UNIVERSITY OF CAGLIARI

Ph.D. School in Mathematics and Computer Science
Ph.D. Course in Computer Science – XXV cycle

Doctoral Thesis
Scientific fields
INF/01
MAT/09

Models and algorithms for the empty container repositioning and its integration with routing problems

Michela Lai

Advisor
Prof. Paola Zuddas

Co-advisor
Dr. Maria Battarra

May 2013

Dedicated to my family

Thesis introduction

The introduction of containers has fostered intermodal freight transportation. A definition of intermodality was provided by the European Commission as

"a characteristic of a transport system whereby at least two different modes are used in an integrated manner in order to complete a door-to-door transport sequence" EUROPA 2005.

The intermodal container transportation leads to several benefits, such as higher productivity during handling phases and advantages in terms of security, losses and damages. However, the distribution of containers comes with a drawback: due to directional imbalances in freight flows, some areas tend to accumulate unnecessary empty containers, while others face container shortages. For example, in 2007 the Trans-Pacific trade container flow from Asia to North America was estimated at 14.5 million TEUs (Twenty foot Equivalent Unit), while the opposite direction only had 5.6 million TEUs "Review of maritime transport, United Nations, New York, 2008". As a consequence, carries were requested to reposition about 9 million TEUs from North America to Asia, in order to meet future transportation opportunities.

Several planning models were developed for carriers in order to manage both loaded and empty containers profitably, Crainic 2003. However, they were built to operate under normal circumstances, neglecting the fact that networks are increasingly affected by both uncertainty and vulnerability, which may result in disruptions. These disruptions affect the normal operations of facilities, and make them partially or totally non-functional. Disruptions may be generated by unexpected events, due to both natural causes (hurricanes, earthquakes, etc.) and human-generated ones (accidents, failures, etc.). To make matters worse, widely adopted "just-in-time" supply chains make manufacturers vulnerable to disruptions if containers do not reach assembly plants in time. "The lack of key parts could reduce output, employment, and income for individual companies by amounts larger than the value of the delayed part and in areas and businesses far removed from the facility where a disruption occurred" Office 2006.

Therefore, it is not enough for carriers to serve customers while operating at the lowest costs, because unexpected future evolutions cannot be ignored at all, even if they will not be observed. Proactive strategies should be adopted to become

more resilient, i.e., deploying contingency plans for smoothly continuing operations whenever disruptions occur. Considering a hurricane for example, a forecast may preventively stop all activities of a port, but other forecasts may not exclude weather improvements which allow the port to continue its activities. Therefore, both disruptions and normal operations represent possible futures, and there is no certainty about which future scenario will occur.

In Part I the maritime repositioning of empty containers is described and modelled, in order to evaluate the effects of uncertainty on this problem. The objective is to survey whether the impact of uncertainty can be mitigated by a stochastic programming approach, in which disruptions and normal operations are both foreseen as possible futures or scenarios. This approach is carried out by a multi-scenario optimization model in which scenarios are linked by non-anticipativity conditions, which enforce here-and-now decisions to be identical over all scenarios.

The empty container repositioning becomes even more challenging and difficult when integrated with routing problems. In fact, carriers often face problems in which they must determine simultaneously how many empty containers are carried by a fleet of vehicles and which routes must be followed by these vehicles. These problems typically arise in inland networks, in which one must plan the distribution by trucks of loaded and empty containers to customers.

Part II and III address this type of vehicle routing problems, which are motivated by a real case study occurred during the collaboration with a carrier that operates in the Mediterranean Sea in door-to-door modality. The carrier manages a fleet of trucks based at the port. Trucks and containers are used to service two types of transportation requests, the delivery of container loads from the port to import customers, and the shipment of container loads from export customers to the port.

Part II addresses the problem involving a heterogeneous fleet of trucks that can carry one or two containers. We present a Vehicle Routing Problem with backhauls, load splits into multiple visits, and the impossibility to separate trucks and containers during customer service. Then, we formalize the problem by an Integer Linear Programming formulation and propose an efficient meta-heuristic algorithm able to solve it. The meta-heuristic determines the initial solution by a variant of the Clarke-and-Wright algorithm, and improves it by several local search phases, in which both node movements and truck swaps are implemented.

Part III addresses the problem involving a homogeneous fleet of trucks that can carry more than a container per truck. As a consequence, the identification of routes can be more difficult. We present a Vehicle Routing Problem with backhauls, load splits into multiple visits, the opportunity to carry more than a containers per truck, and the impossibility to separate trucks and containers during customer service. Then, we formalize the problem by an Integer Linear Programming formulation and

propose an efficient meta-heuristic algorithm able to solve it. The meta-heuristic determines an initial feasible solution by a Tabu Search step, and next improves this solution by appropriate adaptive guidance mechanisms.

Contents

List of Figures

List of Tables

Part I

Seaside

Chapter 1

Problem description

Ports are intermodal transit facilities for full and empty containers. Full containers are transported from departure ports to arrival ports according to the requests of customers. Due to trade imbalances in their flows, import-dominant ports tend to accumulate unnecessary empty containers, while export-dominant ones face container shortages. Therefore, shipping companies must periodically reposition empty containers between ports.

The empty container surpluses and shortages in ports can be expressed in terms of supply and demand. The supply can be defined as the number of empty containers available in a given port at any given time. These containers can be stored to meet future requests from the landside or moved to other ports. The demand can be defined as the number of empty containers requested in a given port at any given time. The demand in a port must be met by empty containers in stock in the same port or moved there from other ports.

The empty container repositioning is performed by vessels sailing well-established shipping lines according to tight schedules, which specify arrival and departure times at ports. Vessels carry both full and empty containers. The paths of full containers from departure ports to arrival ports (through possible transshipment ports) are supposed known to shipping companies, in fact these data are available from the moment they accept the transportation requests of customers. Customers typically book the transportation of full containers several days before the arrival of vessels at the departure ports. Despite what happens with full containers, empty containers have no fixed origin-destination pairs. As explained by Song and Dong 2011a, "this type of practice tends to be more flexible to handle the uncertain and dynamic events, since part of the repositioning decisions can be done when more reliable information becomes available".

The empty container repositioning problem aims to pick up unused empty containers from ports where they are in surplus and providing them in ports

where they are in shortage. Therefore, shipping companies must decide how many empty containers to load/unload to/from vessels, and how many empty containers transport in the residual transportation capacity (left by full containers). An optimal repositioning plan minimizes both operational costs and shortage costs.

In this problem setting, decisions must be taken one day before the arrival of vessels. This leads ports to organize their internal activities on time, in order to implemented the decisions and serve vessels as requested. Moreover, shipping companies want to decide the overall number of empty containers loaded and unloaded from vessels during the whole time spent in port in their berthing. These decisions must be taken one day before the arrival of vessels, because ports need to timely plan their internal activities. In other words, when a vessel spends more than one day in a port, decisions do not concern the number of empty containers loaded and unloaded in each day during its berthing, decisions concern the overall number of empty containers loaded or unloaded, leading ports to plan their internal activities properly in order to serve all requests. The exact time in which containers are loaded or unloaded will be determined in a different issue, which is called *Quay Crane Scheduling Problem*, Meisel and Bierwirth 2011. Moreover, since the time spent by vessels in ports is limited, the number of empty containers loaded or unloaded from vessels is limited as well.

Usually, in the real world, when empty containers are unloaded from a vessel, they are moved to temporary areas, where they are kept while loading and discharging operations are performed on that vessel. Next, when the vessel leaves, empty containers are moved to dedicated areas for their storage. Similarly, when empty containers must be shipped, they are picked from the dedicated storage areas at least one day before the arrival of vessels, moved to temporary areas and loaded onto vessels. Therefore, shipping companies must determine the number of empty containers stored in ports taking into account the space available in the dedicated storage areas.

It is important to note that demands in a port may not be completely satisfied, especially when: *(i)* the supply of empty containers over all ports is lower than the overall demand; *(ii)* storage, transportation and handling capacities limit the number of empty containers repositioned. Therefore, it is crucial for shipping companies to determine how many empty containers lack until the full demand satisfaction. This information helps shipping companies to take, and perform if this is the case, corrective actions, such as renting containers from leasing companies.

Decisions on empty container repositioning are taken daily, even if data are affected by uncertainty. As an example, the uncertainty may concern the empty container demand arising from unexpected transportation opportunities. Generally speaking, the uncertainty leads to multiple possible future evolutions. Each possible

future evolution have same characteristics for certain data, and changes can be represented by different values for the uncertain parameters (for example the number of empty containers requested in a port). Therefore, all possible future evolutions can be described in different scenarios, each one representing all the characteristics of the possible future evolution described. In this study the maritime repositioning of empty containers is modelled in order to evaluate the effects of the uncertainty. Two formulations will be proposed and evaluated in terms of demand satisfaction and operating costs: *(i)* a deterministic model with a single scenario; *(ii)* a multi-scenario model in which multiple scenarios are are linked by non-anticipativity conditions, which force identical decisions for all scenarios in the first periods.

The objective is to survey whether the impact of uncertainty can be mitigated by a stochastic programming approach, in which disruptions and normal operations are both foreseen as possible futures or scenarios. The two formulations, deterministic and multi-scenario, will be proposed in Chapter 2. Numerical experiments are carried out in Chapter 3 to show the value of the multi-scenario formulation in this problem, and its computational viability. Finally, conclusions are summarized in Chapter 4.

1.1 Literature review

Many studies have been done on empty container repositioning under stochastic demand. Research on empty containers was carried out by both dynamic programming and integer programming. The first approach aims at examining the characteristics of optimal control policies for empty containers in shipping networks, which typically have specific topological structures. Lai, Lam, and Chan 1995 evaluated several allocation policies to reduce operational costs and prevent empty container shortage when their demand is uncertain. Li et al. 2004 investigated a single port with a discrete time system and stochastic demand. Their research was extended later to a multi-port system Li et al. 2007. Lam, Lee, and Tang 2007 studied a two-ports two voyages system, Song and Earl 2008 investigated a two depot system and Song and Zhang 2010 worked on a single port in a continuous-time system. Specific work on liner shipping systems was carried out by Song and Dong 2011a on both fleet sizing and empty container repositioning under uncertain demand.

This thesis belongs to the line of integer programming formulations for the management of empty containers and extends the body of literature in the field, because it is investigated under the assumption of uncertain port disruptions. Many papers in the field assume perfect knowledge on future system evolution and propose deterministic formulations, Crainic 2003. This is not to say that they

ignore uncertainty at all. They typically work with forecasted data and use models in a rolling horizon fashion. Deterministic models on empty container management have been proposed by Shen and Khoong 1995, Choong, Cole, and Kutanoglu 2002, Erera, Morales, and Savelsbergh 2005, Olivo et al. 2005, Feng and Chang 2008, Song and Carter 2009, Moon, Do Ngoc, and Hur 2010 and Song and Dong 2011b. However, these formulations may not be appropriate when dissimilar futures might occur and preventive measures must be adopted to deal with possible disruptions.

An alternative approach to address the problem at hand is the Adjustable Robust Optimization, Ben-Tal et al. 2004. A robust plan is defined as one in which constraints are satisfied for the nominal values and feasibility can be re-established for any outcome by recovery actions. Erera, Morales, and Savelsbergh 2009 adopted this approach for the maritime repositioning of empty containers. In their setting, all demands must be met and larger-than-expected demands are faced by keeping safety stocks and moving additional containers on vessels. However, there may be some limits to the adjustment generated by recovery actions, in the case of limited capacities in storage and transportation activities. Hence, sometimes only a part of the empty container demand can be satisfied.

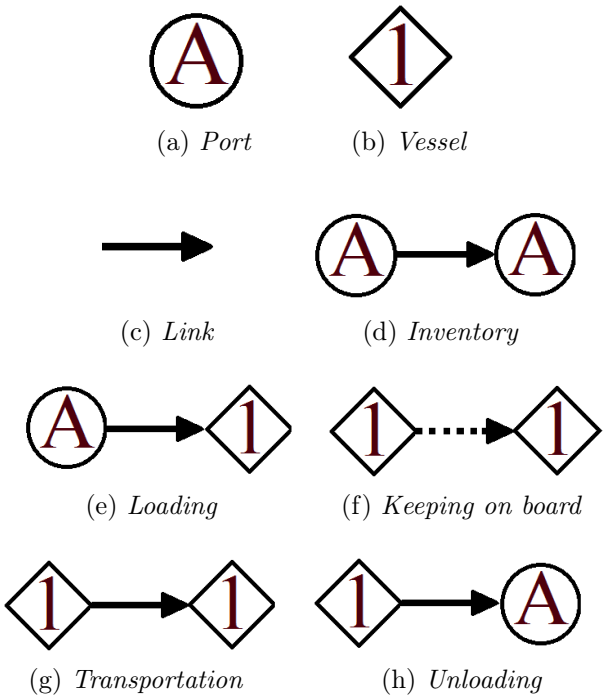
The approach presented in this study is in the field of stochastic integer programming Powell and Topaloglu 2003, whose goal is to find a policy optimizing the expected value of a function depending on random variables. Although this approach requires a good knowledge of the distributions of uncertain parameters, it can be adopted even for this problem. The different scenarios represent the different future evolutions and subjective weights can be used to characterize their relative importance. Non-anticipativity conditions guarantee that decisions do not depend on which scenario will occur, because this information is not yet available.

Stochastic programming for empty container management has already been used to account for the uncertainty of demand and transportation capacities. Crainic, Gendreau, and Dejax 1993 proposed a two-stage stochastic model with partial network recourse for inland repositioning problems. Specific papers on maritime networks have been proposed by Cheung and Chen 1998, Leung and Wu 2004 and Di Francesco, Crainic, and Zuddas 2009. This is, to the best of our knowledge, the first paper using stochastic programming to address empty container repositioning under port disruptions.

Chapter 2

Modeling

The elementary components of the system are ports and vessels. Ports are represented by circular shapes and are denoted by letters, as in Figure 2.1a. Vessels are represented by rhombus shapes and are denoted by numbers, as in Figure 2.1b. All elementary components are connected to each other by links, represented by arrows as in Figure 2.1c, that concern possible decisions for the shipping company.



(a) *Port* (b) *Vessel*

(c) *Link* (d) *Inventory*

(e) *Loading* (f) *Keeping on board*

(g) *Transportation* (h) *Unloading*

Figure 2.1: Components of the system.

The temporal horizon, with respect to which the planning horizon is referred, is divided in several periods. Periods can represent hours, days, weeks, and so on, depend on the company needs. Moreover, decisions refer to periods in which they are taken, and may have effects in other periods. Therefore, the connection

between elementary components lead to a multi-period complex system, in which are represented all dynamics.

An example of a system can be observed in Figure 2.2. The example is made up of five ports, two vessels and a time horizon of five periods. Periods are denoted by numbers from 1 to 5. Ports are replicated in every period of the planning horizon and are represented by circular shape, denoted by letters from A to E. Vessels are represented by rhombus shape above ports in all periods of their berth, denoted by numbers 1 and 2.

The set of decisions involved in Figure 2.2 concerns:

Inventory Number of empty containers located in the port between two periods. These decisions are represented by arrows that have origin in a port in a period, and destination in the same port in the next period. Figure 2.1d;

Loading Number of empty containers loaded on a vessel during its berthing. These decisions are represented by arrows that have origin in a port in a period, and destination in a vessel that is berthing in that port and in the same period. Figure 2.1e;

Keeping on board Number of empty containers kept on board on a vessel during its berthing.
These decisions are represented by arrows that have origin in a vessel, that is berthing in a port, and destination in the same vessel during the same berthing (in the same port and in the next period). Figure 2.1f;

Transportation Number of empty containers carried by a vessel between two berthing.
These decisions are represented by arrows that have origin in a vessel, that is berthing in a port, and destination in the same vessel on next berthing (in a different port and in a following period, at least the next one). Figure 2.1g;

Unloading Number of empty containers unloaded from a vessel during its berthing. These decisions are represented by arrows that ave origin in the vessel berthing in a port, and destination in the same port and in the same period. Figure 2.1h.

In Figure 2.2 there are two shipping lines, line 1 and line 2. The vessel deployed on line 1 arrives in port B at period 1, in port C at period 2, in port A at period 3, next in port B at period 5, and so on, repeating its schedule from this point onwards. The vessel deployed on line 2 arrives in port D at period 2, in port C at period 3, in port E from period 4 up to period 5, and so on, repeating its schedule from this point onwards. Loading and unloading connections represent the number

of empty containers loaded and unloaded from vessels in each period of their berth, as in Cheung and Chen 1998.

According to the classical notation of the graph theory Ahuja, Magnanti, and Orlin 1993: *external arrows* entering and leaving the port represent respectively the supply and the request of empty containers in that port. Moreover, A *"super-sink"* node is introduced at the end of the planning horizon in order to collect empty containers that have their destination port beyond the last period.

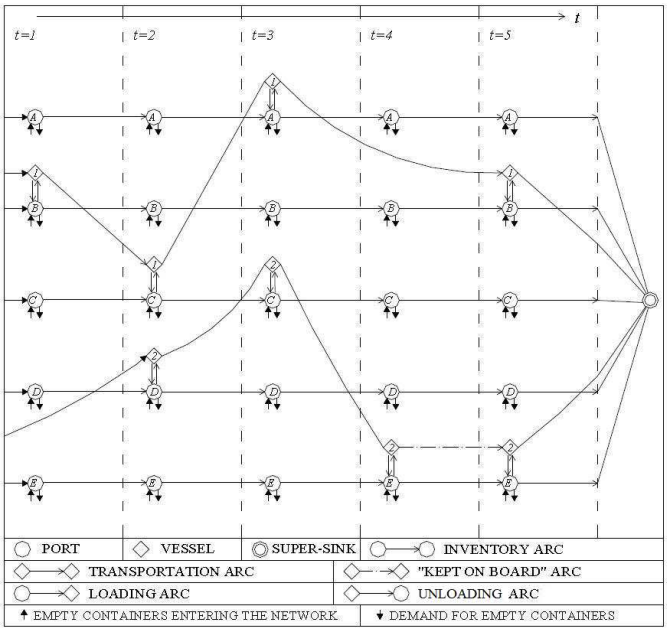


Figure 2.2: An example of system.

To represent the whole system dynamics, we adopt the following choices: *(i)* vessels are represented above the port only in the arrival period of their berthing; *(ii) keeping on board* arrows are not taken into account; *(iii)* unloading arrows have origin in the vessel, in the arrival period, and destination in the port, in the departure period of the berthing. As a consequence, when unloading arrows have their origin and destination in the same period, the vessel stays in port one period only. If the destination is in the next period, the vessel stays in port two periods, and so on. Moreover, according to the requirement of loading only empty containers available in ports at least one day before the arrival of vessels, an additional choice adopted is the following: *(iv)* loading arcs have destination in the vessel, in the arrival period, and origin in the port, in the previous period of the arrival period of the berthing. It is important to note that even if the decision on how much empty containers load onto vessels is taken one day before their arrival period at ports, empty containers loaded onto vessels use the available storage capacity of the ports

until the arrival period of vessels.

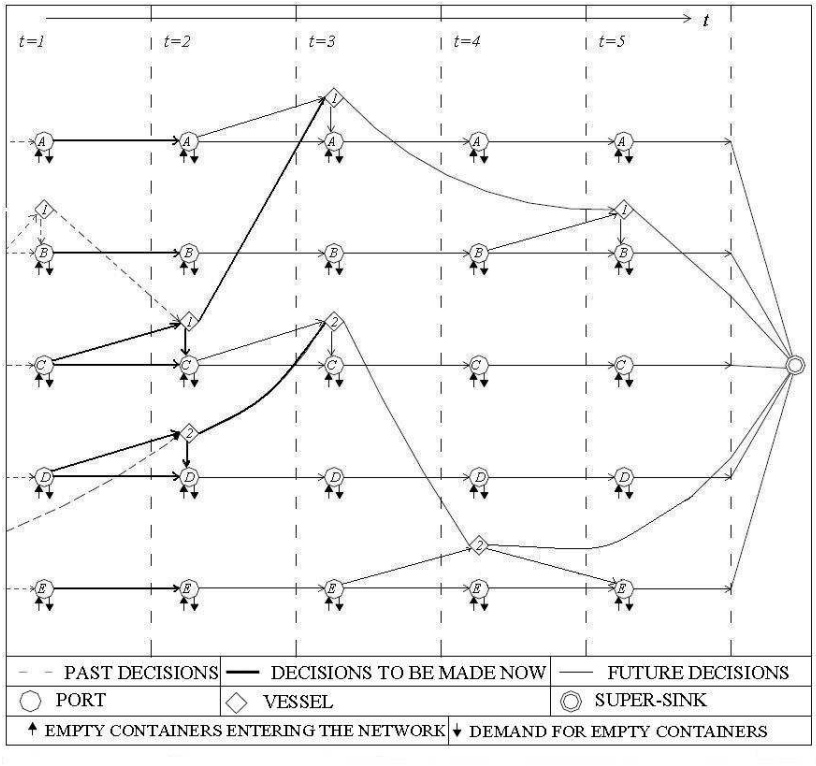


Figure 2.3: The system dynamics.

Figure 2.3 shows an example of the system dynamics made up of five periods, five ports and two vessels. Despite what represented in Figure 2.2 vessels are not replicated in each period of their berth, but are represented only in the periods when they arrive at ports. As a consequence, also loading and unloading connections differ form Figure 2.2, and, even if the vessel berth for more than one period, only one loading connection and one unloading connection are represented for each berth of the vessel.

In Figure 2.3 thick lines represent decisions to take now (*here-and-now decisions*), dotted line represent past decisions and thin lines represent future decisions. The here-and-now decisions of Figure 2.3 concern:

- the number of empty containers loaded onto vessels arriving in period 2;
- the number of empty containers transported by vessels after the departure from ports reached at period 2;
- the number of empty containers unloaded from vessels arriving in ports at period 2;
- the number of empty containers stored between period 1 and period 2 in each port.

In this study, two different formalizations will be introduced for the problem at hand: the deterministic model, to be presented in Section 2.1, and the multi-scenario model to be presented in Section 2.2.

2.1 Deterministic model

This section concerns the formalization of the deterministic model. The deterministic model involves a single scenario, and it is appropriate when data are known with a high level of certainly for all periods of the planning horizon, since the repositioning of empty container is determined for the whole planning horizon.

The deterministic model involves sets, data and parameters. We now present an overview of the notation to understand better the model. All sets, data and parameters are listed following their alphabetical order.

SETS

H**:** Set of ports;

T**:** Set of contiguous time periods, each of which represents a day. The time period index $t \in T$ takes values from 1 to $|T|$. All decisions made at period $t = 1 \in T$ represent the here-and-now decisions;

V**:** Set of vessels;

$V(i,t)$**:** Set of vessels arriving in port $i \in H$ at time $t \in T$;

$W(i,t)$**:** Set of vessels leaving from port $i \in H$ at time $t \in T$.

INPUT DATA

$b_t(i)$**:** Net supply of empty containers in port $i \in H$ at period $t \in T$. Positive value of $b_t(i)$ represents the number of empty containers which become available in port $i \in H$ at time $t \in T$. Nevative value of $b_t(i)$ represents the number of empty containers required in port $i \in H$ at time $t \in T$. If $b_t(i)$ is equal to zero, port $i \in H$ acts as a transshipment node at time $t \in T$;

R_v**:** Schedule of vessel $v \in V$. The schedule consists of a sequence of ordered triplets $< i,t,t^{'} >$, the triplets means that vessel $v \in V$ arrives in port $i \in H$ at period $t \in T$ and stays in this port up to $t^{'} \in T$. The next triplet in the sequence provides analogous data for the following port reached by vessel $v \in V$ after $i \in H$;

$U_{(t+1)v}(i)$**:** Handling capacity. Represents the number of load/unload operations of empty containers that can be successfully executed during the time spent

in port by vessel v arriving at port i at time $t+1$. In this study, the time needed to execute a load operation is assumed to be equal to the time needed to execute an unload operation. Given the time spent in port by the vessel, and the time needed to complete an operation, the number of containers that can be loaded/unloaded during the berthing is known. Moreover, the volume of full containers to load/unload during the berth is also known. Taking into account the number of full containers that must be loaded/unloaded during the berth, the handling capacity is supposed updated in order to represent the number of empty containers only that can be loaded/unloaded. Therefore, the handling capacity represents an upper bound of the number of empty containers that can be loaded/unloaded from a vessels during its berthing;

$Uh_t(i)$**:** Maximum number of empty containers that can be stored in port $i \in H$ at period $t \in T$;

$Um_{tv}(i)$**:** Residual transportation capacity for empty containers moved by vessel $v \in V(i,t)$ from port $i \in H$ to the next port in its schedule R_v. Taking into account the volume occupied by full containers, the transportation capacity is updated in order to represent the number of empty containers that can be transported by the vessel to the next port.

DECISION VARIABLES

$xh_t(i)$**:** Decision made at period $t \in T$ on the number of empty containers stored in port $i \in H$ at period $t \in T$ and not assigned for loading to any vessel arriving at time $t+1 \in T$; $ch_t(i)$ represents the related unitary cost. Loading and unloading decisions do not take into account the precisely periods in which these operation are performed. Therefore, storage costs are supposed charged only when empty containers are kept in the dedicated storage area;

$xl_{tv}(i)$**:** Decision made at period $t \in T$ on the number of empty containers available in port $i \in H$, which must be loaded on vessel $v \in V(i,t+1)$; $cl_{tv}(i)$ represents the related unitary cost;

$xm_{tv}(i)$**:** Decision made at period $t \in T$ on the number of empty containers transported by vessel $v \in V(i,t+1)$ from port $i \in H$ to the following port in its schedule R_v. It is important to note that this variable considers all empty containers carried by vessel $v \in V(i,t+1)$ regardless when and where they were loaded and when and where they will be unloaded; $cm_{tv}(i)$ represents the related unitary cost.

$xu_{tv}(i)$**:** Decision made at period $t \in T$ on the number of empty containers unloaded from vessel $v \in V(i,t+1)$ in port $i \in H$; $cu_{tv}(i)$ represents the related unitary cost.

PARAMETERS

α_{vit}**:** Periods spent in port by vessel $v \in V(i,t)$: it takes value 0 if the vessel stays in the port for less than one day, it takes the value 1 if it stays between one and two days and, so on. Index *vit* will be omitted in the model in order to facilitate its reading;

τ_{vit}**:** Number of periods elapsed between two arrival times of two consecutive berth by vessel $v \in V(i,t)$: it takes the value 1 if the arrival time in port $i-1 \in H$ is equal to $t-1$, it takes the value 2 if the arrival time in port $i-1 \in H$ is equal to $t-2$, and so on. The arrival time of vessel $v \in V(i,t)$ in the previous port can be found in the triplets of its schedule R_v. Index *vit* will be omitted in the model in order to facilitate its reading.

Additional informations concern the number of empty containers in shortage in port $i \in H$ at period $t \in T$ and the number of empty containers that cannot be stored in port $i \in H$ at period $t \in T$, due to its limited storage capacity. In order to facilitate the comprehension of the model, these quantities are represented explicitly by variables usually called as *artificial variables* or *dummy variables*. For the sake of clarity, from this point onward, empty containers in shortage in port $i \in H$ at time $t \in T$ are represented by *shortage variables*, denoted by $xsh_t(i)$, whereas empty containers that cannot be stored in port $i \in H$ at time $t \in T$ are represented by *exceed capacity variables*, denoted by $xec_t(i)$. $csh_t(i)$ and $cec_t(i)$ represents their related unitary cost.

Finally, the problem can be formalized as follow:

$$min \sum_{t \in T} \sum_{i \in H} \Bigg[\sum_{v \in V(i,t+1)} \Big(cl_{tv}(i)\ xl_{tv}(i) + cm_{tv}(i)\ xm_{tv}(i) + cu_{tv}(i)\ xu_{tv}(i) \Big) + ch_t(i)\ xh_t(i) + csh_t(i)\ xsh_t(i) + cec_t(i)\ xec_t(i) \Bigg] \quad (2.1)$$

$$xh_t(i) + \sum_{v \in V(i,t+1)} xl_{tv}(i) + xec_t(i) - xh_{t-1}(i) - \sum_{v \in W(i,t)} xu_{(t-1-\alpha)v}(i) - xsh_t(i) = b_t(i) \quad \forall i \in H, \forall t \in T \quad (2.2)$$

$$xm_{(t-\tau)v}(i-1) - xm_{tv}(i) - xu_{tv}(i) + xl_{tv}(i) = 0 \qquad \forall v \in V(i,t+1), \forall i \in H, \forall t \in T \quad (2.3)$$

$$xh_t(i) + \sum_{v \in V(i,t+1)} xl_{tv}(i) + xsh_t(i) - xec_t(i) \leq Uh_t(i) \qquad \forall i \in H, \forall t \in T \quad (2.4)$$

$$xl_{tv}(i) + xu_{tv}(i) \leq U_{(t+1)v}(i) \qquad \forall v \in V(i,t+1), \forall i \in H, \forall t \in T \quad (2.5)$$

$$xm_{tv}(i) \leq Um_{tv}(i) \qquad \forall v \in V(i,t), \forall i \in H, \forall t \in T \quad (2.6)$$

In addition to constrains 2.2-2.6 all decision variables must be non-negative integers.

The objective function in 2.1 minimizes the costs of inventory, loading, transportation, unloading operations over the planning horizon. Shortage and extra capacity costs are also taken into account.

Constraints 2.2 concern the flow balance constraint for each port $i \in H$ and each period $t \in T$. A positive value of $b_t(i)$ represents a surplus of empty containers, available in port $i \in H$ at period $t \in T$. Empty containers in surplus: *(i)* can be stored in the same port in the same period, $xh_t(i)$; *(ii)* can be loaded onto a vessel $v \in V(i,t+1)$ arriving in the same port in the following period, $xl_{tv}(i)$; *(iii)* can exceed the storage capacity of the port in the same period, $xec_t(i)$. When empty containers exceed the storage capacity, the empty containers that are not able to stay in port in that period are given away by a most costly way, like transported in other port. Negative value of $b_t(i)$ represent a request of empty containers. Empty container demands: *(iv)* can be met by containers stored in the same port in the previous period, $xh_{t-1}(i)$; *(v)* can be met by containers unloaded from a vessel $v \in V(i,t-\alpha)$ departing from the same port in the same period, $xu_{(t-1-\alpha)v}(i)$; *(vi)* can lead to shortages in that period, $xsh_t(i)$. When empty container demands lead to shortages, empty containers are provided by a most costly way, like renting empty containers from other port.

Constraints 2.3 concern the flow balance constraint for each vessel $v \in V(i,t+1)$, each port $i \in H$ and each period $t \in T$. A vessel $v \in V(i,t+1)$ arrives in port $i \in H$ at period $t+1 \in T$, and according to its schedule R_v: *(i)* in the previous berth the vessel was in port $i-1 \in H$ at period $t-\tau \in T$; *(ii)* in the next berth the vessel will be in port $i+1 \in H$ at period $t+\tau \in T$. The total number of empty containers shipped by vessel v, $xm_{tv}(i)$, from port i to the next port $i+1$, is given by the number of empty containers shipped by the same vessel in the previous berth, $xm_{(t-\tau)v}(i-1)$, *PLUS* the number of empty containers loaded on the same vessel in the same berth, $xl_{tv}(i)$, *MINUS* the number of empty containers unloaded from the same vessel in the same berthing, $xu_{tv}(i)$. Loading and unloading operations are supposed performed in mutual exclusion, and the constraint does not force any

operation. If no operation is need, the number of empty containers shipped by vessel v, from port i to the next port $i+1$, must be equal to the number of empty containers shipped by the same vessel in the previous berth, from port $i-1$ to port i.

Constraints 2.4 concern the storage capacity of each port $i \in H$ in each period $t \in T$. Empty containers in stock can be: *(i)* stored in the same port in the same period, $xh_t(i)$; *(ii)* loaded on a vessel $v \in V(i,t+1)$ arriving in the same port in the following period, $xl_{tv}(i)$; *(iii)* provided by a most costly way in the same port and in the same period, $xsh_t(i)$; *(iv)* given away by a most costly way in the same port and in the same period, $xec_t(i)$. The operations involved in point *(i)* , *(ii)* and *(iii)* occupy a part of the storage capacity of the port, whilst the operation involved in point *(iv)* make available a part of the same storage capacity. Therefore, all operations must be taken into account to not violate the storage capacity, $Uh_t(i)$.

Constraints 2.5 concern the handling capacity of the each port $i \in H$ for each vessel $v \in V(i,t+1)$, in each period $t \in T$. The handling capacity of a port $i \in H$ for a vessel $v \in V(i,t+1)$ represents the number of empty containers that can be successfully loaded/unloaded during the berthing. Therefore, the number of empty containers loaded on vessel $v \in V(i,t+1)$, $xl_{tv}(i)$, $PLUS$ the number of empty containers unloaded from the same vessel, $xu_{tv}(i)$, must be lower than the maximum number of empty containers that can be successfully loaded/unloaded during the berthing of the vessel v in port i at time $t+1$, $U_{(t+1)v}(i)$.

Constraints 2.6 concern the residual transportation capacity for each vessel $v \in V(i,t)$, each port $i \in H$ and each period $t \in T$. A vessel $v \in V(i,t)$ depart from port $i \in H$ at period $t \in T$, and according to its schedule R_v in the next berth the vessel will be in port $i+1 \in H$ at period $t-\alpha+\tau \in T$. The total number of empty containers shipped by vessel v, $xm_{tv}(i)$, from port i to the next port $i+1$, must not exceed the residual transportation capacity of vessel v in port i at time t, $Um_{tv}(i)$.

