Mathematical models and solution approaches for chemical tanker scheduling and allocation problems

Submitted in partial fulfillment of the requirements

of the degree of

Doctor of Philosophy

of the

Indian Institute of Technology Bombay, India

and

Monash University, Australia

by

Anurag Ladage

(Roll No. 154194001)

Supervisors:

Prof. Andreas Ernst

Dr. Ashutosh Mahajan

Prof. Mohan Krishnamoorthy

The course of study for this award was developed jointly by

Monash University, Australia and the Indian Institute of Technology Bombay, India

and was given academic recognition by each of them.

The programme was administrated by The IITB-Monash Research Academy.

June 9, 2023

This thesis is dedicated to my parents and my wife, who have been extremely supportive throughout my PhD journey.

Thesis Approval

This thesis entitled **Mathematical models and solution approaches for chemical tanker scheduling and allocation problems** by **Anurag Ladage** is approved for the degree of **Doctor of Philosophy**.

Examiners:

. . . .

. . . .

. . . .

. . . .

Supervisor: Chairperson:

. . . .

Date: . . . .

Place: . . . .

Declaration

I declare that this written submission represents my ideas in my own words and where others ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Date: June 9, 2023

Anurag Ladage

Roll No. 154194001

INDIAN INSTITUTE OF TECHNOLOGY BOMBAY, INDIA

CERTIFICATE OF COURSE WORK

This is to certify that **Anurag Ladage** (Roll No. 154194001) was admitted to the candidacy of Ph.D. degree on 16 July 2015, after successfully completing all the courses required for the Ph.D. programme. The details of the course work done are given below.

S.No	**Course Code**	**Course Name**	**Credits**
1	IE 601	Optimization Techniques	6
2	IE 611	Introduction to Stochastic Models	8
3	IE 688	R & D Project	6
4	IE 716	Integer Programming: Theory and Computations (Audit)	0
5	IES801	Seminar	4
6	HS 791	Communication Skills -I	PP
7	IE 792	Communication Skills -II	PP
		Total Credits	**24**

IIT Bombay

Date: June 9, 2023

Dy. Registrar (Academic)

Abstract

Optimization models and solution methods have been used to improve the efficiency of transportation systems for quite some time. However, their application to maritime transportation problems has gathered interest only in the last couple of decades. These techniques are often used to build decision support systems that optimize critical planning decisions leading to a more efficient system. Optimization-related research in maritime transportation has primarily revolved around scheduling operations. Scheduling problems incorporate cargo ship activities related to transporting cargoes between different locations. These locations are either intra-port (multiple berths within a port) or inter-port (a network of multiple ports). The ships service different categories of cargoes like bulk cargoes (grain, oil, or chemicals), containers, or people. Based on the planning requirements and the type of cargo transported, cargo shipping is classified into liner and tramp shipping.

Liner shipping (also known as container shipping) is analogous to bus service, meaning it transports large volumes of cargo (inside containers) at a relatively cheaper cost, but has a pre-planned schedule. The tramp shipping segment is like a cab service. It is scheduled according to the needs of the customer. Tramp ships transport bulk cargoes that are directly stored in the compartments of the ships. Different structural specifications of different cargo ships, varied operational and tactical planning requirements of different shipping segments, and numerous safety regulations result in different scheduling problems. Our research primarily revolves around scheduling problems related to the chemical tanker industry. Chemical tankers are one of the most sophisticated types of cargo tramp ships. The sophisticated design of the tanker enables it to transport multiple chemicals simultaneously. However, it also complicates the scheduling operations and presents a challenging research opportunity from a practical and academic perspective.

We study *a chemical tanker scheduling problem* in which a multi-compartment chemical tanker picks up chemicals (within specific time windows) and delivers them to their destinations. Important scheduling decisions included in the problem definition are the arrival times of the chemical tanker at different ports on its route and the chemicals to be serviced (either pick-ups or drop-offs) at these ports. Additionally, a chemical tanker operator has to also generate a cargo-compartment assignment plan at every port. The cargo compartment assignment plan details the quantity of cargoes stored in every compartment while considering multiple safety and ship balancing requirements. Traditionally, the scheduling decisions like route planning and cargo-to-ship allocations and the cargo-compartment assignment decisions are considered two different problems. However, solving them individually can lead to cargo pick-ups that cannot be feasibly assigned to the compartments.

Consequently, we refer to the combined problem of scheduling the route, assigning cargoes to ship, and assigning the cargoes to compartments as the *single ship pick-up and delivery problem with pick-up time windows, tank allocations, and changeovers* (s-PDP-TWTAC). The s-PDP-TWTAC also allows flexibility in terms of cargo-compartment changeovers to incorporate newly loaded cargoes. Including all these compartment-related decisions as part of the scheduling requirements makes our research problem unique. Moreover, for a fixed route and fixed cargoes to ship allocations, the cargo-compartment assignment problem will be referred to as *multi-period cargo assignment problem* (mp-CAP).

We propose a mixed integer linear programming (MILP) formulation for the s-PDP-TWTAC. The MILP formulation has a significantly fewer number of decision variables and constraints in comparison to the previous formulation. The number of variables is reduced primarily due to the altered definition of the changeover decision variable. The MILP formulation gives a tighter linear relaxation than the formulations in the literature. We also introduce a more practical definition of cargo pick-up time windows. All these factors help improve our MILP formulation, making it more tractable and practically more relevant.

During our research, it became evident that there was a dearth of data related to chemical tanker scheduling problems. A comprehensive library of compartment-related data like the number of compartments, the compartment capacities, and the ship's structure was also scarce. We introduce an instance generator built on real-world data capable of generating randomised test instances for the s-PDP-TWTAC. A set of test instances have also been presented to help future researchers in this area.

Even though the MILP formulation for the s-PDP-TWTAC requires significantly lesser computational effort than previous formulations, it is still challenging to solve for large realistic networks. We propose neighbourhood search methods for generating feasible schedules for the s-PDP-TWTAC. The proposed heuristics have two phases. Phase 1 generates one or more initial solutions for the problem. The second phase improves the initial solution through a neighbourhood search that explores the solution space by either serving a new cargo or removing a port from the existing chemical tanker's route. Two heuristics use linear relaxation to restrict the solution space and generate an initial solution. Three heuristics depend on a unique integer relaxation of the s-PDP-TWTAC to generate good quality initial solutions during Phase 1. We also propose a greedy randomised adaptive search procedure (GRASP) meta-heuristic framework for our problem. These heuristics find good quality solutions in a reasonably less amount of time. The fastest heuristic generates solutions within 1 % optimality gap for more than 80 % of the test instances in under 10 seconds.

We study the structure and complexity of the mp-CAP. The MILP formulation for the mp-CAP is already considered part of the s-PDP-TWTAC. The structural analysis enabled us to design a Dantzig-Wolfe reformulation for the mp-CAP. The DW reformulation reduces the total number of constraints for the model but increases the number of decision variables exponentially. As a result, we designed a customised delayed column generation framework to solve the DW reformulation. This framework is solved only at the root node as designing an entire branch-cut & price algorithm would require advanced knowledge of branching rules, cutting planes, and tree pruning strategies, which are out of the scope of this chapter. The experiments showed that the CG framework could find good MILP solutions using fewer variables. DW reformulation helps us decompose the mp-CAP into a master problem and multiple sub-problems. The sub-problems are independent and are solved in parallel. The master problem also helps us exploit a unique shortest path structure within the mp-CAP. This eliminates the need to solve the Branch & Bound tree to get a MILP solution from a given set of master problem columns. This framework also found tighter lower bounds than the LP relaxation of the MILP for the mp-CAP. However, for bigger instances, the DW-CG framework converges slowly.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Global trade depends, to a large extent, on maritime transportation. It is the cheapest mode of transport available to haul a large volume of cargo internationally. According to Sieminski (2016), the total annual demand for solid fuels (coal) is decreasing while that for liquid fuels (crude oil, petroleum and chemicals) is increasing. Sieminski (2016) states that during 2016, the total liquid fuel consumption in the world was 93.85 million barrels per day at an average cost of $48 per barrel. It also predicts that the total liquid fuel consumption by 2017 will increase to 96.78 million barrels per day. Moreover, Sieminski (2016) states that the daily consumption of liquid fuels in India will be 4.88 million barrels per day. These liquid fuels are transported regularly via very intricate and robust transportation networks. Incredibly complex strategic, tactical and operational problems need to be solved for these transportation networks to function efficiently and profitably.

The transportation industry (for all types of cargoes; solid, liquid, gaseous or otherwise) can be broadly classified under *Vehicle Related, Airline Related, Railway Related, Maritime Related* and *Pipeline Related*. Among these, the maritime industry is responsible for hauling the greatest quantity of cargo by volume. The total volume of world seaborne shipments has increased from 4 billion tonnes (in 1990) to 9.84 billion metric tonnes (in 2014). The maritime industry accounts for about 80% of the world merchandise trade (and more than 90% in developing countries) according to United Nations (2019). Additionally, shipping is the cheapest mode of transport after the pipeline and much more flexible than the same while hauling high volume cargoes over long distances Christiansen et al. (2007).

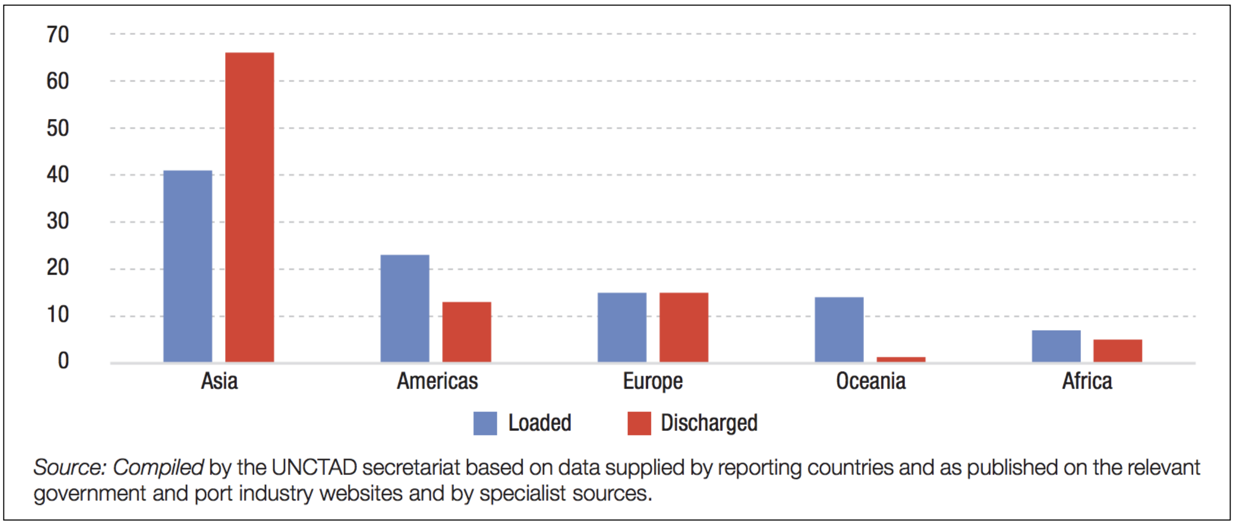

Figure 1.1: Distribution of world seaborne trade (percentage share in world tonnage) as per United Nations (2021).

Figure 1.1 shows that Asia is responsible for majority of the total world seaborne trade. However, there are enormous infrastructure deficits in developing countries like India. According to United Nations (2015), their annual infrastructure improvement budget should double from \$0.9 trillion (2013) to about \$1.8-\$1.9 trillion by 2020. Further, there has been a steady decrease in the manufacturing of new vessels, and the fleet's average age has also increased. As per Dimitrakiev and Gunes (2019), this trend of reduction in buying of new ships will continue in the years to come. They state that factors like steep increase in fuel cost, low cargo prices, and high capital cost have lead to difficult operating conditions. As such, there is an ever-increasing need to optimise the pre-existing maritime transportation networks and make them as efficient as possible.

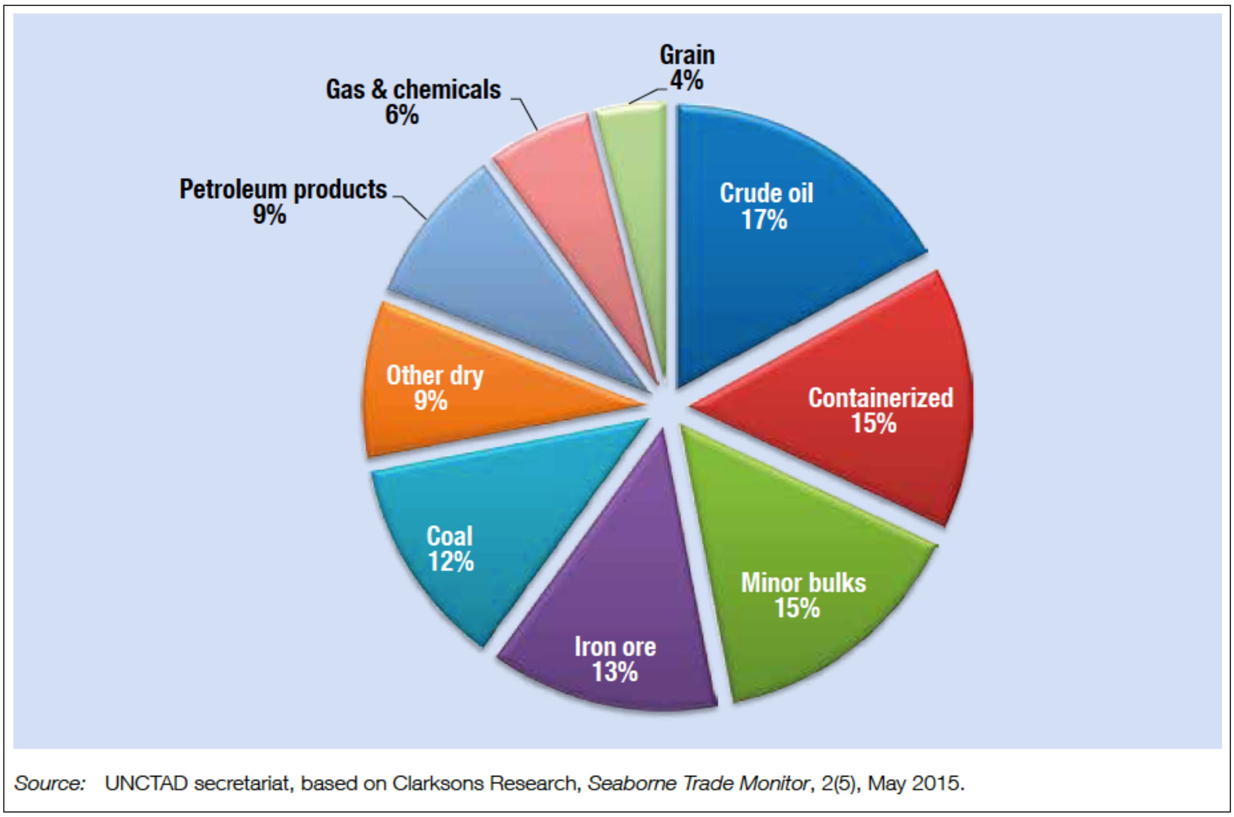

Figure 1.2: Structure of the international seaborne trade during 2014 as shown in United Nations (2015).

