^k_{lp} into constraints (4.21), (4.22), (4.23), (4.24). Constraint (4.21) results be an identity:

$$\sum_{j \in I} x^k_{jp} + \sum_{j \in E} x^k_{jp} - \sum_{i \in I} x^k_{pi} - \sum_{i \in E} x^k_{pi} = 0 \qquad \forall k \in K$$

$$\sum_{j \in I} x^k_{jp} + \sum_{i \in E} \sum_{j \in \{p\} \cup I} x^k_{ji} - \sum_{i \in I} \sum_{l \in \{p\} \cup E} x^k_{il} - \sum_{i \in E} x^k_{pi} = 0 \qquad \forall k \in K$$

$$\sum_{j \in I} x^k_{jp} + \sum_{i \in E} x^k_{pi} + \sum_{i \in E} \sum_{j \in I} x^k_{ji} - \sum_{i \in I} x^k_{ip} - \sum_{i \in I} \sum_{l \in E} x^k_{il} - \sum_{i \in E} x^k_{pi} = 0$$

$$\forall k \in K$$

$$\sum_{i \in E} \sum_{j \in I} x^k_{ji} - \sum_{i \in I} \sum_{l \in E} x^k_{il} = 0 \qquad \forall k \in K$$

Therefore, the first reformulation can be rewritten as follows in the so-called second reformulation:

$$\min \sum_{k \in K} \Big(\sum_{i \in I; l \in E} (c^k_{pi} + c^k_{il} + c^k_{lp} + h^k_{pi}) x^k_{il} + \sum_{i \in I} (c^k_{pi} + c^k_{ip} + h^k_{pi}) x^k_{ip} + \sum_{l \in E} (c^k_{pl} + c^k_{lp} + h^k_{pl}) x^k_{pl} \Big) \tag{4.26}$$

$$\sum_{k \in K} (x^k_{ip} + \sum_{l \in E} x^k_{il}) = d_i \qquad \forall i \in I \tag{4.27}$$

$$\sum_{k \in K} (x^k_{pl} + \sum_{j \in I} x^k_{jl}) = d_l \qquad \forall l \in E \tag{4.28}$$

$$\sum_{i \in I} \sum_{l \in E} x^k_{il} + \sum_{i \in I} x^k_{ip} + \sum_{i \in E} x^k_{pi} \leq 1 \qquad \forall k \in K \tag{4.29}$$

$$x^k_{ij} \in \{0, 1\} \tag{4.30}$$

The objective function (4.26) is equal to (4.18). Constraints (4.27) and (4.28) enforce demand satisfaction for each importer and exporter, respectively. Constraints (4.29) guarantee that trucks are not used more than once. Constraints (4.30) describe the domain of decision variables.

4.4.2 Case with one-container per truck and homogeneous vehicles (third reformulation)

If routing costs are equal for each truck, we can modify the definition of decision variables as follows:

$t_{ij} = \sum_{k \in K} x_{ij}^k$: number of trucks moved along arc $(i, j) \in A$

We redefine costs accordingly:

c_{ij} routing cost of arc $(i, j) \in A$

h_{pi} handling cost for each container put on a truck at port p to serve customer i.

Therefore, in the case of one-container per truck and homogeneous vehicles, the previous model can be rewritten as follows:

$$\min\Big(\sum_{i \in I; l \in E} (c_{pi} + c_{il} + c_{lp} + h_{pi}) t_{il} + \sum_{i \in I} (c_{pi} + c_{ip} + h_{pi}) t_{ip} + \sum_{l \in E} (c_{pl} + c_{lp} + h_{pl}) t_{pl} \Big) \tag{4.31}$$

$$t_{ip} + \sum_{l \in E} t_{il} = d_i \qquad \forall i \in I \tag{4.32}$$

$$t_{pl} + \sum_{j \in I} t_{jl} = d_l \qquad \forall l \in E \tag{4.33}$$

$$\sum_{i \in I} \sum_{l \in E} t_{il} + \sum_{i \in I} t_{ip} + \sum_{i \in E} t_{pi} \leq |K| \tag{4.34}$$

$$t_{ij} \in \mathbb{Z} \tag{4.35}$$

The objective function (4.31) minimize routing and handling costs. Constraints (4.32) and (4.33) enforce demand satisfaction at each importer and exporter, respectively; constraint (4.34) guarantees that the number of routes is not larger than the number of available vehicles; constraints (4.35) describe the domain of decision variables.