2.2 Uncertainty and Multi-scenario model

The deterministic formulation, presented in section 2.1, has a single scenario and is appropriate when data are known with a high level of certainly for all periods of the planning horizon, and there is confidence in the data. When data are not known with certainty, or a level of uncertainty affects some data, a multi-scenario formulation can be more appropriate to represent better the situation of the system. The aim is to introduce a formulation that ensure an effective repositioning of empty container for each different possible future evolutions.

Usually, all data are known with a high level of certainty or are based on historical data. Expert opinions are taken into account to identify some statistical

component for each uncertain parameter and construct subjective distributions. The multi-scenario formulation is effective also when distributions, based on historical data, are not available. Moreover, when statistical informations are known with a low level of certainty, or there is no confidence in the data, expert opinions are taken into account to assign to each scenario a weight that characterizes its importance.

Generally speaking, the information of the uncertain parameters are as much accurate as much the period in which are foreseen is close the first period of the planning horizon. Therefore, all parameters are certain up a given period θ, and $\theta + 1$ represents the first period in which uncertain parameters may appear.

All scenarios are collected in an overall mathematical model linked by the "*non-anticipativity*" constraints, which forces identical decisions up to time θ. Their addition in the model guarantee that current decisions do not take advantage of information not yet available.

The multi-scenario model belongs to the class of stochastic integer programming, and it can be easily converted into the deterministic equivalent of a two-stage stochastic model, in which here-and-now decisions are in the first stage and the remaining decisions in the second stage Powell and Topaloglu 2003.

Referring to Figure 2.3 that illustrates the system dynamics, Figure 2.4 represents an example of two scenarios in a time-extended network. The uncertainty can affect, potentially, all parameters of the system. Figure 2.4 represents an example in which uncertainty concerns the numbers of empty containers offered or demanded in ports. Positive numbers close to nodes represent the surplus of empty containers, whereas negative numbers represent their demand. The two scenario in Figure 2.4 differs in the number of empty containers requested in port C at period 4, respectively 15 for *scenario* 1 and 20 for *scenario* 2.

The following part of the section illustrates the multi-scenario model for the most general case: uncertainty in all parameters. In this problem setting, the ability of decisions to withstand against uncertainty is sought from the point of view of shipping companies, which must plan the repositioning of empty containers in order to meet all customer demands.

With respect to the deterministic formulation, presented in section 2.1, in the multi-scenario formulation there is the need to introduce scenarios and weights. The following notation will be used:

SETS

G**:** Set of scenarios. A single scenario g represents a possible future evolution based on the value taken by the uncertain parameters.

DATA

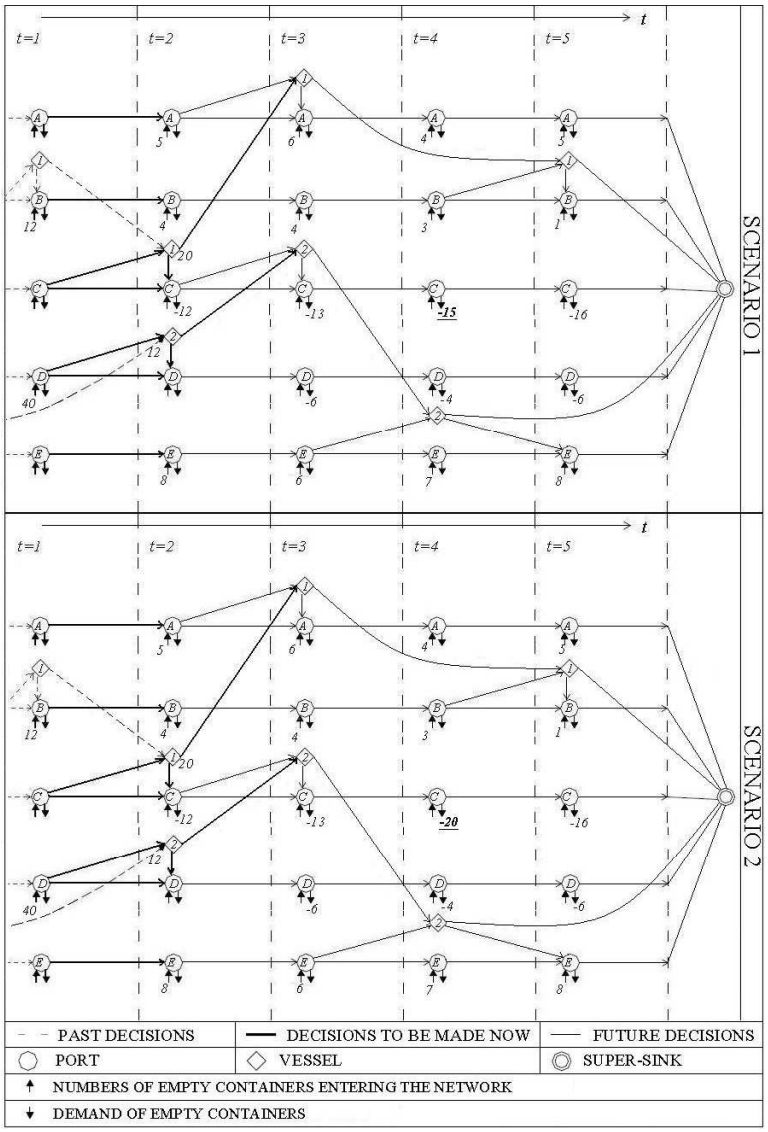


Figure 2.4: Two scenarios in a time-extended network.

θ**:** Period up to which all parameter are certain. $\theta + 1$ represents the first period in which uncertain parameters may appear;

w_g**:** Weight of the scenario $g \in G$.

The differences between the deterministic model and the multi-scenario one concern the introduction of scenarios and non-anticipativity constraints. The multi-scenario model can be formalized as follow:

$$min \sum_{g \in G} w^g \sum_{t \in T} \sum_{i \in H} \Bigg[\sum_{v \in V(i,t+1)} \Big(cl^g_{tv}(i)\ xl^g_{tv}(i) + cm^g_{tv}(i)\ xm^g_{tv}(i) + cu^g_{tv}(i)\ xu^g_{tv}(i) \Big) + ch^g_t(i)\ xh^g_t(i) + csh^g_t(i)\ xsh^g_t(i) + cec^g_t(i)\ xec^g_t(i) \Bigg] \quad (2.7)$$

$$xh^g_t(i) + \sum_{v \in V(i,t+1)} xl^g_{tv}(i) + xec^g_t(i) - xh^g_{t-1}(i) - \sum_{v \in W(i,t)} xu^g_{(t-1-\alpha)v}(i) - xsh^g_t(i) = b^g_t(i) \quad \forall i \in H, \forall t \in T, \forall g \in G \quad (2.8)$$

$$xm^g_{(t-\tau)v}(i-1) - xm^g_{tv}(i) - xu^g_{tv}(i) + xl^g_{tv}(i) = 0 \quad \forall v \in V(i,t+1), \forall i \in H, \forall t \in T, \forall g \in G \quad (2.9)$$

$$xh_t(i)^g + \sum_{v \in V(i,t+1)} xl^g_{tv}(i) + xsh^g_t(i) - xec^g_t(i) \leq Uh^g_t(i) \quad \forall i \in H, \forall t \in T, \forall g \in G \quad (2.10)$$

$$xl^g_{tv}(i) + xu^g_{tv}(i) \leq U^g_{(t+1)v}(i) \qquad \forall v \in V(i,t+1), \forall i \in H, \forall t \in T, \forall g \in G \quad (2.11)$$

$$xm^g_{tv}(i) \leq Um^g_{tv}(i) \qquad \forall v \in V(i,t), \forall i \in H, \forall t \in T, \forall g \in G \quad (2.12)$$

$$xh^g_t(i) = xh^f_t(i) \qquad \forall i \in H, t \in \{1, \ldots, \theta\}, \forall g, f \in G, g \neq f \quad (2.13)$$

$$xl^g_{tv}(i) = xl^f_{tv}(i) \qquad \forall v \in V(i,t), \forall i \in H, t \in \{1, \ldots, \theta\}, \forall g, f \in G, g \neq f \quad (2.14)$$

$$xm^g_{tv}(i) = xm^f_{tv}(i)) \qquad \forall v \in V(i,t), \forall i \in H, t \in \{1, \ldots, \theta\}, \forall g, f \in G, g \neq f \quad (2.15)$$

$$xu^g_{tv}(i) = xu^f_{tv}(i) \qquad \forall v \in V(i,t), \forall i \in H, t \in \{1, \ldots, \theta\}, \forall g, f \in G, g \neq f \quad (2.16)$$

In addition to constrains (2.8)-(2.16) all decision variables must be non-negative integers.

The objective function in (2.7) minimizes the costs of inventory, loading, transportation, unloading operations over the planning horizon for all scenarios. Shortage and extra capacity costs are also take into account.

Constraints (2.8) concern the flow balance constraint for each port $i \in H$, each period $t \in T$ and each scenarios $g \in G$. Constraints (2.9) concern the flow balance constraint for each vessel $v \in V(i,t+1)$, each port $i \in H$, each period $t \in T$ and each scenarios $g \in G$. Constraint 2.10 concerns the storage capacity of each port

$i \in H$ in each period $t \in T$ and each scenarios $g \in G$. Constraints 2.11 concern the handling capacity of the each port $i \in H$ for each vessel $v \in V(i,t+1)$, in each period $t \in T$ and each scenarios $g \in G$. Constraints 2.12 concern the residual transportation capacity for each vessel $v \in V(i,t)$, each port $i \in H$, each period $t \in T$ and each scenarios $g \in G$.

Constraints (2.13) concern the non-anticipativity condition for the number of empty containers stored in ports between period 1 and period 2. The number of empty container stored in port must be the same for each scenario.

Constraints (2.14) concern the non-anticipativity condition for the number of empty containers loaded onto vessels that arrive in ports at period 2. The number of empty container loaded onto vessels arriving in port in the second period of the planning horizon must be the same for each scenario.

Constraints (2.15) concern the non-anticipativity condition for the number of empty containers transported by vessels that arrive in ports at period 2. The number of empty container transported by vessels arriving in port in the second period of the planning horizon must be the same for each scenario.

Constraints (2.16) concern the non-anticipativity condition for the number of empty containers unloaded from vessels that arrive in ports at period 2. The number of empty container unloaded from vessels arriving in port in the second period of the planning horizon must be the same for each scenario.

It is important to note that the non-anticipativity constrains do not concern both shortage and extra capacity variables. In fact, these information may differ between scenarios since the value taken by the uncertain parameters is different.

Chapter 3

Experimentation

Numerical tests are carried out to evaluate the relevance of both formulations, the deterministic and the multi-scenario one, for the repositioning of empty containers under uncertainty.

In both formulations decisions are taken in a *"rolling horizon fashion"* to comprehend their impacts in the future. In a rolling horizon fashion the repositioning of empty container is determined for the whole planning horizon, but only the decision involved in the first period are taken into account and implemented. Next, after decisions are implemented, all periods are shifted by one unit. The second period becomes the first, the third one becomes the second, and so on until the end of the planning horizon, where new information become available. The data of the problem are updated and the problem will be solved again to find the optimal solution under new conditions.

The two formulations are compared in terms of operating costs (OPC) and customer satisfaction, represented by the demand fulfilment percentage (DFP). Both indicators, OPC and DFP, are defined in each rolling for each port $i \in H$ only in the fist period of the planning horizon. The OPC and the DFP indicators are computed as follows:

$$OPC = \sum_{v \in V(i,2)} \Big(cl_{1v}(i) + cm_{1v}(i) + cu_{1v}(i) \Big) + ch_1(i) \qquad \forall i \in H \qquad (3.1)$$

$$DFP = 100 \; \frac{|b_1(i)| - xsh_1(i)}{|b_1(i)|} \qquad \forall i \in H \qquad (3.2)$$

Referring to Equation (3.1), operating costs are computed as the sum of all costs generated by here-and-now decisions in the first period of the planning horizon. According to Equation (3.2), the demand fulfilment percentage: *(i)* takes value 100 if there is no shortage in port $i \in H$ in the first period of the planning horizon; *(ii)* takes value zero if the number of empty containers in shortage in port $i \in H$ in the first period of the planning horizon equals the demand of the port (no demand is

met). It is important to note that in ports characterized by a surplus of empty containers DFP takes always value 100.

Tests are carried out according the following procedure for a number of days:

- Models are solved and here-and-now decisions are implemented;
- Operating costs generated by here-and-now decisions are determined, and the demand fulfilment percentage is computed for each port in the first period of the planning horizon;
- A new period is added at the end of the planning horizon, which is rolled forward in time once (i.e., the second period "becomes" the first in the next run of the models, the third "becomes" the second and so on) and the forecasts are updated. In this phase, called from now onward *"rolling"*, may happen that some uncertain parameter becomes certain, since their period becomes more close to period θ with the rolling.

In this experimentation θ takes value 2. This is a reasonable choice motivated by the following points: *(i)* customers book their requests several days before the arrival of the vessel; *(ii)* the number of empty containers available at ports in the first period is known by observation; *(iii)* the number of the empty container in the second period can be estimated accurately.

Demand and supply values are generated for each port, and each period, by several customer transportation requests. Each request consists of a number of full containers, which must be shipped, on behalf of a customer, from an origin port, in a period, to a destination port, in another period. Therefore, each customer request refers to number of empty containers required in the origin port and offered in the destination port.

The experimentation is based on the extended-time network represented in Figure 3.1, which shows the network in the first day of the simulation. The network has 8 periods, 5 ports and 2 vessels. Ports A and B are import-dominant, C is a hub, D and E are export-dominant. As a result, the problem consists in the repositioning of empty containers from A and B to D and E. The number of empty containers of each customer request is randomly generated by a uniform distribution from 1 to 3. For each request the time in which empty containers are requested in the origin port and become available in the destination port is generated by a uniform distribution over 50 periods. 2000 customer requests has been generated and the number of empty containers that are totally offered and required in each port and each period is computed. The difference between supplies and demands is equal to $b_t(i)$. Furthermore, the average daily supply in ports A and B is 80 containers, whereas the average daily demand in ports D and E is 100 containers.

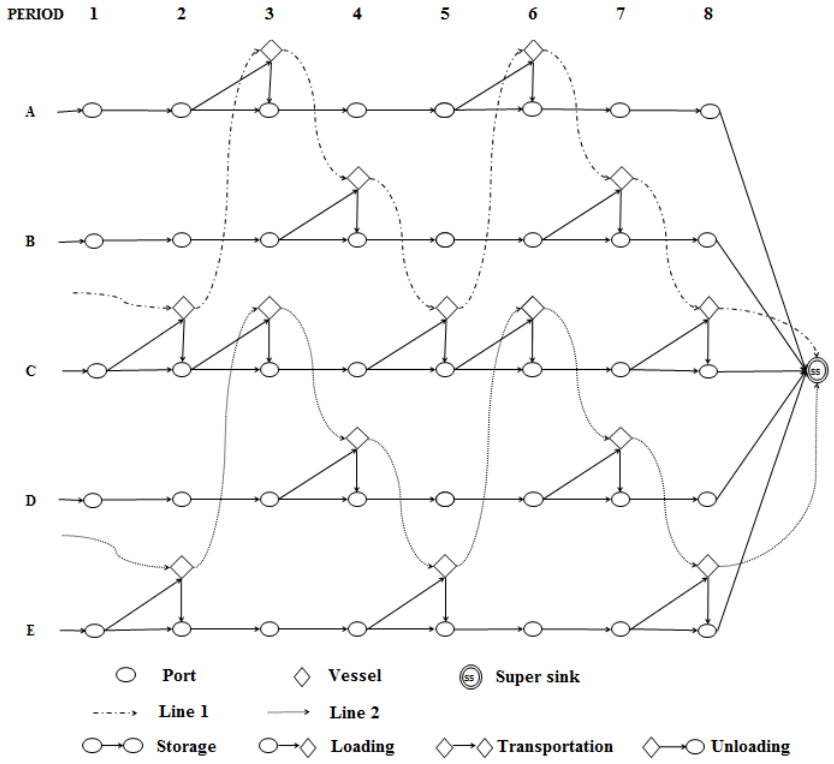


Figure 3.1: Experimentation time-extended network.

To reduce both beginning and end effects, due to the data generation of demands and supplies, the 50 period planning horizon is truncated removing the first 15 periods and the final 15 periods.

This simulation process can be performed with any value of costs. This experimentation considers the case of a shipping company repositioning its empty containers in the space available on its vessels: it pays ports for storage, loading and unloading operations. Since the shipping company owns vessels, no additional cost is paid for containers while on-board. According to meeting with industrial experts, the following unitary costs are adopted:

- Storage ($ch_t(i)$): \$10;
- Loading ($cl_{tv}(i)$): \$100;
- Transportation ($cm_{tv}(i)$): \$0;
- Unloading ($cu_t(i)$): \$100.

It is important to note that these costs do not lead to unnecessary volumes of empty containers repositioned on vessels. For example, if we consider a container available in a port in the first period of the planning horizon, since the storage cost in the whole planning horizon (\$10/period multiplied by 8 periods = \$80) is lower than

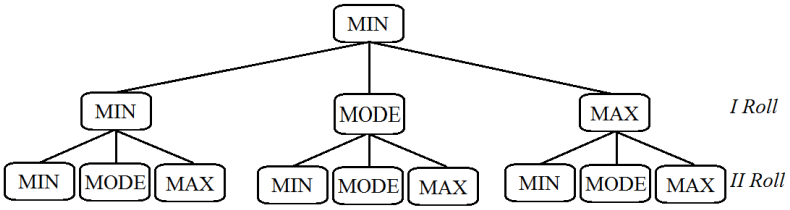
the loading cost ($100), it would be never put on a vessel if there is no demand in the network. Finally, $csh_t(i)$ and $cec_t(i)$ are set equal to a high value, $10000, in order to minimize both shortages and storage capacity violations.

3.1 Scenario definition

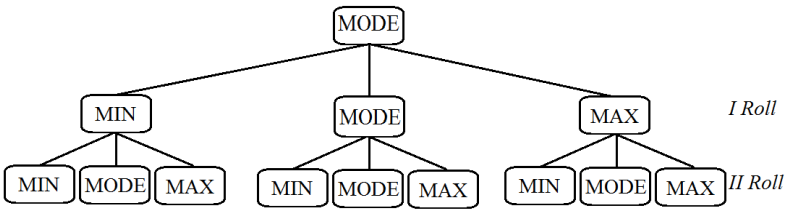
A set of scenarios is generated to consider all different possible realizations of uncertain parameters. The set of scenarios generated characterize each possible future evolutions, and differ each other on the value taken by uncertain parameters. Moreover, there is the complication that the values foreseen or the weights assigned to the uncertain parameters may change during the time, because models are implemented in a rolling horizon fashion and at each rolling data forecast are updated. Several procedures can be adopted for the scenarios generation. An easy procedure is to take into account the expert opinions in order to identify the minimum, the most plausible, and the maximum values for each uncertain parameter. Then, these three values are combined for each uncertain parameter and one scenario is generated for each possible future.

The deterministic formulation considers one scenario at a time. In order to compare in the same conditions the deterministic formulation with the multi-scenario one there is the need to solve all possible combinations of the uncertain parameters during the time. All possible combinations of the uncertain parameters generates a decision tree for the deterministic policies. The *deterministic decision tree* represents all scenarios, and at each level i there are the values foreseen for the uncertain parameters in the i^{th} rolling. Figure 3.2 shows an example of how the deterministic decision tree is generated. The example involves a single uncertain parameter that can take three different values: the minimum, MIN, the most plausible, $MODE$, and the maximum value, MAX. The uncertain parameter is foreseen the first time in the fourth period of the planning horizon. In the first level of the decision tree (root node) there is the value foreseen for the uncertain parameter when it appears the first time. Next, the first rolling is performed and the uncertain parameter is foreseen in the third period of the planning horizon. As the previous run of the model, the uncertain parameter can take three different values (minimum, most plausible or maximum). Next, the second rolling is performed and the uncertain parameter is foreseen in the second period of the planning horizon, it becomes certain and it is observed with one of the three different values. Figures 3.2a, 3.2b and 3.2c represent the three decision trees generated. Each of them differs in the value foreseen for the uncertain parameter when it appear the first time in the planning horizon, when none rolling is performed. As Figure 3.2 shows, each value foreseen for the uncertain parameter, when it appears the first time (first

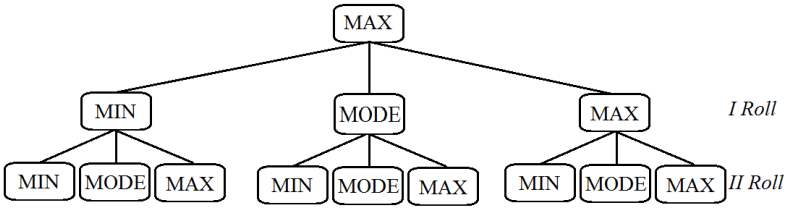
level of the decision tree), leads to 9 different policies, each of them differs in the value foreseen for uncertain parameter during the time, and taken by the uncertain parameter when it becomes certain in the last rolling.



(a) *Decision tree of the minimum value*



(b) *Decision tree of the most plausible value*



(c) *Decision tree of the maximum value*

Figure 3.2: A deterministic decision tree with a single parameter.

The decision tree grows exponentially with the number of the uncertain parameters and the values that uncertain parameters can take. In this experimentation, for the sake of clarity in the comprehension of the comparison, is represented the case in which the uncertain parameters can take two values only (minimum or maximum). Moreover, only four deterministic policies will be compared with the multi-scenario model, the first two policies and the last two policies of the decision tree, as showed in Figure 3.3 in which grey nodes represent the policies taken into account for the comparison.

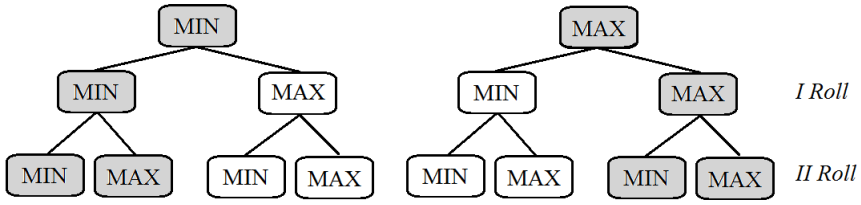


Figure 3.3: Deterministic policies taken into account for the comparison.

Two scenarios are considered: in scenario g_1 "normal" operations are foreseen,

whereas in scenario g_2 "uncertain" operations are forecast as a possible future. Three policies can be adopted:

p_1 which uses the deterministic model with scenario g_1 as a point forecast;

p_2 which uses the deterministic model with scenario g_2 as a point forecast;

p_3 which uses the multi-scenario model. Scenarios g_1 and g_2 are incorporated in the multi-scenario formulation and linked by non-anticipativity constraints. In these tests, the weights of scenarios are supposed equal, in order to face the most critical situation, in which the shipping company has no idea about which scenario is more likely to occur.

These policies are compared for a number of days by the simulation of two possible future evolutions:

e_1 in which normal operations occur in each period of the planning horizon;

e_2 in which uncertain operations occurs at some point in time.

The three policies (p_1, p_2, and p_3) are compared in both evolutions (e_1 and e_2) in terms of costs, generated by here-and-now decisions, and demand fulfilment percentage.

The uncertainty can involve potentially any data or parameter. The contribution of this study is to illustrate the effectiveness of a multi-scenario approach in facing three significant types of uncertainty:

Uncertain demand in which customer request are supposed to increase significantly after unexpected event. This is the most realistic type of uncertainty, in fact as the planning horizon growth as difficult becomes the correct estimation of the customer requests. In the multi-scenario model presented in Section 2.2, this type of uncertainty involves Equation (2.8);

Uncertain handling capacity in which empty containers cannot be loaded and unloaded from vessels, because berthing cannot be performed. Although containers cannot be handled in the seaside, they can be stored, shipped and received from the landside. In the multi-scenario model presented in Section 2.2, this type of uncertainty involves Equation (2.11). From this point onward, this type of uncertainty will be called as uncertain handling capacity or partial disruption;

Uncertain demand and handling capacity in which empty containers can neither be handled in the seaside nor the landside. Vessels do not berth and

containers cannot be stored, shipped and received from the landside. The volume of empty containers in stock in the port does not change during this type of uncertainty. Empty containers can be re-used only after the uncertainty conclusion. In the multi-scenario model presented in Section 2.2, this type of uncertainty involves Equations (2.8) and (2.11). From this point onward, this type of uncertainty will be called as uncertain demand and handling capacity or complete disruption.

3.2 Data management

The deterministic and the multi-scenario model were coded using IBM ILOG CPLEX Optimization Studio 12.5 and solved by the Branch & Bound of ILOG CPLEX 12.2 which employs state-of-the-art algorithms and techniques to solve mixed integer programming problems. Experiments are performed on a Linux four-CPU server 2.67GHz 64GB RAM, with default parameter settings.

The ILOG Optimization Programming Language (*OPL*) provides an easy way to represent models. Moreover, other features concern: *(i)* the possibility to use advanced data structures; *(ii)* the possibility to manage Mixed-Integer Linear Programming; *(iii)* the possibility to initialize decision variables both integer or float.

The ILOG OPL Studio is made up of:

- The Optimization Programming Language used to develop optimization models;
- The Integrated Development Environment used to perform and test in an easy way optimization models;
- Application Programming Interfaces (API) that incorporates optimization models in stand-alone applications.

OPL is supported in various platforms, such as UNIX (AIX, Solaris and Linux) and Windows. It is important to note that Unix platforms do not have a graphic user interface that perform and test the optimization model easily. In a Unix platform an optimization model is executed by the *oplrun* command line. Notwithstanding, all options included in the Windows graphic user interface are available also in the command line version.

The are mainly two file in OPL: *(i)* a file concerning the optimization model (.mod) and *(ii)* one or more files concerning the initialization of data declared in the model file (.dat). The model file includes the declaration of data and decision variables, the objective function and the definition of all constraints. The input file

includes the initialization of all data involved in the model, such as vessels, ports and periods in this particular case.

As mentioned before, the rolling phase requires the shift of the planning horizon by one period. In order to facilitate the rolling phase, two output files are generated with extension .dat by a script that personalizes all data included in it. The first file concerns all solution details, such as objective function and number of containers stored, loaded, transported and unloaded in each period of the planning horizon. The second output file has the same structure included in the input one. The differences between them concern the following points:

- All components that are characterized by the period in which they are considered, such as demands, scheduling of vessels, storage capacities and costs, are reported in the output file with their period decreased by one unit;
- All components involved in period 0 after the shift operation are deleted;
- All components involved in the first period of the planning horizon are updated in order to take into account the decisions made in the current execution of the model;
- All components involved in the last period of the planning horizon ($t = T$) are set to 0.

In order to find the optimal solution under new conditions, only information involved in the last period of the planning horizon must be updated based on new information available. Moreover, when new information becomes available, changes may be applied also to uncertain parameters updating their point forecasts.

All components that are not characterized by a period, such as the total storage capacity of a vessel, are reported as in the input file.

The idea of create a second output file with the same structure of the input one leads to an important benefit: the saving of time spent in managing all components included in the input file. Instead of modify all periods for all components, only a little part of them must be updated.

Next sections show the computational results for each type of uncertainty presented in Section 3.1

3.3 Uncertain demand

Referring to Figure 3.1, this test evaluates the effects of an abnormal growth of the demands in ports D and E, at periods 4 and 5 respectively. If uncertain events occur, the demand is supposed to grow of a quantity δ. In this experimentation δ is

defined as the maximum demand that characterize the port in the whole planning horizon. The capability of facing the uncertainty consists in the ability to increase the number of empty containers unloaded in port D and E during the first and second call of the vessel. These additional containers represent preventive measures reducing the risk of shortages if the uncertainty occurs.

Policies p_2 and p_3 foresee the anomaly growth of the demand, which could be observed during the second rolling of the planning horizon. In order to evaluate the policies also after the realization of the uncertainty, the simulation is performed for 8 days, rolling the planning horizon 7 times. Results are illustrated in Table 3.1, where each row describes a combination of policies and evolutions. Columns show the demand fulfilment percentages, DFP, in the first period of the planning horizon from day 4 up to day 8, computed as in Equation (3.2). The demand is completely met from day 1 up to day 3 in all ports. The last column, OPC, represents total costs generated by here-and-now decisions, computed as in Equation (3.1). These costs are computed for each row as the sum of storage, loading, transportation and unloading costs in all days of the simulation.

The next subsections analyse the results obtained for the two different evolutions of the future: e_1 in which normal operations occur in each period of the planning horizon, and e_2 in which uncertain operations occurs at some point in time.

Evolution e_1 - Normal Operations

When normal operations occur (evolution e_1), the policy that provides highest demand fulfilment percentages is policy p_3. Operating costs amount to \$210040 and, with respect to the demand fulfilment percentages, only 81.25% of demand in E, and 100% elsewhere is met at day 7, whereas all demands are satisfied in each port in the other periods of the planning horizon.

Deterministic policies are less effective in terms of demand fulfilment percentages. More precisely, with respect to the policy p_3:

- the deterministic policy p_1 satisfies all demands in the first 7 days, but the demand fulfilment percentage of day 8 in port E is lower. In fact this policy satisfies only the 56.25% of the demands in port E, and 100% elsewhere at day 8.

- the deterministic policy p_2 presents the worst demand fulfilment percentages for this evolution. This policy satisfies all demand in each port from day 6 up to day 8, whereas low demand fulfilment percentages characterize day 4 and 5. In fact, the policy satisfies only the 51.00% of the demands in port E, and 100% elsewhere at day 4, and 0.00% of the demands in port E, and 100% elsewhere at day 5.

	DFP					OPC
	Day 4	Day 5	Day 6	Day 7	Day 8	
$p_1(e_1)$	100% in all ports	100% in all ports	100% in all ports	100% in all ports	56.25% in E, 100% elsewhere	\$194640
$p_2(e_1)$	51.00% in E, 100% elsewhere	0.00% in E, 100% elsewhere	100% in all ports	100% in all ports	100% in all ports	\$195320
$p_3(e_1)$	100% in all ports	100% in all ports	100% in all ports	81.25% in E, 100% elsewhere	100% in all ports	\$210040
$p_1(e_2)$	100% in all ports	100% in all ports	0.00% in D, 22.00% in E, 100% elsewhere	100% in all ports	25.00% in E, 100% elsewhere	\$192940
$p_2(e_2)$	51.00% in E, 100% elsewhere	0.00% in E, 100% elsewhere	100% in all ports	100% in all ports	25.00% in E, 100% elsewhere	\$211350
$p_3(e_2)$	100% in all ports	100% in all ports	0.00% in D, 82.00% in E,100% elsewhere	100% in all ports	25.00% in E, 100% elsewhere	\$207640

$p_1(e_1)$ - Policy p_1 in evolution e_1
$p_2(e_1)$ - Policy p_2 in evolution e_1
$p_3(e_1)$ - Policy p_3 in evolution e_1
$p_1(e_1)$ - Policy p_1 in evolution e_2
$p_2(e_2)$ - Policy p_2 in evolution e_2
$p_3(e_2)$ - Policy p_3 in evolution e_2

Table 3.1: Uncertain demands: Demand fulfilment percentages and operating costs.

From the point of view of the operating cost, the policy p_1 is the best one with an amount of \$194640. Policies p_2 and p_3 provide higher costs. However, since this comparison is based only on operating costs without take into account shortage costs, this consideration can be wrong. More precisely, with respect to the operating costs the policy p_3 presents the highest value with an amount of \$210040 but, with respect to policy p_2, the company has more day to find an *alternative* to improve the demand fulfilment percentages that characterize port E. If the possibility to find a better alternative leads to less shortage costs, then the policy p_3 can be more convenient than policy p_2.

Evolution e_2 - Uncertain Operations

When the uncertain operations occur (evolution e_2), the policy that provides highest demand fulfilment percentages is the policy p_3. The total operating cost amounts to \$207640 and, with respect to the demand fulfilment percentages, only the 0.00% of the demands in port D, 82.00% of the demands in port E, and 100% elsewhere is met at day 6, and 25.00% of the demands in port E, and 100% elsewhere is met at day 8. All demand are satisfied in each port in the other periods of the planning horizon.

Deterministic policies are less effective in terms of demand fulfilment percentages. More precisely, with respect to the policy p_3:

- the deterministic policy p_1 provides lower demand fulfilment percentage at day 6. In fact this policy satisfies only 0.00% of the demands in port D, 22.00% of the demands in port E, and 100% elsewhere.

- the deterministic policy p_2 present the worst demand fulfilment percentages. This policy satisfies all demand in each port from day 6 and 7, whereas low demand fulfilment percentages characterize day 4 and 5. Indeed, this policy satisfies only 51.00% of demands in port E, and 100% elsewhere at day 4, and 0.00% of the demands in port E, and 100% elsewhere at day 5.

From the point of view of operating costs, the policy p_1 is the best one with an amount of \$192240. Policies p_2 and p_3 provide higher costs. In spite of what happen for the evolution e_1, the policy p_2 present the highest operating costs with an amount of \$211350.

Remarks

Generally speaking, the multi-scenario policy p_3 leads to higher demand fulfilment percentages than policy p_1 and policy p_2 and, of course, larger operating costs must be paid. Moreover, considering the evolution e_2 the policies p_1 and p_3 lead to a better distribution of the shortages between ports D and E, whereas the deterministic policy p_2 concentrate all shortages in port E.

Figure 3.4 represents for each policy the decisions taken on the number empty containers stored, loaded, kept on board, and unloaded in port E during the first call of the vessel of line 2.