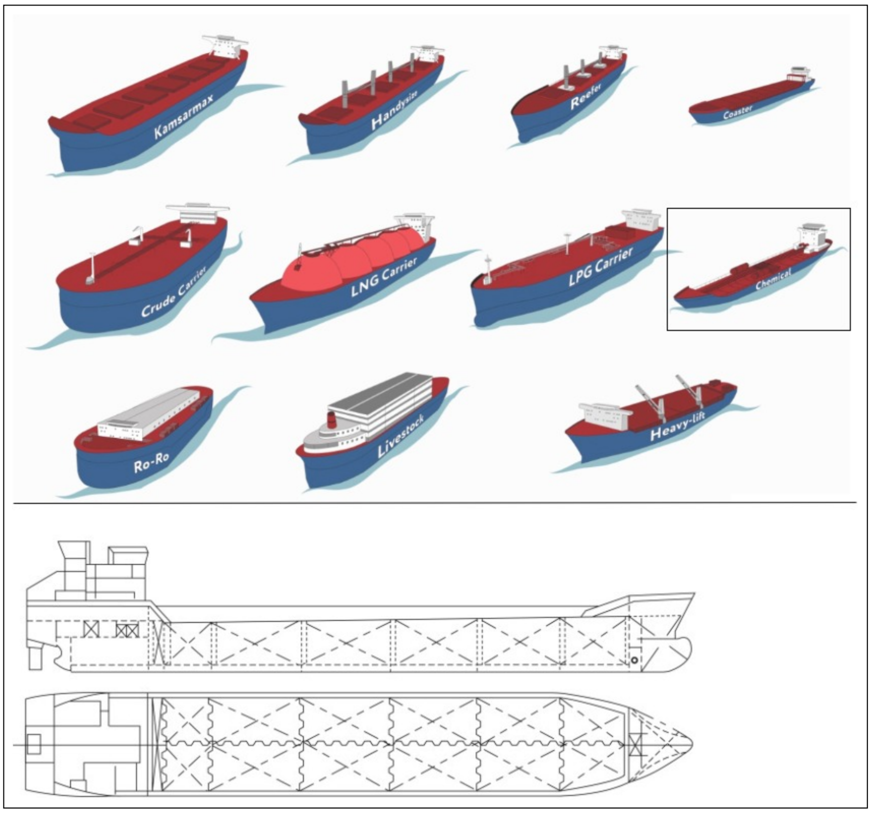

Figure 1.3: Different types of tankers, and the structural layout of a typical chemical tanker.

Based on the planning requirements and the type of cargo transported, cargo shipping is classified into liner and tramp shipping. Liner shipping (also known as container shipping) is analogous to bus service. It transports large volumes of cargo (inside containers) at a relatively cheaper cost, but it has a pre-planned schedule. The tramp shipping segment is like a cab service. It is scheduled according to the needs of the customer. Generally, liner ships transport cargoes within containers, while tramp ships transport bulk cargoes. Bulk cargoes comprising crude oil, petroleum, and chemical shipping showed a volatile freight rate environment in 2014-15. Figure 1.2 shows that 32% of the world's seaborne shipment falls under the tramp shipping segment, which transports crude oil, petroleum, and chemicals throughout the globe. A minor change in the efficiency of these massive and highly complex tramp shipping networks will result in savings amounting to billions of dollars.

Tramp ships transporting liquid bulk cargo like oil, petroleum or chemicals are known as parcel tankers. Figure 1.3 shows the different types of parcel tankers, with the chemical tanker in the highlighted box. According to United Nations (2021), the total world chemical tanker fleet stands at 48,858 at the end of 2021. Table 1.1 shows the fleet sizes owned by the most influential companies in the chemical tanker shipping segment.

Table 1.1: Global leaders in the chemical tanker segment and their fleet sizes as stated by Dimitrakiev and Gunes (2019)

Company Name	Stolt	Odjfell	Navig8	Sinochem	MOL Chemical	Nordic	Uni chartering	Bahri	Team Tankers	Womar
Fleet size	119	85	75	60	60	46	39	38	36	34

Our research focuses on scheduling problems related to the chemical tanker industry. Even amongst the parcel tankers, chemical tankers are smaller in size. However, compared to other liquid bulk cargoes, chemicals are quite volatile by nature, and it is dangerous to transport them. As such, the chemical tankers transporting them are highly sophisticated and have more compartments than other parcel tankers. The number of compartments can typically vary from 10 to 55. Figure 1.4 depicts the Bow Cecil chemical tanker used by the Odfjell shipping company[1]. It also shows a planar network representation of the ship's compartment structure. The sophis-

[1] https://www.odfjell.com

ticated design of the tanker enables it to transport multiple chemicals simultaneously. However, it also complicates the scheduling operations and presents a challenging research opportunity from a practical and academic perspective. In our research, we study two problems often faced by chemical tanker operators while transporting chemicals from one port to another. The first problem is related to generating the most profitable schedule for a chemical tanker. A schedule of a chemical tanker includes decisions like route of the tanker, arrival times at each port, cargoes to be serviced, and their allocation to different compartments of the tanker. The second problem is a special case of the scheduling problem. It deals only with decisions related to the allocation of cargo to compartments at each port on the route of the chemical tanker. We now present a detailed discussion of these problems.

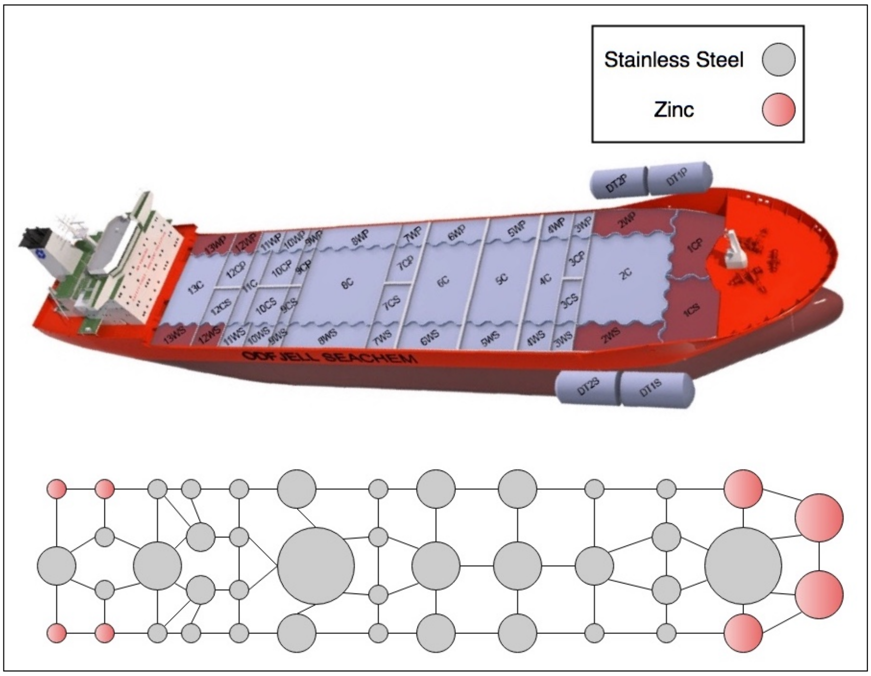

Figure 1.4: This figure shows the structure of the Bow Cecil chemical tanker managed by the Odfjell Ship Management.

We primarily study *a chemical tanker scheduling problem* in which a multi-compartment chemical tanker picks up chemicals (within specific time windows) and delivers them to their destinations. Important tactical decisions included in the problem definition are the tanker's route, the tanker's arrival time at each port on its route, and the chemicals to be serviced (either pick-ups or drop-offs) at these ports. In order to transport several non-mixable chemicals together, the chemical tanker has many compartments or cargo holds. A higher number of compartments allows for more flexible cargo-compartment allocation plans. This flexibility, in turn, increases the complexity of the problem. The compartment walls of the chemical tanker are made of stainless steel, zinc or epoxy. The compartment material might restrict the possible cargoes that can be stored in the compartment. Additionally, safety regulations may restrict the type of chemicals stored in adjacent compartments. Furthermore, the unbalanced distribution of the cargo weight in various compartments may lead to instability of the tanker. These domain-specific planning requirements make our problem quite unique and complex.

Traditionally, the tactical decisions of our scheduling problem and the cargo-compartment assignment-related operational decisions are considered two different problems. However, solving them individually can lead to cargo pick-ups that cannot be feasibly assigned to the compartments. We refer to the combined problem of scheduling the route, assigning cargoes to ship, and assigning the cargoes to compartments as the *single ship pick-up and delivery problem with pick-up time windows, tank allocations, and changeovers* (s-PDP-TWTAC). The solution to the s-PDP-TWTAC generates a schedule that is feasible for the tactical as well as operational planning requirements of the chemical tanker operators.

We also separately study the operational problem of generating the cargo-compartment assignment plan. For a fixed route and fixed cargoes to ship allocations, the cargo-compartment assignment problem will be referred to as *multi-period cargo assignment problem* (mp-CAP). Both these problems are formally defined and described in detail in Chapters 4 and 6. We present a brief overview of our main research requirements, followed by an outline for the rest of the thesis.

1.2 Main research contributions

We study the *single ship pick-up and delivery problem with pick-up time windows, tank allocations, and changeovers* (s-PDP-TWTAC) and the *multi-period cargo assignment problem* (mp-CAP). The main research contributions related to these problems are as follows:

1. We develop a mixed integer linear programming model capable of generating a schedule based on significant operational and tactical constraints for a single chemical tanker. This formulation substantially reduces the memory requirements for solving the problem, reduces the solving time compared to previous formulations in the literature, and improves

the practical usage of the solution.

2. Additionally, motivated by the unavailability of freely accessible data, we built an instance generator and a library of instances that may help develop better models and solution techniques.

3. Empirical computational results related to the MILP formulation show that even the improved model is insufficient for medium-sized benchmark instances. Thus, a systematic study of the s-PDP-TWTAC formulation is performed to gain additional insight. Subsequently, to generate good quality solutions in less time, we present a set of linear programming guided neighbourhood search heuristics and integer relaxation guided neighbourhood search heuristics.

4. We also study the mp-CAP problem. We present a MILP formulation for this problem and perform a structural analysis of the problem.

5. This analysis helps us present a Dantzig-Wolfe (DW) reformulation solved using a delayed column generation (CG) framework for the mp-CAP. This framework helps us exploit multiple special structures within the mp-CAP.

1.3 Thesis outline

Chapter 2 - Cargo shipping background This chapter starts with a short introduction to various terminologies used in the maritime transportation industry. It then presents various taxonomies used to classify the shipping industry, and a short introduction to liner shipping followed by a detailed discussion of the tramp shipping segment. We discuss some of the issues hindering the research in this area.

Chapter 3 - Operations research background This chapter focuses on chemical tanker shipping literature from an operations research perspective. We discuss the evolution of tramp shipping and various mathematical models related to the chemical tanker industry. We also talk about the different solution approaches implemented in the literature. We present the mathematical background of the Dantzig-Wolfe (DW) decomposition and the column generation (CG) algorithm. This background is essential to understand the customised DW-CG framework applied to the mp-CAP in Chapter 6.

Chapter 4 - The single ship pick-up delivery problem with time-windows, tank allocations and changeovers (s-PDP-TWTAC) This chapter describes the s-PDP-TWTAC, including the various assumptions and unique features in the problem definition. It also presents the revised MILP formulation proposed by us and highlights its major differences compared to the previous formulations. Then, this section discusses details about the instance generator's construction and design. The computational section of this chapter discusses computational details related to the generation of the test instances, the effects of improvements to the MILP formulation, and the sensitivity of some performance parameters to the input data.

Chapter 5 - Neighbourhood search heuristics for the s-PDP-TWTAC The fifth chapter presents six heuristics designed to solve the s-PDP-TWTAC. The heuristics are grouped into two main categories: linear programming guided neighbourhood search heuristics and integer relaxation guided neighbourhood search heuristics. Complete details related to the design philosophy and heuristic design are also presented in this chapter. The final section of this chapter presents an empirical study discussing the performance of various heuristics revolving around two major performance parameters, total solution time and quality.

Chapter 6 - The multi-period cargo assignment problem (mp-CAP) This chapter starts by describing the mp-CAP, followed by its MILP formulation. The chapter then discusses the Dantzig-Wolfe (DW) reformulation and the delayed column generation (CG) framework. It also discusses some special structures present within the mp-CAP, which can be exploited with the help of the DW-CG framework. Further, the chapter notes practical implementation related to termination criteria, generating initial and multiple columns, embedding heuristics within the framework, symmetry breaking, and parallel computing. The computational section of this chapter presents experiments related to the performance of the DW-CG framework and makes some comparisons with the MILP formulation.

Chapter 7 - Summary and Conclusions This chapter gives a short recap of the thesis research. It also presents key results and important conclusions that can be drawn from our research.

Chapter 2

Cargo shipping background

Operations researchers working on maritime-related problems are relatively fewer than in other application domains. As such, we think it would be useful to develop some background about the terminologies used in the maritime transportation industry. We define a few terms to familiarize the readers, and encourage them to go through Christiansen et al. (2007), Chew et al. (2015), Ronen (1983) or Christiansen et al. (2013) for more background and in-depth explanations.