4.5 Algebraic properties of the new formulations

In order to investigate if the matrix constraint of the model (4.32)-(4.35) is TU, its representation is reported in Table 4.2, where n, m are the cardinalities of sets I, E, respectively. Therefore, lines from i_1 to i_n are associated with constraints (4.32), lines from j_1 to j_m with constraints (4.33), the last line with constraints (4.34). The

	i_1j_1	i_1j_2	...	i_2j_1	i_2j_2	...	i_nj_m	i_1p	i_2p	...	i_np	pj_1	pj_2	...	pj_m
i_1	1	1	...	0	0	...	0	1	0	...	0	0	0	...	0
i_2	0	0	...	1	1	...	0	0	1	...	0	0	0	...	0
⋮															
i_n	0	0	...	0	0	...	1	0	0	...	1	0	0	...	0
j_1	1	0	...	1	0	...	...	0	0	...	0	1	0	...	0
j_2	0	1	...	0	1	...	...	0	0	...	0	0	1	...	0
⋮															
j_m	0	0	...	0	0	...	1	0	0	...	0	0	0	...	1
k_1	1	1	...	1	1	...	1	1	1	...	1	1	1	...	1

Table 4.2: Constraint matrix A of the model (4.32)-(4.35)

following theorem represents the main result of this thesis.

Theorem 2. *The constraint matrix of reformulation* (4.32) - (4.35) *is totally unimodular.*

Proof. Let us denote with A the constraint matrix of (4.32)-(4.35). This matrix is depicted in Tab.4.2 as the restriction of matrix M limited to rows indexed by $I \cup J \cup \{k_1\}$, where $I = \{i_1, \ldots, i_n\}$ and $J = \{j_1, \ldots, j_m\}$, and the columns indexed by set $A_1 \cup A_2 \cup A_3$, where $A_1 = \{(i, j, k_1) | i \in I, j \in J\}$, $A_2 = \{(i, p, k_1) | i \in I\}$, $A_3 = \{(p, j, k_1) | j \in J\}$. Let I' and J' be two subsets of I and J respectively.

We will use Theorem 1 to prove that A is totally unimodular. We distinguished two cases.

- Case 1 : The subset of A is indexed by the set $I' \cup J' \cup \{k_1\}$
  We can choose the partition: $C_1 = I' \cup J', C_2 = \{k_1\}$ In Table 4.3 we indicated the sums of the elements of each column. For every column indexed

		A_1	A_2	A_3
	$\sum_{c_1 \in I'} a_{c_1 j}$	1 ∨ 0	1 ∨ 0	0
+	$\sum_{c_1 \in J'} a_{c_1 j}$	1 ∨ 0	0	1 ∨ 0
-	$\sum_{c_2 \in C_2} a_{c_2 j}$	1	1	1
=		1 ∨ 0 ∨ -1	0 ∨ -1	0 ∨ -1

Table 4.3: Case: $C = I' \cup J' \cup \{k_1\}$

by $(i, j, k) \in A_1 \cup A_2 \cup A_3$ the sum of the elements in $I' \cup J'$ may be $2, 1$ or 0. Subtracting the row indexed by k_1, we obtain $1, 0, -1$. Then Theorem 1 applies.

- Case 2: The subset of A is indexed by the set $I' \cup J'$
  We can consider the partition: $C_1 = I', C_2 = J'$

		A_1	A_2	A_3
	$\sum_{c_1 \in C_1} a_{c_1 j}$	1 ∨ 0	1 ∨ 0	0
-	$\sum_{c_2 \in C_2} a_{c_2 j}$	1 ∨ 0	0	1 ∨ 0
=		1 ∨ 0 ∨ -1	0 ∨ -1	0 ∨ -1

Table 4.4: Case: $C = I' \cup J'$

We obtain

$$|\sum_{c_1 \in C_1} a_{c_1 j} - \sum_{c_2 \in C_2} a_{c_2 j}| \leq 1 \qquad \forall j.$$

For every column indexed by $(i, j, k) \in A_1 \cup A_2 \cup A_3$ the sum of the elements in I' may be 1 or 0. Subtracting the row indexed by J', we obtain $1, 0, -1$. Then Theorem 1 applies.