Figure 3.4 shows that there is no empty container in stock for each policy before the arrival of the vessel, and several containers are unloaded in order to meet demands. If policy p_1 is adopted, the number of empty containers unloaded is lower than the number of empty containers unloaded by policies p_2 and p_3. Indeed,

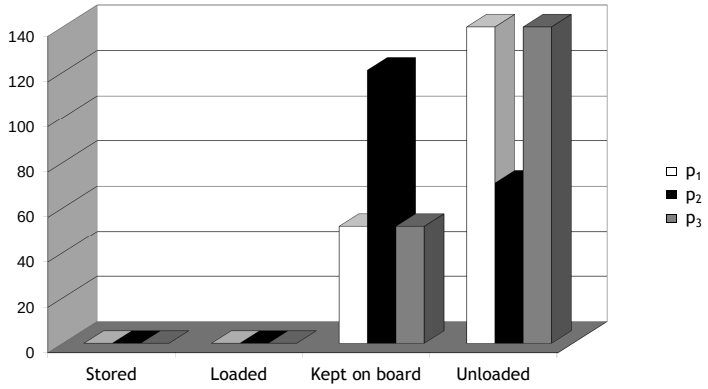


Figure 3.4: Uncertain demands: Decisions taken in the first call of the vessel in port E.

policy p_1 does not foresee the disruption and relies on the second call to unload an additional number of empty containers. In order to minimize costs, policy p_1 kept on board the volume of empty containers unloaded by policies to p_2 and p_3.

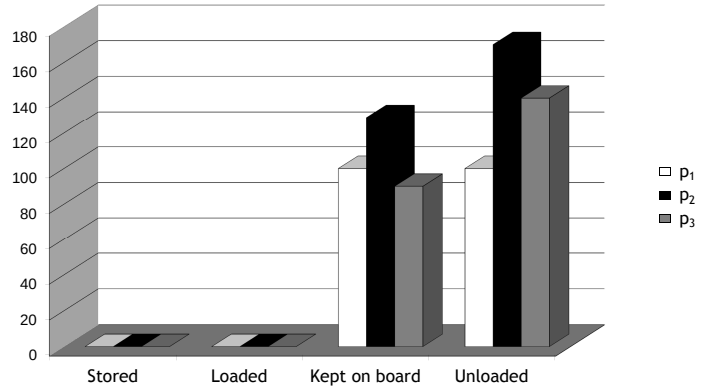


Figure 3.5: Uncertain demands: Decisions taken in the second call of the vessel in port E. Evolution e_1.

Figure 3.5 and Figure 3.6 represent for each policy the decisions taken on the number of empty containers stored, loaded, kept on board, and unloaded in port E during the second call of the vessel, according to evolution e_1 and evolution e_2 respectively.

When uncertain operations do not occur and policy p_1 is adopted, there are

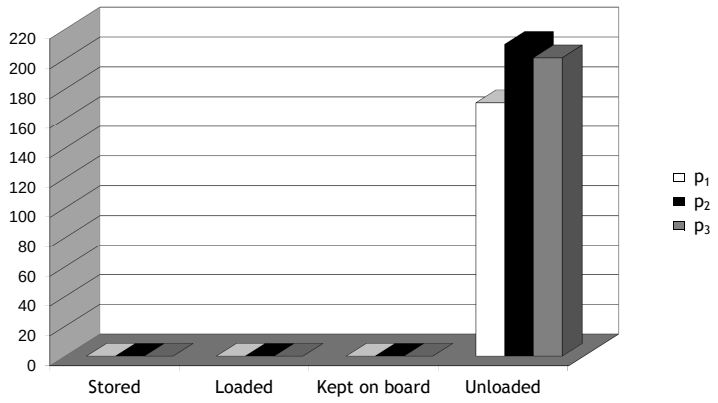


Figure 3.6: Uncertain demands: Decisions taken in the second call of the vessel in port E. Evolution e_2.

no containers in stock before the arrival of the vessel, and empty containers are unloaded in order to meet demands up to the third call of the vessel in port E. No empty containers are unloaded according to policy p_2, even if this opportunity is allowed by the realization of normal operations. Generally speaking, deterministic policies seem to not exhibit the flexibility of exploit new information, whereas additional containers are unloaded during second call of the vessel according to policy p_3.

Figure 3.6 shows that no empty containers are unloaded in port E when uncertain operations occur. The least effective policy is p_1, because many empty containers are kept on board, whereas they were supposed unloaded. Moreover, policies p_2 and p_3 can face container shortages by empty containers in stock in port E, unloaded during the first call of the vessel.

3.4 Uncertain handling capacity

Referring to Figure 3.1, this test evaluates the effects of a partial disruption in ports D and E, at periods 4 and 5 respectively. If the partial disruption occurs, empty containers cannot be unloaded from the vessel deployed on line 2 during the first call in port D, and the second call in port E. The capability of facing uncertainty consists in the ability to increase the number of empty containers unloaded in port E during the first call. These additional containers represent preventive measures reducing the risk of container shortages in case of partial disruption.

Policies p_2 and p_3 foresee the partial disruption, which could be observed during the second rolling of the planning horizon. In order to evaluate the policies also after the realization of the partial disruption, the simulation is performed for 8 days, rolling the planning horizon 7 times. Results are illustrated in Table 3.2, where each row describes a combination of policies and evolutions. Columns show the demand fulfilment percentages, DFP, in the first period of the planning horizon from day 4 up to day 8, computed as in Equation (3.2). The demand is completely met from day 1 up to day 3 in all ports. The last column, OPC, represents total costs generated by here-and-now decisions, computed as in Equation (3.1). These costs are computed for each row as the sum of storage, loading, transportation and unloading costs in all days of the simulation.

	DFP					OPC
	Day 4	Day 5	Day 6	Day 7	Day 8	
$p_1(e_1)$	100% in all ports	100% in all ports	100% in all ports	100% in all ports	56.25% in E, 100% elsewhere	\$194640
$p_2(e_1)$	100% in all ports	100% in all ports	100% in all ports	100% in all ports	25.00% in E, 100% elsewhere	\$166980
$p_3(e_1)$	100% in all ports	100% in all ports	100% in all ports	100% in all ports	43.75% in E, 100% elsewhere	\$189000
$p_1(e_2)$	100% in all ports	100% in all ports	38.33% in D, 0.00% in E, 100% elsewhere	100% in all ports	25.00% in E, 100% elsewhere	\$146380
$p_2(e_2)$	100% in all ports	100% in all ports	41.67% in D, 100% elsewhere	100% in all ports	25.00% in E, 100% elsewhere	\$165930
$p_3(e_2)$	100% in all ports	100% in all ports	38.33% in D, 100% elsewhere	100% in all ports	26.25% in E, 100% elsewhere	\$164420

$p_1(e_1)$ - Policy p_1 in evolution e_1
$p_2(e_1)$ - Policy p_2 in evolution e_1
$p_3(e_1)$ - Policy p_3 in evolution e_1
$p_1(e_1)$ - Policy p_1 in evolution e_2
$p_2(e_2)$ - Policy p_2 in evolution e_2
$p_3(e_2)$ - Policy p_3 in evolution e_2

Table 3.2: Partial disruption: Demand fulfilment percentages and operating costs.

The next subsections analyse the results obtained for the two different evolutions of the future: e_1 in which normal operations occur in each period of the planning horizon, and e_2 in which the partial disruption occurs at some point in time.

Evolution e_1 - Normal Operations

When normal operations occur (evolution e_1), the policy that provide highest demand fulfilment percentages is policy p_1. Operating costs amount to \$194640 and, with respect to the demand fulfilment percentages, only 56.25% of the demand in E, and 100% elsewhere is met at day 8, whereas all demand are satisfied in each port in the other periods of the planning horizon.

The deterministic policy p_2 and the multi-scenario one p_3 are less effective in terms of demand fulfilment percentages. More precisely, with respect to policy p_1:

- the deterministic policy p_2 is the worst policy. This policy satisfies all demands in the first 7 days, but the demand fulfilment percentage of day 8 in port E is lower. In fact this policy satisfies only the 25.00% of the demands in port E, and 100% elsewhere at day 8.

- the multi-scenario policy p_3 satisfies all demands in the first 7 days, but the demand fulfilment percentage of day 8 in port E is lower. The policy p_3 satisfies only the 43.75% of the demands in port E, and 100% elsewhere at day 8.

From the point of view of operating costs, the policy p_2 is the best one with an amount of \$166980. Policies p_1 and p_3 provide higher costs with an amount of \$194640 and \$189000 respectively. However, since this comparison is based only on the operating cost without take into account the shortage ones, this consideration can be wrong. More precisely, with respect to the operating costs, the policy p_3 presents an amount of \$189000 but, with respect to policy p_2, the company has a better demand fulfilment percentage. If the company gives a great importance to the customer satisfaction, then shortage costs may be to high and can lead the policy p_3 more convenient than policy p_2.

Evolution e_2 - Uncertain Operations

When uncertain operations occur (evolution e_2), the policy that provide highest demand fulfilment percentages are the policy p_2 and the policy p_3. The total operating cost amounts to \$165930 and \$164420 respectively, that leads the multi-scenario policy p_3 slightly better than the deterministic one p_2. The demand fulfilment percentages of policy p_2 are only the 41.67% of the demands in port D, and 100% elsewhere at day 6, 25.00% of the demands in port E, and 100% elsewhere is met at day 8, whereas all demand are satisfied in each port in the other periods of the planning horizon. The demand fulfilment percentages of policy p_3 are only the 38.33% of the demands in port D, and 100% elsewhere at day 6, 26.25%

of the demands in port E, and 100% elsewhere is met at day 8. All demands are satisfied in each port in the other periods of the planning horizon.

The deterministic policy p_1 is less effective in terms of demand fulfilment percentages. More precisely, the deterministic policy p_1 provides lower demand fulfilment percentage at day 6. In fact this policy satisfies only 38.33% of the demands in port D, 0.00% of the demands in port E, and 100% elsewhere at day 6, 25.00% of the demands in port E and 100% elsewhere at day 8. All demand are satisfied in each port in the other periods of the planning horizon.

From the point of view of the operating costs, the policy p_1 is the best one with an amount of \$146380. Policies p_2 and p_3 provide higher costs with an amount of \$165930 and \$164420 respectively. However, since this comparison is based only on operating costs without take into account the shortage costs, this consideration can be wrong. More precisely, with respect to operating costs, the policy p_3 presents an amount of \$164420 but, with respect to policy p_1, the company has better demand fulfilment percentages. If the company gives a great importance to the customer satisfaction, then shortage costs may be too high and can lead policy p_3 more convenient than policy p_1.

Remarks

Generally speaking, deterministic policies seem effective only when the future will evolve as expected, and seem unable to reposition empty containers reliably.

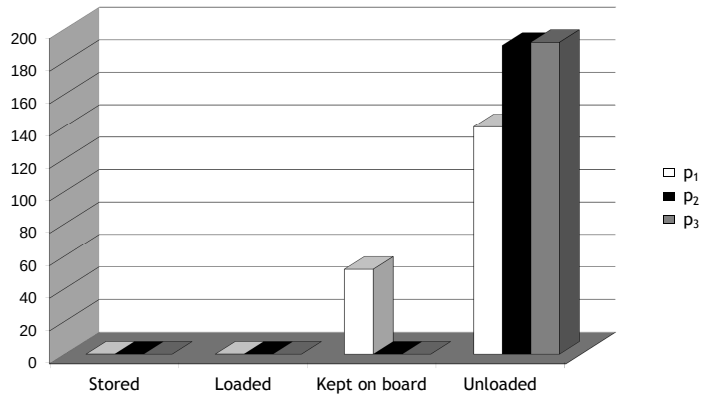


Figure 3.7: Partial disruption: Decisions taken in the first call of the vessel in port E.

Figure 3.7 represents for each policy the decisions taken on the number empty containers stored, loaded, kept on board, and unloaded in port E during the first

call of the vessel of line 2.

Figure 3.7 shows that there are no empty containers in stock before the arrival of the vessel, and several containers are unloaded in order to meet demands. If policy p_1 is adopted, the number of empty containers unloaded is lower than the number of containers unloaded by policies p_2 and p_3. Policy p_1 does not foresee the partial disruption and relies on the second call of the vessel to unload an additional number of empty containers. In order to minimize costs, policy p_1 kept on board the volume of empty containers unloaded by policies to p_2 and p_3.

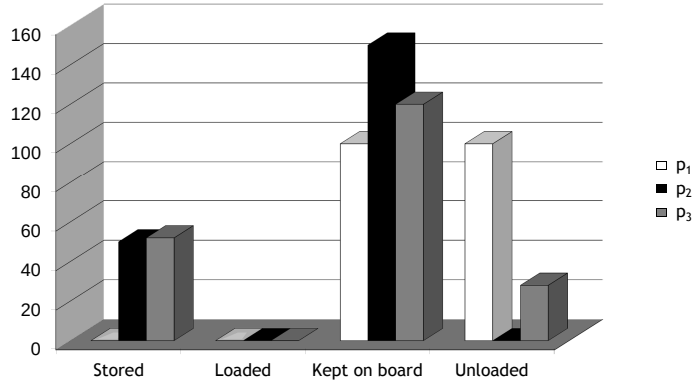


Figure 3.8: Partial disruption: Decisions taken in the second call of the vessel in port E. Evolution e_1.

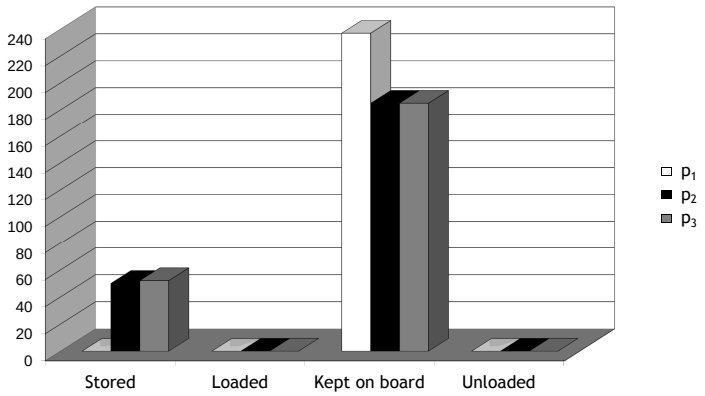


Figure 3.9: Partial disruption: Decisions taken in the second call of the vessel in port E. Evolution e_2.

Figure 3.8 and Figure 3.9 represent for each policy the decisions taken on the number of empty containers stored, loaded, kept on board, and unloaded in port E during the second call of the vessel, according to evolution e_1 and evolution e_2 respectively.

When the partial disruption does not occur and policy p_1 is adopted, there are no containers in stock before the arrival of the vessel, and empty containers are unloaded in order to meet demands up to the third call of the vessel in port E. No empty containers are unloaded according to policy p_2, even if this opportunity is allowed by the realization of normal operations. Generally speaking, deterministic policies seem to not exhibit the flexibility of exploit new information, whereas additional containers are unloaded during second call of the vessel according to policy p_3.

Figure 3.9 shows that no empty containers are unloaded in port E when the partial disruption occurs. The least effective policy is p_1, because many empty containers are kept on board, whereas they were supposed unloaded. Moreover, policies p_2 and p_3 can face container shortages by empty containers in stock in port E, unloaded during the first call of the vessel.

3.5 Uncertain demand and handling capacity

Referring to Figure 3.1, this section evaluates the effect of a complete disruption in port E at periods 4 and 5. If the complete disruption occurs, empty containers cannot be unloaded from the vessel deployed on line 2 during the second call in this port. Moreover, empty containers cannot be stored, shipped and received from the landside. The capability of facing uncertainty consists in the ability to increase the number of empty containers unloaded in port E during the first call of the vessel. These additional containers reduce the risk of container shortages in case of complete disruption.

Policies p_2 and p_3 foresee the complete disruption, which could be observed during the second rolling of the planning horizon. In order to evaluate the policies also after the realization of the complete disruption, the simulation is performed for 8 days, rolling the planning horizon 7 times. Results are illustrated in Table 3.3, where each row describes a combination of policies and evolutions. Columns show the demand fulfilment percentages, DFP, in the first period of the planning horizon from day 4 up to day 8, computed as in Equation (3.2). The demand is completely met from day 1 up to day 3 in all ports. The last column, OPC, represents total costs generated by here-and-now decisions, computed as in Equation (3.1). These costs are computed for each row as the sum of storage, loading, transportation and unloading costs in all days of the simulation.

	DFP					OPC
	Day 4	Day 5	Day 6	Day 7	Day 8	
$p_1(e_1)$	100% in all ports	100% in all ports	100% in all ports	100% in all ports	56.25% in E, 100% elsewhere	\$194640
$p_2(e_1)$	100% in all ports	100% in all ports	100% in all ports	100% in all ports	40.63% in E, 100% elsewhere	\$171850
$p_3(e_1)$	100% in all ports	100% in all ports	100% in all ports	100% in all ports	43.75% in E, 100% elsewhere	\$193370
$p_1(e_2)$	100% in all ports	100% in all ports	100% in all ports	75.00% in E, 100% elsewhere	0.00% in E, 100% elsewhere	\$157040
$p_2(e_2)$	100% in all ports	100% in all ports	100% in all ports	100% in all ports	40.63% in E, 100% elsewhere	\$172750
$p_3(e_2)$	100% in all ports	100% in all ports	100% in all ports	100% in all ports	40.63% in E, 100% elsewhere	\$192420

$p_1(e_1)$ - Policy p_1 in evolution e_1
$p_2(e_1)$ - Policy p_2 in evolution e_1
$p_3(e_1)$ - Policy p_3 in evolution e_1
$p_1(e_1)$ - Policy p_1 in evolution e_2
$p_2(e_2)$ - Policy p_2 in evolution e_2
$p_3(e_2)$ - Policy p_3 in evolution e_2

Table 3.3: Complete disruption: Demand fulfilment percentages and operating costs.

The next subsections analyse the results obtained for the two different evolutions of the future: e_1 in which normal operations occur in each period of the planning horizon, and e_2 in which the complete disruption occurs at some point in time.

Evolution e_1 - Normal Operations

When normal operations occur (evolution e_1), the policy that provide highest demand fulfilment percentages is policy p_1. Operating costs amount to \$194640 and, with respect to the demand fulfilment percentages, only 56.25% of demands in E, and 100% elsewhere is met at day 8, whereas all demand are satisfied in each port in the other periods of the planning horizon.

The deterministic policy p_2 and the multi-scenario one p_3 are less effective in

terms of demand fulfilment percentages. More precisely, with respect to policy p_1:

- the deterministic policy p_2 represents the worst policy. This policy satisfies all demands in the first 7 days, but the demand fulfilment percentage of day 8 in port E is lower. Indeed, policy p_2 satisfies only the 40.63% of demands in port E, and 100% elsewhere at day 8.
- the multi-scenario policy p_3 satisfies all demands in the first 7 days, but the demand fulfilment percentage of day 8 in port E is lower. In fact this policy satisfies only the 43.75% of demands in port E, and 100% elsewhere at day 8.

From the point of view of the operating cost, policy p_2 is the best one with an amount of \$171850. Policies p_1 and p_3 provide higher costs with an amount of \$194640 and \$193370 respectively. However, since this comparison is based only on operating costs without take into account shortage costs, this consideration can be wrong. More precisely, with respect to operating costs, policy p_3 presents an amount of \$193370 but, with respect to policy p_2, the company has better demand fulfilment percentages. If the company gives a great importance to the customer satisfaction, then shortage costs may be too high and can lead policy p_3 more convenient than policy p_2.

Evolution e_2 - Uncertain Operations

When the complete disruption occur (evolution e_2), the policy that provide highest demand fulfilment percentages are both policy p_2 and policy p_3. Operating costs amount to \$172750 and \$192420 respectively, that leads the deterministic policy p_2 better than the multi-scenario one p_3. The demand fulfilment percentages of policies p_2 and p_3 are the same: only 40.63% of demands in port E, and 100% elsewhere, are met at day 8, whereas all demands are satisfied in each port in the other periods of the planning horizon.

The deterministic policy p_1 is less effective in terms of demand fulfilment percentages. More precisely, policy p_1 provides lower demand fulfilment percentage at day 7 and day 8. In fact this policy satisfies only 75.00% of demands in port E, and 100% elsewhere at day 7 and 0.00% of demands in port E, and 100% elsewhere at day 8. All demand are satisfied in each port in the other periods of the planning horizon.

From the point of view of operating costs, policy p_1 is the best one with an amount of \$157040. Policies p_2 and p_3 provide higher costs with an amount of \$172750 and \$192420 respectively. However, since this comparison is based only on operating costs without take into account shortage costs, this consideration can be wrong. More precisely, with respect to operating costs, policy p_3 presents an

amount of \$192420 but, with respect to policy p_1, the company has better demand fulfilment percentages. If the company gives a great importance to the customer satisfaction, then shortage costs may be too high and can lead policy p_3 more convenient than policy p_1.

Remarks

Generally speaking, as in Section 3.4, deterministic policies seem effective only when the future will evolve as expected, and seem unable to reposition empty containers reliably.

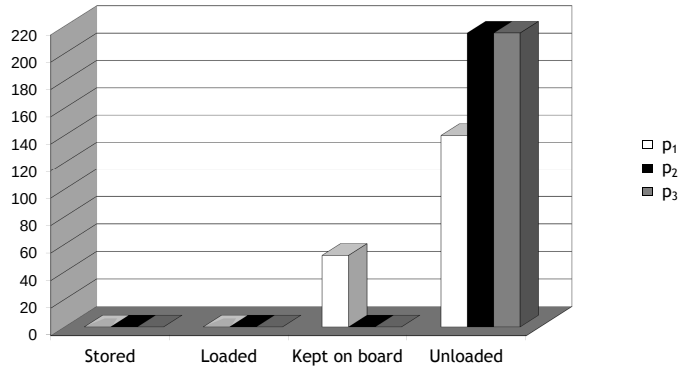


Figure 3.10: Complete disruption: Decisions taken in the first call of the vessel in port E.

Figure 3.10 represents for each policy the decisions taken on the number empty containers stored, loaded, kept on board, and unloaded in port E during the first call of the vessel of line 2.

Figure 3.10 shows that there are no empty containers in stock before the arrival of the vessel, and several containers are unloaded in order to meet demands. If policy p_1 is adopted, the number of empty containers unloaded is lower than the number of containers unloaded by policies p_2 and p_3. Policy p_1 does not foresee the complete disruption and relies on the second call of the vessel to unload an additional number of empty containers. In order to minimize costs, policy p_1 kept on board the volume of empty containers unloaded by policies to p_2 and p_3.

Figure 3.11 and Figure 3.12 represent for each policy the decisions taken on the number of empty containers stored, loaded, kept on board, and unloaded in port E during the second call of the vessel, according to evolution e_1 and evolution e_2 respectively.

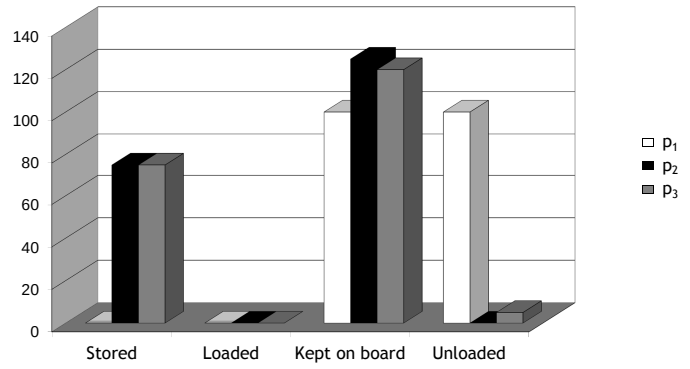


Figure 3.11: Complete disruption: Decisions taken in the second call of the vessel in port E. Evolution e_1.

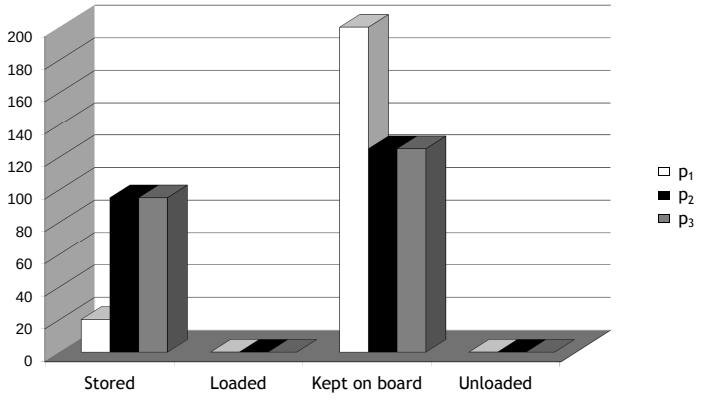


Figure 3.12: Complete disruption: Decisions taken in the second call of the vessel in port E. Evolution e_2.

When the complete disruption does not occur and policy p_1 is adopted, there are no containers in stock before the arrival of the vessel, and empty containers are unloaded in order to meet demands up to the third call of the vessel in port E. No empty containers are unloaded according to policy p_2, even if this opportunity is allowed by the realization of normal operations. This confirm the lack of flexibility to exploit new information, whereas additional containers are unloaded during second call of the vessel according to policy p_3.

Figure 3.12 shows that no empty containers are unloaded in port E when the complete disruption occurs. The least effective policy is p_1, because many empty

containers are kept on board, whereas they were supposed unloaded. Moreover, policies p_2 and p_3 can face container shortages by empty containers in stock in port E, unloaded during the first call of the vessel.

Chapter 4

Conclusions

Disruptions generate shocks in the shipping industry, as well as in several planning activities. This part of the thesis, dedicated to the seaside problem, investigated the planning of empty containers repositioning when uncertain events or disruptions may take place. They are supposed to be temporary events occurring in a port that prevent the empty containers from being properly repositioned in the maritime network.

We investigated three repositioning policies, and for each policy two different possible evolutions were taken into account, one with normal operations and the other in which uncertain operations occur. Tests show that deterministic formulations are effective only when events occur as forecast. Generally speaking, the deterministic formulations lead to unsuitable decisions whenever the future differs from the point forecast. However, the future is uncertain and shipping companies cannot trust to luck, or use a "crystal ball" to observe the values of uncertain parameters. A viable method for repositioning empty containers under uncertainty is provided by a multi-scenario model, in which scenarios are linked by non-anticipativity conditions. Tests show that the multi-scenario model provides highest demand fulfilment percentages for different future evolutions with respect to deterministic approaches.

Most of concepts presented in this first part of the thesis are included in Di Francesco, Lai, and Zuddas 2013, and were presented in Di Francesco, Lai, and Zuddas 2010 and Lai, Di Francesco, and Zuddas 2011.

Part II

Landside - Heterogeneous fleet size

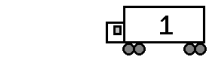
Chapter 5

Problem description

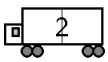
This part of the thesis addresses a vehicle routing problem, which is motivated by a real case study. A carrier must plan the distribution of container loads by trucks and containers based at the port. Trucks can carry one container, as in Figure 5.1a, or two containers, as in Figure 5.1b. Trucks and containers are used to service two types of transportation requests: the delivery of container loads from the port to importers, Figure 5.1c, and the shipment of container loads from exporters to the port, Figure 5.1d. Typically customers need to ship or receive more than one container load. Therefore, usually each customer must be serviced by multiple containers and must be visited by more than one truck. Figure 5.1e represents an example of multiple visits in which customers are importers.

According to the carrier's policy, trucks and containers cannot be uncoupled during customer service, because truck drivers are in charge of checking the right execution of packing and unpacking operations. As a result, when importers receive container loads by trucks, containers are emptied and moved away by the same trucks used for providing container loads. Similarly, when exporters are served by empty containers, containers are filled and moved away by the same trucks used for providing empty containers. It is important to note that there are no pickups or deliveries of loaded and empty containers: during customer service containers are filled or emptied and moved away by the same trucks used for moving containers to customers.

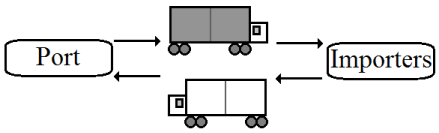
Moreover, since the container loads of exporters are typically not ready before the afternoon, the carrier policy is to service importers before exporters. As a result, routes may consist in the shipment of container loads from the port to importers, the allocation of empty containers from importers to exporters, and the final shipment of container loads from exporters to the port. Trucks with one container can service up to two customers in a route (one importer and one exporter). Trucks with two containers can service up to four customers in a route



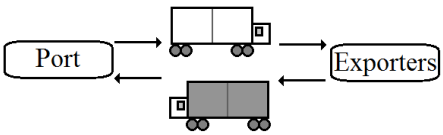
(a) *Truck of capacity one container*



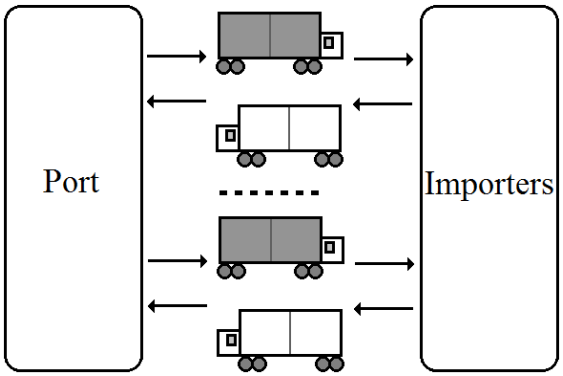
(b) *Truck of capacity two containers*



(c) *Transportation request of importers*



(d) *Transportation request of exporters*



(e) *Multiple visits example*

Figure 5.1: Heterogeneous landside problem description.

(two importers and two exporters). Every pair of containers can be shipped in a truck, all containers leaving from importers can be used to service exporters and no incompatibility occurs between customers and trucks, which can service almost any customer. Figure 5.2 represents an example in which an importer requests two container loads. Next, containers are emptied and moved to an exporter in which one of those containers is filled and shipped to the port.

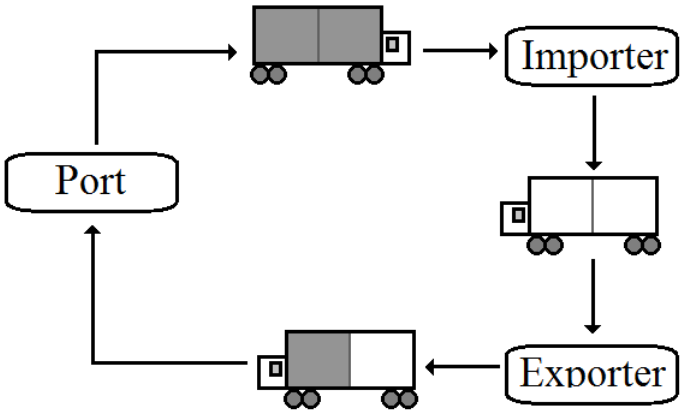


Figure 5.2: An example of route

It is important to note that the number of container loads to be picked up and

delivered is generally different. When the number of container loads delivered to importers is larger than the number of container loads shipped by exporters, a number of empty containers must be moved back to the port. When the number of container loads delivered to importers is lower than the number of container loads shipped by exporters, a number of empty containers must be put on trucks leaving from the port, in order to service all customers.

According to Parragh, Doerner, and Hartl 2008, this problem belongs to the class of Vehicle Routing Problems with Clustered Backhauls (*VRPCB*), because in each route all deliveries must be performed before all pickups. However, in classical *VRPCB*, each customer must be visited only once, whereas in this problem multiple visits at each customer are allowed.

This part of the thesis aims to propose an optimization model accounting for the original characteristics of this problem. The movement of trucks generates routing costs. In this problem setting, trucks with two containers capacity lead to higher costs per unitary distance than trucks with one container capacity. Moreover, handling costs are paid to put containers on trucks at the port. The objective is to determine a set of routes in order to minimize routing and handling costs, such that customers are serviced as requested, truck capacity constraints hold, and importers are serviced before exporters.

A linear integer programming formulation for the problem will be proposed in Chapter 6, and tested in Chapter 8. As tests will show, the complexity of the model leads the exact formulation able to solve only instances with few customers. Moreover, in the real case study, solutions must be determined rapidly. Therefore, another objective of this part of the thesis, is to propose an efficient meta-heuristic for the problem at hand. The scheme of the meta-heuristic, presented in Chapter 7, is to find an initial feasible solution and, next, to improve this solution by several local search phases. In Chapter 8, the results of the computational experience are presented, and a comparison between the performance of the exact and the meta-heuristic algorithm is reported. Finally, conclusions are summarized in Chapter 9.

5.1 Literature review

This problem belongs to the field of pickup and delivery problems, because there are two types of customers needing to ship or receive container loads Savelsbergh and Sol 1995. According to Parragh, Doerner, and Hartl 2008, the problem can be referred to as a Vehicle Routing Problem with Backhauls, because container loads must be shipped from the port to importers (or linehaul customers) and from exporters (or backhaul customers) to the port. More precisely, the thesis investigates a Vehicle Routing Problem with Clustered Backhauls (VRPCB), because in each

route all importers must be serviced before all exporters. In Berbeglia et al. 2007 the problem is referred to as a one-to-many-to-one pick up and delivery problem.

However, in traditional VRPCB each customer must be visited exactly once, whereas in this problem the demand of each customer may be greater than the vehicle capacity and may be serviced by several vehicles. Therefore, multiple visits are allowed Archetti, Speranza, and Hertz 2006. To our knowledge, Vehicle Routing Problems with Backhauls and splits have been investigated only in Mitra 2005 and Mitra 2008. Unlike this problem setting, the fleet of trucks is homogeneous and the delivery and pickup of loads can be in any sequence.

In the field of freight transportation, container transportation by truck between customers and intermodal terminals is known as "drayage". According to Macharis and Bontekoning 2004, it involves the delivery of a full container from an intermodal terminal to a receiver and the following collection of an empty container, as well as the provision of an empty container to the shipper and the subsequent transportation of a full trailer or container to the intermodal terminal.