Shipping is an act of moving cargo through ships, while a *shipper* is a person or entity that provides the cargo. We refer to *routing* as assigning a series of ports to individual ships without considering the time-related activities (space network). On the other hand, *scheduling* considers routing as well as time assignments for different activities (space-time network). The inclusion of temporal activities in a problem definition often makes the problem difficult to solve. The first problem considered in this thesis is a scheduling problem with pick-up time windows. Thus, both the schedule optimises not only the sequence of ports followed by the ship, but also generates arrival times for each of these ports. Additionally, the problem considered in this thesis is made more complicated due to the presence of time-windows restricting the pick-up of new cargoes.

A *loading port* or a *unloading port* is a port at which a cargo is picked up or dropped off, respectively. The journey between the harbour where the ship loads or unloads its first cargo, and the port where the ship unloads its last cargo is referred to as its *voyage*. The scheduling problem presented in Section 4.2 has exactly one loading and one unloading port for each cargoes. Meaning, the entire quantity of a single cargo is picked up a single port and the entire quantity has to be discharged at its unloading port.

A *sailing leg* is the journey of a ship between two consecutive ports it visits, while a *loaded leg* is the load carrying leg in a ship's voyage. Legs are analogous to arcs in a network while voyage is similar to a path in a network. The placement of the cargoes in a ship in a manner that provides ship stability is called *stowage*. A good stowage plan would ensure the ship's stability, as well as easy offloading of the cargoes. The capacity of a ship is often calculated in terms of dead-weight tonnage (DWT). *DWT* is the difference between loaded and unloaded ship. Another factor affecting the capacity of the ship is the draft. *Draft* is the vertical distance measured between the waterline and the bottom of the ship's hull.

We refer to *cargoes* as a set of goods shipped from its source to its destination and *load* is the set of cargoes on board a ship at any time. A *product* is defined as a set of goods or cargoes that can be stored together in the same compartment, and have the same loading and unloading port. Cargoes can also broadly be classified as bulk cargoes and containerized cargoes.

There are many different types of ships in the maritime industry. According to Christiansen et al. (2007), some of the important categorizations are as follows:

- Tankers: These ships carry liquid in bulk. They are usually in three sizes: crude oil tankers, large parcel tankers and small parcel tankers.
- Bulk carriers: Carry dry bulk commodities like iron ore or coal.
- Liquefied gas carriers: Carry refrigerated gas under pressure.
- Container ship: Standardised metal containers.
- General cargo: These carry all kinds of goods in their holds and decks.
- Reefers or refrigerated vessels: Carries cargo that needs temperature control.
- Roll on- roll off: These ships have ramps for vehicles like trucks and cars.

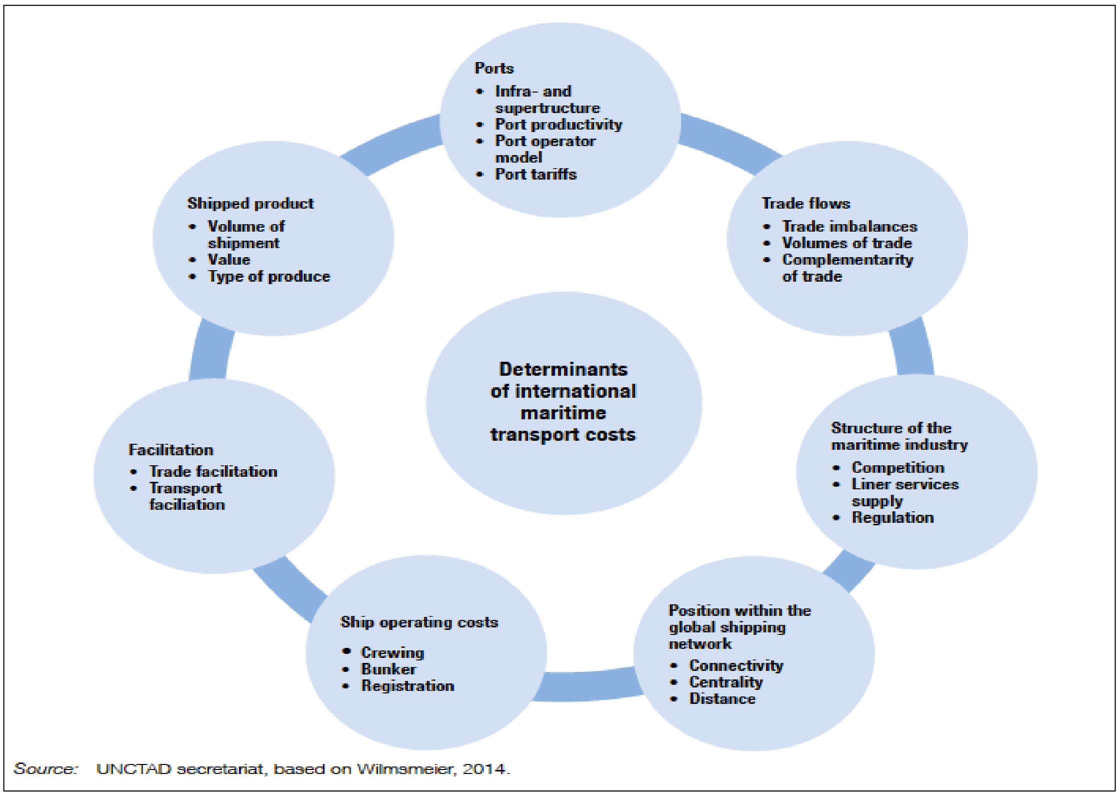

Figure 2.1: Various sources of costs incurred by cargo ship operators as mentioned in United Nations (2015).

Christiansen et al. (2007) give a useful classification of various operations research (OR) models and solution methods used in maritime transportation. Maritime OR problems are classified as follows:

- Type of shipping industry (Liner, Tramp or Industrial Shipping)
- Type of planning decisions (Strategic, Tactical or Operational)
- Fleet type (Homogeneous or Heterogeneous)
- Types of cargo (Bulk (Dry or liquid), Container or Passenger)
- Geographical characteristics (Deep Sea, Short Sea, Coastal or Inland Waterways)

From an operations research perspective, the industrial and tramp shipping sectors are very similar. They are often grouped together in the literature, but they do have their own unique characteristics. In industrial shipping, the cargo is often owned by the shipping company. Thus, their primary objective is to transport the given set of cargoes at minimum cost. On the other hand, in tramp shipping, the shipping company choose to transport cargoes that maximize their revenue. The focus of this thesis is on problems faced by chemical tankers, which are a special type of tramp ship.

However, liner and tramp shipping are very different. Liner shipping is similar to a bus service. It has fixed schedules and routes. In this case, cargo follows the schedule of the ships. Some large-scale, long-term contracted cargo decides the liner schedule. However, it might also be decided due to strategic or political decisions to maximise the profit. It is quite the opposite of tramp/industrial shipping. Tramp shipping is very similar to a taxicab service, meaning they travel to the cargo as per demand. Along with the different types of cargoes transported, the cargo shipping industry has to also manage different types of transportation costs shown in Figure 2.1. Thus, the models and solution methods that are required in each of these areas are different. The literature on maritime transportation suggests that liner shipping is analogous to container shipping, while liquid products like crude oil, chemicals and petroleum are transported using tramps.

Christiansen et al. (2007) further divides the operations research models for the optimisation of design and operation functions in the shipping industry as follows:

- Strategic planning problems including – ship designing problems, fleet size and mix problems, liner network design problems, maritime transportation design issues, and contract evaluation problems.
- Tactical planning problems including – scheduling/routing problems in industrial and tramp shipping, maritime supply chain/inventory routing problems, liner fleet deployment problem, barge scheduling problem, schedule of marine vessels and ship management problems.
- Operational planning problems including – operations scheduling, environmental routing (waves or ocean currents), optimum, speed selection, ship loading and single order bookings.

Let us first understand the difference between problems in the shipping industry and traditional vehicle routing problems. Vilhelmsen et al. (2015) mention the following modelling characteristics that separate these two domains.

- Continuous operations: Ships operate around the clock. As such, a delay in the schedule might not be absorbed due to a lack of downtime in ship operations. Additionally, this also points to the fact that ships might have different starting points during the start of the schedule.

- Lack of common depot: In most VRPs, the trucks must return to their respective depots at the end of the planning horizon. Thus, the starting points are fixed. In tramp shipping, on the other hand, a ship generally never returns to its original point immediately. The ships might also be in transit or at their refuelling ports at the start of the planning horizon.

- Compatibility issues: These issues are much more complex in the shipping industry than in vehicle routing. Many restrictions between ship-cargo, ship-port and ship route can arise due to safety issues, lack of equipment, incompatible draft conditions, unavailability of necessary cargo or political ties between two nations.

- Optional cargoes: The presence of optional cargo leads to a prioritised list of cargoes, which is generally not used in vehicle routing problems.

The above list of differences is non-exhaustive. Additionally, differences between ship routing with other modes of transport like airways and trains have been discussed in Ronen (1983), Ronen (1993) and Christiansen et al. (2004). We request the readers to refer to these papers to better understand the difference between maritime transportation models and others. This thesis focuses on chemical tanker operations, a subset of the tramp shipping sector. We will briefly discuss some aspects of liner shipping before focusing on the tramp shipping domain and the chemical tanker industry.

2.1 Liner Shipping

Liner shipping, as explained earlier, is similar to a bus service. A liner shipping company publishes its routes and schedule at the beginning of the time horizon. These routes and schedules are finalised based on long-term shipping commitments. The routes might also be decided on factors like maximising the utility of ships and the percentage of spot cargoes. Once these schedules are published, the cargo owners have to synchronise their exports according to these schedules. The schedule's restrictions are balanced by the low cost incurred while using liner services. It can be seen from the literature that liner shipping is analogous to container shipping.

A standard 20 feet or 40 feet container is used to transport the cargo, and the shipping capacities are mentioned in *Twenty feet equivalent units (TEUs)*. Tran and Haasis (2015) mention that the ports on a liner route must be visited every week. Kjeldsen (2011) and Tran and Haasis (2015) give excellent classifications of the liner shipping industry. Tran and Haasis (2015) review more than 120 papers in the liner shipping segment published during 1979-2013. Tran and Haasis (2015) attempt to classify the entire liner shipping domain while Kjeldsen (2011) focus only on the liner routing and scheduling problems. Kjeldsen (2011) review a total of 24 papers from 1969 to 2010. Liner shipping problems can be classified as shown in Table 2.1.

Table 2.1: Liner shipping problem classification[Kjeldsen (2011), Tran and Haasis (2015)]

1. Container routing	2. Liner routing and scheduling	3. Network design
1.1 Single period 1.2 Multiple period	2.1 Routing without fleet management 2.2 Routing with fleet management 2.3 Scheduling without fleet management 2.4 Scheduling without fleet management	3.1 Optimal single route 3.2 Optimal multiple routes 3.3 Hub and spoke network

2.2 Tramp Shipping

This section briefly reviews the literature on the tramp shipping sector. Tenold and Murphy (2007) give an excellent description of the shipping industry from 1960 to 1985. They discuss the history of the parcel tanker industry, which includes the growth of three prominent shipping companies (Stolt-Nielson, Panocean-Anco, and Odfjell Group). The paper also explains how, despite substantial financial backing, the pan-ocean company bowed out of the parcel tanker industry around 1983. The paper states two main reasons for this: (a) fleet management and (b) strategic timing.

One of the first maritime transportation review papers was published by Ronen (1983). Christiansen et al. (2007) specifically focuses on the evolution of scheduling problems in tramp shipping. Further, Christiansen et al. (2013), Vilhelmsen et al. (2015), Pache et al. (2019) and Pache et al. (2020) extensively review the research carried out in the field of tramp shipping since 1983. Together they review more than 50 research papers, most of which have been published in the last decade.

Optimization-related research in *Tramp shipping routing and scheduling problems* (TSRSP) only started picking up pace around a decade ago, even though the industry has been around for more than 100 years. According to Vilhelmsen et al. (2015), some of the reasons are as follows:

- Conservative and competitive industry: The shipping industry has a relatively small number of players involved. As such, the industry is highly competitive and secretive regarding its operations. There has been less exchange of real-world problems and data between the industry and researchers. This has been a significant hindrance in the digitization of shipping-related operations.
- Industry under pressure: Till recently, there was a considerable gap between industry research and academic research. The research outputs were either too theoretical or were unable to handle the complexities of practical models. This uncertainty of results, coupled with the traditional outlook of the industry, has led to the industry focusing its resources (that are already scarce) on more pressing matters.
- Highly uncertain and disruptive operating conditions: Stochasticity in maritime transportation is much higher when compared to other modes of transport. Many unforeseen events like the change in weather or daily variation in demand coupled with very long voyages made it almost impossible to plan for entire voyages ahead of time. Thus, the traditional vehicle routing models could not be directly translated to TSRSP.
- Simplified Problems: Most of the problems tackled by academicians were oversimplified versions of their real-world counterparts. Additionally, until recently, researchers were not equipped with hardware and software advanced enough to handle real-world problems.

However, recent advancements in the field of mathematics and improved computer software and hardware capabilities are gradually offsetting these issues. Both researchers and industry personnel are willing to work together with a common goal of tackling practical problems. Advancements in the tramp shipping industry are also being propelled by the fact that most domains have a wave of digitization and automation.

Pache et al. (2019) and Pache et al. (2020) present an overview of the recent trends and advances related to the tramp shipping segment. They derive their finding from thirty nine research papers published in the last decade. They mention that strategic problems tackled by researcher mostly deal with optimal fleet sizing decisions, while tactical problems revolve around generating routes and scheduling for tramp ships. Moreover, they also mention that operational problems focus on cargo routing or cargo-compartment assignment related decisions. Relatively few papers tackle problem combining tactical and operational decisions. Pache et al. (2020) divide the research related to tramp shipping domain as follows:

1. Planning horizon: Tramp shipping problems are classified as short term, medium-term and long term. Short term problems generally occur while dealing with operational decision, medium term problems generally occur when dealing with tactical decisions, while long term problems are problems focusing on strategic decisions.

2. Vessel type: Pache et al. (2020) segregate research based on vessel types. Vessel type are defined as bulk carriers, tankers and others.