We can conclude that A is totally unimodular. □

Therefore, one may wonder if the assumption of one-container pr truck is necessary and sufficient (without any assumption on vehicle costs)to prove the total unimodularity of the constraint matrix in the formulation (4.26)-(4.30). To answer this question, we consider the representation of the model (4.26)-(4.30) shown in tab 4.5, when we have only two trucks. As in the previous case, lines from i_1 to i_n are associated with constraints (4.27), lines from j_1 to j_m with constraints (4.28), lines k_1, k_2 with constraints (4.29).

	k_1				k_1			k_1			k_2				k_2			k_2		
	$i_1 j_1$	$i_1 j_2$	…	$i_n j_m$	$i_1 p$	…	$i_n p$	$p j_1$	…	$p j_m$	$i_1 j_1$	$i_1 j_2$	…	$i_n j_m$	$i_1 p$	…	$i_n p$	$p j_1$	…	$p j_m$
i_1	1	1	…	0	1	…	0	0	…	0	1	1	…	0	1	…	0	0	…	0
i_2	0	0	…	0	0	…	0	0	…	0	0	0	…	0	0	…	0	0	…	0
⋮																				
i_n	0	0	…	1	0	…	1	0	…	0	0	0	…	1	0	…	1	0	…	0
j_1	1	0	…	…	0	…	0	1	…	0	1	0	…	…	0	…	0	1	…	0
j_2	0	1	…	…	0	…	0	0	…	0	0	1	…	…	0	…	0	0	…	0
⋮																				
j_m	0	0	…	1	0	…	0	0	…	1	0	0	…	1	0	…	0	0	…	1
k_1	1	1	…	1	1	…	1	1	…	1	0	0	…	0	0	…	0	0	…	0
k_2	0	0	…	0	0	…	0	0	…	0	1	1	…	1	1	…	1	1	…	1

Table 4.5: Constraint matrix M in the case $|K| = 2$

More generally, if the cardinality of K is equal to r, the constraint matrix can be represented as follows in Table 4.6. The constraint matrix of this model can be

	k_1							k_1				k_1				$k_2 \ldots$
	i_1j_1	i_1j_2	$\ldots$	i_2j_1	i_2j_2	$\ldots$	i_nj_m	i_1p	i_2p	$\ldots$	i_np	pj_1	pj_2	$\ldots$	pj_m	$\ldots$
i_1	1	1	$\ldots$	0	0	$\ldots$	0	1	0	$\ldots$	0	0	0	$\ldots$	0	$\ldots$
i_2	0	0	$\ldots$	1	1	$\ldots$	0	0	1	$\ldots$	0	0	0	$\ldots$	0	$\ldots$
$\vdots$													$\ldots$			
i_n	0	0	$\ldots$	0	0	$\ldots$	1	0	0	$\ldots$	1	0	0	$\ldots$	0	$\ldots$
j_1	1	0	$\ldots$	1	0	$\ldots$	$\ldots$	0	0	$\ldots$	0	1	0	$\ldots$	0	$\ldots$
j_2	0	1	$\ldots$	0	1	$\ldots$	$\ldots$	0	0	$\ldots$	0	0	1	$\ldots$	0	$\ldots$
$\vdots$													$\ldots$			
j_m	0	0	$\ldots$	0	0	$\ldots$	1	0	0	$\ldots$	0	0	0	$\ldots$	1	$\ldots$
k_1	1	1	$\ldots$	1	1	$\ldots$	1	1	1	$\ldots$	1	1	1	$\ldots$	1	$\ldots$
k_2	0	0	$\ldots$	0	0	$\ldots$	0	0	0	$\ldots$	0	0	0	$\ldots$	0	$\ldots$
$\vdots$				$\vdots$					$\vdots$						$\vdots$	$\ldots$
k_r	0	0	$\ldots$	0	0	$\ldots$	0	0	0	$\ldots$	0	0	0	$\ldots$	0	$\ldots$