According to the previous definition of drayage, it is possible to separate or not trucks and containers. The separation of trucks and containers was investigated by Jula et al. 2005, Chung et al. 2007, Zhang, Yun, and Moon 2011, Zhang, Yun, and Kopfer 2010, Vidovic, Radivojevic, and Rakovic 2011. Less attention was devoted to the case in which the tractor and truck drivers stay with containers during packing or unpacking operations. Due to the lower productivity in this use of trucks, this case was investigated only in papers motivated by specific technical restrictions Imai, Nishimura, and Current 2007 or particular regulatory policies Cheung et al. 2008. In this thesis, trucks and containers cannot be uncoupled because the carrier aims at providing a higher quality service, in which truck drivers are involved in the control of packing and unpacking operations.

To our knowledge, most of the papers on drayage assume that vehicles transport exactly one container at a time Jula et al. 2005, Namboothiri and Erera 2008, Zhang, Yun, and Moon 2011, Zhang, Yun, and Kopfer 2010. However, if a given weight limitation is not exceeded or special combined chassis are used, carrying two containers per truck is allowed in many countries, . Hence, moving two containers instead of one is an opportunity to improve the efficiency of the distribution. It is important to note that this opportunity makes routing problems more difficult to solve due to their combinatorial characteristics: a truck with a single container moves one loaded or one empty container, whereas a truck with two containers moves two loaded containers or two empty containers or a loaded container and an empty container. The possibility of moving two containers per truck was considered only in Chung et al. 2007, who worked on an one-to-one pick up and delivery problem, and Vidovic, Radivojevic, and Rakovic 2011, who determined routes from

the matching of pick up and delivery nodes.

The closest problem setting was probably faced by Imai, Nishimura, and Current 2007, who studied the optimal assignment of trucks to a set of delivery and pickup pairs. As in this problem setting, trucks and containers cannot be uncoupled, but the capacity of trucks is limited to one container only. Caris and Janssens 2009 extended the paper by Imai, Nishimura, and Current 2007 with time-windows at customers and depot. Their problem was modelled as a full truckload pickup and delivery problem with time windows. Even this approach is viable only if the transportation capacity of trucks is one container.

Chapter 6

Modeling

This chapter concerns the formalization of the mathematical model. The model involves sets, data and parameters. All sets, data and parameters are listed following their alphabetical order.

SETS

I**:** Set of importers;

E**:** Set of exporters;

K**:** Set of trucks.

INPUT DATA

d_i**:** Containers demand of customer $i \in I \cup E$. When $i \in I$, d_i represents the number of loaded containers used to service import customer $i \in I$. Due to the problem setting, d_i is also equal to the number of empty containers available after the service to customer $i \in I$. When $i \in E$, d_i represents the number of empty containers used to service export customer $i \in E$, and d_i is also equal to the number of loaded containers shipped by customer $i \in E$ to port p;

h_{pj}^k**:** Handling cost of a container put on truck $k \in K$ at the port p to service customer $j \in I \cup E$;

p**:** The port;

u^k**:** Capacity of truck $k \in K$.

Consider a directed graph $G = (N,A)$. The set of nodes is defined as the union of the port with the sets of importers and exporters:

$$N = \{p \cup I \cup E\}.$$

The set of arcs A includes all allowed ways to move trucks:

- from the port to any importer and any exporter;
- from an importer to the port, any other importer and any exporter;
- from an exporter to the port and any other exporter.

More formally, the set A is defined as $A = A_1 \cup A_2$, where

$$A_1 = \{(i,j)|i \in p \cup I, j \in N, i \neq j\}$$
$$A_2 = \{(i,j)|i \in E, j \in p \cup E, i \neq j\}.$$

DECISION VARIABLES

x_{ij}^k**:** Routing selection variable equal to 1 if arc $(i,j) \in A$ is traversed by truck $k \in K$, 0 otherwise. c_{ij}^k represents its routing cost;

y_{ij}^k**:** Number of loaded containers moved along arc $(i,j) \in A$ by truck $k \in K$;

z_{ij}^k**:** Number of empty containers moved along arc $(i,j) \in A$ by truck $k \in K$.

The problem can be formulated as follows:

$$min \sum_{k \in K} \left[\sum_{(i,j) \in A} c_{ij}^k x_{ij}^k + \sum_{j \in N} h_{pj}^k (y_{pj}^k + z_{pj}^k) \right] \tag{6.1}$$

s.t.

$$\sum_{k \in K} \sum_{l \in N} y_{il}^k = \sum_{k \in K} \sum_{j \in p \cup I} y_{ji}^k - d_i \qquad \forall i \in I \tag{6.2}$$

$$\sum_{k \in K} \sum_{l \in N} z_{il}^k = \sum_{k \in K} \sum_{j \in p \cup I} z_{ji}^k + d_i \qquad \forall i \in I \tag{6.3}$$

$$\sum_{l \in N} y_{il}^k \leq \sum_{j \in p \cup I} y_{ji}^k \qquad \forall i \in I, \forall k \in K \tag{6.4}$$

$$\sum_{l \in N} z_{il}^k \geq \sum_{j \in p \cup I} z_{ji}^k \qquad \forall i \in I, \forall k \in K \tag{6.5}$$

$$\sum_{k \in K} \sum_{l \in p \cup E} y_{il}^k = \sum_{k \in K} \sum_{j \in N} y_{ji}^k + d_i \qquad \forall i \in E \tag{6.6}$$

$$\sum_{k \in K} \sum_{l \in p \cup E} z_{il}^k = \sum_{k \in K} \sum_{j \in N} z_{ji}^k - d_i \qquad \forall i \in E \tag{6.7}$$

$$\sum_{l \in p \cup E} y_{il}^k \geq \sum_{j \in N} y_{ji}^k \qquad \forall i \in E, \forall k \in K \tag{6.8}$$

$$\sum_{l \in p \cup E} z_{il}^k \leq \sum_{j \in N} z_{ji}^k \qquad \forall i \in E, \forall k \in K \tag{6.9}$$

$$\sum_{(ji)\in A}(y_{ji}^k + z_{ji}^k) = \sum_{(il)\in A}(y_{il}^k + z_{il}^k) \qquad \forall i \in I \cup E, \forall k \in K \qquad (6.10)$$

$$y_{ij}^k + z_{ij}^k \leq u^k x_{ij}^k \qquad \forall (i,j) \in A, \forall k \in K \qquad (6.11)$$

$$\sum_{j\in N} x_{ji}^k - \sum_{l\in N} x_{il}^k = 0 \qquad \forall i \in N, \forall k \in K \qquad (6.12)$$

$$\sum_{j\in N} x_{pj}^k \leq 1 \qquad \forall k \in K \qquad (6.13)$$

$$\sum_{k\in K}\sum_{i\in I\cup E} z_{ip}^k - \sum_{k\in K}\sum_{i\in I\cup E} z_{pi}^k = \sum_{i\in I} d_i - \sum_{i\in E} d_i \qquad (6.14)$$

$$x_{ij}^k \in \{0,1\} \qquad \forall (i,j) \in A, \forall k \in K \qquad (6.15)$$

$$y_{ij}^k \in \{0,1,2\} \qquad \forall (i,j) \in A, \forall k \in K \qquad (6.16)$$

$$z_{ij}^k \in \{0,1,2\} \qquad \forall (i,j) \in A, \forall k \in K \qquad (6.17)$$

Costs are minimized in the objective function (6.1). The objective function takes into account both sources of cost: those relative to the travelled arcs and those concerning the handling of containers at the port.

Constraints (6.2)-(6.5) concern the movement of containers to importers. Constraints (6.2) and (6.3) are the flow conservation constraints of loaded and empty containers, respectively, at each importer node. Constraints (6.4) and (6.5) check the number of loaded and empty containers in each truck entering and leaving from importers: the number of loaded containers cannot be increased after a service at each importer, whereas the number of empty containers cannot be reduced.

Constraints (6.6)-(6.9) concern the allocation of containers to exporters. Constraints (6.6) and (6.7) are the flow conservation constraints of loaded and empty containers, respectively, for each exporter node. Constraints (6.8) and (6.9) control the number of loaded and empty containers in each truck entering and leaving from exporters: the number of loaded containers cannot be reduced after a service at each exporter, whereas the number of empty containers cannot be increased.

Constraints (6.10) guarantee that the number of containers carried by each truck do not change after visiting a customer.

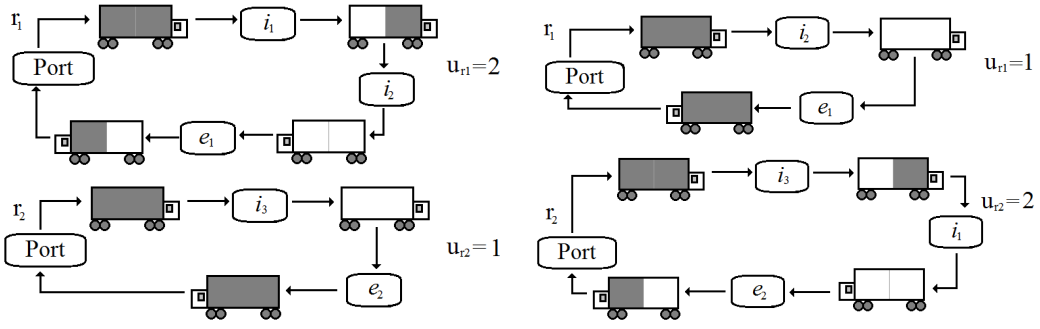
Constraints (6.11) impose that the number of containers moved by each truck is not larger than its transportation capacity u^k. In this problem, u^k takes value 1 or 2. Constraints (6.12) concerns the flow conservation constraints for trucks at each node. Constraints (6.13) guarantee that trucks are not used more than once. Constraint (6.14) represents the flow conservation of empty containers at the port p.

Finally, constraints (6.15), (6.16) and (6.17) define the domain of the decision variables.

Chapter 7

Solution method

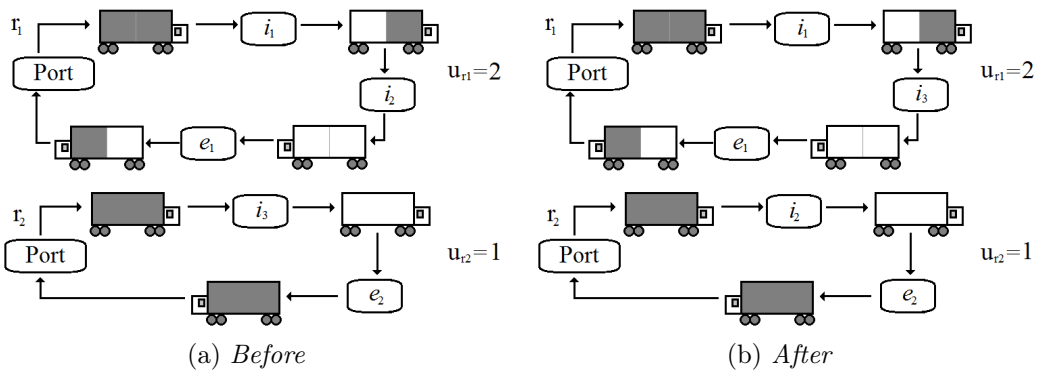
Vehicle Routing Problems are known to be difficult and this problem is not an exception. Since exact methods may not be able to solve real problem instances, this thesis propose a meta-heuristic, in which solutions are defined in terms of truck routes. The meta-heuristic consists of two phases: *(i) Constructive phase*, determines a feasible solution by a variant of the Clarke and Wright method, in which routes are merged and assigned to trucks; *(ii) Improvement phase*, improves the solution by several local search phases. The search space of the local search phases is the set of truck assignments to routes satisfying all constraints (6.2)-(6.17). Two neighbourhoods are used: *(i) Node Relocate* , in which a node is moved from its current route, inserted into another route by the best-insertion method, and trucks are reassigned to routes involved in the local move, as in Figure 7.1; *(ii) Node Exchange* , in which two nodes are swapped between two different routes, and trucks are reassigned to routes involved in the local move, as in Figure 7.2.



(a) *Before* (b) *After*

Figure 7.1: Improvement phase: *Node Relocate* example.

In both phases, the *Constructive phase* and the *Improvement phase*, the meta-heuristic determines new solutions evaluating pairs of routes in the current solution, selecting one of them and assigning trucks to the selected pair of routes. In the *Constructive phase*, the meta-heuristic selects which pair of routes should be merged



(a) *Before* (b) *After*

Figure 7.2: Improvement phase: *Node Exchange* example.

and which truck should be used in the new route. In the *Improvement phase*, the meta-heuristic evaluates how nodes are moved between any pair of routes, and selects the pair of routes involved in the best local movement. The trucks assigned to the routes of the selected pair can be swapped, if the solution improves.

It is important to note that there is not a single way to compare pairs of routes. Most importantly, different comparison criteria may result in the selection of different pairs of routes at any step. In this thesis, two criteria are proposed for the selection of the best pair of routes: *(i)* $Criterion1$, selects pairs of routes such that the objective function in the new solution is minimum; *(ii)* $Criterion2$, selects pairs of routes such that the total travelled distance in the new solution is minimum. Since it is hardly possible know a priori which criterion finally determines the best solution in terms of objective function, both of them are considered in the *Constructive phase* and the *Improvement phase*.

The pseudo-code of the meta-heuristic algorithm is illustrated in Table 7.1, in which the following notation is adopted:

initSol Initial solution;

initList List of savings that can be obtained merging routes in the initial solution;

cwSol(Criterion) Current solution of the *Constructive phase*, for the considered criterion;

cwList(Criterion) List of savings of the current solution $cwSol(Criterion)$ in the *Constructive phase*, for the considered criterion;

ClarkeWright($cwSol(Criterion)$,$cwList(Criterion)$**)** Function that implements the variant of the Clarke and Wright method in the *Constructive phase*. The input parameters are the current solution, $cwSol(Criterion)$, and the current list of savings, $cwList(Criterion)$. The output is the new current solution, $cwSol(Criterion)$;

Sol(Criterion) Current solution in the *Improvement phase*, for the considered criterion;

improve Boolean variable taking value true if the *Improvement phase* improves the solution, false otherwise. It is initialized to true, in order to perform the *Improvement phase* at least once.

improveNodeRelocate Boolean variable taking value true if the *Node Relocate* search phase improves the current solution in the *Improvement phase*, false otherwise. It is initialized to false.

improveNodeExchange Boolean variable taking value true if the *Node Exchange* search phase improves the current solution in the *Improvement phase*, false otherwise. It is initialized to false.

nodeRelocate($Sol(Criterion)$,$improveNodeRelocate$**)** Function that implements the *Node Relocate* search phase. The input parameters are the current solution $Sol(Criterion)$, and the variable $improveNodeRelocate$.
It returns the new current solution, $Sol(Criterion)$, and the updated value of $improveNodeRelocate$.

nodeExchange($Sol(Criterion)$,$improveNodeExchange$**)** Function that implements the *Node Exchange* search phase. The input parameters are the current solution, $Sol(Criterion)$, and the variable $improveNodeExchange$.
It returns the new current solution, $Sol(Criterion)$, and the updated value of $improveNodeExchange$.

$\mathbf{S}^*$ Best solution found.

7.1 Initialization

The initial solution *cwSol* is made up with direct trips, in which vehicles carrying one container start from the port, go directly to a customer, and then turn back directly to the port. Although in this solution all customers are served as requested, it is unfeasible when the number of available trucks with capacity one-container is not sufficient.

initList is initialized in order to implement the variant of the Clarke and Wright method, Section 7.2. Each entry of *initList* represents:

- The identifiers of the pair of routes which may be merged;

```
procedure MAIN
    Initialize                                                    ▷ Section 7.1
    Create initSol with direct trips performed by trucks carrying one container only
    Create the savings list initList
    EndInitialize
    for Criterion = 1, 2 do
        cwSol(Criterion) ← initSol
        cwList(Criterion) ← initList
        ClarkeWright(cwSol(Criterion),cwList(Criterion))          ▷ Section 7.2
        improve ← true
        Sol(Criterion) ← cwSol(Criterion)
        while improve == true do                                  ▷ Section 7.3
            improve ← false
            improveNodeRelocate ← false
            improveNodeExchange ← false
            nodeRelocate(Sol(Criterion),improveNodeRelocate)
            nodeExchange(Sol(Criterion),improveNodeExchange)
            if improveNodeRelocate == true OR improveNodeExchange == true
then
                improve ← true
            end if
        end while
    end for
    S* ← ∅
    S* ← min{Sol(1), Sol(2)}
    return S*
end procedure
```

Table 7.1: The structure of the meta-heuristic.

- The maximum saving deriving from their merging. To clarify, since a pair of routes can sometimes be merged in several ways, this thesis consider only the new route resulting in the maximum saving;

- The truck capacity $\check{v}$ of the new route, generated after the merging. In this specific problem setting, if the new route consists of 1 importer and 1 exporter demanding 1 container each, $\check{v}$ takes value 1, whereas it takes value 2 otherwise.

Given two generic routes r_1 and r_2, the maximum saving generated by their merging is computed as:

$$cost(r_1) + cost(r_2) - cost(r).$$

In which: $cost(r_1)$ and $cost(r_2)$ represent the cost of routes r_1 and r_2, respectively, and $cost(r)$ the cost of the route obtained by merging r_1 and r_2. Savings are ordered in the *initList* in a non-increasing fashion.

7.2 Constructive phase

In order to reduce the number of routes to the number of available trucks, the variant of the Clarke and Wright method is implemented as follows:

$Step_0$ The current solution, *cwSol*($Criterion$), and the associated list of savings, *cwList*($Criterion$), are considered;

$Step_1$ If the number of routes is lower than the number of trucks, or if there are no entries in *cwList*($Criterion$), save the current solution, *cwSol*($Criterion$), and stop, otherwise go to $Step_2$;

$Step_2$ Consider the pair of routes at the top of *cwList*($Criterion$) and check the truck capacity $\check{v}$. If there is an available truck having capacity $\check{v}$, merge the pair of routes, assign the truck to the new route and go to $Step_5$, otherwise go to $Step_3$;

$Step_3$ If a route of the pair has capacity $\check{v}$, merge the pair of routes, assign the truck to the new route and go to $Step_5$, otherwise go to $Step_4$;

$Step_4$ Scroll the *cwList*($Criterion$) and check the next entry. If *cwList*($Criterion$) is not empty go to $Step_2$, otherwise return the current solution, *cwSol*($Criterion$), and stop;

$Step_5$ Save the new current solution, *cwSol*($Criterion$), re-compute the list of savings, *cwList*($Criterion$), and repeat from $Step_1$.

After previous steps, the solution may not be feasible, because the final number of routes was not controlled. If the final number of routes is lower than or equal to the number of available trucks, the current solution is feasible and the meta-heuristic switches to the *Improvement phase* (section 7.3). If this is not the case, the algorithm turns all routes performed by trucks with capacity one-container into direct trips. The new current solution, *cwSol*($Criterion$), is made up of all routes performed by trucks with capacity two-containers, as in the previous solution, and all direct trips performed by trucks with capacity one-container. Next, *cwList* is re-computed and the previous steps of the Clarke and Wright's method are repeated. In order to not cycle back to the previous solution, $Step_3$ does not take into account trucks with capacity one-container. After this phase, the solution is feasible and the meta-heuristic proceeds with the *Improvement phase*.

7.3 Improvement phase

The *Improvement phase* consists of several local search phases using two neighbourhoods. The first neighbourhood is the set of feasible solutions obtained by relocating a node from a route to another route by the best insertion method. When a node is relocated without reassigning trucks to routes, solutions may be infeasible. In such cases, sometimes feasible solutions can be restored swapping the trucks previously assigned to the routes involved in the move. Therefore, every time a node is moved into another route, trucks involved in the local move are reassigned. The second neighbourhood is the set of feasible solutions obtained by exchanging two nodes between two routes, and reassigning trucks to routes. As in the first neighbourhood, trucks assigned to routes can be swapped, in order to determine feasible solutions.

The *Improvement phase* starts from the feasible solution determined in the *Constructive phase* and, first of all, performs the *Node Relocate* search phase. It starts from the first route, which is in the top of the list of routes. If there are improving moves for the first route, the best one is selected and implemented. Next, the two routes involved in the move are inserted into the bottom of the list of routes in the new solution. The list of routes is scrolled down from the top, in order to not limit local moves to a subset of routes. After an improving move, the list of routes is updated and the search restarts from the top of the list. If there are not improving moves for the first route, the best not improving move is saved and the search goes on with the following route.

After a series of local moves, a local optimum is encountered in which improving moves are no longer available. In such case, the best not-improving move is implemented, the route list is updated, and the search restarts from the top of the list. In order to not repeat moves during the search, moves already implemented are denied and saved in two separate memories: one for the *improving moves* and one for the *not improving moves*. In order to not reject possible moves for too long, *improving moves* are deleted from their memory whenever a *not improving move* is implemented, whereas *not improving move* are deleted when a new local optimum is determined.

After the consecutive implementation of the maximum number of not improving moves, the search in the first neighbourhood stops, and goes on in the second neighbourhood. If the search in at least a neighbourhood improves the solution, the search is carried out again starting from the first neighbourhood and, next, in the second one.

The outline of the local search in a neighbourhood is the following:

$Step_0$ $notImpIt = 0$;

$Step_1$ If $notImpIt$ is equal to $maxnotImpIt$, the method stops its execution. Otherwise increase $notImpIt$ by 1 and go to the next step;

$Step_2$ If any, search the best *improving move* and go to the next step. Otherwise go to $Step_4$ (during this search, the best *not improving move* is saved);

$Step_3$ Implement the best *improving move* and go to $Step_2$;

$Step_4$ If the solution has been improved save the new local optimum, $Sol(Criterion)$, and set $notImpIt = 0$. Next, go to $Step_5$;

$Step_5$ If any, select the best *not improving move*, implement it, and go to $Step_1$. Otherwise the method stops its execution.

$Where$:

notImpIt is the number of iterations for which a not improving move has been consecutively implemented. When a new local optimum is found, $notimpIt$ is set to 0; when a not improving move is implemented, $notimpIt$ is increased by 1.

maxnotImpIt is the maximum number of iterations for the not improving moves.

Chapter 8

Experimentation

The experimentation proposed in this chapter aims to analyse the performance of the proposed solution method, and to verify that problems of real-life dimensions can be solved in reasonable time. The meta-heuristic is tested on five real instances provided by a carrier operating in the area of Vado Ligure (Italy), as well as on several randomly generated problem instances, that have structures closely similar to real problems. Moreover, according to meeting with the carrier, the time estimated in which solutions must be provided is 10 minutes.

This experimentation considers 70 randomly generated artificial instances, which are divided into five classes:

- 6 instances with 10 customers;
- 10 instances with 20 customers;
- 14 instances with 30 customers;
- 18 instances with 40 customers;
- 22 instances with 50 customers.

The number of containers requested by each customer is uniformly generated from 1 to 5. In each class the coordinates of nodes are fixed. The instances of a class differ in the number of importers, in the number of exporters, and in the number of available trucks for each container capacity. The available trucks in each instances are equal to the minimum number of trucks needed to service all container load requests. Moreover, in each class: *(i)* in the first half of the instances, trucks with capacity 2-containers service two-thirds of all container loads requests, whilst the remaining part is serviced by trucks with capacity 1-container; *(ii)* in the second half of instances, trucks with capacity 2-containers service all containers load requests, and trucks with capacity 1-container are not available.

8.1 Data management

The exact formulation is coded using IBM ILOG CPLEX Optimization Studio 12.5 and solved by the Branch & Bound of ILOG CPLEX 12.2, which employs state-of-the-art algorithms and techniques to solve mixed integer programming problems. The meta-heuristic is coded in the programming language C++. Experiments are performed on a Linux four-CPU server 2.67GHz 64GB RAM, with default parameter settings.

Although a major requirement for the carrier is to determine solutions in about 10 minutes, ILOG CPLEX 12.2 is set to stop after 3 hours. This choice allows to evaluate the quality of solutions provided by the meta-heuristic when no feasible solution is determined by ILOG CPLEX 12.2 in 10 minutes. For the sake of clarity, we refer to the solutions obtained by the exact algorithm as CPLEX.

Two parameters must be calibrated in the meta-heuristic algorithm:

***maxnotImpIt*(*Node Relocate*):** maximum number of consecutive *not improving move* in the *Node Relocate* search phase;

***maxnotImpIt*(*Node Exchange*):** maximum number of consecutive *not improving move* in the *Node Exchange* search phase.

Denoting with $nrRouteSol$ the number of routes in the solution, the two parameters, $maxnotImpIt(Node\ Relocate)$ and $maxnotImpIt(Node\ Exchange)$, are defined as follows:

$$maxnotImpIt(Node\ Relocate) = nrRouteSol \cdot \alpha$$
$$maxnotImpIt(Node\ Exchange) = nrRouteSol \cdot \beta$$

Tests are performed by running all artificial instances with the chosen values of α and β. Table 8.1 shows the best eight calibrations performed for the two coefficient, α and β. C_1-C_8 denoted the value taken by these coefficients in each calibration. The following part of the experimentation shows the computational results for the best calibration only. The selected values of α and β are shown in the last column of the table.

	Calibrations								*Selected*
	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_5
α	1	2	3	4	5	6	7	8	5
β	1	2	3	4	5	6	7	8	5

Table 8.1: Tested value of α and β

8.2 Artificial instances

Computational results are presented in Tables 8.2, 8.3, 8.4, 8.5, and 8.6, with the following notation:

- $|I|$: Number of importers;
- $|E|$: Number of exporters;
- $|K_1|$: Number of available trucks with capacity 1-container;
- $|K_2|$: Number of available trucks with capacity 2-containers;

META-HEURISTIC

- *o.f.*: Objective function returned by the meta-heuristic.
- *Node Relocate*: Objective function improvements obtained in the *Node Relocate* search phases;
- *Node Exchange*: Objective function improvements obtained in the *Node Exchange* search phases;
- *Criterion*: Criterion that provided the best solution: 1 for *Criterion*1, 2 for *Criterion*2. When both criteria provide the same objective function *Criterion* takes value "X".
- *Gap criteria*: The difference between the two criteria, in terms of objective function;
- *t(s)*: Time in seconds before the best solution is found;
- *% Gap from CPLEX*: Percentage gap with respect to the best solution provided by CPLEX. When solutions of the meta-heuristic are better than CPLEX upper bounds, or the meta-heuristic provides the optimal solution, gaps are reported in bold. Moreover, when the solutions of the meta-heuristic are better than CPLEX upper bounds, gaps are reported with a negative value;
- *n.a.*: means comparison gap *not available*;

CPLEX

- *% Opt. Gap*: Gap between upper and lower bounds determined by CPLEX. The elapsed time in seconds is reported only when CPLEX finds the optimal solution;

- *n.s.*: means *no solution* determined by CPLEX within 3*hours*, nor optimal nor feasible.

				META-HEURISTIC							CPLEX
$\|I\|$	$\|E\|$	$\|K_1\|$	$\|K_2\|$	o.f	Node Relocate	Node Exchange	Criterion	Gap criteria	t(s)	% Gap from CPLEX	% Opt. Gap
2	8	2	9	20,135.28	426.54	0.00	2	44.48	0.00	**0.00**	0.00 (376.08s)
5	5	2	7	20,381.79	247.52	54.57	1	14.40	0.00	0.19	0.00 (5.21s)
8	2	5	9	21,072.44	0.00	271.45	1	90.22	0.00	0.80	0.00 (277.09s)
2	8	0	10	20,677.64	0.00	248.63	1	5.63	0.00	0.03	3.76
5	5	0	8	19,960.83	0.00	136.49	X	0.00	0.00	**0.00**	0.00 (29.70s)
8	2	0	12	20,697.67	131.29	232.18	X	0.00	0.00	0.00	3.83

Table 8.2: Artificial instances: 10 customers

Table 8.2 shows the results for the class of instances with 10 customers. CPLEX is able to solve optimally 4/6 instances within 10 minutes, as required by the carrier. The meta-heuristic can solve all instances in less than one second, and is able to solve optimally 2/6 instances.

The *Node Exchange* search phase improves the solution more often than the *Node Relocate*. Moreover, for half of the instances the best solution is returned by $Criterion1$, whereas the other times the two criteria return the same solution. In one instance only $Criterion2$ is better.

CPLEX provides a slightly better solution in 3/6 instances with:

- 5 importers, 5 exporters, 2 trucks of capacity 1-container, and 7 trucks of capacity 2-containers (0.19%);
- 8 importers, 2 exporters, 5 trucks of capacity 1-container, and 9 trucks of capacity 2-containers (0.80%);
- 2 importers, 8 exporters, none truck of capacity 1-container, and 10 trucks of capacity 2-containers (0.03%).

Generally speaking, gaps between the best upper bounds provided by CPLEX within 3 hours and the meta-heuristic solutions are lower than 1%.

Table 8.3 shows the results for the class of instances with 20 customers. CPLEX is able to solve optimally 1/10 instances within 3 hours, and none within 10 minutes. The meta-heuristic can solve most of instances in less than one second, and is able to solve optimally 1/10 instances.

The *Node Exchange* search phase improves the solution in few instances, whereas the *Node Relocate*one is not able to improve none of them. Moreover, most of times $Criterion1$ and $Criterion2$ return the same solution, or $Criterion2$ is better. $Criterion1$ provides the best solution in only one instance.

				META-HEURISTIC							CPLEX
$\|I\|$	$\|E\|$	$\|K_1\|$	$\|K_2\|$	o.f	Node Relocate	Node Exchange	Criterion	Gap criteria	t(s)	% Gap from CPLEX	% Opt. Gap
2	18	8	22	41,214.33	0.00	0.00	X	0.00	0.00	0.00	2.51
5	15	7	19	36,607.42	0.00	0.00	X	0.00	0.00	**0.00**	0.00 (5,845.77s)
10	10	5	14	32,333.11	0.00	306.83	1	426.18	1.00	0.56	2.26
15	5	7	19	37,657.58	0.00	0.00	2	15.10	1.00	0.02	2.87
18	2	5	24	42,781.65	0.00	461.78	2	7.17	0.00	0.00	2.53
2	18	0	26	40,806.04	0.00	0.00	X	0.00	0.00	**-0.91**	5.74
5	15	0	23	36,048.28	0.00	0.00	X	0.00	0.00	0.00	2.99
10	10	0	17	31,691.35	0.00	144.39	2	8.11	0.00	0.06	3.91
15	5	0	23	37,301.67	0.00	0.00	X	0.00	1.00	**-0.75**	6.01
18	2	0	27	43,015.11	0.00	0.00	2	8.76	0.00	**-0.04**	4.96

Table 8.3: Artificial instances: 20 customers

CPLEX provides a slightly better solution in 3/10 instances with:

- 10 importers, 10 exporters, 5 trucks of capacity 1-container, and 14 trucks of capacity 2-containers (0.56%);
- 15 importers, 5 exporters, 7 trucks of capacity 1-container, and 19 trucks of capacity 2-containers (0.02%);
- 10 importers, 10 exporters, none truck of capacity 1-container, and 17 trucks of capacity 2-containers (0.06%).

Generally speaking, gaps between the best upper bounds provided by CPLEX within 3 hours and the meta-heuristic solutions are lower than 1%.

It is important to note that the meta-heuristic provides a better solution than CPLEX in 3/10 instances. The best improvement of the meta-heuristic is found for the instance with 2 importers, 18 exporters, 0 truck with capacity 1-container, and 26 truck with capacity 2-containers, in which the improvement is about 0.91%.

Table 8.4 shows the results for the class of instances with 30 customers. CPLEX is not able to solve optimally any instance. The meta-heuristic can solve most of instances in less than five seconds.

The *Node Exchange* search phase improves the solution more often than the *Node Relocate*. Moreover, most of times the best solution is returned by $Criterion1$, whereas few times the two criteria return the same solution or $Criterion2$ is better. Furthermore, sometimes the best solution is much better than the other one. Therefore, the intuition of considering both criteria in the algorithm seems to be appropriate.

CPLEX provides a slightly better solution in 2/14 instances with:

- 10 importers, 20 exporters, 10 trucks of capacity 1-container, and 25 trucks of capacity 2-containers (0.43%);

				META-HEURISTIC							CPLEX
$\|I\|$	$\|E\|$	$\|K_1\|$	$\|K_2\|$	o.f	Node Relocate	Node Exchange	Criterion	Gap criteria	t(s)	% Gap from CPLEX	% Opt. Gap
2	28	13	33	64,525.58	0.00	0.00	1	294.72	3.00	**-5.06**	7.54
5	25	12	30	60,416.51	0.00	0.00	1	52.44	8.00	n.a.	n.s.
10	20	10	25	52,645.37	0.00	0.00	1	601.20	2.00	0.43	2.98
15	15	8	19	50,158.37	1,502.78	556.68	1	158.14	2.00	1.13	2.69
20	10	10	26	53,571.86	445.20	449.54	1	307.32	2.00	**-6.38**	9.08
25	5	12	32	62,111.73	0.00	0.00	1	127.90	1.00	**-0.26**	3.21
28	2	14	35	67,227.58	0.00	0.00	1	114.90	1.00	n.a.	n.s.
2	28	0	40	62,782.75	0.00	0.00	X	0.00	1.00	**-3.43**	7.02
5	25	0	36	58,974.78	0.00	333.19	2	96.96	1.00	**-7.96**	11.43
10	20	0	30	51,763.93	0.00	436.31	2	932.64	2.00	**-4.72**	9.38
15	15	0	23	48,940.66	0.13	909.49	2	184.18	1.00	**-0.85**	6.61
20	10	0	31	53,095.51	322.75	565.86	1	279.02	1.00	**-1.99**	7.16
25	5	0	38	60,565.79	148.25	0.00	1	58.92	0.00	**-3.71**	7.07
28	2	0	42	65,305.66	0.00	0.00	X	0.00	1.00	**-1.34**	3.79

Table 8.4: Artificial instances: 30 customers

- 15 importers, 15 exporters, 8 truck of capacity 1-container, and 19 trucks of capacity 2-containers (1.13%).