3. Voyage distance: Different categories of voyage distances include deep sea and short sea shipping.

Pache et al. (2019) also state that compared to liner shipping, tramp shipping is easier to enter, which makes this segment highly competitive. Additionally, absence of fixed schedules, short term contractual cargoes and operating models dictated by availability of cargo make tramp shipping highly uncertain. Pache et al. (2020) segregate the tramp shipping literature on basis on research related to variable speed, environmental aspects, extended cargo constraints, bunkering decisions and uncertainties. We now move our discussion towards chemical tanker shipping, and some of its unique aspects.

2.3 Chemical tanker shipping

Chemical tankers are specialized vessels designed to transport a wide range of liquid chemicals and petroleum products, while other types of tankers, such as oil tankers, are specifically designed for the transportation of crude oil or refined petroleum products. The chemical tanker operators transport multiple chemicals over a network of ports. As discussed in Chapter 1, chemical tankers are smaller than oil tankers. However, their highly sophisticated design enables them to transport multiple chemicals (cargoes) simultaneously. Due to the dangerous nature of the chemicals these tankers have to abide by multiple safety related rules and regulations as defined under the international bulk chemical (IBC) code.

Contrary to oil tankers, which are generally classified based on their capacities, the IBC code defines three types of chemical tankers based on the chemicals which they transport. The ST1 category of chemical tankers transport the most dangerous category of products. Due to the hazardous nature of chemicals it transports, these ships are required to be designed in such a way that meet extreme damage resistance. Similarly, categories ST2 and ST3 have to adhere to comparatively milder safety requirements. Multiple storage requirements and compartment coating regulations also have to be adhered to depending on the specifications included in the IBC code. Here are some of the key differences between chemical tankers and other types of tankers:

- Voyage planning: Chemical tankers must carefully plan their voyages to ensure that they comply with international regulations, avoid dangerous weather conditions, and minimize the risk of cargo contamination or spillage. Oil tankers may require less extensive voyage planning.
- Safety systems: Chemical tankers are equipped with specialized safety systems, such as emergency shutdown systems and gas detection systems, to ensure the safe handling and transportation of hazardous chemicals. Oil tankers may require less extensive safety systems due to the less hazardous nature of the cargo.
- Regulatory compliance: Chemical tankers are subject to a range of international regulations governing the transport of hazardous chemicals, such as the International Maritime Dangerous Goods (IMDG) Code, while oil tankers are subject to regulations governing the transportation of petroleum products, such as the International Convention for the Prevention of Pollution from Ships (MARPOL).
- Cargo compatibility: Chemical tankers are designed to carry a wide range of liquid chemicals, such as acids, alkalis, and organic solvents, which require specialized cargo handling equipment and tanks with a high degree of resistance to chemical corrosion. In contrast, oil tankers are designed to carry crude oil or refined petroleum products, which are generally less corrosive than chemicals.

- Tank construction: Chemical tankers typically have a greater number of smaller cargo tanks than oil tankers, which have fewer and larger tanks. Chemical tanks are often made of stainless steel or coated with specialized chemical-resistant coatings, while oil tankers are commonly constructed of mild steel.
- Crew training: Due to the specialized nature of chemical tankers, crew members must receive extensive training on the handling and transportation of hazardous chemicals. In contrast, crew members on oil tankers may require less specialized training due to the more straightforward nature of the cargo.
- Cargo segregation: Chemical tankers must ensure that incompatible chemicals are not stored or transported together, which requires careful planning and segregation of cargo. Oil tankers typically transport a single type of petroleum product and do not require the same level of cargo segregation.
- Cleaning requirements: Chemical tankers must undergo thorough cleaning between cargoes to prevent contamination, which can be a time-consuming and costly process. Oil tankers may require less extensive cleaning between cargoes.
- Cargo value: Chemicals typically have a higher value than petroleum products, which can make chemical tankers more attractive targets for piracy and theft.
- Safety systems: Chemical tankers are equipped with specialized safety systems, such as emergency shutdown systems and gas detection systems, to ensure the safe handling and transportation of hazardous chemicals. Oil tankers may require less extensive safety systems due to the less hazardous nature of the cargo.

Extending the discussion put forth by Pache et al. (2020), the chemical tanker scheduling problem discussed in this thesis can be viewed as follows. We research a short term planning problem that can be solved as a rolling horizon problem to tackle medium term planning horizon problems. This makes it a tactical problem. Our primary research falls under the domain of tanker scheduling problems, with its focus being chemical tanker scheduling. Additionally, we model inter port activities of the tanker, which makes our problem a deep sea shipping problems. A set of problems within chemical tanker scheduling which specifically focuses on intra port activities can be classified as short sea shipping problems.

This chapter discussed various terminologies and classifications of the maritime transportation industry. We briefly discuss various aspects of the liner and tramp shipping industries, followed by a short discussion around chemical shipping and its unique characteristics. The following chapter explores operations research literature on parcel tankers, emphasising on the chemical tanker shipping from an operations research perspective .

This page was intentionally left blank.

Chapter 3

Operations research background

As we have already seen, tramp shipping problems can be classified in multiple ways, depending on the type of shipping industry, the different port operations, and the type of products being transported. However, it is also imperative to understand the different mathematical models and methods used to solve problems faced in the chemical tanker industry. This chapter discusses the chemical tanker literature from an operations research perspective. We discuss various mathematical formulations, heuristics, and exact approaches implemented by researchers to tackle the multitude of problems faced by the chemical tanker industry. The multi-period cargo-assignment problem (mp-CAP) is solved us a customised Dantzig-Wolfe (DW) and column generation (CG) framework. This section also presents the mathematical background required to understand this framework.

3.1 Evolution of tramp shipping and related mathematical models

Tramp shipping problems fall under a broad class of network problems. Network problems are represented as a graph, $G(N,A)$. The N stands for nodes, and A represents directed arcs. A shipping network's loading/unloading ports are described using nodes. The arcs (directed) represent the connections between the different ports. These arcs have some characteristics associated with them. For instance, in our problems, some parameters related to each arc are: fixed cost (for setting up the arcs), variable cost (for transporting a unit quantity of commodities on these arcs), distance travelled on the arcs and so on. Similarly, some parameters associated with the nodes are: the cargoes loaded or unloaded at a particular node, the discharge rate at an individual node, and the earliest and latest pickup times at these nodes (ports).

One of the simplest types of routing-scheduling problems is *Full Shiploads Routing and Scheduling Problem*. The main objective here is to maximize the profit margin. A load is called a full shipload when it consists of only one type of cargo. The one cargo per ship restricted can be due to various reasons like the incompatibility of cargo with other cargoes or contractual agreements. The time-window constraints are levied in addition to total shipload constraints. This problem can be used with minor modifications for full shipload cargoes with variable sizes. A straightforward extension of the formulation mentioned above is *Multiple Cargoes with Fixed Cargo Size*. In this model, the ships are allowed to carry multiple cargoes concurrently. Generally, the cargo delivered by tramps can be classified as contracted and optional cargo. Contracted cargo has to be delivered at all costs, while optional cargoes may be delivered if the schedule allows the ship to do so. Delivering both optional cargo and contracted cargo sometimes requires spot vessels. Thus, the following extension allows the use of spot cargoes in the model. There has been much research on these models, and good usable results have been obtained. In the next chapter, multiple extensions of the TSRSP will be studied. Thus, there is a need to define a base model on which these extensions are defined. We will call this model the basic TSRSP. A basic TSRSP will include multiple cargoes with fixed parcel sizes. It will also allow the use of spot cargoes. Vilhelmsen et al. (2015) state that the inclusion of spot cargoes does not add to the model complexity.

The basic TSRSP is a simplified model and cannot be used to model real-world situations. Thus, we need to include more realistic conditions in our basic TSRSP model. Let us look at some extensions of the basic TSRSP model. The first extension includes *Multiple Cargoes with Flexible Sizes*. Fagerholt and Christiansen (2000a) and Fagerholt and Ronen (2013) study this problem, which allows for flexible cargo holds, meaning the ship capacity occupied by different cargoes are unequal. Additionally, Bronmo et al. (2007), Korsvik and Fagerholt (2010) and Bronmo et al. (2010) are some of the people who have worked on the tanker scheduling problem with adjustable sizes.

The above TSRSP models and extensions ignore one important fact. On numerous occasions, the onboard cargo might not be mixable. The inclusion of this fact leads us to the next extension, *Routing and Scheduling of Multiple Products*. A routing and scheduling problem with multiple products involves ships with multiple tanks/compartments known as parcel tankers. These tanks facilitate the transportation of multiple products. Scott (1995), Bausch et al. (1998), Sherali et al. (1999), Fagerholt and Christiansen (2000a), Fagerholt and Christiansen (2000b) and Jetlund and Karimi (2004) are some of the researchers who have tackled this problem. A particular cargo might be picked up from multiple locations and delivered to multiple locations. Consequently, the next extension allows cargo splitting for pick-up and delivery. Stålhane et al. (2012), Korsvik et al. (2011), Hennig et al. (2015), Nishi and Izuno (2014), Chan et al. (2014) and Hennig et al. (2012) have worked on split pick-up and delivery problems.

Since 2011, another set of problems that has received some attention are models, including speed optimisation. Considering speed as a decision variable makes some constraints of the models mentioned above non-linear. Additionally, the recent increase in fuel cost and increasing awareness of environmental safety have motivated Vilhelmsen et al. (2015), Norstad et al. (2011), Gatica and Miranda (2011), Castillo Villar et al. (2014), and Wen et al. (2016) to explore research opportunities in tanker scheduling problems with speed optimization.

Due to the increasing fuel cost, models with bunkering features have been researched. Refuelling becomes important in tramp shipping due to long voyages of ships. Bunkering becomes a necessity, and the refuelling cost can be high. Vilhelmsen et al. (2014) and Meng et al. (2015) have worked on models that account for the bunkering feature. In some practical problems faced by the industry, both the cargo and the ships are owned by the same company as in industrial shipping. In such a scenario, the owner is responsible not only for the ships' optimal functioning but also for inventory management and other supply chain decisions. This gives rise to a set of problems called *Maritime Inventory Routing Problems*. Christiansen et al. (2013) give a good introduction intro maritime inventory routing problems. Further, Andersson et al. (2010) review various inventory management and routing problems in maritime and road-based systems. In the next section, we will discuss the literature, different mathematical models and solution methods related to the chemical tanker industry.

3.2 Optimization problems in the chemical tanker industry

One of our problems of interest is an extension of the multi-product TSRSP applied to the chemical shipping industry. It simultaneously deals with tactical (routing and scheduling) and operational (tank allocation and safety) decisions. We will refer to this problem as the *single ship pick-up delivery problem with time windows, tank allocations and changeovers (s-PDP-TWTAC)*. Mathematical models for the s-PDP-TWTAC are presented in Section 4.2. In this section, we discuss some of the problems pertaining to the the chemical tanker industry, solved using optimization techniques like mixed integer programming and heuristics.

Depending on the application, problems are either modelled as an extension of the travelling salesman problem with pick-ups and deliveries (TSPPD) or the pick-up delivery problem with time windows (PDPTW). In the context of tramp shipping, the TSPPD type of problems is a special case of the PDPTW type of problems. TSPPD consider important decisions like the sequence of ports to be visited, and the sequence of cargo pickups and drop offs. In addition to the primary decisions considered by TSPPD, the PDPTW also optimizes the set of ports to be visited, and the set of cargoes to be serviced. Our formulation is an extension of the PDPTW.

Extensions of the TSPPD optimise the intra-port movement of the chemical tankers and have time windows related to the loading/ unloading of cargoes. TSPPD-based formulations assume that all the berths (analogous to ports) must be visited. Additionally, all cargoes are loaded or unloaded at their origins and destinations. As output, the problem generates a sequence of berths to visit and the arrival times at each berth. Unlike the second variation, the TSPPD-based extensions also generate the sequence in which every cargo is picked up or discharged. This makes the TSPPD-based extensions different in comparison to the PDPTW-based extensions.

A recent study extending the TSPPD-based formulations has been carried out by Elgesem et al. (2018). They consider stochastic waiting times at different terminals within the port. Additionally, they perform experiments to prove that including uncertainty in their models directly affects the optimal route of the chemical tankers. However, unlike our problem, Elgesem et al. (2018) do not generate a cargo-compartment allocation plan as part of their problem output.

Another study related to TSPPD-based formulations is presented by Wang et al. (2018). There are some similarities between the problems presented in this paper and the one presented by Wang et al. (2018). Similar to our problem, Wang et al. (2018) incorporate draft requirements and compartment-related decisions into their problem definition. The compartment-related decisions include compartment capacities, ship stability criteria, cargo-cargo and cargo-compartment incompatibility guidelines. However, our problem allows more feasible cargo compartment allocations by permitting cargo swaps between the compartments. Further, because their formulation is based on TSPPD, some higher-level decisions, like generating an optimal set of ports and cargoes, are fixed.

The second set of variations of the chemical tanker scheduling problems is an extension of the PDPTW problem. This set of problems is defined over a set of ports (inter port) instead of a set of berths/terminals (intra-port). In addition to generating a sequence of ports to visit, the problems in this set also decide the subset of ports to visit and the subset of cargoes to pick up. Fagerholt and Christiansen (2000a) propose a problem in dry bulk shipping by modifying the PDPTW formulation. They generate a feasible cargo-compartment allocation based on the capacities of the compartments. However, they do not consider tanker stability conditions and cargo-cargo incompatibilities.

Jetlund and Karimi (2004), and more recently by Lin and Liu (2011), Cóccola et al. (2015) and Cóccola and Méndez (2015) modify the PDPTW to propose a multi-ship pick-up delivery problem with pickup time windows. Their formulations generate feasible schedules for a fleet of heterogeneous chemical tankers, which considers all the decisions shown in the first column of Table 3.1. Hennig et al. (2015) and Homsi et al. (2020) solve the multi-ship pick-up delivery problem with pick-up time windows with an added complexity of splitting cargoes between chemical tankers. Furthermore, Homsi et al. (2020) also includes time-windows related to drop-offs. However, the second variations of problems discussed until now do not consider compartment-related decisions (mentioned earlier).

Neo et al. (2006) and Giavarina dos Santos et al. (2020) have worked on PDPTW based formulations that incorporate the compartment-related decisions in their problem definition. The problem discussed by Giavarina dos Santos et al. (2020) is transporting fertilisers (chemicals) from their origins to their destinations. Although they include the compartment capacity in their problem, they do not consider the ship stability conditions, the incompatibility norms, the cargo distribution into multiple compartments, or cargo swapping activity. Moreover, Giavarina dos Santos et al. (2020) fix the cargoes that must be picked up, including soft time windows as opposed to hard time windows, considered by us. However, unlike us, they solve a multi-ship problem with split loads.