Table 4.6: Constraint matrix M in the case $|K| = r$

written with the following blockstructure :

B	B	B	$\ldots$	B
E_1	E_2	E_3	$\ldots$	E_r

Where $B \in \mathbb{R}^{(n+m)\times(n\times m+m+n)}$ and $E_1, ..., E_r \in \mathbb{R}^{r\times(n\times m+m+n)}$.
The part of the matrix in Table 4.6 corresponding to rows indexed by i_1 to j_m and by all the columns associated with a vehicle of set K, represents the block B. The block E_1 is represented by the rows indexed by $k_1, \ldots, k_r$ and by all the columns associated with vehicle k_1. In general, the matrix E_r has all rows equal to zero, except the row k that has all entries equal to 1.
At this stage it is worth recalling that if there exists a subdeterminant of M, different to $0, 1, -1$, M is not TU. We consider the submatrix of M reported in Table 4.7. Its

	$\mathbf{x^{k_1}_{i_1p}}$	$\mathbf{x^{k_1}_{pj_1}}$	$\mathbf{x^{k_2}_{i_1j_1}}$
$\mathbf{i_1}$	1	0	1
$\mathbf{j_1}$	0	1	1
$\mathbf{k_1}$	1	1	0

Table 4.7: A submatrix of M

determinant is -2, then M is not totally unimodular. Hence, vehicles are required to have the same costs to guarantee that the TU of the matrix M.

However, we can prove that also the continuous relaxations of formulation (4.18)-(4.25) and of formulation (4.26)-(4.30) admit an integer optimal solution when the arc costs are equal for all vehicles [31].

Proposition 3. *If the costs c_{ij}^k and h_{pj}^k do not depend on k, then the continuous relaxation of formulation* (4.26)-(4.30) *admits an integer optimal solution.*

Proof. If the costs c_{ij}^k and h_{pj}^k do not depend on the choice of the truck k, it follows that we can define $f_{il} = f_{il}^k$, $f_{pi} = f_{pi}^k$, and $f_{lp} = f_{lp}^k$ for any $k \in K$, where

- $f_{il}^k = (c_{pi}^k + h_{pi}^k) + c_{il}^k + c_{lp}^k$, route $p \to i \to l \to p$, for $i \in I, l \in E, k \in K$;
- $f_{ip}^k = (c_{pi}^k + h_{pi}^k) + c_{ip}^k$, route $p \to i \to p$, for $i \in I, k \in K$;
- $f_{pl}^k = (c_{pl}^k + h_{pl}^k) + c_{lp}^k$, route $p \to l \to p$, for $l \in E, k \in K$.

By Theorem 2, (4.31)-(4.35) admits an integer optimal solution t^* for its relaxation. Associated with t^* there exists a dual optimal solution (λ^*, μ^*) with $\lambda^* \in \mathbb{R}^{|I|+|E|}$ and $\mu^* \in \mathbb{R}$ such that

$$\sum_{i\in I; l\in E} f_{il}t_{il}^* + \sum_{i\in I} f_{ip}t_{ip}^* + \sum_{l\in E} f_{pl}t_{pl}^* = \sum_{i\in I} d_i\lambda_i^* + \sum_{l\in E} d_l\lambda_l^* + |K|\mu^*, \tag{4.36}$$

$$f_{il} - \lambda_i^* - \lambda_l^* - \mu^* \geq 0 \qquad i \in I, l \in E, \tag{4.37}$$

$$f_{ip} - \lambda_i^* - \mu^* \geq 0 \qquad i \in I, \tag{4.38}$$

$$f_{pl} - \lambda_l^* - \mu^* \geq 0 \qquad l \in E, \tag{4.39}$$

where condition (4.36) states that primal objective function value of (4.31)-(4.35) is equal to its dual objective function value, and (4.37)-(4.39) are the dual constraints for variables t_{il}, t_{ip}, t_{pl}, $i \in I$, $l \in E$, respectively.