Generally speaking, gaps between the best upper bounds provided by CPLEX within 3 hours and the meta-heuristic solutions are lower than 1.5%.

It is important to note that the meta-heuristic provides a better solution than CPLEX in 10/14 instances. Comparison gaps are not available for 2/14 instances, because CPLEX does not provide a feasible solution.

The best improvement of the meta-heuristic is found for the instance with 5 importers, 25 exporters, 0 truck with capacity 1-container, and 36 truck with capacity 2-containers, in which the improvement is about 7.96%.

Table 8.5 shows the results for the class of instances with 40 customers. CPLEX is not able to solve optimally any instance. The meta-heuristic can solve most of instances in less than twenty seconds.

The *Node Exchange* search phase improves the solution more often than the *Node Relocate*. Moreover, most of times the best solution is returned by $Criterion2$, whereas the other times the two criteria return the same solution or $Criterion1$ is better. Furthermore, sometimes the best solution is much better than the other one. Therefore, as in the previous case, the intuition of considering both criteria in the algorithm seems to be appropriate.

It is important to note that the meta-heuristic provides a better solution than CPLEX in 2/18 instances .Comparison gaps are not available for 16/18 instances, because CPLEX does not provide a feasible solution.

Gaps between the best upper bounds provided by CPLEX and the meta-heuristic solutions are negative for both instances solved by CPLEX. Therefore, the meta-heuristic provides better solution than CPLEX. The best improvement of the

				META-HEURISTIC							CPLEX
$\|I\|$	$\|E\|$	$\|K_1\|$	$\|K_2\|$	o.f	Node Relocate	Node Exchange	Criterion	Gap criteria	t(s)	% Gap from CPLEX	% Opt. Gap
2	38	20	49	96,786.96	182.60	0.00	2	11.26	9.00	n.a.	n.s.
5	35	18	45	89,790.61	79.02	166.66	1	210.38	14.00	n.a.	n.s.
10	30	14	38	75,791.53	298.07	264.11	1	147.79	11.00	n.a.	n.s.
15	25	12	31	67,958.61	0.00	853.21	1	437.45	10.00	n.a.	n.s.
20	20	12	29	69,465.86	688.83	837.90	2	699.74	9.00	n.a.	n.s.
25	15	14	36	76,566.41	0.00	477.06	1	574.50	9.00	n.a.	n.s.
30	10	17	43	87,129.18	330.23	331.50	1	446.68	11.00	n.a.	n.s.
35	5	19	48	96,135.51	0.00	0.00	X	0.00	18.00	n.a.	n.s.
38	2	20	51	100,694.59	278.59	15.50	2	146.47	26.00	n.a.	n.s.
2	38	0	59	94,185.86	0.00	155.79	X	0.00	7.00	n.a.	n.s.
5	35	0	54	87,477.82	0.00	136.83	X	0.00	10.00	n.a.	n.s.
10	30	0	45	74,220.82	172.50	679.59	2	261.49	11.00	n.a.	n.s.
15	25	0	37	66,582.24	0.00	853.21	1	9.65	8.00	n.a.	n.s.
20	20	0	35	68,060.10	0.00	690.89	2	267.46	9.00	n.a.	n.s.
25	15	0	43	74,941.72	0.00	1,045.09	2	201.28	7.00	**-19.16**	18.69
30	10	0	51	85,442.64	0.00	57.17	2	105.16	7.00	n.a.	n.s.
35	5	0	58	93,870.67	20.22	0.00	X	0.00	25.00	n.a.	n.s.
38	2	0	61	97,832.13	0.00	178.24	X	0.00	23.00	**-13.26**	13.68

Table 8.5: Artificial instances: 40 customers

meta-heuristic is found for the instance with 25 importers, 15 exporters, 0 truck with capacity 1-container, and 43 truck with capacity 2-containers, in which the improvement is about 19.16%.

				META-HEURISTIC							CPLEX
$\|I\|$	$\|E\|$	$\|K_1\|$	$\|K_2\|$	o.f	Node Relocate	Node Exchange	Criterion	Gap criteria	t(s)	% Gap from CPLEX	% Opt. Gap
2	48	22	56	129,245.05	2.69	0.00	1	142.91	39.00	n.a.	n.s.
5	45	21	54	124,640.83	0.00	187.81	1	229.38	44.00	n.a.	n.s.
10	40	18	50	115,950.50	0.00	178.44	1	378.54	26.00	n.a.	n.s.
15	35	17	42	101,052.21	0.00	652.91	1	1,617.57	36.00	n.a.	n.s.
20	30	13	37	91,561.14	0.00	495.20	1	1,280.39	17.00	n.a.	n.s.
25	25	11	32	83,273.59	25.25	408.15	2	857.91	17.00	n.a.	n.s.
30	20	12	32	85,351.20	0.00	210.60	2	663.43	16.00	n.a.	n.s.
35	15	15	39	97,350.77	0.00	57.34	2	296.27	20.00	n.a.	n.s.
40	10	17	46	108,653.18	0.00	446.74	1	54.47	21.00	n.a.	n.s.
45	5	20	50	116,059.43	15.87	310.08	1	164.20	40.00	n.a.	n.s.
48	2	22	55	126,499.99	15.87	0.00	1	169.28	42.00	n.a.	n.s.
2	48	0	67	124,536.09	0.00	317.35	2	494.13	25.00	n.a.	n.s.
5	45	0	65	120,406.58	165.01	371.93	2	159.23	31.00	n.a.	n.s.
10	40	0	59	112,487.21	0.00	0.00	2	196.24	19.00	n.a.	n.s.
15	35	0	51	98,374.91	577.13	566.67	1	994.41	19.00	n.a.	n.s.
20	30	0	44	89,079.62	218.64	703.37	1	782.34	19.00	n.a.	n.s.
25	25	0	38	81,241.01	0.00	1,551.66	2	180.11	17.00	n.a.	n.s.
30	20	0	38	83,161.03	0.00	3,172.85	1	493.30	17.00	n.a.	n.s.
35	15	0	47	93,779.99	921.21	1,127.10	1	253.52	29.00	n.a.	n.s.
40	10	0	55	104,644.95	0.44	255.56	X	0.00	19.00	n.a.	n.s.
45	5	0	60	111,911.64	0.00	431.58	2	50.18	37.00	n.a.	n.s.
48	2	0	66	121,785.65	0.00	431.58	2	50.18	38.00	n.a.	n.s.

Table 8.6: Artificial instances: 50 customers

Table 8.6 shows the results for the class of instances with 50 customers. CPLEX is not able to solve optimally any instance. The meta-heuristic can solve most of

instances in less than thirty seconds.

The *Node Exchange* search phase improves the solution more often than the *Node Relocate*. Moreover, most of times the best solution is returned by $Criterion1$, whereas the other times the two criteria return the same solution or $Criterion2$ is better. Furthermore, sometimes the best solution is much better than the other one. Therefore, as in the previous cases, the intuition of considering both criteria in the algorithm seems to be appropriate.

It is important to note that comparison gaps are not available in all instances, because CPLEX does not provide a feasible.

Remarks

Tables 8.2-8.6 show that only few instances with 10 customers can be optimally solved by CPLEX within 10 minutes, as required by the carrier.

The meta-heuristic can solve all instances in less than one minute. Generally speaking, the *Node Exchange* search phase improves the solution more often than the *Node Relocate*. Sometimes the best solution is returned by $Criterion1$, whereas at times $Criterion2$ is better. Furthermore, sometimes the best solution is much better than the other one. Therefore, the intuition of considering both criteria in the algorithm seems to be appropriate.

Gaps between the best upper bounds provided by CPLEX within 3 hours and the meta-heuristic solutions are lower than 1% in the case of instances with 10 and 20 customers. As problem sizes increases, the meta-heuristic provides significantly better solutions than CPLEX.

8.3 Real instances

Real instances, which have about 40 customers, cannot be solved by CPLEX within 10 minutes. In this case the meta-heuristic is compared to the performance of the carrier's decisions in terms of total travelled distances. Results are shown in Table 8.7, in which the following notation is adopted:

- *Instances* The instance considered;
- $|I|$ Number of importers;
- $|E|$ Number of exporters;
- $|K_1|$ Number of available trucks with capacity 1-container;
- $|K_2|$ Number of available trucks with capacity 2-containers;

- *Carrier Distances* The total travelled distance according to the carrier's decisions (Km);
- *Distances* The total travelled distance according to the meta-heuristic (Km);
- *Saving(km)*: Difference between the meta-heuristic and the carrier's decisions (Km);
- *Saving(%)*: Percentage gap between the meta-heuristic and the carrier's decisions. When the solutions of the meta-heuristic are better than the carrier's decisions both values, differences and gaps, are reported in bold.

Instances	$\|I\|$	$\|E\|$	$\|K_1\|$	$\|K_2\|$	Carrier	Meta-heuristic		
					Distances	Distances	Saving(km)	Saving(%)
Instance 1	9	34	2	40	16891	16403	**488**	**2.89**
Instance 2	13	32	8	31	14986	13097	**1889**	**12.61**
Instance 3	4	34	4	39	15094	14437	**657**	**4.35**
Instance 4	8	35	6	36	14232	12961	**1271**	**8.93**
Instance 5	4	31	3	41	14251	13660	**591**	**4.15**

Table 8.7: Real instances.

Table 8.7 shows that, for all instances, the meta-heuristic improves the carrier's performance. The improvement seems to be particularly relevant when $|I|$ increases and becomes closer to $|E|$, due to the larger search space of feasible routes.

Chapter 9

Conclusions

This part of the thesis investigated a vehicle routing problem with a number of original characteristics, such as backhauls, multiple visits, heterogeneous fleets of trucks, the opportunity to carry two containers per truck, and the impossibility to separate trucks and containers during customer service. All characteristics were formalized by an integer linear programming model, and an exact algorithm was used to solve several artificial instances. As tests showed, the exact method was able to solve only instances with few customers.

The proposed meta-heuristic determines a feasible solution by a variant of the Clarke-and-Wright algorithm, in which routes are merged and assigned to trucks. Next, the solution is improved by by several local search phases, in which both node movements and truck swaps are implemented. Test show that the meta-heuristic is more effective than the exact algorithm in solving artificial instances similar to real problem instances. The comparison with the carrier's decisions shows that the meta-heuristic represents a promising instrument to improve the current decision-making process, because it leads to significant savings and determines routes quickly.

Most of concepts presented in this second part of the thesis are included in Lai et al. 2013, and were presented in Lai et al. 2012 and Di Francesco et al. 2012.

Part III

Landside - Homogeneous fleet size

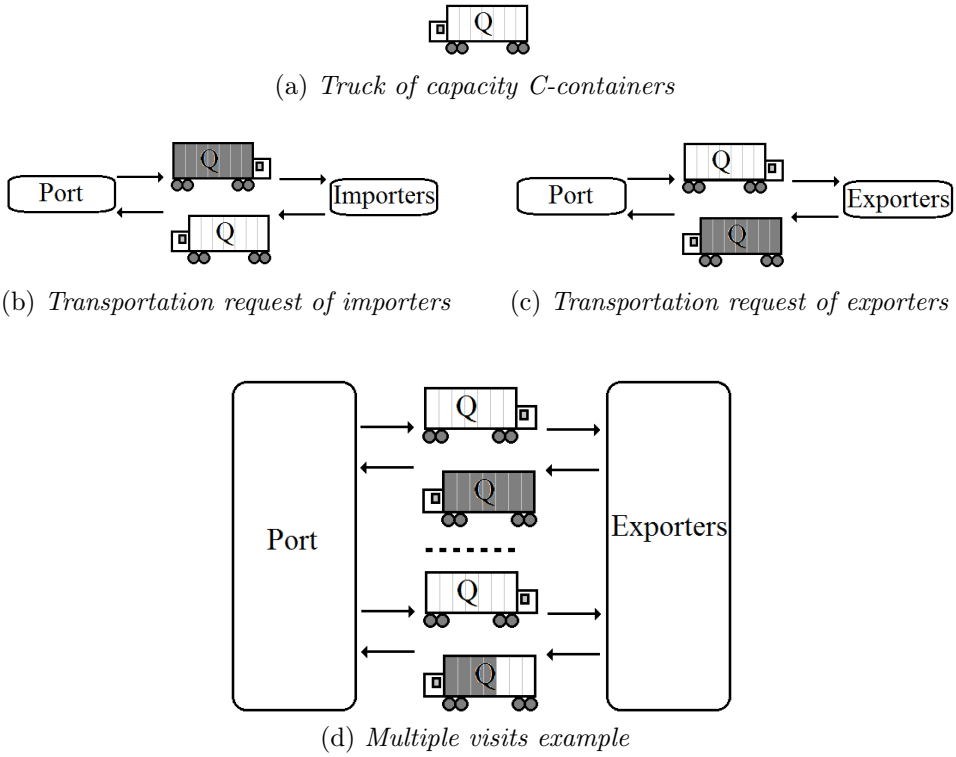
Chapter 10

Problem description

This third part of the thesis addresses a variant of the vehicle routing problem described in Part II. The difference, between the problem described hereafter and the previous one, concerns the composition of the fleet of trucks: the carrier manages a homogeneous fleet of trucks that can carry more than a container per truck. For the sake of clarity, in the following problem description, both new and old concepts are presented.

The carrier manages a homogeneous fleet of trucks based at the port. Trucks have capacity C-containers, as in Figure 10.1a. Trucks and containers are used to service two types of transportation requests: the delivery of container loads from the port to import customers, Figure 10.1b, and the shipment of container loads from export customers to the port, Figure 10.1c. Typically, customers may ask for the delivery and collection of large number of container loads, therefore more than a truck frequently visits the same customer. Split of the load is therefore allowed, even when it is not necessary. Figure 10.1d represents an example of multiple visits in which customers are exporters.

The carrier policy prescribes that container loads are packed and unpacked at customer facilities. More precisely, container loads are unpacked at importer locations in the presence of the driver, and emptied containers are immediately collected. In the same way, trucks bring empty containers to export customers and container loads are packed at the customer location in the presence of the driver. This policy, in which trucks and containers are always *coupled* substantially increases the time spent by drivers in the distribution activity and may lead to more costly solutions. From the carrier's point of view, empty containers are never left at customer locations and this will improve containers safety and integrity. From the customer's point of view, drivers participate to all packing and unpacking operations and promptly inform the carrier of possible problems during the service, and this practice is perceived as a high-quality service.



(a) *Truck of capacity C-containers*

(b) *Transportation request of importers*

(c) *Transportation request of exporters*

(d) *Multiple visits example*

Figure 10.1: Homogeneous landside problem description

In addition, since container loads of export customers are typically not ready for collection before the afternoon, the carrier policy establishes that import customers are serviced before exporters. As a result, containers emptied at importers can be filled at subsequent export customers, where container loads are packed to be brought to the port. Trucks with capacity one-container can service up to two customers in a route (one importer and one exporter). Trucks with C containers can service up to 2C customers in a route (C importers and C exporters). All containers leaving from importers can be used to service exporters and no incompatibility occurs between customers and trucks, which can service almost any customer. Figure 10.2 shows an example in which an importer receives $C - 1$ loaded containers from the port. Next, containers are emptied and moved to an exporter in which three of those containers are filled and shipped to the port.

The number of container loads to be picked up and delivered is generally different. If the number of container loads to deliver at importers and to collect from exporters is not balanced in a route, empty containers for exporters have to be loaded on the truck at the port (if export demand is higher than import demand) or emptied containers at the importers have to be returned to the port (if import demand is higher than export demand). Therefore, when the number of container loads delivered to importers is larger than the number of container loads shipped by exporters, a number of empty containers must be moved back to the port. When

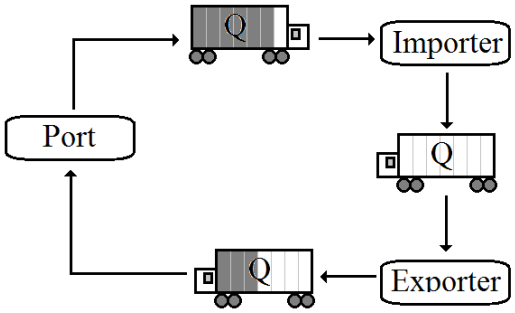


Figure 10.2: An example of route

the number of container loads delivered to importers is lower than the number of container loads shipped by exporters, a number of empty containers must be put on trucks leaving from the port, in order to service all customers.

The objective is to determine a set of routes in which routing costs are minimized, all customers are serviced (importers before exporters), and the trucks' capacity is never exceeded. According to Parragh, Doerner, and Hartl 2008 this problem belongs to the class of Vehicle Routing Problems with Clustered Backhauls (*VRPCB*), because in each route all deliveries must be performed before all pickups. However, in classical *VRPCB*, each customer must be visited only once, whereas in this problem multiple visits at each customer are allowed. This problem is called hereafter the Split Delivery Vehicle Routing Problem with Clustered Backhauls (SD-VRPCB).

In this study, linehaul customers are referred as import customers, delivery customers or simply importers. In the same way, backhaul customers are export customers, pickup customers or simply exporters. Similarly, we refer to delivery routes or pickup routes to identify those routes in which all visited customers are importers or exporters, respectively. In addition, we may refer to the delivery or pickup of container loads as delivery or pickup of containers.

The objective of this part of the thesis is to propose an optimization model accounting for the characteristics of this problem. A linear integer programming formulation for the problem will be proposed in Chapter 11, and tested in Chapter 14. As tests will show, the complexity of the model leads the exact formulation able to solve only instances with few customers. Moreover, solutions must be determined rapidly. Therefore, another objective of this part of the thesis, is to propose an efficient meta-heuristic for the problem at hand. The meta-heuristic proposed is the result of a series of improvements achieved during the time spent abroad in the University of Southampton. The scheme of the meta-heuristic, presented in Chapter 13, is to find an initial feasible solution and, next, to improve this solution by some adaptive guidance mechanisms. Furthermore, in order to better understand the improvements achieved during the time spent abroad, the previous constructive

heuristic is presented in Chapter 12. In Chapter 14, the results of the extensive computational experience are presented, and a comparison between the performance of the exact and the meta-heuristic algorithm is reported. Finally, conclusions are summarized in Chapter 15.

10.1 Literature review

The SD-VRPCB belongs to the *one-to-many-to-one* pickup and delivery problems, because container loads must be shipped from the port to importers, and from exporters to the same port (Berbeglia et al. 2007). In traditional *VRPCB* each customer must be visited exactly once, whereas in this problem the demand of each customer may be greater than the vehicle capacity and may be serviced by several vehicles. Therefore, multiple visits are allowed and the SD-VRPCB allows for *split* of units of the load (Archetti, Speranza, and Hertz 2006).

In the field of freight transportation, containers distributions by truck between customers and intermodal terminals is known as "drayage". According to Macharis and Bontekoning 2004, drayage involves the delivery of a full container from an intermodal terminal to a receiver and the subsequent collection of an empty container, as well as the provision of an empty container to the shipper and the subsequent transportation of a full trailer or container to the intermodal terminal.

The definition of drayage allows solutions in which trucks and containers are coupled (as in SD-VRPCB) as well as solutions in which they are not. Problems in which trucks and containers are not coupled have been investigated by Jula et al. 2005, Chung et al. 2007, Zhang, Yun, and Moon 2011, Zhang, Yun, and Kopfer 2010, and Vidovic, Radivojevic, and Rakovic 2011. The coupled problem received less attention; due to the long waiting times at the customer and consequent low productivity of the trucks, this class of problems has been investigated only in papers motivated by specific technical restrictions (i.e., Imai, Nishimura, and Current 2007) or regulation policies (Cheung et al. 2008). In this paper, trucks and containers are coupled, because the carrier aims at providing a high quality service, in which truck drivers are responsible for controlling the packing/unpacking operations and containers' integrity is favoured.

Another typical characteristic of drayage problems in the literature is that vehicles are assume to transport at most one container (see for example Jula et al. 2005, Namboothiri and Erera 2008, Zhang, Yun, and Moon 2011, and Zhang, Yun, and Kopfer 2010). In practice, truck's capacity could be higher than one container and carrying two or more containers per truck is allowed in many countries (Nagl 2007). Hence, solutions in which trucks have larger capacities represent an opportunity to remarkably increase the efficiency of the distribution. It is important

to note that this opportunity increases substantially the difficulty of SD-VRPCB, given the additional complexity of the underlying packing problem.

Imai, Nishimura, and Current 2007 present a problem in which the optimal assignment of trucks to a set of delivery and pickup pairs is performed. The authors develop a subgradient heuristic based on a Lagrangian relaxation which enables to identify a near optimal solution. The heuristic consists of two sub-problems: the classical assignment problem and the generalized assignment problem. As in our setting, the container distribution is divided into two activities: pickup and delivery. These activities are essentially independent, and containers are emptied/loaded at customer locations. Main differences concern the composition of the fleet of vehicles and the possibility to assign a truck to more than one trip (if the working time length is not exceeded). The available trucks have same type and size, but cannot visit more than one pickup customer (or delivery customer) in a single trip before coming back to the depot. Moreover, split deliveries are not taken into account.

Caris and Janssens 2009 model the drayage of containers in the service area of an intermodal terminal as a full truckload pickup and delivery problem with time windows. The authors propose a two-phase insertion heuristic to construct an initial solution, and next the solution is improved with a local search heuristic based on three neighbourhoods. Main differences concern the composition of the fleet of vehicles and the introduction of time windows. The available trucks have the same capacity, but cannot visit more than one pickup customer (or delivery customer) in a single trip since in the full truckload pickup and delivery problem a vehicle carries a single load. As in our setting, the vehicles' capacity is expressed in terms of containers.

The closest problem to SD-VRPCB was faced by Mitra 2005 and Mitra 2008. These two papers consider a similar problem: vehicles located at a depot serve delivery and pickup demands of a set of customers. The available vehicles have the same capacity, and, as in our setting, they can perform at most one trip daily. Moreover, split deliveries are allowed and the demand of a customer may exceed the capacity of the vehicles. Main differences concern the order of visit in a trip and the possibility for a customer to have both delivery and pickup demands. Unlike SD-VRPCB, importers and exporters are allowed to be visited in any order and each customer may be visited more than once by the same vehicle. Mitra 2005 develop a Mixed Integer Linear Programming (MILP) formulation for the problem and develop a route construction heuristic improving the best known solutions obtained by the MILP formulation. Mitra 2008 improves further the solution quality by developing a parallel clustering technique and route construction heuristic for the same problem.

Chapter 11

Modeling

This chapter introduces first the mathematical notation and then presents an Integer Linear Programming (ILP) model for the SD-VRPCB. All sets, data and parameters are listed following their alphabetical order.

SETS

I**:** Set of importers;

E**:** Set of exporters;

K**:** Set of available trucks, each one of capacity C-containers.

INPUT DATA

d_i**:** Containers demand of customer $i \in I \cup E$. When $i \in I$, d_i represents the number of loaded containers used to service the importer $i \in I$. Due to the problem setting, d_i is equal to the number of empty containers available after the service to customer $i \in I$ as well. When $i \in E$, d_i represents the number of empty containers used to service exporter $i \in E$, and d_i is equal to the number of loaded containers shipped by customer $i \in E$ to port p as well;

p**:** The port;

C**:** The truck capacity.

Consider a directed graph $G = (N,A)$. The set of nodes is defined as the union of the port with the sets of importers and exporters:

$$N = \{p \cup I \cup E\}.$$

The set of arcs A includes all allowed ways to move trucks:

- from the port to any importer and any exporter;

- from an importer to the port, any other importer and any exporter;
- from an exporter to the port and any other exporter.

The backhaul constraints allow for reducing our network and do not consider those arcs connecting export customers to importers. More formally, the set A is defined as $A = A_1 \cup A_2$, where:

$$A_1 = \{(i,j)|i \in p \cup I, j \in N, i \neq j\}$$
$$A_2 = \{(i,j)|i \in E, j \in p \cup E, i \neq j\}.$$

DECISION VARIABLES

x_{ij}^k**:** Routing selection variable equal to 1 if arc $(i,j) \in A$ is traversed by truck $k \in K$, 0 otherwise. c_{ij} represents its non negative routing cost, which is equal for each truck $k \in K$;

y_{ij}^k**:** Number of loaded containers moved along arc $(i,j) \in A$ by truck $k \in K$;

z_{ij}^k**:** Number of empty containers moved along arc $(i,j) \in A$ by truck $k \in K$.

The problem can be formulated as follows:

$$min \sum_{k \in K} \sum_{(i,j) \in A} c_{ij}\ x_{ij}^k \tag{11.1}$$

$s.t.$

$$\sum_{k \in K} \sum_{l \in N} y_{il}^k = \sum_{k \in K} \sum_{j \in p \cup I} y_{ji}^k - d_i \qquad \forall i \in I \tag{11.2}$$

$$\sum_{k \in K} \sum_{l \in N} z_{il}^k = \sum_{k \in K} \sum_{j \in p \cup I} z_{ji}^k + d_i \qquad \forall i \in I \tag{11.3}$$

$$\sum_{l \in N} y_{il}^k \leq \sum_{j \in p \cup I} y_{ji}^k \qquad \forall i \in I, \forall k \in K \tag{11.4}$$

$$\sum_{l \in N} z_{il}^k \geq \sum_{j \in p \cup I} z_{ji}^k \qquad \forall i \in I, \forall k \in K \tag{11.5}$$

$$\sum_{k \in K} \sum_{l \in p \cup E} y_{il}^k = \sum_{k \in K} \sum_{j \in N} y_{ji}^k + d_i \qquad \forall i \in E \tag{11.6}$$

$$\sum_{k \in K} \sum_{l \in p \cup E} z_{il}^k = \sum_{k \in K} \sum_{j \in N} z_{ji}^k - d_i \qquad \forall i \in E \tag{11.7}$$

$$\sum_{l \in p \cup E} y_{il}^k \geq \sum_{j \in N} y_{ji}^k \qquad \forall i \in E, \forall k \in K \tag{11.8}$$

$$\sum_{l \in p \cup E} z_{il}^k \leq \sum_{j \in N} z_{ji}^k \qquad \forall i \in E, \forall k \in K \tag{11.9}$$

$$\sum_{(ji)\in A}(y^k_{ji}+z^k_{ji})=\sum_{(il)\in A}(y^k_{il}+z^k_{il}) \qquad \forall i\in I\cup E,\forall k\in K \qquad (11.10)$$

$$y^k_{ij}+z^k_{ij}\leq C\ x^k_{ij} \qquad \forall (i,j)\in A,\forall k\in K \qquad (11.11)$$

$$\sum_{j\in N}x^k_{ji}-\sum_{l\in N}x^k_{il}=0 \qquad \forall i\in N,\forall k\in K \qquad (11.12)$$

$$\sum_{j\in N}x^k_{ij}\leq 1 \qquad \forall i\in N,\forall k\in K \qquad (11.13)$$

$$\sum_{k\in K}\sum_{i\in I\cup E}z^k_{ip}-\sum_{k\in K}\sum_{i\in I\cup E}z^k_{pi}=\sum_{i\in I}d_i-\sum_{i\in E}d_i \qquad (11.14)$$

$$x^k_{ij}\in\{0,1\} \qquad \forall (i,j)\in A,\forall k\in K \qquad (11.15)$$

$$y^k_{ij}\in\{0,1,\ldots,C\} \qquad \forall (i,j)\in A,\forall k\in K \qquad (11.16)$$

$$z^k_{ij}\in\{0,1,\ldots,C\} \qquad \forall (i,j)\in A,\forall k\in K \qquad (11.17)$$

Routing costs are minimized in the objective function (6.1).

Constraints (11.2)-(11.5) concern the movement of containers to importers. Constraints (11.2) and (11.3) are the flow conservation constraints of loaded and empty containers, respectively, at each importer node. Constraints (11.4) check that the number of loaded containers cannot increase after servicing an importer, whereas constraints (11.5) check that the number of empty containers does not decrease.

Constraints (11.6)-(11.9) concern the allocation of containers to exporters. Constraints (11.6) and (11.7) are the flow conservation constraints of loaded and empty containers, respectively, for each exporter. Constraints (11.8) and (11.9) control the number of loaded and empty containers in each truck entering and leaving from exporters: the number of loaded containers cannot decrease after a service at an exporter, whereas the number of empty containers cannot increase.

Constraints (11.10) guarantee that the number of containers carried by each truck does not change after visiting a customer. In other words, each node must be a transshipment node with respect to the total amount of containers transiting in it. Constraints (11.11) imposes that the number of containers on each truck does not exceed the capacity C.

Constraints (11.12) represent the flow conservation constraints for each truck at each node and Constraints (11.13) impose that, for each node and for each truck, can exist at most one successor. Constraints (11.13) together with Constraints (11.12) imply that the degree of each node must be at most 2. This forces a vehicle to visit the same customer at most once in a route. Moreover, if there is a successor for a node i and a truck k, Constraints (11.12) impose that there must exist also a predecessor for the same node and the same truck. Constraints (11.13) also guarantee that trucks are not used more than once.

Constraints (11.14) represent the flow conservation of empty containers at the port p. Finally, Constraints (11.15), (11.16) and (11.17) define the domain of the decision variables.

The model was coded using IBM ILOG CPLEX Optimization Studio 12.5 and solved by ILOG CPLEX 12.2 solver. Extensive computational results are presented in Chapter 14, however the model was not capable of solving instances of large size and with high trucks' capacity. Therefore, we decided to develop a meta-heuristic to better exploit the possibility of solving difficult instances in limited computing time. Next chapters presents this contribution.

Chapter 12

Heuristic solution method

This chapter illustrates the constructive heuristic algorithm designed before the improvements achieved during the collaboration with the University of Southampton. The constructive heuristic consists of two phases: *(i) SplitVRP phase*, determines an initial solution in which all routes service either importers or exporters; *(ii) Merging phase*, improves the solution by some heuristics, each one implements a different rule to merge routes determined in the *SplitVRP phase*. Finally, the best heuristic, in terms of objective function provided, is selected.

Table 12.1 illustrates the pseudo-code of the constructive heuristic, in which the following notation is adopted:

initSolImp Initial solution servicing importers only;

initSolExp Initial solution servicing exporter only;

initSMatrix Matrix of all savings that can be obtained merging routes;

Heu Identifier of the heuristic method;

Sol(Heu) Current solution of the considered heuristic;

SolImp Initial solution servicing importers of the considered heuristic;

SolExp Initial solution servicing exporters of the considered heuristic;

SMatrix(Heu) Matrix of all savings of the considered heuristic;

Solve($Sol(Heu)$, $SolImp(Heu)$, $SolExp(Heu)$, $SMatrix(Heu)$**)** Function that implements the heuth heuristic method. The input parameters are the current solution, $Sol(Heu)$, the initial solution for importers and exporters, $SolImp(Heu)$ and $SolExp(Heu)$, and the saving matrix, $SMatrix(Heu)$. The output is the new current solution for the considered heuristic method, $Sol(Heu)$;

S^* Best solution found.

```
procedure MAIN
    Found initSolImp                                  ▷ SplitVRP phase Section 12.1
    Found initSolExp
    Create the savings matrix initSMatrix
    for Heuristic = 1, 2, ..., 7 do                   ▷ Merging phase Section 12.2
        SolImp(Heu) ← initSolImp
        SolExp(Heu) ← initSolExp
        SMatrix(Heu) ← initSMatrix
        SOLVE(Sol(Heu), SolImp(Heu), SolExp(Heu), SMatrix(Heu))
    end for
    S* ← ∅
    S* ← min{Sol(1), Sol(2), ..., Sol(7)}
    return S*
end procedure
```

Table 12.1: The structure of the constructive heuristic.

12.1 SplitVRP phase

The *SplitVRP phase* faces two vehicle routing problems with splits: the first has importers only, whilst the second has exporters only. Split Vehicle Routing Problems are known to be difficult. Therefore, since an efficient heuristic for this class of problems was proposed by Archetti, Speranza, and Hertz 2006, their algorithm is adopted to determine routes for the initial solution. The algorithm is composed of three phases: *(i)* the first phase determines the initial feasible solution constructing a giant tour by applying the GENIUS algorithm, and imposing trucks to return to the depot whenever their load equals the capacity, *(ii)* the second phase consists of a tabu search algorithm and *(iii)* the third phase improves the solution found by removing t-split cycles and by re-optimizing each route using the GENIUS algorithm. The tabu search is based on relocation moves, where this neighbourhood has been extended to take into account "split" moves. More precisely, a customer is either relocated into another route or copied into an alternative route. In the latter case, its original demand is split between the two routes.