Over the years, researchers have introduced numerous formulations to tackle the problems faced by the chemical tanker industry. This has necessitated using different heuristics and exact methods to solve these formulations. Recall that the s-PDP-TWTAC is an extension of the PDPTW problem. As such, we discuss some of the existing solution approaches for tackling some extensions of the PDPTW problem within the tramp shipping industry.

Lin and Liu (2011) study the problem of transporting dry bulk cargoes over a network of ports. They view the problem as a combination of two different flows: the flow of the ship through different ports and the cargoes. They apply a genetic algorithm that improves the solution by mutating the cargoes assigned to the tanker. However, they do not consider the pick-up time windows and model dry bulk cargoes.

Jetlund and Karimi (2004) presents a chemical tanker scheduling problem to transport chemicals optimally over a network of ports. Hard pick-up time windows and multi-product delivery add to the problem's complexity. They propose a MILP formulation for the single-ship version of their problem. Further, they propose an approximation heuristic to construct the solution to the multi-ship problem based on the solution to the single-ship problem. The approximation heuristic solves the multi-ship problem as multiple single-ship problems. If a cargo is shared by more than one ship, it is assigned to the ship where it can generate the most profit. The scheduling problems for the remaining ships are resolved with the common cargo. This process is repeated till no conflicting assignments remain.

Cóccola et al. (2015) and Cóccola and Méndez (2015) also work on the multi-ship problem introduced by Jetlund and Karimi (2004). However, Jetlund and Karimi (2004), Cóccola et al. (2015) and Cóccola and Méndez (2015) do not consider the compartment-related decisions that have been considered by us. The compartment-related decisions include the compartment capacities, the ship stability criteria, the cargo-cargo, and cargo-compartment incompatibility norms. Cóccola et al. (2015) propose a set-partitioning master problem and a MILP sub-problem. Subsequently, the column generation framework and a set of custom branching rules are employed by Cóccola et al. (2015), which reduces the total solution time. Including the compartment-related decisions in the model increases the complexity of the MILP sub-problem proposed by Cóccola et al. (2015), rendering the column generation framework unusable for the s-PDP-TWTAC. Cóccola and Méndez (2015) further improve the total solution time by introducing an iterative algorithm, which solves the single-ship MILP repeatedly. During each iteration, a subset of integer variables is fixed, and an improved feasible solution is generated.

Korsvik et al. (2011), Hennig et al. (2015) and Homsi et al. (2020) solve a multi-ship extension of the PDPTW problem with split loads. Korsvik et al. (2011) present a large neighbourhood search heuristic to generate feasible solutions for the problem. Their heuristic primarily works on the principle of destroy-and-repair. It destroys a feasible solution by excluding certain cargoes at each iteration and re-constructs a different solution by employing a constructive insertion heuristic. The *insertNewCargo* core sub-routine of our heuristics described in Section 5.1.1, is motivated by Korsvik et al. (2011). However, compartment-related decisions in the s-PDP-TWTAC require a different kind of implementation, unlike the type of implementation used by Korsvik et al. (2011). Hennig et al. (2015) study the effectiveness of two different path-flow formulations within a column generation framework. Homsi et al. (2020) work on a similar problem with larger instances and an added complexity of discharge time windows for the cargoes. Homsi et al. (2020) introduce a novel branch-and-price algorithm along with a hybrid genetic meta-heuristic, which solves larger instances in significantly less time. However, none of these researchers have considered compartment-related constraints.

An extension of the multi-ship problem with split loads is introduced by Giavarina dos Santos et al. (2020). They include compartment capacities in their problem definition and formulate a matheuristic. Their matheuristic consists of three phases: the relaxation algorithm phase, the modified relax-and-fix algorithm phase, and the improvement phase. Motivated by the relax-and-fix algorithm that is implemented by Giavarina dos Santos et al. (2020), we introduce Heuristic *H2* in Section 5.3.2. Although Giavarina dos Santos et al. (2020) incorporate compartment capacities in their problem definition, they do not include the rest of the compartment-related decisions. Additionally, unlike our model, the Giavarina dos Santos et al. (2020) works with a fixed set of cargoes with soft time windows.

]

Table 3.1: Problem characteristics tackled by researchers working on problems similar to the s-PDP-TWTAC

Decisions optimised by researchers	**Researchers tackling problems similar to the s-PDP-TWTAC**					
	Jetlund and Karimi (2004), Lin and Liu (2011), Cóccola et al. (2015), Cóccola and Méndez (2015)	Hennig et al. (2015) Homsi et al. (2020)	Hvattum et al. (2009), Vilhelmsen et al. (2016)	Wang et al. (2018)	Fagerholt and Christiansen (2000a)	Neo et al. (2006), **s-PDP-TWTAC (Our work)**
Single ship			✓	✓	✓	✓
Multiple ships	✓	✓				
Port set	✓	✓			✓	✓
Port sequence	✓	✓		✓	✓	✓
Cargo pick-ups and drop-offs	✓	✓			✓	✓
Cargo pick-up and drop off sequence	✓	✓		✓	✓	✓
Pick-up time window	✓	✓		✓	✓	✓
Drop off time window		✓			✓	
Compartment capacity			✓	✓	✓	✓
Ship stability			✓	✓		✓
Ship's draft				✓		✓
Cargo split between ships		✓				
Cargo-cargo compatibility			✓	✓		✓
Cargo-compartment compatibility			✓	✓		✓
Cargo swapping						✓

Since the s-PDP-TWTAC problem incorporates all compartment-related decisions, it is structurally different from the previously mentioned problems tackled by the researchers. Neo et al. (2006) and Ladage et al. (2021) are the only researchers who solve the s-PDP-TWTAC problem. Neo et al. (2006) were the first to incorporate compartment-related decisions into a chemical tanker scheduling problem. However, they could not perform an extensive computational study due to hardware limitations. Table 3.1 highlights the literature's contributions and differentiates the s-PDP-TWTAC from the rest.

Individually, the problem of generating a cargo-compartment assignment plan based on compartment related decisions is complex. The literature on solving cargo-assignment problems in the chemical tanker industry is very scarce. Hvattum et al. (2009) and Vilhelmsen et al. (2016) provide some insight into the problem complexity, and refer to the problem of generating cargo-compartment allocations as the *Tank Allocation Problem* (TAP). They introduce multiple TAP variants and prove that the problem is NP-complete. Hvattum et al. (2009) present a MILP formulation for the simplified version of the tank allocation problem. However, the solution time for the MILP is quite large. They also propose an innovative constraint programming approach. However, this approach cannot find a feasible solution within their termination criteria. Vilhelmsen et al. (2016) presents a hybrid method for solving the allocation problem. Their constructive heuristic solves the test instances quite fast. However, both, Hvattum et al. (2009) and Vilhelmsen et al. (2016) solve single period cargo-assignment problems. Ostermeier et al. (2021) present a more general study on multi-compartment vehicle routing problems.

The assignment plans are generated individually during every sailing leg. One major drawback of generating a cargo-compartment allocation plan at every sailing leg separately is that it might unnecessarily re-arrange chemicals between compartments to make room for new cargo that might be picked up. Even worse, re-arranging chemicals might not be allowed in many practical applications as they can prove quite time-consuming and expensive. However, if cargo re-arrangements are not allowed, cargo-compartment allocations during previous sailing legs might make future profitable cargo un-serviceable due to the unavailability of compartments. Re-arrangements or not, a cargo assignment problem that considers possible future cargo-compartment placements is needed from a practical point of view. We study an extension of this problem called the *multi-period cargo-assignment problem (mp-CAP)*. As per our knowledge, this problem has not been studied previously in the literature.

Having explored the literature from an operations research perspective, we present the mathematical background necessary to understand the DW reformulation and the column generation framework used to solve the mp-CAP.

3.3 Dantzig-Wolfe reformulation and column generation theory

The concept of Dantzig-Wolfe decomposition and reformulation was first introduced by Dantzig and Wolfe (1960). Dantzig-Wolfe reformulation and decomposition are often employed when the set of constraints for a given MIP formulation can be separated into *easy* and *complicated* constraints. Ideally, the set of *easy* constraints would define a feasible region over which one can quickly evaluate the solution to a given objective function. From a practical perspective, the *niceness* of the feasible region defined by the *easy* constraints is due to some unique structure or the availability of good heuristics to evaluate a given objective over it.

$$\textbf{MILP Formulation:} \quad \begin{aligned} &\text{Min } c^T x \\ &A_1 x \geq b_1, \\ &A_2 x \geq b_2, \\ &x_j \in \mathbb{Z} \ \ \forall j = 1, \ldots, p, \\ &x_j \geq 0 \quad \forall j = 1, \ldots, n. \end{aligned} \tag{3.1}$$

Consider the MILP (3.1), where $p \leq n$, and all the data is assumed to be rational. Assume that $A_1 x \geq b_1$ are *complicating* constraints, while $A_2 x \geq b_2$ are *nice* constraints. Dantzig-Wolfe's (DW) reformulation technique uses Minkowski's Double Description Theorem, also known as the Minkowski-Weyl Theorem, to reformulate the feasible region represented by the complicating constraints. Formally, the Minkowski-Weyl theorem is stated as follows.

Theorem 3.1 ***(Minkowski-Weyl Theorem)****: Any polytope $P \in \mathbb{R}^n$ can be defined by either*

- *$P = \{x \in \mathbb{R}^n : Ax \geq b$ for some $m \times n$, matrix A and vector $b \in \mathbb{R}^m\}$, or*
- *For some set of vertices $f_1, \ldots, f_M \in \mathbb{R}^n$ and extreme rays $r_1, \ldots, r_N \in \mathbb{R}^n$ (where potentially M=0 or N=0): $P = \{\sum_{i=1}^{M} \lambda_i f_i + \sum_{j=1}^{N} \alpha_j r_j : \lambda \in \mathbb{R}^M, \alpha \in \mathbb{R}^N, \sum_{i=1}^{M} \lambda_i = 1, \lambda_i, \alpha_j \geq 0 \ \forall i, j\}$*

Theorem 3.1 states that the solution to any linear program can be represented as a convex combination of the vertices (corner points) and extreme rays. The authors assume that the feasible region is always bounded for all further discussions, which is the case in most practical integer formulations. This means there are no extreme rays (N=0). The feasible region of the mp-CAP is also bounded.

DW Relaxation:

$$
\begin{aligned}
\text{Min } \sum_{v \in V} c f^v \lambda^v, \\
\sum_{v \in V} A_1 f^v \lambda^v \geq b_1, \\
\sum_{v \in V} \lambda^v = 1, \\
\lambda \in \mathbb{R}^{V+}.
\end{aligned}
\tag{3.2}
$$

$$
\sum_{v \in V} f_j^v \lambda^v \in \mathbb{Z}^+ \quad \forall j = 1, \ldots, p. \tag{3.3}
$$

Let V be a finite set of corner points of the feasible region defined by $Q := \{x \in \mathbb{R}^n_+ : A_2 x \geq b_2 \; \& \; x \in \mathbb{Z}\}$. Let $f^v \in V$ represent a corner point. Assuming the absence of extreme rays, formulation (3.2) represents the LP relaxation of the DW reformulation of the MILP (3.1). Equation (3.3) is used enforce the integrality condition.

This transformation is simply obtained by substituting $x_j = \sum_{v \in V} f_j^v \lambda^v$, where $\sum_{v \in V} \lambda^v = 1$ and f^v are vertices of the feasible region Q. Readers must observe that the number of constraints has reduced. In addition to reducing the number of constraints under favourable conditions, the DW reformulation also provides tighter bounds than the LP relaxation. However, the number of decision variables increased exponentially. A DW reformulation has at least as many decision variables as the number of corner points (and extreme rays, if present).

Consider the LP (3.2). Based on the knowledge of the simplex algorithm, readers can observe that the number of columns in the simplex table is equivalent to the number of corner points in set V. Generating all the columns beforehand is computationally quite expensive for practical problems. Nevertheless, readers may recall the following optimality condition defined by the simplex algorithm.

Theorem 3.2 *Consider a basic feasible solution x associated with a basis matrix B, and let $\bar{rc}$ be the vector of reduced costs. Then, the following statements are true.*

- *If $\bar{rc} \geq 0$, then x is optimal.*
- *If x is optimal, then $\bar{rc} \geq 0$.*

Additionally, also recall that if the above optimality conditions are met, then the optimal solution $x^* = B^{-1}b$, where b is the R.H.S of the constraints defining the feasible region. The readers should realize that only a small subset of columns with non-negative reduced costs are required to prove optimality and calculate the optimal solution.

Column generation (also often referred to as delayed column generation) uses this fact to generate only those columns that improve the solution iteratively. Column generations algorithm was first discussed by Gilmore and Gomory (1961) in the context of the cutting stock problem. Theoretically, this method terminates when there are no more columns that can enter the basis B, meaning no additional column with a negative reduced cost exists.

Restricted master problem (RMP):

$$\text{Min} \sum_{v \in V'} c f^v \lambda^v, \tag{3.4}$$

$$\sum_{v \in V'} A_1 f^v \lambda^v \geq b_1, \tag{3.5}$$

$$\sum_{v \in V'} \lambda^v = 1, \tag{3.6}$$

$$\lambda \in \mathbb{R}^{V'+}. \tag{3.7}$$

In the context of the DW reformulation (3.2), the relaxation (3.4)-(3.7), where $V' \subseteq V$ such that the problem is feasible. Formulation (3.4)-(3.7) is referred to as the restricted master problem (RMP). If the RMP is unbounded then the DW relaxation (3.2) is also unbounded.

Otherwise, for a given subset of column V', let $\bar{\lambda}$ be the optimal solution for the RMP (3.4). Moreover, let $\bar{\mu_1}$ and $\bar{\mu_2}$ be the optimal dual solution corresponding to the RMP Constraints (3.5) and (3.6), respectively. Observe that $\bar{\mu_1}$ is a vector, while $\bar{\mu_2}$ is a scalar. Populating V' with any new columns with negative reduced costs can only improve the solution. In other words, if the RMP has an optimal solution, then it is an upper bound on the full DW reformulation. New columns with negative reduced cost can be generated by solving the Formulation (3.8).