Then we can define a solution $\tilde{x}$ and a corresponding dual solution for (4.26)-(4.30) as follows:

- for each variable $t_{il}^* > 0$ ($t_{ip}^* > 0$, $t_{pl}^* > 0$) define a set of vehicles K_{il} (K_{ip}, K_{pl}, respectively) such that $|K_{il}| = t_{il}^*$ ($|K_{ip}| = t_{ip}^*$, $|K_{pl}| = t_{pl}^*$, respectively); each vehicle $k \in K$ belongs to at most one set in the family $\mathcal{F}_K = \{K_{il} | i \in I, l \in E\} \cup \{K_{ip} | i \in I\} \cup \{K_{pl} | l \in E\}$;
- set $\tilde{x}_{il}^k = 1$ for $k \in K_{il}$, $\tilde{x}_{ip}^k = 1$ for $k \in K_{ip}$, $\tilde{x}_{pl}^k = 1$ for $k \in K_{pl}$, and $\tilde{x}_{il}^k = 0, \tilde{x}_{ip}^k = 0, \tilde{x}_{pl}^k = 0$, otherwise;
- define the dual solution for the relaxation of (4.26)-(4.30) as $(\tilde{\lambda}, \tilde{\mu}) \in \mathbb{R}^{|I|+|E|+|K|}$ such that $\tilde{\lambda} = \lambda^*$ and $\tilde{\mu}^k = \mu^*$ for each $k \in K$.

We check that $(\tilde{x}, \tilde{\lambda}, \tilde{\mu})$ satisfies primal and dual constraints for (4.26)-(4.30) and attains the same objective function value for the relaxation of (4.26)-(4.30) and for its dual, respectively. Primal feasibility for (4.27) and (4.28) is straightforward because $\sum_{k\in K} \tilde{x}_{il}^k = t_{il}^*$, $\sum_{k\in K} \tilde{x}_{ip}^k = t_{ip}^*$, $\sum_{k\in K} \tilde{x}_{pl}^k = t_{pl}^*$; primal feasibiliy for (4.29) comes from the condition that the sets K_{il}, K_{ip}, K_{pl} are pairwise disjoint for all

$i \in I, l \in E$, so at most one variable in (4.29) takes value 1. The dual constraints of the continuous relaxation (4.26)-(4.30) are the following:

$$\begin{aligned} f^k_{il} - \tilde{\lambda}_i - \tilde{\lambda}_l - \tilde{\mu}^k \geq 0 \qquad & i \in I, l \in E, k \in K, \\ f^k_{ip} - \tilde{\lambda}_i - \tilde{\mu}^k \geq 0 \qquad & i \in I, k \in K, \\ f^k_{pl} - \tilde{\lambda}_l - \tilde{\mu}^k \geq 0 \qquad & l \in E, k \in K. \end{aligned}$$

Dual feasibility of $(\tilde{\lambda}, \tilde{\mu})$ comes from definition of $\tilde{\lambda}$ and $\tilde{\mu}^k$ and from conditions (4.37)-(4.39). Finally, we check that with respect to reformulation (4.26)-(4.30) the dual objective value for $(\tilde{\lambda}, \tilde{\mu})$ is equal to its primal objective value for $\tilde{x}$:

$$\begin{aligned} & \textstyle\sum_{k \in K} \tilde{\mu}^k + \sum_{i \in I} \tilde{\lambda}_i d_i + \sum_{l \in E} \tilde{\lambda}_l d_l = \\ = \; & \mu^* |K| + \textstyle\sum_{i \in I} \lambda^*_i d_i + \sum_{l \in E} \lambda^*_l d_l = \\ = \; & \textstyle\sum_{i \in I; l \in E} f_{il} t^*_{il} + \sum_{i \in I} f_{ip} t^*_{ip} + \sum_{l \in E} f_{pl} t^*_{pl} = \\ = \; & \textstyle\sum_{i \in I; l \in E} f_{il} \sum_{k \in K} \tilde{x}^k_{il} + \sum_{i \in I} f_{ip} \sum_{k \in K} \tilde{x}^k_{ip} + \sum_{l \in E} f_{pl} \sum_{k \in K} \tilde{x}^k_{pl}, \end{aligned}$$

and as consequence $\tilde{x}$ and $(\tilde{\lambda}, \tilde{\mu})$ are a pair of primal-dual optimal solutions for the relaxation of (4.26)-(4.30) and $\tilde{x}$ is integer by construction. □