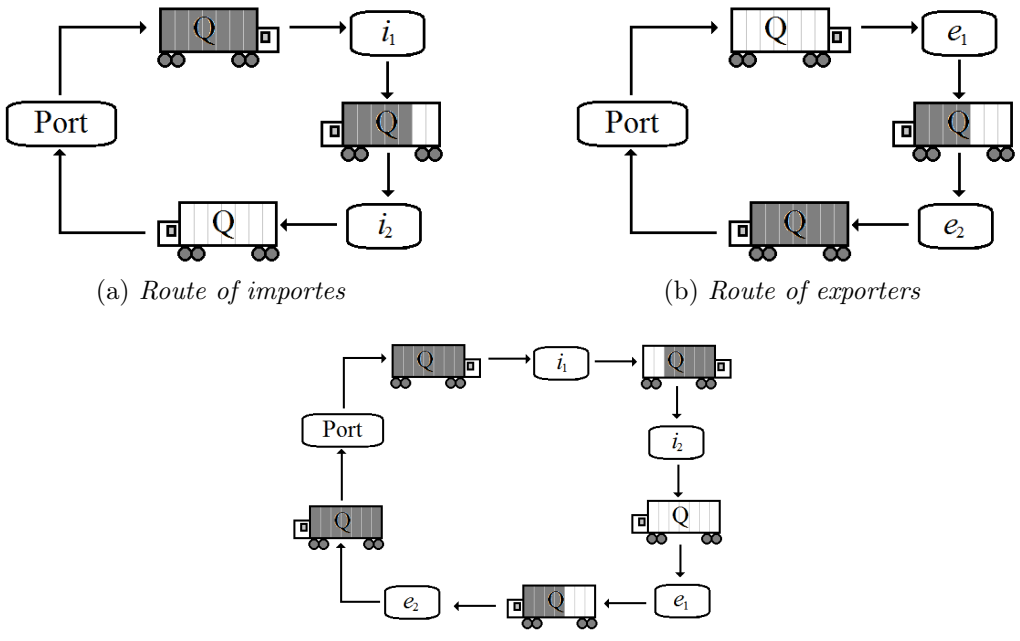
12.2 Merging phase

Routes determined in the *SplitVRP phase* are merged in the *Merging phase* according to different saving-based heuristics. Savings represent routing costs achieved merging two routes, instead of leaving them separately. Given an importer

route r_i and an exporter route r_j, the saving generated by their merge is computed as $s_{ij} = c(r_i) + c(r_j) - c(r_{ij})$. In which $c(r_i)$ and $c(r_j)$ represent the cost of routes r_i and r_j respectively, and $c(r_{ij})$ represent the cost of the merged route r_{ij}.

Savings are recorded in a matrix, in which the number of rows is equal to the number of routes servicing importers, and the number of columns is equal to the number of routes servicing exporters. Moreover, only positive savings are taken into account. All negative savings are rejected and recorded in the matrix with value 0, that means *merging not allowed*. Whenever two routes r_i and r_j are merged by a heuristic, all savings that involve the route r_i and/or the route r_j in the matrix are set to 0. Routes r_i and r_j are no longer available for other merges. Therefore, each importer route can be merged with at most an exporter route, and vice-versa.

The order of visit between importers and exporters does not change during the merge. To clarify, consider for instance two importers, i_1 and i_2, and two exporters, e_1 and e_2. Assume that routes determined in the initial solution are p, i_1, i_2, p, Figure 12.1a, and p, e_1, e_2, p, Figure 12.1b. If these routes are merged, the final route will be p, i_1, i_2, e_1, e_2, p as in Figure 12.1c. Therefore, the possibility of visiting the importer i_2 before the importer i_1, and the exporter e_2 before the e_1, is not taken into account.



(a) *Route of importes* (b) *Route of exporters*

(c) *Merged route*

Figure 12.1: Merging method.

The *Merging phase* performs a number of heuristics, which can be used to tackle the problem at hand. All proposed heuristics implement a different rule to merge routes determined in the *SplitVRP phase*. In order to facilitate the illustration,

hereafter there is the adopted notation:

Row i or $\mathbf{r}_i$: represents the $i-th$ importer route in the saving matrix;

Column j or $\mathbf{r}_j$: represents the $j-th$ exporter route in the saving matrix;

Entry $\mathbf{s}_{ij}$: saving generated merging routes r_i and r_j;

$\mathbf{m}_i$: Number of exporter routes (columns) that can be merged with route r_i;

$\mathbf{m}_j$: Number of importer routes (rows) that can be merged with route r_j;

$\mathbf{avrg}_i$: Average of all savings available for route r_i;

$\mathbf{avrg}_j$: Average of all savings available for route r_j.

Eight heuristics are proposed, each one provides a solution denoted by Sol_0-Sol_7.

H_0 - Heuristic 0

The heuristic H_0 returns routes as determined in the *SplitVRP phase*. The heuristic takes care of the union of the two separate sets of routes, without any sort of check.

The outline of Heuristic 0 is:

$Step_0$ $Sol_0 = \emptyset$.

$Step_1$ For each row i, insert r_i into Sol_0.

$Step_2$ For each column j, insert r_j into Sol_0.

H_1 - Heuristic 1

The heuristic H_1 examines the importer routes one at a time. The order in which routes are examined is their creation order. For each route examined, selects the best available route servicing exporters. The best route is the one with the highest value of saving. If a best route exists, merges the two routes and inserts the merged route in the final solution. Next, goes on by the examination of the next importer route, until all importer routes are examined.

After the routes examination, it may happen that some route is not merged. To avoid the possibility to not service customers involved in such routes, all routes not merged are inserted in the final solution as determined in the *SplitVRP phase*.

The outline of Heuristic 1 is:

$Step_0$ $Sol_1 = \emptyset$.

$Step_1$ For each row i, select the highest s_{ij}. Merge routes r_i and r_j, if any, and insert the new route into Sol_1.

$Step_2$ For each row i not involved in any merging, insert r_i into Sol_1.

$Step_3$ For each column j not involved in any merging, insert r_j into Sol_1.

H_2 - Heuristic 2

The heuristic H_2 examines the exporter routes one at a time. The order in which routes are examined is their creation order. For each route examined, selects the best available route servicing importers. The best route is the one with the highest value of saving. If a best route exists, merges the two routes and inserts the merged route in the final solution. Next, goes on by the examination of the next exporter route, until all exporter routes are examined.

After the routes examination, it may happen that some route is not merged. To avoid the possibility to not service customers involved in such routes, all routes not merged are inserted in the final solution as determined in the *SplitVRP phase*.

The outline of Heuristic 2 is:

$Step_0$ $Sol_2 = \emptyset$.

$Step_1$ For each column j, select the highest s_{ij}. Merge routes r_i and r_j, if any, and insert the new route into Sol_2.

$Step_2$ For each row i not involved in any merging, insert r_i into Sol_2.

$Step_3$ For each column j not involved in any merging, insert r_j into Sol_2.

H_3 - Heuristic 3

The heuristic H_3 examines the importer routes one at a time. The order in which routes are examined depends on the number of available merges. All routes r_i, determined in the *SplitVRP phase*, are ordered by their value of m_i in a increasing fashion. For each route examined, selects the best available route servicing exporters. The best route is the one with the highest value of saving. If the best route exists, merges the two routes, inserts the merged route in the final solution, and reorder all routes based on the updated value of m_i in a increasing fashion. Next, goes on by the examination of the next importer route until all importer routes are examined.

After the routes examination, it may happen that some route is not merged. To avoid the possibility to not service customers involved in such routes, all routes not merged are inserted in the final solution as determined in the *SplitVRP phase*.

The outline of Heuristic 3 is:

$Step_0$ $Sol_3 = \emptyset$.

$Step_1$ Search for row i with the lowest value of m_i. If any, go to $Step_2$, otherwise go to $Step_3$.

$Step_2$ Select the highest s_{ij} available in row i, merge routes r_i and r_j and insert the new route into Sol_3. Go to $Step_1$.

$Step_3$ For each row i not involved in any merging, insert r_i into Sol_3.

$Step_4$ For each column j not involved in any merging, insert r_j into Sol_3.

H_4 - Heuristic 4

The heuristic H_4 examines the exporter routes one at a time. The order in which routes are examined depends on the number of available merges. All routes r_j, determined in the *SplitVRP phase*, are ordered by their value of m_j in a increasing fashion. For each route examined, selects the best available route servicing importers. The best route is the one with the highest value of saving. If the best route exists, merges the two routes, inserts the merged route in the final solution, and reorder all routes based on the updated value of m_j in a increasing fashion. Next, goes on by the examination of the next exporter route until all exporter routes are examined.

After the routes examination, it may happen that some route is not merged. To avoid the possibility to not service customers involved in such routes, all routes not merged are inserted in the final solution as determined in the *SplitVRP phase*.

The outline of Heuristic 4 is:

$Step_0$ $Sol_4 = \emptyset$.

$Step_1$ Search for column j with the lowest value of m_j. If any, go to $Step_2$, otherwise go to $Step_3$.

$Step_2$ Select the highest s_{ij} available in column j, merge routes r_i and r_j and insert the new route into Sol_4. Go to $Step_1$.

$Step_3$ For each row i not involved in any merging, insert r_i into Sol_4.

$Step_4$ For each column j not involved in any merging, insert r_j into Sol_4.

H_5 - Heuristic 5

The heuristic H_5 examines the routes one at a time. The order in which routes are examined depends on the number of available merges. All importer routes, r_i, and all exporter routes, r_j, determined in the *SplitVRP phase*, are ordered by their value of m_i and m_j respectively in a increasing fashion. If the value of $m_i \leq m_j$ (the importer route has lower possibilities to be merged), the route to be examined is the importer route, whereas if the value of $m_i > m_j$ the route to be examined is the exporter route. For each route examined, selects the best available route servicing importers/exporters.The best route is the one with the highest value of saving. If a best route exists, merges the two routes, inserts the merged route in the final solution, and reorder all routes based on the updated value of m_i and m_j in a increasing fashion. Next, goes on by the examination of the next route until all routes are examined.

After the routes examination, it may happen that some route is not merged. To avoid the possibility to not service customers involved in such routes, all routes not merged are inserted in the final solution as determined in the *SplitVRP phase*.

The outline of Heuristic 5 is:

$Step_0$ $Sol_5 = \emptyset$.

$Step_1$ Search for row i with the lowest value of m_i and the column j with the lowest value of m_j. If $m_i \leq m_j$, go to $Step_2$, otherwise go to $Step_3$. If no routes can be merged, go to $Step_4$.

$Step_2$ Select the highest s_{ij} for i, merge routes r_i and r_j and insert into Sol_5. Go to $Step_1$.

$Step_3$ Select the highest s_{ij} for j, merge routes r_i and r_j and insert into Sol_5. Go to $Step_1$.

$Step_4$ For each row i not involved in any merging, insert r_i into Sol_5.

$Step_5$ For each column j not involved in any merging, insert r_j into Sol_5.

H_6 - Heuristic 6

This heuristic works in the same way as Heuristic 5, but differs in the selection of savings. The best route is the one with the value of saving closest to the average of all available savings, instead of the highest one.

The outline of Heuristic 6 is:

$Step_0$ $Sol_6 = \emptyset$.

$Step_1$ Search for row i with the lowest value of m_i and the column j with the lowest value of m_j. If $m_i \leq m_j$, go to $Step_2$, otherwise go to $Step_3$. If no routes can be merged, go to $Step_4$.

$Step_2$ Select the closest s_{ij} to $avrg_i$, merge routes r_i and r_j and insert into Sol_6. Go to $Step_1$.

$Step_3$ Select the closest s_{ij} to $avrg_j$, merge routes r_i and r_j and insert into Sol_6. Go to $Step_1$.

$Step_4$ For each row i not involved in any merging, insert r_i into Sol_6.

$Step_5$ For each column j not involved in any merging, insert r_j into Sol_6.

H_7 - Heuristic 7

This heuristic merges routes according to the largest available saving in the matrix.

The heuristic H_7 implements the Clarke and Wright's method into the selection of routes. The heuristic examines all entries of the saving matrix, and selects the best one. The best entry is the one with the highest value of saving, s_{ij}. If a best entry exists, merges the importer route r_i with the exporter route r_j, and inserts the merged route in the final solution. Next, goes on the search of the best entry of the matrix, until all entries are examined.

After the routes examination, it may happen that some route is not merged. To avoid the possibility to not service customers involved in such routes, all routes not merged are inserted in the final solution as determined in the *SplitVRP phase*.

The outline of Heuristic 7 is:

$Step_0$ $Sol_7 = \emptyset$.

$Step_1$ Select for the largest s_{ij} in the saving matrix. If any, go to $Step_2$, otherwise go to $Step_3$.

$Step_2$ Merge routes r_i and r_j and insert the new route into Sol_7. Go to $Step_1$.

$Step_3$ For each row i not involved in any merging, insert r_i into Sol_7.

$Step_4$ For each column j not involved in any merging, insert r_j into Sol_7.

Remarks

After the execution of all heuristics, the algorithm selects the best one in terms of objective function.

It is important to note that, none of the illustrated heuristics checks the feasibility of the solution. Therefore, it may happen that some of them leads to infeasible solutions in terms of available trucks.

Chapter 13

Meta-heuristic solution method

This chapter illustrates the improvements, from a methodological point of view, achieved during the time spent abroad in the University of Southampton with the supervision of Dr. Maria Battarra. The meta-heuristic proposed is based on Adaptive Guidance mechanisms. The motivation behind the choice of such methodology is that adaptive guidance algorithms allow for *recycling* non-specialized heuristics at hand. The company may easily adapt modules of code already in use, minimizing the inconvenience of adopting a new software. Easy pieces of code are also easier to maintain and possibly adapt to incorporate more advanced problem features.

Hart 2005 exhibits a large class of simple rules of behavior, called adaptive heuristics. In adaptive guidance algorithms, the different code components are coordinated and unified by guidance mechanisms: simple rules are applied to check the quality of the current solution and detect possibly improvements. The input parameters of the code components are then perturbed in order to achieve the desired diversification. Examples of successful implementations of adaptive guidance algorithms are presented in Battarra, Monaci, and Vigo 2009, Bai et al. 2007, Kramer 2008 and Olivera and Viera 2007.

In our implementation, the code at hand was a Tabu Search (TS) algorithm (Glover and Laguna 1998) for the Split and Delivery Vehicle Routing Problem (SDVRP) proposed by Archetti, Speranza, and Hertz 2006.

The overall meta-heuristic consists of three phases: *(i) SplitDeliveryVRP phase*, a constructive heuristic decomposes the SD-VRPCB into two SD-VRP sub problems, one for importers and one for exporters; *(ii) Merging phase*, merges routes determined in the first phase using an ILP model based on the saving concept; *(iii) Adaptive guidance phase*, analyses the current solution, detects areas of improvement and adjusts the input parameters of the *SplitDeliveryVRP phase* by using penalization mechanisms. The three phases are repeated sequentially until a stop

criterion is satisfied and the best solution found is returned.

Table 13.1 illustrates the pseudo-code of the adaptive guidance meta-heuristic, in which the following notation is adopted:

tExe Execution time;

MAXTIME Execution time limit;

it Number of consecutive iterations performed during the whole execution;

MAXIT Maximum number of consecutive iterations allowed during the whole execution;

notImpIt Number of consecutive iterations performed since the best solution known was found;

Sol Current solution of the meta-heuristic;

S* Best solution found;

SolImp Routes servicing importers. Routes are generated by a Tabu Search algorithm for the SD-VRP. The input parameters are the set of importers to be served and the associated perturbed data;

SolExp Routes servicing exporters. Routes are generated by a Tabu Search algorithm for the SD-VRP. The input parameters are the set of exporters to be served and the associated perturbed data;

SMatrix Matrix of all savings that can be obtained by merging routes;

Merge(*Sol*, *SolImp*, *SolExp*, *SMatrix***)** Function that merges routes determined in the *SplitDeliveryVRP phase* by an ILP model. The input parameters are the current solution, *Sol*, the initial solution for importers and exporters, *SolImp* and *SolExp*, and the saving matrix, *SMatrix*. The output is the new current solution, *Sol*;

AdaptiveGuidance(*Sol*, *SolImp*, *SolExp*, *it***)** Function that analyses the solution with respect to the adaptive guidance mechanisms and perturbs the data of the *SplitDeliveryVRP phase* by applying penalization mechanisms. The input parameters are the current solution, *Sol*, the initial solution for importers and exporters, *SolImp* and *SolExp*, and the current number of iterations, *it*.

In the following, the three phases of the algorithm are described in detail.

```
procedure MAIN
    Start tExe
    it = 0
    notImpIt = 0
    S* ← ∅
    while tExe ≤ MAXTIME & notImpIt ≤ MAXIT do
        it = it + 1
        notImpIt = notImpIt + 1
        SolImp ← TS(I);                          ▷ SplitDeliveryVRP phase Section 13.1
        SolExp ← TS(E);
        Create the savings matrix SMatrix
        Sol ← ∅
        MERGE(Sol, SolImp, SolExp, SMatrix)      ▷ Merging phase Section 13.2
        if Sol ≤ S* || S* == ∅ then
            S* ← ∅
            S* ← Sol
            notImpIt ← 0
        end if
        ADAPTIVEGUIDANCE(Sol, SolImp, SolExp, it) ▷ Adaptive guidance phase Section 13.3
    end while
    return S*
end procedure
```

Table 13.1: The structure of the meta-heuristic.

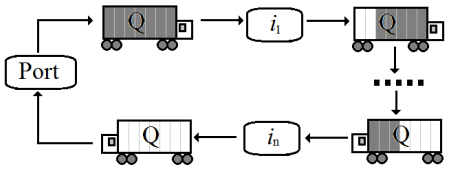
13.1 SplitDeliveryVRP phase

The *SplitDeliveryVRP phase* consists of solving two distinct SD-VRP: the first involves importers only, whilst the second exporters only. As previously stated, the TS algorithm proposed by Archetti, Speranza, and Hertz 2006 is employed to solve this NP-hard problem. The algorithm is composed of three phases: *(i)* the first phase determines the initial feasible solution constructing a giant tour by applying the GENIUS algorithm, and imposing trucks to return to the depot whenever their load equals the capacity, *(ii)* the second phase consists of a tabu search algorithm and *(iii)* the third phase improves the solution found by removing t-split cycles and by re-optimizing each route using the GENIUS algorithm. The tabu search is based on relocation moves, where this neighbourhood has been extended to take into account "split" moves. More precisely, a customer is either relocated into another route or copied into an alternative route. In the latter case, its original demand is split between the two routes.

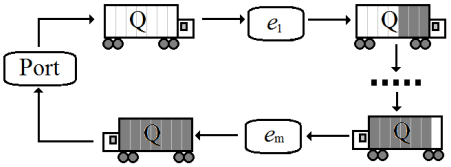
13.2 Merging phase

Routes determined in the *SplitDeliveryVRP phase* are merged in the *Merging phase* according to an ILP model based on the saving concept inspired by the

Clarke and Wright savings algorithm (Clarke and Wright 1964). Savings represent routing costs achieved merging a route servicing importers with a route servicing exporters, instead of leaving them separately. The order of visits of customers involved in a route can change during the merging, therefore we consider the order on which routes are created and the opposite one. For each pair of routes that can be merged, we compute the four possible ways to merge routes and the largest saving is selected as the best merge for the pair of routes.



(a) *Route of importes*



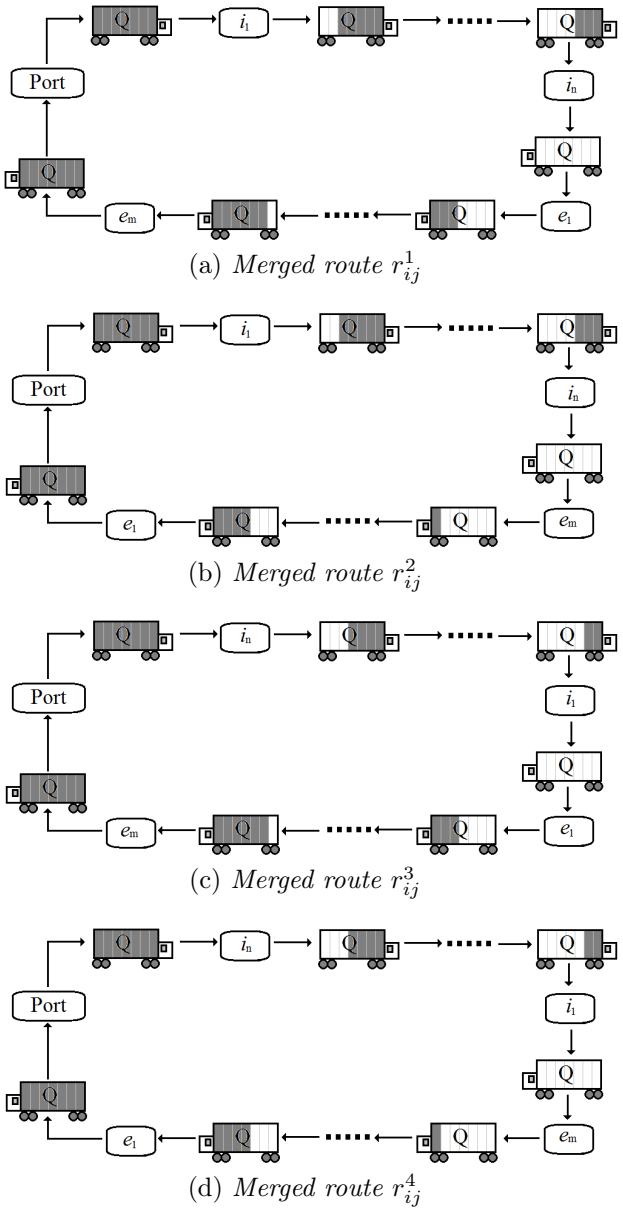
(b) *Route of exporters*

Figure 13.1: Improved merging method: example route.

To clarify, consider for instance n importers, serviced by route $r_i = \{p, i_1, \ldots, i_n, p\}$, as in Figure 13.1a, and m exporters serviced by route $r_j = \{p, e_1, \ldots, e_m, p\}$, as in Figure 13.1b, denoting with $c(i_n,e_1)$ the cost associate to arc $(i_n,e_1) \in A$, and so on. When evaluating the merge of the two routes, r_i and r_j, the algorithm computes four different savings based on the extra mileage formula:

- $s^1_{ij} = c_{(i_n,\ p)} + c_{(p,\ e_1)} - c_{(i_n,\ e_1)}$ in which in the route r_i and r_j are merged using their original direction. Figure 13.2a;
- $s^2_{ij} = c_{(i_n,\ p)} + c_{(p,\ e_1)} + c_{(e_m,\ p)} - c_{(i_n,\ e_m)} - c_{(e_1,\ p)}$ in which in the final route r_i has its original direction and r_j has the opposite direction. Figure 13.2b;
- $s^3_{ij} = c_{(p,\ i_1)} + c_{(i_n,\ p)} + c_{(p,\ e_1)} - c_{(p,\ i_n)} - c_{(i_1,\ e_1)}$ in which in the final route r_i has the opposite direction and r_j has its original direction. Figure 13.2c;
- $s^4_{ij} = c_{(p,\ i_1)} + c_{(i_n,\ p)} + c_{(p,\ e_1)} + c_{(e_m,\ p)} - c_{(p,\ i_n)} - c_{(i_1,\ e_m)} - c_{(e_1,\ p)}$ in which in the routes r_i and r_j have the opposite direction with respect to their original one. Figure 13.2d.

For each pair of routes, the best saving is defined as the maximum of all computed extra mileages as $s_{ij} = max\{s^1_{ij},\ s^2_{ij},\ s^3_{ij},\ s^4_{ij}\}$. Savings are recorded in a matrix, in which the number of rows is equal to the number of routes servicing



(a) *Merged route* r_{ij}^1

(b) *Merged route* r_{ij}^2

(c) *Merged route* r_{ij}^3

(d) *Merged route* r_{ij}^4

Figure 13.2: Improved merging method: merged routes.

importers, and the number of columns is equal to the number of routes servicing exporters.

Routes determined in *SplitDeliveryVRP phase* are merged in the *Merging phase* according to an ILP model. The *SplitDeliveryVRP phase* returns two sets of routes: the set of delivery routes, defined as $SolImp$, and the set of pickup routes, defined as $SolExp$. Binary variables x_{ij},$\forall i \in SolImp$,$j \in SolExp$ assume value 1 if routes r_i and r_j are merged, 0 otherwise. Note that s_{ij} represents the best saving obtained by the merging of routes r_i and r_j, computed as described above.

The problem can be formulated as follows:

$$max \sum_{i \in SolImp} \sum_{j \in SolExp} s_{ij}\ x_{ij} \tag{13.1}$$

$s.t.$

$$\sum_{j \in SolExp} x_{ij} \leq 1 \qquad \forall i \in SolImp \tag{13.2}$$

$$\sum_{i \in SolImp} x_{ij} \leq 1 \qquad \forall j \in SolExp \tag{13.3}$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in SolImp,\ j \in SolExp \tag{13.4}$$

The overall gain is maximized in the objective function (13.1).

Constraints (13.2) and (13.3) limit the maximum number of merges allowed for each route. For each route in $SolImp$ at most one merge with a $SolExp$ route is possible, and vice-versa. Therefore, we do not consider merge operations involving more than an importer and an exporter route, and we merge routes only at their extreme points (the first or last routes' position). Despite this limitation, the solution quality achieved does not seem to suffer from the simplification. In fact, it is rare that multiple merges would generate feasible solutions (because of capacity restrictions), and adaptive guidance mechanisms will force the TS algorithm to choose the most suitable customers in the first and last positions of the routes (see Section 13.3).

13.3 Adaptive guidance phase

The *Adaptive guidance phase* analyses the incumbent solution with respect to some adaptive guidance mechanisms. If drawbacks are detected in the solution, the input data of the TS algorithm are suitably perturbed by applying penalization mechanisms. The aim is therefore to identify drawbacks in the incumbent solution, define quantitative measures for them and design suitable penalization mechanisms that would achieve the desired diversification effect, without corrupting the original input data.

The meta-heuristic is guided by four *Adaptive Guidance Mechanisms*:

(i) $A.G.M.1$ - Exaggerated Split

This mechanism aims at minimizing multiple visits to customers, because we realized that the TS algorithm tends to generate routes in which split of the load is frequent. The overall solutions would likely to be not high quality. Given a customer i, the minimum number of visits required to satisfy its demand is

$minTrip_i = \lceil d_i/C \rceil$. In the incumbent solution, $visit_i$ is the number of visits to customer i in the current solution, and $exceed_i = visit_i - minTrip_i$. The guidance mechanism selects the importer customer $i_I \in I$ and the exporter customer $i_E \in E$ with the largest positive value of $exceed_i$. If any, a penalization is introduced for all arcs connecting the selected customers, $i_I \in I$ and $i_E \in E$, in the next γ iterations. This will guide the TS algorithm to use a lower number of edges connecting these customers and therefore split the load in a lower number of routes.

(ii) $A.G.M.2$ - Extreme importer

This mechanism aims at obtaining the most suitable *extreme* customers in importer routes. Given that the merge of routes is always performed on the first or last customer of a route, it is important to define such extreme customers in a suitable way. More precisely, the first and last importers in a route should be as close as possible to export customers.

Given a importer $i_I \in I$, we denote with α_{i_I} the number of times in which $i_I \in I$ is visited as extreme node of a delivery route in the incumbent solution ($SolImp$), net of the number of times that $i_I \in I$ is not visited as extreme node. Negative values of α_{i_I} indicate that, considering the set of delivery routes in the incumbent solution that visit the selected importer, $i_I \in I$ is frequently visited as an internal node. Moreover, we denote as σ_{i_I} the sum of all distances between the selected importer $i_I \in I$ and all export customers $j_E \in E$. This total distance is usually lower when many exporters are close to the selected importer $i_I \in I$. The guidance mechanism selects, if any, the importer $i_I \in I$ with a negative value of α_{iI} and the highest number of nearest exporters (minimum value of σ_{i_I}). If any, a penalization is introduced for all arcs entering or leaving the selected node, i_I, in the next γ iterations.

(iii) $A.G.M.3$ - Extreme exporter

This mechanism works in the same way as $A.G.M.2$, but considering export customers.

(iv) $A.G.M.4$ - Expensive arc

Given a set of customers, this mechanism aims at diversifying the solution imposing a penalization for the most expensive arcs used in the incumbent solution. More precisely, the most expensive arcs among the connections between importers

$(l_I, m_I) \in SolImp$ and between exporters $(l_E, m_E) \in SolExp$ are identified as:

$$c_{(l_I,\ m_I)} = max\{c_{(i_I,\ j_I)} | (i_I, j_I) \in SolImp,\ i_I \in I,\ j_I \in I\}$$
$$c_{(l_E,\ m_E)} = max\{c_{(i_E\ j_E)} | (i_E, j_E) \in SolExp,\ i_E \in E,\ j_E \in E\}$$

The guidance mechanism introduces a penalization for arcs $(l_I,\ m_I)$, $(m_I,\ l_I)$, $(l_E,\ m_E)$ and $(m_E,\ l_E)$ in the next γ iterations.

13.4 Penalizations

Once the incumbent solution's arcs have been analysed according to the guidance mechanisms, penalizations are applied to the suitable arc costs for the subsequent γ iterations. For all arcs for which the guidance mechanisms have suggested a penalization, the cost matrix is updated as $c_{ij} = c_{ij} + RandomCoef \cdot M$ for those arcs connecting customers, and as $c_{ij} = c_{ij} + RandomCoef \cdot M + (|N| - 1) \cdot M$ for those arcs that involve the port. The latter formula differs from the first: in order to minimize the number of trucks in the solution, a larger penalty is added to the cost of those arcs connecting to the port. *RandomCoef* is a coefficient that randomly decreases/increases the penalties during the overall execution of the algorithm, according to the formula:

$$RandomCoef = (Random(1, \ldots, \alpha) + \beta)/100 \tag{13.5}$$

β is a self-adapting parameter, that starts with value 0 and increases its value by α after each α iterations. Each time a new local optimum is reached, β is reinitialized to 0. The value M is defined as the largest entry of the cost matrix.

Whenever a new best known solution is obtained, β is set to 0 (in order to refresh our penalties). Moreover, the penalties associated to the port are removed for the set of customers requesting a lower number of routes (this should allow for solutions with a better routing cost, without increasing the overall number of combined routes).

Regarding the introduction of penalties in the next γ iterations, three different methods are proposed. Each method leads to a different version of the algorithm. All methods are implemented and the best one is presented in the computational experience (Chapter 14). The three methods are:

(i) **Unchecked penalties** Penalties are always added to the cost matrix. If the penalty associated to arc $(i,\ j) \in A$ is found at iterations it and $it + \delta$ (with $\delta \leq \gamma$), the penalty is added twice;

(ii) **Unique penalties** Penalties are inserted in the next γ iterations only if there have not been inserted before. If a penalty is already inserted, the penalty is rejected and the adaptive guidance mechanism is executed again, until a penalty not yet inserted is found, or no more penalties are available;

(iii) **Incremental unique penalties** Penalties are inserted in the next γ iterations, but further penalties are explored. If the penalty is already inserted, the old penalty is incremented and the adaptive guidance mechanism is executed another time until a penalty not yet inserted is found or no more penalties are available. Therefore, if a penalty concerning a generic arc $(i, j) \in A$ is found at iterations it and $it + \delta$ (with $\delta \leq \gamma$), the penalty is inserted twice and the adaptive guidance mechanism is executed again looking for additional penalties.

Chapter 14

Experimentation

The results proposed in this chapter aim to analyse the performance of the proposed meta-heuristic. The test set consists of 175 randomly generated instances with 10, 20, 30, 40, and 50 customers. Given that large-sized instances proved to be the most challenging, we generated a larger number of instances of large size. The number of instances for each instance-size is 15, 25, 35, 45, and 55, respectively. The number of containers requested by each customer is uniformly generated between 1 and 5, whereas the customer coordinates are uniformly generated between -1000 and $+1000$. Instances with the same number of customers consider the same containers' demand values and the customers coordinates, whereas the truck capacity and the number of importers and exporters vary. A fifth of the instances for each problem-size considers vehicles with capacity 1-container, 2-containers, 4-containers, 6-containers and 8-containers, respectively. The proportion of importers and exporters in each instance is computed as follow. Denoting with n the number of customers involved, we generate $n/5+1$ instances, in which the number of importers increases by 5 in each instance (starting from value 0), and as a consequence the number of exporters decreases by 5 (starting from value n). In order to have at least two importers/exporters in each instance, the number of importers/exporters involved is altered when equal to zero: we increase by 2 number those customers that are not represented in the instance, and consequently we decrease the number of customers of the other type by the same quantity. The number of available trucks in each instances is fixed, and is equal to the minimum number of trucks needed to service all container load requests computed as the the bin packing lower bound $\lceil dem/C \rceil$, where dem represents the maximum number of container load requested by importers or exporters defined as $dem = max\{\sum_{i \in I} d_i, \sum_{i \in E} d_i\}$.

14.1 Data management

The integer programming formulation (11.1)-(11.17) was coded using IBM ILOG CPLEX Optimization Studio 12.5 and solved by the Branch & Bound of ILOG CPLEX 12.2. The meta-heuristic presented in Chapter 13 was coded in C++, and the integer model (13.1)-(13.4) is solved using the Callable Libraries of CPLEX 12.2. Experiments were performed on a Linux four-CPU server 2.67GHz 64GB RAM, with default parameter settings.

Although a major requirement for the carrier is to determine solutions in about 10 minutes, ILOG CPLEX 12.2 is set to stop also after 3 hours. This choice allowed the solver to produce upper and lower bounds and provides a base of comparison for assessing the quality of the meta-heuristic.

The meta-heuristic algorithm depends on two parameters:

φ strategy used to update penalties: takes value 1 for "Unchecked penalties", 2 for "Unique penalties", and 3 for "Incremental unique penalties" (see Section 13.4);

θ**:** probability that a guidance approach will be used in the next iteration.

Moreover, we set the $MAXTIME$ value to 600 seconds as suggested by the company, the $MAXIT$ value to 10000, the γ value to $|I|$ for penalties that involve import customers and $|E|$ for penalties that involve export customers. Finally, the coefficient α used to evaluate penalties in Equation (13.5) takes value 10.

Table 14.1 presents the validation of θ and φ, two crucial components of the adaptive guidance algorithm. The table presents the best five calibrations obtained during our preliminary experiments, denoted by C_1-C_5. The aim is to select φ and α such that the meta-heuristic generates solutions as close as possible to those provided by the exact algorithm. Therefore, the third row of Table 14.1, $\sum_{Gap}$, reports the sum of the positive average percentage gaps between the solutions provided by the meta-heuristic with respect to those provided by the exact algorithm.