Sub problem (SP):

$$Min_{f \in V} \quad f^T c - f^T {A_1}^T \bar{\mu_1} - \bar{\mu_2} \tag{3.8}$$

$$Min_{x \in Q} \quad (c - {A_1}^T \bar{\mu_1})^T x - \bar{\mu_2} \tag{3.9}$$

As f is just a corner point (or a basic solution) of Q, Formulation (3.8) can be re-written in terms of x as Formulation (3.9). Thus, the sub-problem is nothing but our original problem (3.1) without the constraints $A_1x \leq b_1$ and the objective function function equivalent to the reduced cost in terms of the dual values ($\bar{\mu}_1$, $\bar{\mu}_2$). As the authors have assumed the feasible region to be bounded, only two possible results can be obtained by solving the SP (3.9). If the objective is non-negative, then ($\bar{\lambda}$) is the optimal solution to the DW relaxation (3.2), where the values of variables $\lambda_v, v \in V \backslash V'$ are set to zero. Otherwise, one or more column (variable) with negative reduced cost enters the restricted master problem.

It is imperative to understand the bounds generated by the reformulation. If the RMP has an optimal solution, then it is an upper bound on the full DW reformulation. In the absence of degeneracy, this upper bound keeps improving with each iteration of the column generation algorithm. However, how does the optimal solution to the full DW reformulation compare to the MILP formulation (3.1) and its LP relaxation? Let IP^*, LP^* and DW^* be the optimal solutions of the MILP formulation (3.1), its relaxation and the full DW reformulation, respectively. From the theory of integer programming, readers are already aware that for a minimization problem, $IP^* \geq LP^*$. The following theorem gives the relationship between the LP^* and DW^*.

Theorem 3.3 *For minimization problems defined as (3.1), $DW^* \geq LP^*$. Furthermore, $DW^* = LP^*$ if $conv(Q)$ = $\{x \in \mathbb{R}^n : A_2x \geq b_2\}$*

The above theorem effectively means that tighter bounds from the DW reformulation may be achieved if the sub-problems are harder to solve (harder in the sense that at least branching within the branch & bound tree is required to get an integer solution for the sub-problem). However, even this does not assure a better bound. If readers are aware of the Lagrangian relaxation, then they should derive and observe the fact that the Lagrangian dual is the LP dual of the DW relaxation (3.2). However, this proof is outside the scope of this chapter.

To solve the MILP (3.1), the DW relaxation is solved at each branch node and bound tree node. This procedure is called Branch & Price (B&P). B&P can be considered an extension of the Branch & Bound (B&B) algorithm. Let Z_{ip}, Z_{lp}, and Z_{dw} denote the MILP formulation (3.1), its LP relaxation and its Dantzig Wolfe relaxation (3.2), respectively. Recall that $x^*_j \in \mathbb{Z}, j = \{1, \ldots, p\}$ defined for Z_{ip} are integer variables. B&B solves Z_{lp} at each node of the tree while B&P solves Z_{dw} at each tree node.

An implementation of the B&P starts by solving the Z_{dw} at the root node. B&B employs the Simplex method to solve Z_{lp}, while B&P uses column generation to solve Z_{dw}. Theoretically, Z_{dw} is a form of linear relaxation and can be solved using the Simplex method. However, column generation or delayed column generation is very effective at handling formulations with an exponential number of variables, which is typically the case for Z_{dw}. At the root node, Z_{dw} is solved to optimality using column generation, which generates fractional λ's. The corresponding fractional x variable vector can be obtained from the Equation $x_j = \sum_{v \in V} f_j^v \lambda^v$, where f^v are the corner points are defined in formulation (3.2).

If any $x_j^* \in \mathbb{Z}, j = \{1, \ldots, p' | p' \subseteq p\}$ take fractional values in the optimal solution of Z_{dw}, then the B&P algorithm creates $2p'$ branches to create as many additional Dantzig Wolfe relaxations. Every fractional integer variable x_j^* gives rise to two new DW relaxations, which have bounding constraints, $x_j^* \leq \lfloor x_j^* \rfloor$ and $x_j^* \geq \lceil x_j^* \rceil$. In terms of λ, these constraints can be written as $\sum_{v \in V} f_j^v \lambda_v^* \leq \lfloor x_j^* \rfloor$ and $\sum_{v \in V} f_j^v \lambda_v^* \geq \lceil x_j^* \rceil$. For example, if $x_1 = 2.5$ in the optimal solution of the Z_{dw}, then the bounding constraints are $x_1 <= 2$ and $x_1 >= 3$.

The B&P algorithm solves the new DW relaxations and the branches generated from them. All nodes resulting in optimal solutions with variables $x_j^* \in \mathbb{Z}, j = \{1, \ldots, p\}$ having integer values are known as the leaf node. The optimal solution at any leaf node is a feasible solution for Z_{ip}. Each leaf node generates a feasible solution or an upper bound, while each DW relaxation generates a lower bound for the Z_{ip}. Depending on the termination criteria, the B&P algorithm terminates after a certain time or relative gap (%). An outline of the Branch & Price procedure is given below.

1. Solve the DW relaxation at a node to obtain the optimal solution λ^*, and calculate the corresponding optimal solution $x^*(\sum_{v \in V} f^v \lambda_v^*)$ for the LP relaxation of the MILP (3.1).

2. If x^* satisfied the integrality constraint $x_j^* \in \mathbb{Z}, j = 1, \ldots, p$ then a leaf node has been reached.

3. Otherwise create new nodes by branching on every fractional x_j^* variables by introducing the following constraints for every new node j.

$$\sum_{v \in V} f_j^v \lambda_v^* \leq \lfloor x_j^* \rfloor$$
$$\sum_{v \in V} f_j^v \lambda_v^* \geq \lceil x_j^* \rceil \tag{3.10}$$

Practical implementations of B&P have to be significantly improved using problem-specific cuts, heuristics to generate better upper bounds, and better formulations to reduce the computational effort required to solve DW relaxation at every node. Readers should also note that the sub-problem's structure changes every time a bounding constraint is added to the problem. This could destroy any special structure inherently present in the DW reformulation at the root node. These challenges must be addressed carefully while implementing column generation and the B&P algorithm.

Gilmore and Gomory (1961) and Barnhart et al. (1998) have been influential in the study of the B&P algorithm. It is worth noting that the B&P algorithm presented above would in most practical scenarios converge extremely slowly. As such, over the years lot of work has been put into developing various tricks to improve its practical computational performance. For example, most Branch & Price implementations avoid the branching scheme mentioned above as it typically destroys the structure of the pricing problem. Hence, it is common to use a custom, problem-specific branching scheme. Yildiz et al. (2022) discuss one such branching strategy which significantly reduces the number of nodes explored for their B&P implementation.

Some other customisations include stabilisation methods, custom cutting planes, primal heuristics, pricing problem heuristics, generating multiple columns at each iteration during the column generation procedure, supplying good quality initial columns/initial solutions using a quick heuristic, and using parallel or distributed computing. Vaclavik et al. (2018) present an implementation of Branch & Price accelerated using machine learning. Readers may also refer to Vanderbeck (2000), Desrosiers and Lübbecke (2005), Lubbecke and Desrosiers (2005) and Ralphs and Galati (2010) for more information on DW reformulation, CG and B&P algorithm implementations. Desaulniers et al. (2006) present application of B&P algorithm to different practical problems, while Cóccola and Méndez (2015), Menezes et al. (2017) and Hellsten et al. (2022) present B&P implementations on maritime problems.

To summarize the general procedure of the DW reformulation as applied to MILPs, the readers must start by identifying a problem for which the constraint set can be separated into *easy* and *complicated* constraints. The problem can be separated into a restricted master problem and one or many sub-problem. Subsequently, a column generation approach can be employed within a Branch & Price framework to obtain the integer solution for the MILP. In Chapter 6, we discuss some more customisations generally used to improve B&P implementations. Having understood the basics of DW reformulation and column generation, the authors encourage the readers to ponder upon the following questions.

- Can a decomposed problem have multiple sub-problems?
- Can a formulation be decomposed in multiple ways? Moreover, if multiple decompositions do exists, will they be equally efficient?
- Other than the decomposition into a master problem and a sub-problem, are there any additional special properties or structures worth exploiting?
- Is Branch & Bound algorithm always required to generate a MILP solution from a given set of master problem columns?

3.3.1 Notes for practical implementation

The previous section, revolved around understanding the theory of Dantzig Wolfe decomposition and column generation. However, practical implementation of these methods would require the reader to consider some additional parameters. Some of these parameters are discussed below.

3.3.1.1 Termination criteria

By now, the readers are aware that theoretically, the column generation technique terminates if there are no new columns that can enter the master problem. However, even if more columns are added, the improvement in objective function value can be minimal. As a result, generating all the columns can be pretty time-consuming. Instead, some widely used practical termination criteria are *total number of iterations*, *time limit*, and *gap (%)* between the upper and lower bounds. The total number of iterations states that the algorithm must terminate as soon as a certain number of iterations have elapsed. The time limit criteria terminates the algorithm after a fixed amount of time. Similarly, the gap (%) criteria terminates the algorithm when a particular gap (%) between the bounds is reached.

3.3.1.2 Generation of a set of initial columns

The column generation algorithm requires at least one initial column to begin its execution. Initial columns can be introduced by generating a trivial solution to the sub-problems and constructing the corresponding column. However, generating columns from trivial solutions can lead to extremely slow convergence. It is helpful to have some heuristics that can guide the initial process of column generation. For example, in the proposed decomposition of the mp-CAP, duals from the LP relaxation of the mp-CAP are used during initial iterations to guide the generation of columns.

3.3.1.3 Adding multiple columns per iteration of column generation

Readers must observe that theory dictates the selection of only one entering column per iteration of the column generation algorithm by solving the sub-problems to optimality. However, many practical implementations of column generation apply a fast heuristic that generates multiple alternative columns from the original column. These alternative columns are still feasible for the sub-problems and adhere to the reduced cost criteria. Often for frameworks where MILP sub-problems are solved, traversing through the Branch & Bound tree leads to multiple integer feasible solutions. All the feasible integer solutions that satisfy the reduced cost criteria can be used to generate multiple columns during a single iteration. Adding multiple columns during every iteration of the column generation algorithm can significantly expedite its convergence.

3.3.1.4 Embedding heuristics within the framework

The column generation algorithm helps us split the Dantzig Wolfe reformulation into a restricted master problem (RMP) and sub-problem(s) (SP). In a simple implementation, the RMP and SP are solved using a solver, which selects some linear programming or integer programming algorithm to solve the problem. However, some problems can have additional unique structures within the RMP, SP, or both. These structures might enable the readers to solve the RMP or SP using polynomial-time heuristics or simple arithmetic. For example, authors could leverage the shortest path structure in the master problem for the mp-CAP. This structure eliminated the need to solve the Branch & Bound tree in order to generate the MILP solution from a given set of master problem columns.

3.3.1.5 Symmetry breaking

The DW reformulation and the block diagonal structure of the mp-CAP have been discussed earlier. However, imagine that the feasible regions defined by all the sub-problems were identical. For the sub-problems (6.16)-(6.25), this would be equivalent to saying that all the tanker compartments are identical, and for every sailing leg, the same set of cargoes has to be assigned to the compartments. In essence, the corner points $f^v_{k-1} = f^v_k = f^v_{k+1} \forall k \in K$, during each iteration. Constraint (6.12) disappears because the first and second terms of its L.H.S cancel out each other. Thus, the problem essentially becomes that of placing the cargoes in the least number of compartments during the first leg and maintaining the same placement till the end of the voyage. Readers should observe that the symmetry disappears in the master problem, which significantly simplifies the problem.

3.3.1.6 Parallel computing

The readers may recall, the proposed reformulation helps decompose the problem into one master problem and multiple sub-problems. As the sub-problems are disjoint, they can be solved in parallel by using multiple CPUs. This would accelerate the time for convergence. For large-scale problems, such frameworks can also be designed to leverage distributed computing, where both memory and CPU speeds can be distributed across a network of computers. A recent study by Basso and Ceselli (2022) discusses the application of distributed computing with column generation.

We will answer these questions stated earlier, and present the application of DW-CG framework to the mp-CAP in Chapter 6. In this next chapter, we discuss the single ship pick-up delivery problem with time-windows, tank allocations, and changeovers (s-PDP-TWTAC) and present some empirical experiments related to it.

Chapter 4

The single ship pick-up delivery problem with time-windows, tank allocations and changeovers (s-PDP-TWTAC)

4.1 Problem description

The single ship pick-up and delivery problem with pick-up time windows, tank allocations and changeovers (s-PDP-TWTAC) models the scheduling of a chemical tanker on a network of ports. We consider a chemical tanker with a list of onboard chemicals (cargoes), which need to be transported to their destinations, respectively. At the same time, unassigned cargoes can be picked up by the chemical tanker. Our goal is to generate a schedule for the chemical tanker, which includes cargoes to be picked up, the ports to be visited, the sequence in which these ports are visited, the arrival times at each port, and a feasible cargo-compartment allocation. Our objective tries to maximise the difference between the revenue and four different costs.

[0]Chapter 4 is heavily derived from Ladage et al. (2021)

These costs include the time charter cost, the fuel cost incurred while travelling between ports, the fixed cost associated to a port call, and the cargo swapping cost incurred for every intra-compartment cargo swap.

To make the problem tractable, the intra port activities have been simplified. During each port visit a constant administrative time incorporates activities such as waiting time for berth allocation, repairs, re-fuelling, security clearances, immigration procedures, and delays related to custom inspections. We assume half the administrative time is spent on security checks, following which the cargo-compartment assignments are decided. We term this point as the cargo-compartment assignment point. At this point, we decide the pick-up cargoes and generate a cargo-compartment plan. The cargo-compartment assignment point presented should lie within the pick-up time window of each of the cargoes. The pick-up time windows are specified in units of days (fractional days are allowed). Any cargo that is picked up has to be delivered within the time horizon. All the temporal inter-port and intra-port activities are performed sequentially, one after the other.