An easy consequence of Proposition 3 is that also the continuous relaxation of reformulation (4.18)-(4.25) admits an integer optimal solution when the arc costs do not depend on the vehicle. Indeed, it is sufficient to first determine the solution for (4.26)-(4.30) and then set the variables x^k_{pi} and x^k_{lp} by using constraints (4.19) and (4.20). This can be summarized in the following corollary:

Corollary 1. *If the costs c^k_{ij} and h^k_{pj} do not depend on k, then the continuous relaxation of reformulation* (4.18)-(4.25) *admits an integer optimal solution.*

Chapter 5

Conclusions and Research Developments

Total unimodularity is a desired property for anyone in dealing with the determination of exact solutions of integer programming problems. When the constraint matrix is TU, the feasible region is equal to the convex hull, thus one can ignore the integrality constraint and solve the associated linear programming relaxation, because integral solutions are always obtained in this case. As a consequence, TU allows to optimally solve integer problems by effective linear programming solvers. This thesis was motivated by a conjecture on total unimodularity of the constraint matrix in the formulation of [24], when it is run under specific experimental conditions. To clarify, when all vehicles are supposed to carry one container and have the same costs, the linear programming relaxation was observed to be integer. This thesis aims to provide a theoretical understanding of these results through the analysis of the algebraic properties of this model. Three reformulations are proposed for this model in order to determine necessary and sufficient conditions for TU. The contributions of this thesis can be summarized as follows:

- Carrying one container per truck is a necessary condition for TU of the constraint matrix in [24]. However, in this case the problem is reformulated by a mathematical model (the second reformulation), where TU does not hold. Then, also the constraint matrices of the first and the second reformulations are not TU.
- When all vehicles have the same costs and carry one container each, the problem in [24] can be reformulated by a mathematical model (the third reformulation) where TU holds.
- Even in the first and the second reformulations the optimal solutions of the linear programming relaxation is integer, when all vehicles have the same costs.

At the moment, several research developments are in progress. Firstly, it is of interest to prove if the continuous relaxation of the formulation in [24] admits an

integral optimal solution when all trucks carry one container and have the same costs. Secondly, it will be of interest to reformulate the problem as a equivalent circulation problem, that can be solved by effective network flows algorithms. Thirdly, more complex problem settings will be investigated to account for more realistic attributes of drayage problems, such as multiple trips for each vehicle and planning horizons with several days.

Silvia Schirra gratefully acknowledges Sardinia Regional Government for the financial support of her PhD scholarship (P.O.R. Sardegna F.S.E. Operational Programme of the Autonomous Region of Sardinia, European Social Fund 2007-2013 - Axis IV Human Resources, Objective l.3, Line of Activity l.3.1.)

Bibliography

[1] Bazaraa, M. S., Jarvis, J. J., and Sherali, H. D. *Linear programming and network flows*. John Wiley & Sons, 2011.

[2] Berbeglia, G., Cordeau, J.-F., Gribkovskaia, I., and Laporte, G. Static pickup and delivery problems: a classification scheme and survey. *Top 15*, 1 (2007), 1–31.

[3] Braekers, K., Caris, A., and Janssens, G. Integrated planning of loaded and empty container movements. *OR Spectrum 35*, 2 (2013), 457–478.

[4] Braekers, K., Caris, A., and Janssens, G. K. Bi-objective optimization of drayage operations in the service area of intermodal terminals. *Transportation Research Part E: Logistics and Transportation Review 65* (2014), 50–69.