In the following, results obtained using the calibration C_1 are presented, because this configuration proved to minimize the overall gap from the optimal solutions.

	Calibrations					*Selected*
	C_1	C_2	C_3	C_4	C_5	C_1
θ	33%	50%	25%	33%	50%	33%
φ	1	1	3	3	3	1
$\sum_{Gap}$	26.34%	27.74%	27.74%	26.81%	26.81%	26.34%

Table 14.1: Best calibrations performed.

14.2 Effectiveness of adaptive guidance mechanisms

This section illustrates the improvements achieved by the adaptive guidance mechanisms. Table 14.2 presents the performance of the meta-heuristic adopting adaptive guidance mechanisms with respect to the heuristic solution (the solution obtained by running the algorithm illustrated in Table 13.1 for a single iteration). The *Merging phase* algorithm based on an ILP model illustrated in Section 13.2 detects the optimal solution whenever the truck's capacity is 1-container. Since the aim of this section is to show the effectiveness of the adaptive guidance mechanisms, Table 14.2 disregards instances with capacity 1-container.

Computational results are presented in Table 14.2 where C and *Instances* represent the transportation capacity of the homogeneous fleet of trucks and the number of instances involved in each class respectively, *Average t(s)* is the average time in seconds before the best solution is found and *Average % Gap* is the average percentage deviation of the Adaptive Guidance solution with respect to the solution provided by the heuristic. It is important to note that each row of Table 14.2 represents the average percentage deviation between solutions of the heuristic and the meta-heuristic over a subset of instances.

		HEURISTIC	ADAPTIVE GUIDANCE	
C	Instances	Average t(s)	Avergage t(s)	Average % Gap
		10 CUSTOMERS		
2	3	0.19	0.19	0.00
4	3	5.23	5.23	0.00
6	3	0.17	2.02	0.00
8	3	5.51	5.51	0.00
		20 CUSTOMERS		
2	5	1.11	1.11	0.00
4	5	4.83	244.84	**-0.73**
6	5	4.21	217.78	**-3.51**
8	5	0.73	205.31	**-5.24**
		30 CUSTOMERS		
2	7	8.68	24.77	0.00
4	7	13.12	369.42	**-2.11**
6	7	13.72	110.87	**-1.64**
8	7	2.28	162.00	**-2.64**
		40 CUSTOMERS		
2	9	21.60	145.46	**-0.30**
4	9	18.66	159.87	**-0.69**
6	9	24.42	319.30	**-3.00**
8	9	15.44	234.74	**-1.84**
		50 CUSTOMERS		
2	11	21.10	87.65	**-0.02**
4	11	20.75	127.68	**-0.22**
6	11	19.10	86.72	**-0.19**
8	11	11.98	195.28	**-0.64**

Table 14.2: Adaptive guidance effectiveness.

Tests show that the adaptive guidance mechanisms are effective. Moreover,

the adaptive guidance mechanisms seem to be more effective as the truck capacity increases. Feasible solutions of better quality are provided by the adaptive guidance mechanisms in less than 10 minutes in 68/175 instances.

14.3 Comparison between solution methods

This section aims to compare the two different solution methods proposed in Chapter 12, called *constructive heuristic*, and Chapters 13, called *adaptive guidance meta-heuristic*. Tables 14.3, 14.4, 14.5, 14.6 and 14.7 show the results of the comparison. The aims is to show the improvements achieved by the meta-heuristic based on adaptive guidance mechanisms with respect to the constructive heuristic presented in Chapter 12. Moreover, in order to lead the constructive heuristic more competitive, savings are computed by the extra mileages formula presented in Section 13.1.

Computational results are indicated in Tables 14.3, 14.4, 14.5, 14.6 and 14.7 with the following notation:

- C : Transportation capacity;
- $|I|$: Number of importers;
- $|E|$: Number of exporters;
- $|K|$: Number of available trucks with capacity C-containers;

CONSTRUCTIVE HEURISTIC

- *o.f.*: Objective function returned by the constructive heuristic proposed in Chapter 12;
- $|K_u|$: Number of trucks with capacity C-containers used in the solution. This solution method may leads to infeasible solutions in terms of number of trucks used in the solution. Therefore, for the sake of clarity all infeasible solutions provided are "underlined";
- *t(s)*: Time in seconds before the best solution is found;

ADAPTIVE GUIDANCE META-HEURISTIC

- *o.f.*: Objective function returned by the meta-heuristic proposed in Chapter 13;
- *t(s)*: Time in seconds before the best solution is found;

- *% Gap*: Percentage gap with respect to the best solution provided by the constructive heuristic. When solutions provided by the meta-heuristic are better than those provided by the constructive heuristic, gaps are reported in bold with a negative value.

				CONSTRUCTIVE HEURISTIC			ADAPTIVE GUIDANCE META-HEURISTIC		
C	$\|I\|$	$\|E\|$	$\|K\|$	o.f.	$\|K_u\|$	t(s)	o.f.	t(s)	% Gap
1	2	8	19	29076.63	19	0.11	29064.82	0.26	**-0.04**
	5	5	16	29692.50	16	0.04	29692.50	0.16	0.00
	8	2	23	29607.10	23	0.15	29607.10	0.30	0.00
2	2	8	10	20292.07	10	0.10	20277.64	0.22	**-0.07**
	5	5	8	19640.83	8	0.03	19640.83	0.13	0.00
	8	2	12	20469.85	12	0.09	20237.67	0.22	**-1.14**
4	2	8	5	12709.61	5	0.07	12707	0.17	**-0.02**
	5	5	4	11756.63	4	0.03	11756.63	0.11	0.00
	8	2	6	12896.95	7	0.08	12400.96	15.41	**-3.99**
6	2	8	4	10651.39	4	0.06	9644.93	0.13	**-10.43**
	5	5	3	9443.13	3	0.04	9443.13	0.10	0.00
	8	2	4	10397.93	4	0.07	10397.93	5.85	0.00
8	2	8	3	9627.91	3	0.06	9251.42	0.12	**-4.06**
	5	5	2	8756.39	2	0.02	8756.39	0.09	0.00
	8	2	3	9862.16	4	0.06	9894.46	16.34	0.32

Table 14.3: Comparison between solution methods: 10 Customers.

Table 14.3 shows results for the class of instances with 10 customers. The adaptive guidance meta-heuristic seems to be more effective in those instances in which importers and exporters are not distributed homogeneously and the truck capacity increases.

The constructive heuristic provides a slightly better solution in 1/15 instances with 2 importers, 8 exporters and truck capacity 8-containers (0.32%) but this solution is infeasible. Therefore, since the solution provided by the constructive heuristic exceed the number of available trucks, the solution provided by the adaptive guidance meta-heuristic cannot be considered worst.

The best improvement is found for the instance with 2 importers, 8 exporters and truck capacity 6-containers, in which the improvement, in terms of objective function, is about 10.43%.

The constructive heuristic provides infeasible solutions in 2/15 instances. Moreover, the infeasible instance with truck capacity 4-containers besides being infeasible presents also a worst objective function, compared to the solution provided by the adaptive guidance meta-heuristic.

Table 14.4 shows the results for the class of instances with 20 customers. The adaptive guidance meta-heuristic seems to be more effective in those instances in which importers and exporters are more ore less distributed homogeneously and the truck capacity increases.

				CONSTRUCTIVE HEURISTIC			ADAPTIVE GUIDANCE META-HEURISTIC		
C	$\|I\|$	$\|E\|$	$\|K\|$	o.f.	$\|K_u\|$	t(s)	o.f.	t(s)	% Gap
1	2	18	52	61366.30	52	1.35	61366.30	2.70	0.00
	5	15	45	55129.44	45	0.86	55129.44	1.76	0.00
	10	10	33	48458.95	33	0.26	48122.05	0.96	**-0.70**
	15	5	45	55792.14	45	0.86	55792.14	1.68	0.00
	18	2	53	64705.83	53	1.43	64705.83	2.75	0.00
2	2	18	26	39766.04	26	0.74	39766.04	1.35	0.00
	5	15	23	35714.81	23	0.52	35373.01	0.97	**-0.96**
	10	10	17	31183.84	17	0.21	31019.25	0.75	**-0.53**
	15	5	23	36523.96	23	0.51	36401.67	1.02	**-0.33**
	18	2	27	42077.40	27	0.78	41955.11	1.50	**-0.29**
4	2	18	13	22487.36	14	0.66	22633.91	296.19	0.64
	5	15	12	20298.88	12	0.34	19938.48	275.93	**-1.80**
	10	10	9	18638.57	9	0.16	18309.34	561.17	**-1.79**
	15	5	12	20494.58	12	0.41	20484.61	41.93	**-0.04**
	18	2	14	23433.85	14	0.64	23433.85	48.99	0.00
6	2	18	9	16413.72	9	0.55	16208.59	0.97	**-1.26**
	5	15	8	16187.41	8	0.33	15235.92	490.82	**-6.24**
	10	10	6	14565.74	6	0.13	14372.09	0.56	**-1.34**
	15	5	8	14723.36	8	0.36	14723.36	10.07	0.00
	18	2	9	17791.26	10	0.50	16585.71	586.52	**-7.26**
8	2	18	7	14738.46	8	0.35	14271.76	8.75	**-3.27**
	5	15	6	14984.17	6	0.33	13932	573.21	**-7.55**
	10	10	5	15296.75	5	0.12	13338.96	191.72	**-14.67**
	15	5	6	14483.36	6	0.34	13786.59	248.41	**-5.05**
	18	2	7	14629.61	7	0.80	14596.35	4.50	**-0.22**

Table 14.4: Comparison between solution methods: 20 Customers.

The constructive heuristic provides a slightly better solution in 1/25 instances with 2 importers, 18 exporters and truck capacity 4-containers (0.64%) but this solution is infeasible. Therefore, the solution provided by the adaptive guidance meta-heuristic cannot be considered worst.

The best improvement is found for the instance with 10 importers, 10 exporters and truck capacity 8-containers, in which the improvement in terms of objective function is about 14.67%.

The constructive heuristic provides infeasible solutions in 3/25 instances. Moreover, the infeasible instances with track capacity 6-containers and 8-containers besides being infeasible present also a worst objective function.

Table 14.5 shows the results for the class of instances with 30 customers. The adaptive guidance meta-heuristic seems to be more effective in those instances in which importers and exporters are more ore less distributed homogeneously and the truck capacity increases. However, the constructive heuristic provides a slightly better solution in 3/35 instances with:

- 28 importers, 2 exporters, and truck capacity 2-containers (1.31%);
- 10 importers, 20 exporters and truck capacity 4-containers (3.33%);
- 28 importers, 2 exporters and truck capacity 6-containers (0.72%).

				CONSTRUCTIVE HEURISTIC			ADAPTIVE GUIDANCE META-HEURISTIC		
C	$\|I\|$	$\|E\|$	$\|K\|$	o.f.	$\|K_u\|$	t(s)	o.f.	t(s)	% Gap
1	2	28	79	95786.50	79	5.99	95786.50	11.56	0.00
	5	25	72	89471.34	72	4.27	89471.34	8.36	0.00
	10	20	60	77943.25	60	1.93	77811.59	4.29	**-0.16**
	15	15	45	74845.96	45	0.86	73424.64	3.03	**-1.93**
	20	10	61	80142.17	61	2.13	79524.70	4.34	**-0.77**
	25	5	75	92487.21	75	4.60	92439	8.63	**-0.05**
	28	2	84	100904.53	84	6.61	100904.53	12.66	0.00
2	2	28	40	61202.75	40	2.92	61202.75	5.61	0.00
	5	25	36	57999.63	36	1.98	57999.63	3.91	0.00
	10	20	30	51167.29	30	1.06	51003.60	2.43	**-0.32**
	15	15	23	48117.84	23	0.51	47883.08	1.85	**-0.49**
	20	10	31	53120.94	31	1.03	52118.06	114.88	**-1.92**
	25	5	38	61011.48	38	2.11	59225.19	4.17	**-3.01**
	28	2	42	64619.81	43	3.04	65478.83	40.59	1.31
4	2	28	20	34455.98	20	2.67	33444.53	330.16	**-3.02**
	5	25	18	32721.19	18	1.27	32536.81	291.13	**-0.56**
	10	20	15	29687.28	15	0.66	30712.85	226.89	3.33
	15	15	12	27821.39	12	0.42	27212.92	385.24	**-2.23**
	20	10	16	30261.70	16	0.66	30262.94	230.46	0.00
	25	5	19	32577.30	19	1.44	32569.44	537.97	**-0.02**
	28	2	21	34575.26	22	1.90	34401.26	584.10	**-0.50**
6	2	28	14	24803.84	14	2.29	24547.45	51.71	**-1.04**
	5	25	12	25463.66	12	1.49	24931.40	161.04	**-2.13**
	10	20	10	24559.47	10	0.78	23508.23	481.23	**-4.47**
	15	15	8	20707.75	8	0.37	19676.93	29.78	**-5.23**
	20	10	11	21092.95	11	0.69	20958.28	5.75	**-0.64**
	25	5	13	23080.06	13	1.18	22689.04	24.28	**-1.72**
	28	2	14	25514.31	14	2.20	25700.18	22.33	0.72
8	2	28	10	21389.06	11	1.05	20129.51	198.54	**-6.25**
	5	25	9	21787.45	9	1.11	21228.52	255.26	**-2.63**
	10	20	8	21494.86	8	0.54	19990.34	451.12	**-7.52**
	15	15	6	19101.61	6	0.31	17324.90	204.81	**-10.25**
	20	10	8	19360.10	8	0.57	19359.12	5.57	0.00
	25	5	10	20206.39	10	1.06	20074.81	2.61	**-0.65**
	28	2	11	22709.76	11	1.32	20987.49	16.11	**-8.20**

Table 14.5: Comparison between solution methods: 30 Customers.

It is important to note that, the solution provided for the instance with 28 importers, 2 exporters, and truck capacity 2-containers (1.31%) is infeasible. Therefore, the solution provided by the adaptive guidance meta-heuristic cannot be considered worst.

The best improvement is found for the instance with 15 importers, 15 exporters and truck capacity 8-containers, in which the improvement in terms of objective function is about 10.25%.

The constructive heuristic provides infeasible solutions in 3/35 instances. Moreover, the infeasible instances with track capacity 4-containers and 8-containers besides being infeasible present also a worst objective function.

Table 14.6 shows the results for the class of instances with 40 customers. The adaptive guidance meta-heuristic seems to be more effective in those instances in which importers and exporters are more ore less distributed homogeneously and

				CONSTRUCTIVE HEURISTIC			ADAPTIVE GUIDANCE META-HEURISTIC		
C	$\|I\|$	$\|E\|$	$\|K\|$	o.f.	$\|K_u\|$	t(s)	o.f.	t(s)	% Gap
1	2	38	118	144271.03	118	21.36	144271.03	40.19	0.00
	5	35	108	133117.56	108	15.48	133117.56	29.40	0.00
	10	30	89	112983.25	89	8.45	112554.02	16.77	**-0.38**
	15	25	74	102098.40	74	4.37	101042.34	10.33	**-1.04**
	20	20	70	105464.07	70	2.64	104372.86	8.66	**-1.04**
	25	15	86	113529.81	86	5.44	113490.47	12.18	**-0.03**
	30	10	102	130234.34	102	10.27	130234.34	20.74	0.00
	35	5	115	144120.06	115	17.34	144120.06	33.10	0.00
	38	2	121	149974.73	121	22.26	149974.73	40.19	0.00
2	2	38	59	92010.28	60	14.30	91825.86	21.28	**-0.20**
	5	35	54	86230.88	54	8.21	86198.34	42.97	**-0.03**
	10	30	45	73208.40	45	4.28	72702.31	9.58	**-0.69**
	15	25	37	66360.85	37	2.86	65557.61	10.87	**-1.22**
	20	20	35	68521.51	36	1.48	68216.43	600.09	**-0.44**
	25	15	43	73840.76	43	2.43	73840.76	41.60	0.00
	30	10	51	83381.37	51	4.63	83381.37	10.85	0.00
	35	5	58	93152.85	58	8.04	92748.36	297.70	**-0.43**
	38	2	61	95423.90	61	8.89	95412.13	274.20	**-0.01**
4	2	38	30	49443.72	30	6.69	48557.60	377.49	**-1.82**
	5	35	27	46306.63	27	5.75	46163.18	66.31	**-0.31**
	10	30	23	40473.92	23	2.33	40421.93	22.98	**-0.12**
	15	25	19	37353.95	19	1.26	36237.66	347.10	**-3.08**
	20	20	18	39843.29	18	0.83	37073.56	3.49	**-7.47**
	25	15	22	39702.33	22	1.34	39702.33	327.76	0.00
	30	10	26	47305.16	26	2.34	44546.17	22.75	**-6.19**
	35	5	29	50132.62	29	4.03	49997.79	194.47	**-0.26**
	38	2	31	52052.77	31	6.54	51759.03	76.52	**-0.56**
6	2	38	20	34939.22	20	7.26	34667.29	451.13	**-0.78**
	5	35	18	33544.47	18	3.68	33408.62	24.51	**-0.40**
	10	30	15	30248.45	15	2.29	29012.38	22.30	**-4.26**
	15	25	13	28593.75	13	1.77	27874.28	113.98	**-2.58**
	20	20	12	29216.47	12	1.32	28348.62	79.61	**-3.06**
	25	15	15	31213.14	15	2.16	30656.34	590.16	**-1.81**
	30	10	17	33545.87	18	3.31	34108.71	528.38	1.65
	35	5	20	35416.10	20	5.18	35929.95	480.92	1.43
	38	2	21	37731.12	22	6.71	37495.77	582.79	**-0.62**
8	2	38	15	29232.08	16	3.69	28725.12	52.27	**-1.76**
	5	35	14	28501.99	14	2.66	27698.66	78.21	**-2.90**
	10	30	12	24472.65	12	2.81	24678.09	346.40	0.83
	15	25	10	23283.59	10	1.38	22149.72	429.11	**-5.11**
	20	20	9	24397.49	9	0.66	23158.69	197.34	**-5.34**
	25	15	11	25834.76	12	1.62	25291.06	453.36	**-2.14**
	30	10	14	29062.12	14	3.94	27259.17	24.91	**-6.61**
	35	5	15	29222.67	15	7.51	29207.38	100.50	**-0.05**
	38	2	16	30305.90	16	5.34	30110.53	430.62	**-0.64**

Table 14.6: Comparison between solution methods: 40 Customers.

the truck capacity increases. However, the constructive heuristic provides a slightly better solution in 3/45 instances with:

- truck capacity 6-containers
  - 30 importers and 10 exporters (1.65%);
  - 35 importers and 5 exporters (1.43%);
- 10 importers, 30 exporters and truck capacity 8-containers (0.83%).

It is important to note that, the solution provided for the instance with 30 importers, 10 exporters, and truck capacity 6-containers (1.65%) is infeasible. Therefore, the solution provided by the adaptive guidance meta-heuristic cannot be considered worst.

The best improvement is found for the instance with 20 importers, 20 exporters and truck capacity 4-containers, in which the improvement in terms of objective function is about 7.47%.

The constructive heuristic provides infeasible solutions in 6/45 instances. Moreover, the infeasible instances with track capacity 2-containers, 6-containers (38 importers and 2 exporters) and 8-containers besides being infeasible present also a worst objective function.

Table 14.7 shows the results for the class of instances with 50 customers. The adaptive guidance meta-heuristic seems to be more effective in those instances in which importers and exporters are more ore less distributed homogeneously and the truck capacity increases. However, the constructive heuristic provides a slightly better solution in 5/55 instances with:

- 45 importers, 5 exporters and truck capacity 2-containers (0.01%);
- truck capacity 4-containers
  - 10 importers and 40 exporters (0.02%);
  - 45 importers and 5 exporters (0.67%);
  - 48 importers and 2 exporters (0.64%);
- 48 importers, 2 exporters and truck capacity 6-containers (1.45%).

It is important to note that, the constructive heuristic provides a slightly better solution in 2/55 instances with *(i)* 2 importers, 48 exporters, and truck capacity 2-containers (0.80%) and *(ii)* 30 importers, 20 exporters, and truck capacity 4-containers (0.60%), but these solutions are infeasible. Therefore, the solutions provided by the adaptive guidance algorithm cannot be considered worst.

The best improvement is found for the instance with 25 importers, 25 exporters and truck capacity 4-containers, in which the improvement in terms of objective function is about 7.06%.

The constructive heuristic provides infeasible solutions in 3/55 instances. Moreover, the infeasible instance with 15 importers, 35 exporters and truck capacity 2-containers besides being infeasible presents also a worst objective function.

				CONSTRUCTIVE HEURISTIC			ADAPTIVE GUIDANCE META-HEURISTIC		
C	$\|I\|$	$\|E\|$	$\|K\|$	o.f.	$\|K_u\|$	t(s)	o.f.	t(s)	% Gap
1	2	48	134	192213.62	134	30.10	192213.62	40.27	0.00
	5	45	129	185708.84	129	30.08	185708.84	40.33	0.00
	10	40	118	173619.22	118	23.22	173619.22	40.66	0.00
	15	35	101	150523.14	101	14	150445.56	28.21	**-0.05**
	20	30	87	136255.81	87	7.93	136165.67	19.31	**-0.06**
	25	25	75	124606.20	75	4.38	124365.86	16.15	**-0.19**
	30	20	76	128052.82	76	6.56	127734.01	17.16	**-0.24**
	35	15	93	144637.08	93	12.27	144302.36	25.31	**-0.23**
	40	10	109	161774.28	109	20.39	161753.30	40.11	**-0.01**
	45	5	120	171855.25	120	30.08	171850.36	40.34	0.00
	48	2	132	187325.59	132	30.11	187325.59	40.32	0.00
2	2	48	67	122176.00	68	20.46	123168.38	43	0.80
	5	45	65	117798.48	65	13.03	117705.11	300.55	**-0.07**
	10	40	59	110917.54	59	8.74	110449.35	17.28	**-0.42**
	15	35	51	97746.42	52	6.10	97405.67	103.93	**-0.34**
	20	30	44	87773.99	44	4.98	87633.33	11.41	**-0.16**
	25	25	38	79800.75	38	2.40	79646.66	8.25	**-0.19**
	30	20	38	81199.64	38	6.71	81193.66	15.53	0.00
	35	15	47	92389.21	47	5.95	92097.28	11.40	**-0.31**
	40	10	55	102846.30	55	10.79	102679.92	389.80	**-0.16**
	45	5	60	109666.17	60	13.95	109684.57	26.34	0.01
	48	2	66	119318.58	66	20.90	119318.58	36.74	0.00
4	2	48	34	66101.93	34	9.59	66105.02	16.49	0.00
	5	45	33	64703.01	33	8.40	64659.92	281.18	**-0.06**
	10	40	30	60849.71	30	6.30	60862.98	13.07	0.02
	15	35	26	54224.83	26	4.44	53419.97	10.59	**-1.50**
	20	30	22	48922.36	22	2.05	48443.32	6.69	**-0.98**
	25	25	19	46938.94	19	1.34	43840.72	384.92	**-7.06**
	30	20	19	45540.03	20	2.38	45815.88	41.97	0.60
	35	15	24	53105.83	24	5.18	52388.82	554.36	**-1.36**
	40	10	28	56377.23	28	6.45	55999.16	12.85	**-0.67**
	45	5	30	59640.76	30	7.94	60047.53	41.02	0.67
	48	2	33	64357.17	33	8.01	64778.09	41.39	0.64
6	2	48	23	47229.22	23	11.77	46930.96	374.24	**-0.63**
	5	45	22	48374.83	22	8.50	46698.66	30.07	**-3.58**
	10	40	20	44196.04	20	5.58	43805.92	17.76	**-0.89**
	15	35	17	41109.35	17	4.72	40682.89	9.28	**-1.04**
	20	30	15	38249.86	15	2.44	38055.15	5.66	**-0.51**
	25	25	13	37455.15	13	1.47	35395.70	5.45	**-5.81**
	30	20	13	35386.09	13	1.73	34582.74	4.73	**-2.32**
	35	15	16	38423.12	16	2.77	37992.12	11.07	**-1.13**
	40	10	19	41750.20	19	8.23	41256.29	413.71	**-1.19**
	45	5	20	45527.93	20	8.45	44709.86	40.90	**-1.82**
	48	2	22	46054.44	22	11.01	46736.41	41.09	1.45
8	2	48	17	38400.47	17	6.72	37998.30	361.94	**-1.05**
	5	45	17	39473.97	17	6.69	38450.56	14.59	**-2.66**
	10	40	15	37238.66	15	4.07	36903.76	314.15	**-0.90**
	15	35	13	35310.10	13	4.66	33712.72	9.70	**-4.73**
	20	30	11	31065.09	11	1.14	30799.15	408.85	**-0.86**
	25	25	10	30015.91	10	1.21	29030.66	329.70	**-3.39**
	30	20	10	28764.36	10	2.16	28108.41	524.65	**-2.33**
	35	15	12	33382.06	12	2.45	32038.05	6.96	**-4.19**
	40	10	14	34862.45	14	4.54	34790.30	10.77	**-0.20**
	45	5	15	35623.23	15	6.01	35609.86	26.62	**-0.03**
	48	2	17	35905.74	17	9.30	35869.52	140.19	**-0.10**

Table 14.7: Comparison between solution methods: 50 Customers.

Remarks

Tests show that the adaptive guidance meta-heuristic improves most of instances with respect to the constructive heuristic. It is important to note that savings are

computed by the same formula for both solution methods. Moreover, the improved meta-heuristic seems to be more effective in those instances in which importers and exporters are more or less distributed homogeneously and the truck capacity increases.

The constructive heuristic provides a better solution in 9/175 instances. However, provides also infeasible solutions in 17/175 instances.

From the point of view of the execution time, the constructive heuristic provides a feasible solution in a shortest time. However, feasible solution of a better quality are provided by the adaptive guidance meta-heuristic in less than 10 minutes.

14.4 Comparison with the exact algorithm

This section compares solutions provided by the adaptive guidance meta-heuristic with exact solutions provided by the exact approach, called *CPLEX*. Computational results are reported in Tables 14.8, 14.9, 14.10, 14.11 and 14.12 with the following notation:

- C : Transportation capacity;
- $|I|$: Number of importers;
- $|E|$: Number of exporters;
- $|K|$: Number of available trucks with capacity C-containers;

ADAPTIVE GUIDANCE

- *o.f.*: Objective function returned by the meta-heuristic;
- *t(s)*: Time in seconds before the best solution is found;
- *% Gap 10 min*: Percentage gap with respect to the best solution provided by CPLEX in 10 minutes. When the solutions of the meta-heuristic are better than the best CPLEX upper bounds, or the meta-heuristic provides the optimal solutions, gaps are reported in bold;
- *% Gap 3 h*: Percentage gap with respect to the best solution provided by CPLEX in 3 hours. When the solutions of the meta-heuristic are better than the best CPLEX upper bounds, or the meta-heuristic provides the optimal solutions, gaps are reported in bold;
- *n.a.*: No available gap with respect to CPLEX within its time limit, because the exact algorithm did not find any feasible solution.

CPLEX 10 min and 3h

- *Opt. Gap*: The optimality gap after 10 minutes and $3h$, respectively, between upper and lower bounds determined by CPLEX. The elapsed time in seconds is reported only when CPLEX finds the optimal solution;
- *n.s.*: No feasible solution determined by CPLEX within its time limit.

				ADAPTIVE GUIDANCE				CPLEX	
C	$\|I\|$	$\|E\|$	$\|K\|$	o.f.	t(s)	% Gap 10 min	% Gap 3 h	% Opt. Gap 10 min	% Opt. Gap 3 h
1	2	8	19	29064.82	0.11	**0.00**	**0.00**	0.00 (0.28s)	0.00 (0.29s)
	5	5	16	29692.50	0.07	**0.00**	**0.00**	0.00 (0.14s)	0.00 (0.14s)
	8	2	23	29607.10	0.14	**0.00**	**0.00**	0.00 (0.29s)	0.00 (0.31s)
2	2	8	10	20277.64	0.22	0.00	0.00	5.96	4.34
	5	5	8	19640.83	0.13	**0.00**	**0.00**	0.00 (5.83s)	0.00 (5.79s)
	8	2	12	20237.67	0.22	0.00	0.00	5.20	4.23
4	2	8	5	12707.00	0.17	0.00	**0.00**	4.07	0.00 (2769.84s)
	5	5	4	11756.63	0.11	**0.00**	**0.00**	0.00 (2.27s)	0.00 (2.22s)
	8	2	6	12400.96	15.41	0.00	0.00	7.02	3.03
6	2	8	4	9644.93	0.13	**0.00**	**0.00**	0.00 (27.41s)	0.00 (27.93s)
	5	5	3	9443.13	0.10	**0.00**	**0.00**	0.00 (0.94s)	0.00 (0.93s)
	8	2	4	10397.93	5.85	4.63	4.63	0.00 (43.27s)	0.00 (43.42s)
8	2	8	3	9251.42	0.12	**0.00**	**0.00**	0.00 (13.74s)	0.00 (14.09s)
	5	5	2	8756.39	0.09	**0.00**	**0.00**	0.00 (0.97s)	0.00 (0.81s)
	8	2	3	9894.46	16.34	4.82	4.82	0.00 (32.06s)	0.00 (32.21s)

Table 14.8: Comparison with the exact algorithm: 10 Customers.

Table 14.8 shows the results for the class of instances with 10 customers. The meta-heuristic provides the same solution, with respect to the CPLEX, in most of all instances. However, CPLEX provides slightly better solutions in 2/15 instances with:

- 8 importers, 2 exporters and truck capacity 6-containers (4.63%);
- 8 importers, 2 exporters and truck capacity 8-containers (4.82%).

The adaptive guidance meta-heuristic provides the optimal solution in 10/15 instances. From the point of view of the execution time, the meta-heuristic provides solutions in a shortest time. CPLEX is able to solve optimally 11/15 instances in less than one minute, and 12/15 instances in less than one hour.

Table 14.9 shows the results for the class of instances with 20 customers.

Referring to the time limit of 10 minutes, the adaptive guidance meta-heuristic provides better solutions in 16/25 instances, and provides the optimal solution in 5/25 instances (all instances with truck capacity 1-container).However, CPLEX provides slightly better solutions in 4/25 instances with:

				ADAPTIVE GUIDANCE				CPLEX	
C	$\|I\|$	$\|E\|$	$\|K\|$	o.f.	t(s)	% Gap 10 min	% Gap 3 h	% Opt. Gap 10 min	% Opt. Gap 3 h
1	2	18	52	61366.30	2.70	**0.00**	**0.00**	0.00 (13.56s)	0.00 (13.54s)
	5	15	45	55129.44	1.76	**0.00**	**0.00**	0.00 (8.08s)	0.00 (8.06s)
	10	10	33	48122.05	0.96	**0.00**	**0.00**	0.00 (4.46s)	0.00 (4.42s)
	15	5	45	55792.14	1.68	**0.00**	**0.00**	0.00 (8.59s)	0.00 (8.62s)
	18	2	53	64705.83	2.75	**0.00**	**0.00**	0.00 (17.89s)	0.00 (17.90s)
2	2	18	26	39766.04	1.35	**-0.20**	0.00	5.21	4.95
	5	15	23	35373.01	0.97	0.38	0.63	3.62	3.14
	10	10	17	31019.25	0.75	**-0.22**	0.00	4.32	3.87
	15	5	23	36401.67	1.02	**-1.00**	**-0.01**	6.45	5.45
	18	2	27	41955.11	1.50	**-1.35**	**-0.84**	6.38	5.90
4	2	18	13	22633.91	296.19	**-14.83**	0.00	24.36	11.96
	5	15	12	19938.48	275.93	**-3.62**	0.07	13.44	8.87
	10	10	9	18309.34	561.17	**-11.41**	0.67	20.14	8.56
	15	5	12	20484.61	41.93	**-10.23**	**-0.10**	20.91	11.76
	18	2	14	23433.85	48.99	**-11.23**	**-0.05**	21.19	11.48
6	2	18	9	16208.59	0.97	**-12.24**	**-1.09**	21.31	10.77
	5	15	8	15235.92	490.82	0.24	0.54	10.11	7.66
	10	10	6	14372.09	0.56	6.43	6.43	7.41	5.83
	15	5	8	14723.36	10.07	**-2.39**	**-0.64**	11.31	7.37
	18	2	9	16585.71	586.52	**-22.49**	**-3.37**	27.46	12.42
8	2	18	7	14271.76	8.75	**-24.97**	**-0.11**	32.28	13.37
	5	15	6	13932.00	573.21	**-83.50**	5.51	54.15	8.94
	10	10	5	13338.96	191.72	9.81	9.81	5.80	2.23
	15	5	6	13786.59	248.41	**-12.98**	2.22	26.39	12.28
	18	2	7	14596.35	4.50	**-12.44**	**-3.30**	28.45	19.89

Table 14.9: Comparison with the exact algorithm: 20 Customers.

- 5 importers, 15 exporters and truck capacity 2-containers (0.38%);
- truck capacity 6-containers
  - 5 importers and 15 exporters (0.24%);
  - 10 importers and 10 exporters (6.43%);
- 10 importers, 10 exporters and truck capacity 8-containers (9.81%).

The best improvement is found for the instance with 5 importers, 15 exporters and truck capacity 8-containers, in which the improvement in terms of objective function is about 83.50%.