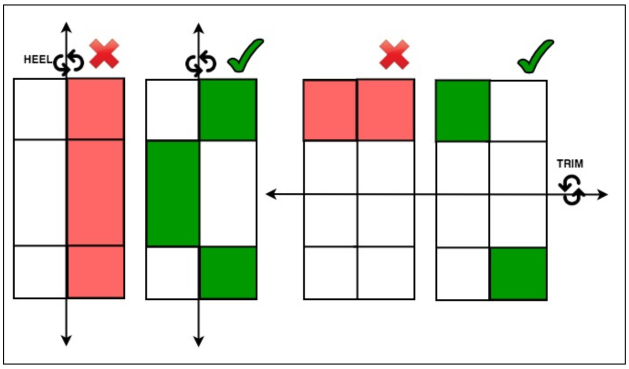

Figure 4.1: Ship balancing requirements: This figure illustrates the trim and heel movements of the ship, along with the possible cargo arrangments affecting them.

Every chemical tanker has a maximum draft limit, which limits its cargo carrying capacity. Moreover, a chemical tanker has multiple compartments or cargo holds. Each compartment can store at most one cargo, but a cargo can be distributed into multiple compartments. A loaded cargo can be moved to a different compartment at an additional cost. This movement provides more flexibility in picking up new cargoes. The cargo-compartment allocation plan must also take into account the ship balancing requirements, and the compartment capacities. As shown in Figure 4.1, the cargoes have to be distributed within permissible limits of trim and heel.

We also consider the cargo-cargo compatibility criteria, which restricts the storage of certain chemicals in neighbouring compartments. Figure 4.2 represents cargo-cargo compatibility for a set of cargoes through an illustrative graph. An edge in the graph means that the two cargoes can be stored in neighbouring compartments. Similarly, the cargo-compartment compatibility criteria restricts the cargo storage to a subset of chemical tanker compartments. The cargo-compartment compatibility can be represented by a bipartite graph, as shown in Figure 4.3.

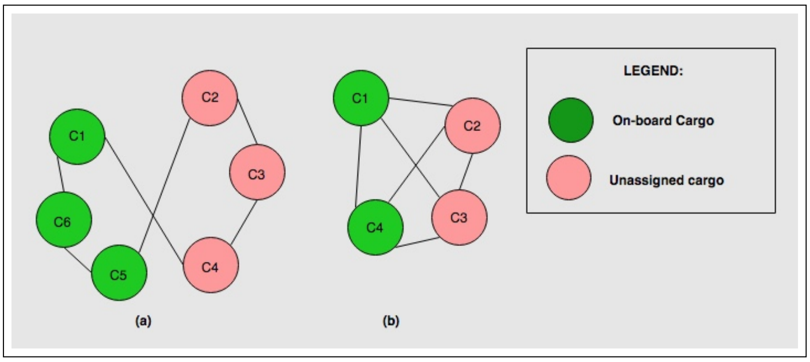

Figure 4.2: Cargo-cargo compatibility graph: Instance (a) shows partial compatibility of cargoes while instance (b) shows complete compatibility of cargoes with each other.

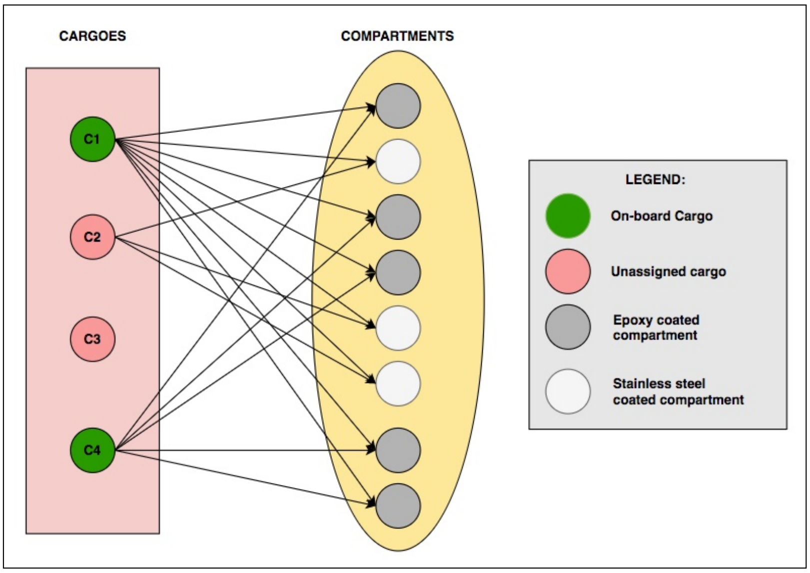

Figure 4.3: Cargo-compartment compatibility graph: Direct connections between cargoes and compartment show compatibility while no connection reports incompatibility.

We believe that the compartment-related decisions are essential while delivering chemicals using chemical tankers. A chemical tanker can have different compartment structures, which dictate the compartment-related decisions. If the compartment-related decisions are ignored, one cannot state with certainty that the schedule will be feasible for a given chemical tanker.

We make the following assumptions in our model. These assumptions have been borrowed from Jetlund and Karimi (2004), Neo et al. (2006), Cóccola et al. (2015) and Cóccola and Méndez (2015). They are listed below.

- We make a simplifying assumption to fix the maximum number of port calls (sailing legs). However, even in the industry, the scheduler is required to generate a schedule for a fixed number of port calls. As such, this is a reasonable assumption.
- The chemical tanker may or may not pick-up all the unassigned cargoes at the port it visits.
- Cargoes cannot be delivered partially.
- The time for loading/unloading cargoes varies only with the total weight of the cargo.
- All port arrival and departure administrative activities are assumed to take 0.25 days.
- Three primary time-consuming activities, namely, travelling between ports, cargo loading, and cargo unloading, are considered in our model. No two of these activities can be performed simultaneously.
- Each compartment can carry only one cargo at any given time. The cargo can be split into multiple compartments of the chemical tanker.
- Changeovers (rearranging) of loaded cargoes within the compartments of the ship are allowed. A fixed penalty cost (changeover cost) is incurred every time an existing cargo is replaced by a different cargo within a compartment. We assume that cargoes can be offloaded from the chemical tankers during re-assignment of these cargoes, and then loaded again.
- Due to safety factors and storage norms, cargoes can only be placed in specific compartments (cargo-compartment compatibility constraints).
- Safety norms also impose certain restrictions on the placement of cargoes in neighbouring compartments, which we model as cargo-cargo compatibility constraints.
- The average speed (nm/hour) of the chemical tanker is assumed constant.

- Fuel consumption is assumed to vary linearly with the distance travelled independently of the load on the ship.
- Onboard cargoes can be re-assigned to compartments only at ports and that this activity can be carried out instantaneously.
- A port can only be visited once in the planning period.

Figure 4.4: This figure illustrates a small instance for the s-PDP-TWTAC along with a possible schedule for that instance.

Further, with the help of an small example, we explain the problem and the importance of considering the compartment-related decisions. Figure 4.4 helps us illustrate our example. At the start of the planning horizon, suppose that the chemical tanker is at Port 0 (Shanghai) and has a list of 7 ports that may be visited. We consider three different chemical tankers as shown in the bottom right corner of Figure 4.4. Cargo *C5* is on board the chemical tankers at time zero. Additionally, cargoes, *C1* to *C4*, are the potential (unassigned) cargoes that are available at the ports. Attributes related to these cargoes, such as origin-destination ports, total volume, and pick-up time windows are displayed in Figure 4.4. Figure 4.4 also depicts the simplified port activities that have been considered in our problem.

Figure 4.5 presents a more detailed representation of the various temporal activities and various port activities defined for the s-PDP-TWTAC. As shown in Figure 4.5, a sailing leg begins when the ship departs from the port visited during sailing leg $k-1$. The ship then travels to the next port which is referred to as the travelling time during leg k. As soon as the ship arrives at the port, a fixed administrative time (T_1^A) is spent for activities like immigration, customs clearance, and generation of a cargo-compartment assignment plan. We assume that a cargo-compartment assignment plan is available as soon as T_1^A is elapsed. Following this, cargoes are discharged and loaded according to the cargo-compartment allocation plan. Following this activity, some administrative time (T_2^A) is reserved for activities like wait time, re-fuelling or any final inspections before port departure. Finally, the sailing leg ends as soon as the ship departs for the next port.

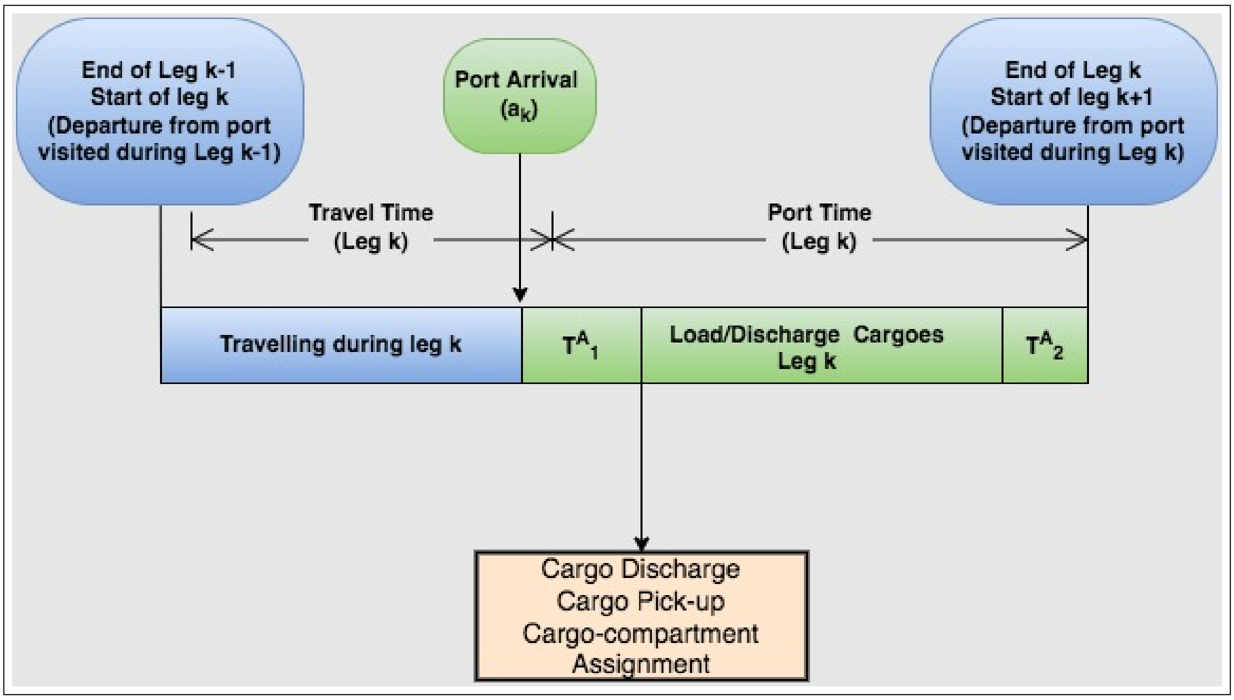

Figure 4.5: This figure illustrates the temporal activities and different port activities included into the s-PDP-TWTAC

Figure 4.4 shows a sample schedule for chemical tanker 1. The cargo *C1* is picked up by the chemical tanker. At Shanghai, cargo *C5* is stored in compartments 2 and 4. At Port 1 (Hong Kong), the chemical tanker picks up cargo *C1*, which is stored in the third compartment. Finally, the voyage ends at Singapore where it delivers both the cargoes. However, due to the compartment structure and related constraints, the same schedule might become infeasible for chemical tanker 2 and 3.

Consider the second chemical tanker, which has two compartments. Each compartment has a storage capacity of 750 tonnes. If we ignore the chemical tanker stability criteria, the entire cargo might be assigned to either compartment 2 or compartment 3. This would jeopardize the safety of the chemical tanker. Consequently, the cargo *C5* is equally distributed in both the compartments of the chemical tanker 2 as shown in Figure 4.4. Further on, if the cargo swapping is not allowed, the chemical tanker would reach port Hong Kong completely full. As a result, the cargo *C1* cannot be picked up, which makes the previous schedule infeasible for the second chemical tanker.

Let us now consider the structure of the third chemical tanker, in which the central compartment is coated with Epoxy. Observe that if both cargo *C1* and *C5* are incompatible with epoxy coated compartments, then the schedule generated for chemical tanker 1 becomes infeasible for chemical tanker 3. Finally, let us assume that cargo *C1* and *C5* are incompatible with each other. Meaning, both these cargoes cannot be stored in adjacent compartments. Then, the schedule presented in Figure 4.4 becomes infeasible for chemical tanker 1. It is easy to observe that neglecting any of the above decisions might generate infeasible schedules for a chemical tanker.

We summarise our problem as follows. Given a set of ports and a set of cargoes, we try to identify the optimal schedule of the chemical tanker. An optimal schedule is one that would transport the most profitable cargoes while adhering to the various problem constraints. Our objective maximises the revenue earned by transporting unassigned (potential) cargoes and minimises the port cost, fuel cost, time chartering cost, and the changeover (cargo swapping) cost. The entire set of feasible cargoes need not be delivered. However, all the cargoes loaded on the ship are required to be delivered before the end of the planning horizon.

The primary decisions that affect the complexity of our problem are the finding of the set of ports to visit, the determining of the sequence in which these ports should be visited, the identification of the set of cargoes to transport, the assigning of the cargoes to compartments and the swapping of cargoes between compartments. The proposed problem is reducible to a *Hamiltonian path problem* by fixing all decisions except the routing of the ship. Thus, the problem is NP-hard. Our model can also be seen as a variation of *the Pickup and Delivery Problem with Time Windows (PDPTW)*, [Jetlund and Karimi (2004)], which itself is an extension of the vehicle routing problem. The tramp scheduling problem without compartment-related decisions is structurally very similar to the PDPTW, as defined by Sun et al. (2018). However, unlike the PDPTW, the s-PDP-TWTAC does not require the vehicle to return to its starting location, and all cargoes need not be served.

At every port, if the route of the chemical tanker and the cargoes to be transported are fixed, we are left with the decision of allocating cargoes to compartments. This sub-problem of cargo-compartment allocations is an extension of the generalised segregation storage problem (GSSP), which is also NP-complete [Barbucha (2004)]. In the following section, we mathematically describe the various parameters and decision variables used in our formulation. We also present the mixed integer linear programming formulation used to model the problem explained in this section.