[5] Caris, A., and Janssens, G. A local search heuristic for the pre- and end-haulage of intermodal container terminals. *Computers and Operations Research 36*, 10 (2009), 2763–2772.

[6] Chandrasekaran, R. Total unimodularity of matrices. *SIAM Journal on Applied Mathematics 17*, 6 (1969), 1032–1034.

[7] Cheung, R., Shi, N., Powell, W., and Simao, H. An attribute–decision model for cross-border drayage problem. *TransportationnResearch Part E: Logistics and Transportation Review 44*, 2 (2008), 217–234.

[8] Chung, K., Ko, C., Shin, J., Hwang, H., and Kim, K. Development of mathematical models for the container road transportation in korean trucking industries. *Computers & Industrial Engineering 53*, 2 (2007), 252–262.

[9] Clarke, G., and Wright, J. W. Scheduling of vehicles from a central depot to a number of delivery points. *Operations research 12*, 4 (1964), 568–581.

[10] Dakin, R. J. A tree-search algorithm for mixed integer programming problems. *The computer journal 8*, 3 (1965), 250–255.

[11] Dantzig, G. B. Programming of interdependent activities: Ii mathematical model. *Econometrica, Journal of the Econometric Society* (1949), 200–211.

[12] Deidda, L., Di Francesco, M., Olivo, A., and Zuddas, P. Implementing the street-turn strategy by an optimization model. *Maritime Policy & Management 35* (2008), 503–516.

[13] Francis, P., Zhang, G., and Smilowitz, K. Improved modeling and solution methods for the multi-resource routing problem. *European Journal of Operational Research 180* (2007), 1045–1059.

[14] Funke, J., and Kopfer, H. A model for a multi-size inland container transportation problem. *Transportation Research Part E: Logistics and Transportation Review 89* (2016), 70–85.

[15] Ghouila-Houri, A. Caractrisation des matrices totalement unimodulaires. *Comptes Rendus Hebdomadaires des Scances de lAcadmie des Sciences (Paris) 254* (1962), 1192–1194.

[16] Gronalt, M., Hartl, R., and Reimann, M. New savings based algorithms for time constrained pickup and delivery of full truckloads. *European Journal of Operational Research 151* (2003), 520–535.

[17] Hoffman, A. J., Kruskal, J. B., Kuhn, H., and Tucker, A. Linear inequalities and related systems. *Annals of Mathematics Studies* (1956), 223–246.

[18] Ileri, Y., Bazaraa, M., Gifford, T., Nemhauser, G., Sokol, J., and Wikum, E. An optimization approach for planning daily drayage operations. *Central European Journal of Operations Research 14* (2006), 141–156.

[19] Imai, A., Nishimura, E., and Current, J. A lagrangian relaxation-based heuristic for the vehicle routing with full container load. *European journal of Operational Research 176*, 1 (2007), 87–105.

[20] Jula, H., Chassiakos, A., and Ioannou, P. Port dynamic empty container reuse. *Transportation Research Part E: Logistics and Transportation Review 42* (2006), 43–60.

[21] Jula, H., Dessouky, M., Ioannou, P., and Chassiakos, A. Container movement by trucks in metropolitan networks: modeling and optimization. *Transportation Research Part E: Logistics and Transportation Review 41*, 3 (2005), 235–259.

[22] Lai, M., Battarra, M., Di Francesco, M., and Zuddas, P. An adaptive guidance meta-heuristic for the vehicle routing problem with splits and clustered backhauls. *Journal of the Operational Research Society 66* (2015), 1222–1236.

[23] Lai, M., Battarra, M., Di Francesco, M., and Zuddas, P. An adaptive guidance meta-heuristic for the vehicle routing problem with splits and clustered backhauls. *Journal of the Operational Research Society 66*, 7 (2015), 1222–1235.

[24] Lai, M., Crainic, T. G., Di Francesco, M., and Zuddas, P. An heuristic search for the routing of heterogeneous trucks with single and double container loads. *Transportation Research Part E: Logistics and Transportation Review 56* (2013), 108–118.