Referring to the time limit of 3 hours, the meta-heuristic provides better solutions in 9/25 instances, provides the optimal solution in 5/25 instances (all instances with truck capacity 1-container). However, CPLEX provides slightly better solutions in 8/25 instances with:

- 5 importers, 15 exporters and truck capacity 2-containers (0.63%);
- truck capacity 4-containers
  - 5 importers and 15 exporters (0.07%);

  - 10 importers and 10 exporters (0.67%);

- truck capacity 6-containers

  - 5 importers and 15 exporters (0.54%);
  - 10 importers and 10 exporters (6.43%);

- truck capacity 8-containers

  - 5 importers and 15 exporters (5.51%);
  - 10 importers and 10 exporters (9.81%);
  - 15 importers and 5 exporters (2.22%).

The best improvement is found for the instance with 18 importers, 2 exporters and truck capacity 6-containers, in which the improvement in terms of objective function is about 3.37%.

From the point of view of the execution time, the meta-heuristic provides solutions in a shortest time. CPLEX is able to solve optimally 5/25 instances in less than twenty seconds.

Table 14.10 shows the results for the class of instances with 30 customers.

Referring to the time limit of 10 minutes, the adaptive guidance meta-heuristic provides the optimal solution in 7/35 instances, all instances with truck capacity 1-container, and improves all other instances for which CPLEX provides a feasible solution. Moreover, comparison gaps are not available for 9/35 instances, because CPLEX does not provide a feasible solution. The best improvement is found for the instance with 5 importers, 25 exporters and truck capacity 8-containers, in which the improvement in terms of objective function is about 115.27%.

Referring to the time limit of 3 hours, the adaptive guidance meta-heuristic provides better solutions in 26/35 instances, and provides the optimal solution in 7/35 instances (all instances with truck capacity 1-container). However, CPLEX provides slightly better solutions in 2/35 instances with:

- truck capacity 2-containers

  - 5 importers and 25 exporters (0.17%);
  - 28 importers and 2 exporters (2.66%).

The best improvement is found for the instance with 2 importers, 28 exporters and truck capacity 6-containers, in which the improvement in terms of objective function is about 87.62%.

				ADAPTIVE GUIDANCE				CPLEX	
C	$\|I\|$	$\|E\|$	$\|K\|$	o.f.	t(s)	% Gap 10 min	% Gap 3 h	% Opt. Gap 10 min	% Opt. Gap 3 h
1	2	28	79	95786.50	11.56	**0.00**	**0.00**	0.00 (305.78s)	0.00 (305.87s)
	5	25	72	89471.34	8.36	**0.00**	**0.00**	0.00 (162.37s)	0.00 (162.02s)
	10	20	60	77811.59	4.29	**0.00**	**0.00**	0.00 (68.75s)	0.00 (69.79s)
	15	15	45	73424.64	3.03	**0.00**	**0.00**	0.00 (42.34s)	0.00 (42.93s)
	20	10	61	79524.70	4.34	**0.00**	**0.00**	0.00 (75.57s)	0.00 (75.15s)
	25	5	75	92439.00	8.63	**0.00**	**0.00**	0.00 (217.93s)	0.00 (216.23s)
	28	2	84	100904.53	12.66	**0.00**	**0.00**	0.00 (373.96s)	0.00 (376.23s)
2	2	28	40	61202.75	5.61	**-7.31**	**-4.14**	10.48	7.75
	5	25	36	57999.63	3.91	n.a.	0.17	n.s.	5.10
	10	20	30	51003.60	2.43	**-13.98**	**-1.19**	17.73	7.13
	15	15	23	47883.08	1.85	**-3.78**	**-0.53**	9.10	6.15
	20	10	31	52118.06	114.88	n.a.	**-0.65**	n.s.	6.55
	25	5	38	59225.19	4.17	n.a.	**-17.13**	n.s.	18.03
	28	2	42	65478.83	40.59	n.a.	2.66	n.s.	2.72
4	2	28	20	33444.53	330.16	**-22.01**	**-15.55**	25.52	21.17
	5	25	18	32536.81	291.13	**-45.69**	**-15.21**	39.61	23.39
	10	20	15	30712.85	226.89	**-23.49**	**-14.11**	33.81	28.32
	15	15	12	27212.92	385.24	**-44.90**	**-21.04**	40.19	27.37
	20	10	16	30262.94	230.46	**-35.23**	**-4.31**	37.50	18.83
	25	5	19	32569.44	537.97	**-31.15**	**-19.97**	31.43	25.03
	28	2	21	34401.26	584.10	**-39.83**	**-25.23**	33.61	25.88
6	2	28	14	24547.45	51.71	n.a.	**-87.62**	n.s.	53.15
	5	25	12	24931.40	161.04	**-66.91**	**-36.26**	50.29	39.08
	10	20	10	23508.23	481.23	n.a.	**-19.35**	n.s.	33.78
	15	15	8	19676.93	29.78	n.a.	**-31.34**	n.s.	31.77
	20	10	11	20958.28	5.75	**-55.08**	**-18.75**	42.69	25.15
	25	5	13	22689.04	24.28	**-90.47**	**-33.22**	51.64	30.85
	28	2	14	25700.18	22.33	**-54.83**	**-35.60**	42.72	34.49
8	2	28	10	20129.51	198.54	n.a.	**-75.72**	n.s.	50.43
	5	25	9	21228.52	255.26	**-115.27**	**-17.22**	63.01	31.83
	10	20	8	19990.34	451.12	**-85.82**	**-18.21**	57.48	33.16
	15	15	6	17324.90	204.81	n.a.	**-71.26**	n.s.	48.70
	20	10	8	19359.12	5.57	**-101.61**	**-11.73**	60.97	28.88
	25	5	10	20074.81	2.61	**-93.41**	**-19.44**	57.17	30.01
	28	2	11	20987.49	16.11	**-97.86**	**-37.51**	55.71	35.92

Table 14.10: Comparison with the exact algorithm: 30 Customers.

From the point of view of the execution time, the adaptive guidance meta-heuristic provides solutions in a shortest time. CPLEX is able to solve optimally 7/35 instances in less than 7 minutes.

Table 14.11 shows the results for the class of instances with 40 customers.

Referring to the time limit of 10 minutes, the adaptive guidance meta-heuristic provides the optimal solution in 2/45 instances, all instances with truck capacity 1-container, and improves all other instances for which CPLEX provides a feasible solution. Moreover, comparison gaps are not available for 38/45 instances, because CPLEX does not provide a feasible solution. The best improvement is found for the instance with 25 importers, 15 exporters and truck capacity 8-containers, in which the improvement in terms of objective function is about 160.11%.

Referring to the time limit of 3 hours, the adaptive guidance meta-heuristic provides the optimal solution in 9/45 instances, all instances with truck capacity

C	$\|I\|$	$\|E\|$	$\|K\|$	ADAPTIVE GUIDANCE o.f.	t(s)	% Gap 10 min	% Gap 3 h	CPLEX % Opt. Gap 10 min	% Opt. Gap 3 h
1	2	38	118	144271.03	40.19	n.a.	**0.00**	n.s.	0.00 (2715.85s)
	5	35	108	133117.56	29.40	n.a.	**0.00**	n.s.	0.00 (1926.40s)
	10	30	89	112554.02	16.77	n.a.	**0.00**	n.s.	0.00 (966.72s)
	15	25	74	101042.34	10.33	**0.00**	**0.00**	0.00 (523.74s)	0.00 (523.28s)
	20	20	70	104372.86	8.66	**0.00**	**0.00**	0.00 (425.73s)	0.00 (425.26s)
	25	15	86	113490.47	12.18	n.a.	**0.00**	n.s.	0.00 (647.39s)
	30	10	102	130234.34	20.74	n.a.	**0.00**	n.s.	0.00 (1127.02s)
	35	5	115	144120.06	33.10	n.a.	**0.00**	n.s.	0.00 (2040.40s)
	38	2	121	149974.73	40.19	n.a.	**0.00**	n.s.	0.00 (2931.59s)
2	2	38	59	91825.86	21.28	n.a.	n.a.	n.s.	n.s.
	5	35	54	86198.34	42.97	n.a.	n.a.	n.s.	n.s.
	10	30	45	72702.31	9.58	n.a.	n.a.	n.s.	n.s.
	15	25	37	65557.61	10.87	n.a.	n.a.	n.s.	n.s.
	20	20	35	68216.43	600.09	n.a.	n.a.	n.s.	n.s.
	25	15	43	73840.76	41.60	n.a.	n.a.	n.s.	n.s.
	30	10	51	83381.37	10.85	n.a.	n.a.	n.s.	n.s.
	35	5	58	92748.36	297.70	n.a.	n.a.	n.s.	n.s.
	38	2	61	95412.13	274.20	n.a.	n.a.	n.s.	n.s.
4	2	38	30	48557.60	377.49	n.a.	**-26.40**	n.s.	24.99
	5	35	27	46163.18	66.31	**-40.18**	**-26.23**	33.91	26.61
	10	30	23	40421.93	22.98	**-37.61**	**-32.84**	33.64	31.22
	15	25	19	36237.66	347.10	**-41.85**	**-26.41**	35.85	27.8
	20	20	18	37073.56	3.49	**-33.27**	**-33.25**	32.02	31.91
	25	15	22	39702.33	327.76	n.a.	n.a.	n.s.	n.s.
	30	10	26	44546.17	22.75	n.a.	n.a.	n.s.	n.s.
	35	5	29	49997.79	194.47	n.a.	n.a.	n.s.	n.s.
	38	2	31	51759.03	76.52	n.a.	**-20.54**	n.s.	23.11
6	2	38	20	34667.29	451.13	n.a.	n.a.	n.s.	n.s.
	5	35	18	33408.62	24.51	n.a.	n.a.	n.s.	n.s.
	10	30	15	29012.38	22.30	n.a.	**-162.18**	n.s.	66.73
	15	25	13	27874.28	113.98	n.a.	n.a.	n.s.	n.s.
	20	20	12	28348.62	79.61	n.a.	**-93.98**	n.s.	57.68
	25	15	15	30656.34	590.16	n.a.	n.a.	n.s.	n.s.
	30	10	17	34108.71	528.38	n.a.	**-149.41**	n.s.	66.04
	35	5	20	35929.95	480.92	n.a.	**-71.52**	n.s.	49.02
	38	2	21	37495.77	582.79	n.a.	n.a.	n.s.	n.s.
8	2	38	15	28725.12	52.27	n.a.	n.a.	n.s.	n.s.
	5	35	14	27698.66	78.21	n.a.	**-110.79**	n.s.	60.12
	10	30	12	24678.09	346.40	n.a.	**-183.27**	n.s.	70.59
	15	25	10	22149.72	429.11	n.a.	**-133.48**	n.s.	62.82
	20	20	9	23158.69	197.34	n.a.	**-176.25**	n.s.	69.85
	25	15	11	25291.06	453.36	**-160.11**	**-66.90**	67.75	49.68
	30	10	14	27259.17	24.91	n.a.	**-60.87**	n.s.	47.19
	35	5	15	29207.38	100.50	n.a.	**-168.13**	n.s.	68.31
	38	2	16	30110.53	430.62	n.a.	**-173.79**	n.s.	68.67

Table 14.11: Comparison with the exact algorithm: 40 Customers.

1-container, and improves all other instances for which CPLEX provides a feasible solution. Moreover, comparison gaps are not available for 18/45 instances, because CPLEX does not provide a feasible solution. The best improvement is found for the instance with 10 importers, 30 exporters and truck capacity 8-containers, in which the improvement in terms of objective function is about 183.27%.

From the point of view of the execution time, the adaptive guidance metaheuristic provides solutions in a shortest time. CPLEX is able to solve optimally 9/45 instances in less than one hour.

				ADAPTIVE GUIDANCE				CPLEX	
C	$\|I\|$	$\|E\|$	$\|K\|$	o.f.	t(s)	% Gap 10 min	% Gap 3 h	% Opt. Gap 10 min	% Opt. Gap 3 h
1	2	48	134	192213.62	40.27	n.a.	n.a.	n.s.	n.s.
	5	45	129	185708.84	40.33	n.a.	**0.00**	n.s.	0.00 (8125.85s)
	10	40	118	173619.22	40.66	n.a.	**0.00**	n.s.	0.00 (5313.18s)
	15	35	101	150445.56	28.21	n.a.	**0.00**	n.s.	0.00 (3136.55s)
	20	30	87	136165.67	19.31	n.a.	**0.00**	n.s.	0.00 (1886.30s)
	25	25	75	124365.86	16.15	n.a.	**0.00**	n.s.	0.00 (1541.31s)
	30	20	76	127734.01	17.16	n.a.	**0.00**	n.s.	0.00 (1807.07s)
	35	15	93	144302.36	25.31	n.a.	**0.00**	n.s.	0.00 (2842.46s)
	40	10	109	161753.30	40.11	n.a.	**0.00**	n.s.	0.00 (4513.38s)
	45	5	120	171850.36	40.34	n.a.	**0.00**	n.s.	0.00 (7020.13s)
	48	2	132	187325.59	40.32	n.a.	n.a.	n.s.	n.s.
2	2	48	67	123168.38	43.00	n.a.	n.a.	n.s.	n.s.
	5	45	65	117705.11	300.55	n.a.	n.a.	n.s.	n.s.
	10	40	59	110449.35	17.28	n.a.	n.a.	n.s.	n.s.
	15	35	51	97405.67	103.93	n.a.	n.a.	n.s.	n.s.
	20	30	44	87633.33	11.41	n.a.	n.a.	n.s.	n.s.
	25	25	38	79646.66	8.25	n.a.	n.a.	n.s.	n.s.
	30	20	38	81193.66	15.53	n.a.	n.a.	n.s.	n.s.
	35	15	47	92097.28	11.40	n.a.	n.a.	n.s.	n.s.
	40	10	55	102679.92	389.80	n.a.	n.a.	n.s.	n.s.
	45	5	60	109684.57	26.34	n.a.	n.a.	n.s.	n.s.
	48	2	66	119318.58	36.74	n.a.	n.a.	n.s.	n.s.
4	2	48	34	66105.02	16.49	n.a.	n.a.	n.s.	n.s.
	5	45	33	64659.92	281.18	n.a.	n.a.	n.s.	n.s.
	10	40	30	60862.98	13.07	n.a.	n.a.	n.s.	n.s.
	15	35	26	53419.97	10.59	n.a.	n.a.	n.s.	n.s.
	20	30	22	48443.32	6.69	n.a.	n.a.	n.s.	n.s.
	25	25	19	43840.72	384.92	n.a.	n.a.	n.s.	n.s.
	30	20	19	45815.88	41.97	n.a.	n.a.	n.s.	n.s.
	35	15	24	52388.82	554.36	n.a.	n.a.	n.s.	n.s.
	40	10	28	55999.16	12.85	n.a.	n.a.	n.s.	n.s.
	45	5	30	60047.53	41.02	n.a.	n.a.	n.s.	n.s.
	48	2	33	64778.09	41.39	n.a.	n.a.	n.s.	n.s.
6	2	48	23	46930.96	374.24	n.a.	n.a.	n.s.	n.s.
	5	45	22	46698.66	30.07	n.a.	n.a.	n.s.	n.s.
	10	40	20	43805.92	17.76	n.a.	n.a.	n.s.	n.s.
	15	35	17	40682.89	9.28	n.a.	n.a.	n.s.	n.s.
	20	30	15	38055.15	5.66	n.a.	n.a.	n.s.	n.s.
	25	25	13	35395.70	5.45	n.a.	n.a.	n.s.	n.s.
	30	20	13	34582.74	4.73	n.a.	n.a.	n.s.	n.s.
	35	15	16	37992.12	11.07	n.a.	n.a.	n.s.	n.s.
	40	10	19	41256.29	413.71	n.a.	n.a.	n.s.	n.s.
	45	5	20	44709.86	40.90	n.a.	n.a.	n.s.	n.s.
	48	2	22	46736.41	41.09	n.a.	n.a.	n.s.	n.s.
8	2	48	17	37998.30	361.94	n.a.	n.a.	n.s.	n.s.
	5	45	17	38450.56	14.59	n.a.	n.a.	n.s.	n.s.
	10	40	15	36903.76	314.15	n.a.	n.a.	n.s.	n.s.
	15	35	13	33712.72	9.70	n.a.	n.a.	n.s.	n.s.
	20	30	11	30799.15	408.85	n.a.	n.a.	n.s.	n.s.
	25	25	10	29030.66	329.70	n.a.	n.a.	n.s.	n.s.
	30	20	10	28108.41	524.65	n.a.	n.a.	n.s.	n.s.
	35	15	12	32038.05	6.96	n.a.	n.a.	n.s.	n.s.
	40	10	14	34790.30	10.77	n.a.	n.a.	n.s.	n.s.
	45	5	15	35609.86	26.62	n.a.	n.a.	n.s.	n.s.
	48	2	17	35869.52	140.19	n.a.	n.a.	n.s.	n.s.

Table 14.12: Comparison with the exact algorithm: 50 Customers.

Table 14.12 shows the results for the class of instances with 50 customers.

Referring to the time limit of 10 minutes, comparison gaps are not available because CPLEX does not provide any feasible solution.

Referring to the time limit of 3 hours, the adaptive guidance meta-heuristic provides the optimal solution in 9/55 instances, all instances with truck capacity 1-container. Comparison gaps are not available for 46/55 instances, because CPLEX does not provide a feasible solution.

From the point of view of the execution time, the adaptive guidance meta-heuristic provides solutions in a shortest time. CPLEX is able to solve optimally 9/55 instances in less than 3 hours.

Remarks

Tests show that the adaptive guidance meta-heuristic improves most of the upper bounds produced by the exact algorithm. Moreover, CPLEX is not able to provide a feasible solutions in 102/175 instances within a time limit of 10 minutes, and 64/175 instances within a time limit of 3 hours.

From the point of view of the execution time, the meta-heuristic provides all feasible solutions in a reasonable time (less than 10 minutes).

Chapter 15

Conclusions

This third part of the thesis investigated a vehicle routing problem with original characteristics, such as backhauls, multiple visits, the opportunity to carry more than a container per truck, and coupled trucks and containers. All characteristics were formalized by an integer linear programming model, and an exact algorithm was used to solve several artificial instances. As tests showed, the exact method was able to solve only instances with few customers.

The proposed adaptive guidance meta-heuristic determines feasible solutions by decomposing the overall problem into an SD-VRP for the import customers and an SD-VRP for the exporters. These problems are solved by a Tabu Search heuristic and the resulting routes are merged by an ILP model. Solutions are iteratively analysed with respect to problem-specific adaptive guidance mechanisms. If drawbacks in the solution are identified, suitable penalizations are applied to the input data at the subsequent iterations. Tests show that the adaptive guidance meta-heuristic is more effective than the exact algorithm in solving artificial instances of size larger than 20-30 customers, yielding significant savings in distances travelled by the trucks. Therefore, the adaptive guidance meta-heuristic represents a promising instrument to improve the decision-making process.

The solution method proposed in Chapter 12 is presented in Lai, Di Francesco, and Zuddas 2012. The solution method proposed in Chapter 13 includes concepts from a work in progress paper, and was presented in Lai, Battarra, and Di Francesco 2012.

Thesis conclusions

Disruptions generate shocks in the shipping industry, as well as in several planning activities. Part I of the thesis investigates the planning of empty containers repositioning when uncertain events or disruptions may take place. They are supposed to be temporary events occurring in a port that prevent the empty containers from being properly repositioned in the maritime network.

Part I investigates three repositioning policies, and takes into account two different possible evolutions for each policy: one with normal operations and the other in which uncertain operations occur. Tests show that deterministic formulations are effective only when events occur as forecast. Generally speaking, the deterministic formulation leads to unsuitable decisions whenever the future is different from the point forecast. However, the future is uncertain and shipping companies cannot trust to luck, or use a "crystal ball" to observe the values of uncertain parameters. A viable method for repositioning empty containers under uncertainty is provided by a multi-scenario model, in which scenarios are linked by non-anticipativity conditions. Tests show that the multi-scenario model provides highest demand fulfilment percentages for different future evolutions with respect to deterministic approaches.

Research is still in progress to investigate more complex problem settings involving cooperation with multiple players, like other carriers and leasing companies, which can provide containers when they are in shortage. Finally, recent advances in Adjustable Robust Optimization indicate the opportunity to explicitly compare this approach to multi-scenario models.

The empty container repositioning becomes even more challenging and difficult when integrated with routing problems. In such cases, carriers often must determine simultaneously how many empty containers are carried by a fleet of vehicles and which routes must be followed by these vehicles. These problems typically arise in inland networks, in which one must plan the distribution by trucks of loaded and empty containers to customers.

Part II and III address this type of vehicle routing problems, which are motivated by a real case study occurred during the collaboration with a carrier that operates in the Mediterranean Sea in door-to-door modality. The carrier manages a fleet

of trucks based at the port. Trucks and containers are used to service two types of transportation requests, the delivery of container loads from the port to import customers, and the shipment of container loads from export customers to the port.

Part II addresses the problem involving a heterogeneous fleet of trucks carrying one or two containers, and investigates a vehicle routing problem with backhauls, multiple visits, and the impossibility to separate trucks and containers during customer service. All characteristics were formalized by an integer linear programming model, and an exact algorithm was used to solve several artificial instances. Tests show that the exact algorithm is able to solve only instances with few customers.

The proposed meta-heuristic determines a feasible solution by a variant of the Clarke-and-Wright algorithm, in which routes are merged and assigned to trucks. Next, the solution is improved by by several local search phases, in which both node movements and truck swaps are implemented. Test show that the meta-heuristic is more effective than the exact algorithm in solving artificial instances similar to real problem instances. The comparison with the carrier's decisions shows that the meta-heuristic represents a promising instrument to improve the current decision-making process, because it leads to significant savings and determines routes quickly.

Research is in progress to face problems with multiple trips for trucks, incompatibilities between container loads and trucks, as well as larger transportation capacities. New solution methods will be developed accounting for the specific characteristics of these problems.

Part III addresses the problem involving a homogeneous fleet of trucks that can carry more than a container per truck, and investigates a vehicle routing problem with backhauls, multiple visits, and coupled trucks and containers. All characteristics were formalized by an integer linear programming model, and an exact algorithm is used to solve several artificial instances. As tests show, the exact method is able to solve only instances with few customers.

The proposed adaptive guidance meta-heuristic determines feasible solutions by decomposing the overall problem into an SD-VRP for the import customers and an SD-VRP for the exporters. These problems are solved by a Tabu Search heuristic and the resulting routes are merged by an ILP algorithm. Solutions are iteratively analysed with respect to problem-specific adaptive guidance mechanisms. If drawbacks in the solution are identified, suitable penalizations are applied to the input data at the subsequent iterations. Tests show that the adaptive guidance meta-heuristic is more effective than the exact algorithm in solving artificial instances of size larger than 20-30 customers, yielding significant savings in distances travelled by the trucks. Therefore, the adaptive guidance meta-heuristic represents a promising instrument to improve the decision-making process.

Research is in progress to face problems with multiple trips for trucks, incompatibilities between container loads and trucks, as well as time window and the possibility to visit customers in any order. New solution methods will be developed accounting for the specific characteristics of these problems.

Bibliography

.

Ahuja, R. K., T. L. Magnanti, and J. B. Orlin (1993). *Network Flows - Theory, Algorithms and Applications*. Prentice-Hall, Englewood Cliffs.

Archetti, C., M. G. Speranza, and A. Hertz (2006). "A Tabu Search Algorithm for the Split Delivery Vehicle Routing Problem". In: *Transportation science* 40.1, pp. 64–73.

Bai, R. et al. (2007). "A Simulated Annealing Hyper-heuristic: Adaptive Heuristic Selection for Different Vehicle Routing Problems". In: *The 3rd Multidisciplinary International Conference on Scheduling: Theory and Applications, MISTA 2007, Paris, France*.

Battarra, M., M. Monaci, and D. Vigo (2009). "An adaptive guidance approach for the heuristic solution of a minimum multiple trip vehicle routing problem". In: *Computers & Operations Research* 36.11, pp. 3041–3050.

Ben-Tal, A. et al. (2004). "Adjustable robust solutions of uncertain linear programs". In: *Mathematical Programming* 99.2, pp. 351–376.

Berbeglia, G. et al. (2007). "Static pickup and delivery problems: a classification scheme and survey". In: *Top* 15.1, pp. 1–31.

Caris, A. and G.K. Janssens (2009). "A Local search heuristic for the pre- and end-haulage of intermodal container terminals". In: *Computers and Operations Research* 36.10, pp. 2763–2772.

Cheung, R. K. and C. Y. Chen (1998). "A two-stage stochastic network model and solutions methods for the dynamic empty container allocation problem". In: *Transportation Science* 32.2, pp. 142–162.

Cheung, R.K. et al. (2008). "An attribute–decision model for cross-border drayage problem". In: *TransportationnResearch Part E: Logistics and Transportation Review* 44.2, pp. 217–234.

Choong, S. T., M. H. Cole, and E. Kutanoglu (2002). "Empty container management for intermodal transportation networks". In: *Transportation Research Part E* 38.6, pp. 423–438.

Chung, K.H. et al. (2007). "Development of mathematical models for the container road transportation in Korean trucking industries". In: *Computers & Industrial Engineering* 53.2, pp. 252–262.

Clarke, G. and J. W. Wright (1964). "Scheduling of vehicles from a central depot to a number of delivery points". In: *Operations Research* 12.4, pp. 568–581.

Crainic, T. G. (2003). "Long-haul freight transportation". In: *Handbook of Transportation Science*. Ed. by RandolphW. Hall. Vol. 56. International Series in Operations Research & Management Science. Springer US, pp. 451–516.

Crainic, T. G., M. Gendreau, and P. Dejax (1993). "Dynamic and stochastic models for the allocation of empty containers". In: *Operations Research* 41.1, pp. 102–126.

Di Francesco, M., T. G. Crainic, and P. Zuddas (2009). "The effect of multi-scenario policies on empty container repositioning". In: *Transportation Research Part E: Logistics and Transportation Review* 45.5, pp. 758–770.

Di Francesco, M., M. Lai, and P. Zuddas (2010). "Maritime repositioning of empty containers under uncertainty". In: *41 Annual Conference of the Italian Operational Research Society, AIRO 2010, Villa San Giovanni, Italy.*

— (2013). "Maritime repositioning of empty containers under uncertain port disruptions". In: *Computers & Industrial Engineering* 64.3, 827–837.

Di Francesco, M. et al. (2012). "A Metaheuristic algorithm for the routing of trucks with single and double container loads". In: *43 Annual Conference of the Italian Operational Research Society, AIRO 2012, Vietri sul Mare, Italy.*

Erera, A. L., J. C. Morales, and M. W. P. Savelsbergh (2005). "Global intermodal tank container management for the chemical industry". In: *Transportation Research Part E: Logistics and Transportation Review* 41.6, pp. 551–566.

— (2009). "Robust optimization for empty repositioning problems". In: *Operations Research* 57.2, pp. 468–483.

EUROPA, ed. (2005). *Intermodal transport: intermodality of goods transport.* URL: http://europa.eu/legislation_summaries/other/l24179_en.htm.

Feng, C. M. and C. H. Chang (2008). "Empty container reposition planning for intra-Asia liner shipping". In: *Maritime Policy & Management* 35.5, pp. 469–489.

Glover, F. and M. Laguna (1998). *Tabu Search*. Springer.

Hart, S. (2005). "Adaptive Heuristics". In: *Econometrica* 73.5, pp. 1401–1430.

Imai, A., E. Nishimura, and J. Current (2007). "A Lagrangian relaxation-based heuristic for the vehicle routing with full container load". In: *European journaltitle of Operational Research* 176.1, pp. 87–105.

Jula, H. et al. (2005). "Container movement by trucks in metropolitan networks: modeling and optimization". In: *TransportationnResearch Part E: Logistics and Transportation Review* 41.3, pp. 235–259.

Kramer, O. (2008). *Self-adaptive heuristics for evolutionary computation*. Springer.

Lai, K. K., K. Lam, and W. K. Chan (1995). "Shipping container logistics and allocation". In: *Journal of the Operational Research Society* 46.6, pp. 687–697.

Lai, M., M. Battarra, and M. Di Francesco (2012). "Heuristics for a split delivery vehicle routing problem with clustered backhauls". In: *43 Annual Conference of the Italian Operational Research Society, AIRO 2012, Vietri sul Mare, Italy*.

Lai, M., M. Di Francesco, and P. Zuddas (2011). "A multi-scenario optimization approach to empty container repositioning under port disruptions". In: *42 Annual Conference of the Italian Operational Research Society, AIRO 2011, Brescia, Italy*.

— (2012). "Heuristics for the routing of trucks with double container loads". In: *3rd Student Conference on Operational Research*. Vol. 22. OpenAccess Series in Informatics (OASIcs). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, pp. 84–93.

Lai, M. et al. (2012). "Heuristic search for the routing of heterogeneous trucks with multiple container loads". In: *EURO working Group on Vehicle Routing and Logistics Optimization, VEROLOG 2012, BOLOGNA, Italy*.

Lai, M. et al. (2013). "Heuristic search for the routing of heterogeneous trucks with single and double container loads". In: *Transportation Research* x.x. Submitted, pp. xx–xx.

Lam, S. W., L. H. Lee, and L. C. Tang (2007). "An approximate dynamic programming approach for the empty container allocation problem". In: *Transportation Research Part C: Emerging Technologies* 15.4, pp. 265–277.

Leung, S. C. and Y. Wu (2004). "A robust optimization model for dynamic empty container allocation problems in an uncertain environment". In: *International Journal of Operations and Quantitative Management* 10.4, pp. 315–334.

Li, J. A. et al. (2004). "Empty container management in a port with long-run average criterion". In: *Mathematical and Computer Modelling* 40.1-2, pp. 85–100.

Li, J. A. et al. (2007). "Allocation of empty containers between multi-ports." In: *Journal of Operational Research* 182.1, pp. 400–412.

Macharis, C. and Y.M. Bontekoning (2004). "Opportunities for OR in intermodal freight transportresearch: A review". In: *European journaltitle of Operational Research* 153.2, pp. 400–416.

Meisel, F. and C. Bierwirth (2011). "A unified approach for the evaluation of quay crane scheduling models and algorithms". In: *Computers & Operations Research* 38.3, pp. 683–693.

Mitra, S. (2005). "An algorithm for the generalized vehicle routing problem with backhauling". In: *Asia-Pacific journaltitle of Operational Research* 22.2, pp. 153–169.

Mitra, S. (2008). "A Parallel Clustering Technique for the Vehicle Routing Problem with Split Deliveries andPickups". In: *journaltitle of the Operational Research Society* 59.11, pp. 1532–1546.

Moon, I. K., A. D. Do Ngoc, and Y. S. Hur (2010). "Positioning empty containers among multiple ports with leasing and purchasing considerations". In: *OR Spectrum* 32.3, pp. 765–786.

Nagl, P. (2007). *Longer combination vehicles (LCV) for Asia and Pacific region: Some economic implications*. UNESCAP Working Papers, United Nations. URL: http://www.unescap.org/pdd/publications/workingpaper/wp_07_02.pdf.

Namboothiri, R. and A.L. Erera (2008). "Planning local container drayage operations given a port access appointment system". In: *Transportation Research Part E: Logistics and Transportation Review* 44.2, pp. 185–202.

Office, Congressional Budget (2006). *The Economic Costs of Disruptions in Container Shipments, Washington DC, 2006*. Accessed 30 December 2010. URL: http://www.cbo.gov/ftpdocs/71xx/doc7106/03-29Container_Shipments.pdf.

Olivera, A. and O. Viera (2007). "Adaptive memory programming for the vehicle routing problem with multiple trips". In: *Computers & Operations Research* 34.1, 28–47.

Olivo, A. et al. (2005). "An operational model for empty container management". In: *Maritime Economics & Logistics* 7.3, pp. 199–222.

Parragh, S. N., K. F. Doerner, and R. F. Hartl (2008). "A survey on pickup and delivery problems. Part I: Transportation between customers and depot". In: *Journaltitle fur Betriebswirtschaft* 58.1, pp. 21–51.

Powell, W. B. and H. Topaloglu (2003). "Stochastic programming in transportation and logistics". In: *Handbooks in Operations Research and Management Science: Stochastic Programming* 10, pp. 555–635.

Savelsbergh, M.W.P. and M. Sol (1995). "The general pickup and delivery problem". In: *Transportation Science* 29.1, pp. 17–29.

Shen, W. S. and C. M. Khoong (1995). "A DSS for empty container distribution planning". In: *Decision Support Systems* 15.1, pp. 75–82.

Song, D. P. and J. Carter (2009). "Empty container repositioning in liner shipping". In: *Maritime policy & Management* 4.36, pp. 291–307.

Song, D. P. and J. X. Dong (2011a). "Effectiveness of an empty container repositioning policy with flexible destination ports". In: *Transport Policy* 18.1, pp. 92–101.

— (2011b). "Flow-balancing-based empty container repositioning in typical shipping service routes". In: *Maritime Economics & Logistics* 13.1, pp. 61–77.

Song, D. P. and C. F. Earl (2008). "Optimal empty vehicle repositioning and fleet-sizing for two depot service systems". In: *European Journal of Operational Research* 185.2, pp. 760–777.

Song, D. P. and Q. Zhang (2010). "A fluid flow model for empty container repositioning policy with a single port and stochastic demand". In: *SIAM Journal on Control and Optimization* 48.5, pp. 3623–3642.

UNCTAD. "Review of maritime transport, United Nations, New York, 2008". In: *UNCTAD - United Nations Conference on Trade And Development*.

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

Zhang, R., W.Y. Yun, and H. Kopfer (2010). "Heuristic-based truck scheduling for inland container transportation". In: *OR Spectrum* 32.3, pp. 787–808.

Zhang, R., W.Y. Yun, and I.K. Moon (2011). "Modeling and optimization of a container drayage problem with resource constraints". In: *International journaltitle of Production Economics* 133.1, pp. 351–359.