[25] Land, A. H., and Doig, A. G. An automatic method of solving discrete programming problems. *Econometrica: Journal of the Econometric Society* (1960), 497–520.

[26] Namboothiri, R., and Erera, A. Planning local container drayage operations given a port access appointment system. *Transportation Research Part E: Logistics and Transportation Review 44*, 2 (2008), 185–202.

[27] Nossack, J., and Pesch, E. A truck scheduling problem arising in intermodal container transportation. *European Journal of Operational Research 230*, 3 (2013), 666–680.

[28] Parragh, S. N., Doerner, K. F., and Hartl, R. F. A survey on pickup and delivery problems. part i: Transportation between customers and depot. *Journal fur Betriebswirtschaft 58*, 1 (2008), 21–51.

[29] Reinhardt, L. B., Pisinger, D., Spoorendonk, S., and Sigurd, M. M. Optimization of the drayage problem using exact methods. *INFOR: Information Systems and Operational Research 54*, 1 (2016), 33–51.

[30] Savelsbergh, M., and Sol, M. The general pickup and delivery problem. *Transportation Science*, 29 (1995), 17–29.

[31] Schirra, S., Di Francesco, M., Gentile, C., Stecca, G., and Zuddas, P. An integral lp relaxation for a drayage problem. *Technical Report.* 2017.

[32] Schrijver, A. *Theory of linear and integer programming.* John Wiley & Sons, 1998.

[33] Sterzik, S., and Kopfer, H. A tabu search heuristic for the inland container transportation problem. *Computers & Operations Research 40*, 4 (2013), 953–962.

[34] Toth, P., and Vigo, D. VRP with backhauls. In *The Vehicle Routing Problem*, P. Toth and D. Vigo, Eds. SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, PA, 2002, pp. 195 – 224.

[35] Vidovic, M., Popovic, D., Ratkovic, B., and Radivojevic, G. Generalized mixed integer and vns heuristic approach to solving the multisize containers drayage problem. *International Transactions in Operational Research* (2016).

[36] Vidovic, M., Radivojevic, G., and Rakovic, B. Vehicle routing in containers pickup up and delivery processes. *Procedia-Social and Behavioral Sciences 20* (2011), 335–343.

[37] Wang, W. F., and Yun, W. Y. Scheduling for inland container truck and train transportation. *International journal of Production Economics 143* (2013), 349–356.

[38] Wang, X., and Regan, A. C. Local truckload pickup and delivery with hard time window constraints. *Transportation Research Part B: Methodological 36* (2002), 97–112.

[39] Wolsey, L. A., and Nemhauser, G. L. Integer and combinatorial optimization. *John Willey & Sons* (1999).

[40] Xue, Z., Lin, W.-H., Miao, L., and Zhang, C. Local container drayage problem with tractor and trailer operating in separable mode. *Flexible Services & Manufacturing 27* (2015), 431–450.

[41] Xue, Z., Zhang, C., Lin, W.-H., Miao, L., and Yang, P. A tabu search heuristic for the local container drayage problem under a new operation mode. *Transportation Research Part E: Logistics and Transportation Review 62* (2014), 136–150.

[42] Zhang, R., Lu, J., and Wang, D. Container drayage problem with flexible orders and its near real-time solution strategies. *Transportation Research Part E: Logistics and Transportation Review 61* (2014), 235–251.

[43] Zhang, R., Yun, W., and Kopfer, H. Heuristic-based truck scheduling for inland container transportation. *OR Spectrum 32*, 3 (2010), 787–808.

[44] Zhang, R., Yun, W., and Kopfer, H. Multi-size container transportation by truck: modeling and optimization. *Flexible Services & Manufacturing 2-3*, 27 (2015), 403–430.

[45] Zhang, R., Yun, W., and Moon, I. A reactive tabu search algorithm for the multi-depot container truck transportation problem. *Transportation Research Part E: Logistics and Transportation Review 45*, 6 (2009), 904–914.

[46] Zhang, R., Yun, W., and Moon, I. Modeling and optimization of a container drayage problem with resource constraints. *International journal of Production Economics 133*, 1 (2011), 351–359.