4.2 MILP formulations for the s-PDP-TWTAC

This section begins with a brief description of the revised (REV) formulation for the *the single ship pick-up and delivery problem with pick-up time windows, tank allocations and changeovers (s-PDP-TWTAC)*. Following this, we present the revised MILP for the s-PDP-TWTAC, which is proposed by us. Section 4.2.2 presents the MILP formulation proposed by Jetlund and Karimi (2004). This formulation is extended by Neo et al. (2006) to include the cargo to compartment decision variables and constraints. The extended formulation presented by Neo et al. (2006) will be referred to as the original (OG) MILP formulation. Finally, we conclude this section by stating some key differences between the revised MILP formulation and the original MILP formulation.

4.2.1 Revised MILP formulation

The s-PDP-TWTAC revised (REV) formulation is described as follows. Let K be the set of indices of the sailing legs, and N^P be the set of feasible ports. The set of cargoes (N^G) divided into on-board cargoes (N^O) and unassigned cargoes (N^U). N^O are the cargoes that are on-board the chemical tanker at the beginning of the planning horizon. Unassigned cargoes (N^U) are the cargoes that can be potentially picked up to maximise the profit. The set N^H is the set of chemical tanker compartments.

Any cargo loaded on the ship has to be delivered. An unassigned cargo $j \in N^U$ can only be picked up within a specified time-window [T_j^E,T_j^L]. A cargo $j \in N^G$ is defined by characteristics like revenue obtained (R_j), origin (P_j^L), destination (P_j^D), weight (W_j) and density (ρ_j). A compartment $h \in N^H$ is defined by compartment volume (V_h), and lateral (κ_h) and longitudinal (ι_h) distance from the centre of the ship.

The set N_h^B defines the structure of the chemical tanker by listing the bordering compartments for every $h \in N^H$. The sets N_j^I and N_h^X helps us define the cargo-cargo incompatibility and the cargo-compartment incompatibility, respectively. The set N_j^I lists all the cargoes that cannot be stored beside the cargo $j \in N^G$. The set N_h^X includes cargoes that cannot be stored in compartment

$h \in N^H$.

Given the starting port (P^I) and the set N^P, our model tries to find an optimal by maximising the difference in the revenue (R_j) and the four different costs; namely, the port cost (C^P_p), the fuel cost ($C^F_{pp'}$), the time chartered cost (C^T) and the changeover cost (C^S). The route of the ship is defined using decision variables like port arrival time (t_k) and routing variable ($z_{kpp'}$). The routing variable equals one if and only if the chemical tanker travels between ports $p, p' \in N^P$ at the end of sailing leg $k \in K$. If the number of profitable port calls are less than the maximum number of sailing legs (|K|), then the chemical tanker is forced to enter a dummy port. Once the chemical tanker enters the dummy port, it stays there till the end of its voyage. The sequence in which the cargoes are serviced are modelled using variables l_{kj} and u_{kj}. The variables l_{kj} and u_{kj} record the sailing leg $k \in K$ at the end of which a cargo is picked up and dropped off, respectively.

The decision variable c_{kjh} equals 1 if cargo $j \in N^G$ is stored in compartment $h \in N^H$ at the end of sailing leg $k \in K$. Moreover, if a cargo $j \in N^G$ is stored in compartment $h \in N^H$ then the variable w_{kjh} gives the cargo weight stored in the compartment. Further, we keep track of the total changeovers by defining variables b_{kjh} and r_{kjh}. We formally define all the sets, decision variables and parameters in Section 4.2.

Sets:

$K =$ Set of indices of sailing legs, $\{0,...,|K|\}$,

$N^P =$ Set of ports,

$N^G =$ Set of all cargoes/goods. Includes cargo 0, a dummy cargo for modelling,

$N^O =$ Set of cargoes already on-board the chemical tanker at time zero, $N^O \subset N^G$,

$N^U =$ Set of potential cargoes that can be picked up, $N^U \subset N^G$,

$N^I_j =$ Set of cargoes incompatible with cargo $j \in N^G, N^I_j \subset N^G$,

$N^H =$ Set of compartments (cargo holds) in the ship,

$N^B_h =$ Set of neighbouring/bordering compartments for compartment $h \in N^H, N^B_h \subset N^H$,

$N^X_h =$ Set of cargoes that cannot be stored in compartment $h \in N^H, N^X_h \subset N^G$.

Indices:

$p, p', p'' =$ Index for port,

k, k' = Index for sailing leg (Index 0 indicates that the chemical tanker is at its starting port),

j = Index for cargo ($j = 0$ signifies dummy cargo with no weight and no incompatibilities),

h, h' = Index for compartment (cargo hold).

Revised decision variables:

t_k = Port arrival time of the chemical tanker at the destination of leg k$\in K$ (Continuous),

$z_{kpp'}$ = 1 if chemical tanker at the end of leg k$\in$K$\setminus\{0\}$ departed from port p$\in N^P$ and arrived at p$'\in N^P$ (Binary),

l_{kj} = 1 if the chemical tanker at the end of leg k$\in K$ loads cargo j$\in N^U$ (Binary),

u_{kj} = 1 if the chemical tanker at the end of leg k$\in K$ unloads cargo j$\in N^U$ (Binary),

c_{kjh} = 1 if the chemical tanker at the end of leg k$\in K$ carries cargo j$\in N^G$ in compartment h$\in N^H$ (Binary),

w_{kjh} = Weight of cargo j$\in N^G$ assigned to compartment h$\in N^H$ of chemical tanker at end of leg k$\in K$ (Continuous),

b_{kjh} = 1 if the chemical tanker at the end of leg k$\in$K$\setminus\{0\}$ replaces any cargo (other than itself) with cargo j$\in N^G\setminus\{0\}$ in compartment h$\in N^H$ (Binary),

r_{kjh} = 1 if chemical tanker at end of leg k$\in$K$\setminus\{0\}$ removes cargo j$\in N^G\setminus\{0\}$ in compartment h$\in N^H$ (Binary).

Parameters:

P^I = Starting port of the ship, $P^I \in N^P$,

P_j^L = Loading port for cargo $j \in N^U$, $P_j^L \in N^P$,

P_j^D = Discharge port for cargo $j \in N^G$, $P_j^D \in N^P$,

$|N^P|$ = Dummy Port, $|N^P| \in N^P$,

R_j = Revenue that can be obtained if cargo j$\in N^G$ is transported by the chemical tanker,

$C_{pp'}^F$ = Cost for travelling between ports p$\in N^P$ and p$'\in N^P$,

C_p^P = Port cost incurred on visiting port p$\in N^P$,

C^T = Cost of time charter of the chemical tanker per day,

$C^S =$ Cost per changeover/swap including cleaning, labour, etc related to swapping cargoes within compartments,

$W_j =$ Weight of the cargo j $\in N^G$,

$\rho_j =$ Density of the cargo j $\in N^G$,

$V_h =$ Volume of compartment h$\in N^H$,

$T_j^E =$ Earliest pick-up time for cargo j$\in N^U$,

$T_j^L =$ Latest pick-up time for cargo j$\in N^U$,

$T_j^P =$ Time required to pick-up cargo j$\in N^U$,

$T_j^D =$ Time required to discharge cargo j$\in N^G$,

$T_{pp'}^T =$ Travel time between port p$\in N^P$ and $p' \in N^P$,

$T_1^A =$ Waiting time for berth allocation, security clearances, immigration procedure,

$T_2^A =$ Time delays incurred due to repairs, bunkering, and customs inspections,

$T^A =$ Total administrative time, $T^A = T_1^A + T_2^A$,

$\kappa_h =$ Lateral distance from compartment h$\in N^H$ to the centre of the chemical tanker,

$\iota_h =$ Longitudinal distance from compartment h$\in N^H$ to the centre of the chemical tanker,

$\alpha =$ Maximum absolute permissible trim causing moment of the chemical tanker,

$\beta =$ Maximum absolute permissible heel causing moment of the chemical tanker,

$DC =$ Draft constant. The total allowable draft (in tonnes) for the chemical tanker (tonnes),

$M =$ A suitably large number for modelling binary decisions.

The objective function of our formulation is as follows:

$$\begin{aligned}\textbf{Maximise} \quad & \sum_{j\in N^U}\left(R_jW_j\sum_{k\in K\setminus\{|K|\}} l_{kj}\right) - \sum_{p\in N^P}\sum_{p'\in N^P}\left(C^F_{pp'}\sum_{k\in K\setminus\{0\}} z_{kpp'}\right) \\ & -\left(C^T(t_{|K|} + T^A(1-\sum_{p\in N^P} z_{|K|p|N^P|}) + \sum_{p\in N^P}\sum_{j\in N^O}(T_j^D z_{|K|pP_j^D}) + \sum_{j\in N^U}(T_j^D u_{|K|j}))\right) \\ & -\sum_{p'\in N^P}\left(C^P_{p'}\sum_{k\in K\setminus\{0\}}\sum_{p\in N^P} z_{kpp'}\right) - \left(C^S\sum_{k\in K\setminus\{0\}}\sum_{j\in N^G\setminus\{0\}}\sum_{h\in N^H} b_{kjh}\right) + \sum_{j\in N^O} R_jW_j - C^P_{P^I} \end{aligned} \tag{4.1}$$

The first term calculates the total revenues generated by picking up a subset of unassigned cargoes. The second term calculates the fuel cost, which is a function of the route of the ship.

The next term including C^T, calculates the total cost of chartering the chemical tanker. C^T is affected by all the temporal actions that the chemical tanker performs. Thus, total C^T is calculated by combining the arrival time at the last port ($t_{|K|}$), with the temporal port activities performed at the last port (T^A and the total unloading time of all the cargoes discharged at the last port)). The total time spent at the port is zero if the last port visited is a dummy port. The succeeding term, calculates the total port cost C^P, which is incurred for every port visited by the chemical tanker. Moreover, the changeover cost is calculated by summing up the total number of changeovers (b_{kjh}). Finally, the last two terms of the Equation 4.1 calculate the revenue obtained from the onboard cargoes, and port cost (C^P) related to visiting the immediate destination. As a result, the objective function (4.1) tries to increase the revenue earned by servicing the cargoes. Simultaneously, the objective function tries to reduce the travel cost, time chartered cost, port cost and changeover cost.

$$\sum_{p \in N^P} z_{kpp'} = \sum_{p'' \in N^P} z_{(k+1)p'p''} \qquad \forall k \in K \setminus \{0, |K|\}, p' \in N^P, \tag{4.2}$$

$$\sum_{k \in K \setminus \{0\}} \sum_{p' \in N^P} z_{kpp'} \leq 1 \qquad \forall p \in N^P \setminus \{|N^P|\}, \tag{4.3}$$

$$\sum_{k \in K \setminus \{0\}} \sum_{p \in N^P} z_{kpp'} \leq 1 \qquad \forall p' \in N^P \setminus \{|N^P|\}, \tag{4.4}$$

$$\sum_{k \in K \setminus \{0\}} \sum_{p \in N^P} z_{kp,P_j^D} = 1 \qquad \forall j \in N^O \setminus \{P_j^D = P^I\}, \tag{4.5}$$

Constraints (4.2) to (4.5) define the path of the ship. Constraint (4.2) ensures that the ship must leave every port it visits, except the last one. However, if the chemical tanker enters the dummy port $|N^P|$ it has to stay there for rest of the voyage. We enforce this during pre-processing by fixing all the routing variables ($z_{k|N^P|p'}, p' \in P \setminus \{|N^P|\}$) to zero. These routing variables correspond to all the arcs originating from dummy port to all other ports. Constraints (4.3) and (4.4) together enforce the assumption that a chemical tanker can visit any port at most once. Constraint (4.5) imposes the condition that discharge ports of each on-board cargo must be visited. Next we formulate constraints related to the pick-up and delivery of cargoes.

$$l_{(k-1)j} \leq \sum_{p \in N^P} z_{kP_j^L p} \qquad \forall k \in K \setminus \{0\}, j \in N^U, \tag{4.6}$$

$$u_{kj} \leq \sum_{p \in N^P} z_{kpP_j^D} \qquad \forall k \in K \setminus \{0\}, j \in N^U, \tag{4.7}$$

$$l_{kj} \leq \sum_{k' \in K \setminus \{k' \leq k+1\}} u_{k'j} \qquad \forall k \in K \setminus \{|K|\}, j \in N^U, \tag{4.8}$$

$$\sum_{k\in K} u_{kj} = \sum_{k\in K} l_{kj} \qquad \forall j \in N^U, \tag{4.9}$$

$$u_{kj} \geq -1 + \sum_{k'\in K} l_{k'j} + \sum_{p\in N^P} z_{kpP_j^D} \qquad \forall k \in K\backslash\{0\}, j \in N^U, \tag{4.10}$$

Constraints (4.6) and (4.7) ensure that the unassigned cargoes are picked up and dropped off at their corresponding loading and unloading ports. Constraint (4.8) states that an unassigned cargo can be dropped off only after pick-up. Constraints (4.9) and (4.10) ensure that an unassigned cargo can be discharged at the end of leg k if and only if it was picked up and its discharge point is visited at the end of leg k. Next, we model the constraints that deal with the temporal activities of the ship.

$$t_k \geq \quad (T_j^E - T_1^A) l_{kj} \qquad \forall k \in K\backslash\{|K|\}, j \in N^U, \tag{4.11}$$

$$t_k \leq \quad (T_j^L - T_1^A) l_{kj} + M(1 - l_{kj}) \qquad \forall k \in K\backslash\{|K|\}, j \in N^U, \tag{4.12}$$

$$\begin{aligned} t_{(k+1)} \geq t_k \;+\; & T^A\Big(1 - \sum_{p\in N^P} z_{(k+1)|N^P|p}\Big) + \sum_{j\in N^O} \Big(T_j^D \sum_{p\in N^P} z_{(k+1)P_j^D p}\Big) + \sum_{j\in N^U} (T_j^P l_{kj}) \\ & + \sum_{j\in N^U} (T_j^D u_{kj}) + \sum_{p\in N^P} \sum_{p'\in N^P} (T_{pp'}^T z_{(k+1)pp'}) \qquad \forall k \in K\backslash\{|K|\}, \end{aligned} \tag{4.13}$$

Constraint (4.11) enforces the condition that if a cargo is picked up then the cargo-assignment time (t_k + T_1^A) should be greater than the earliest pick-up time (T_j^E). Similarly, Constraint (4.12) states that if a cargo is picked up then the cargo-assignment time (t_k + T_1^A) should be less than the latest pick-up time (T_j^L). Constraint (4.13) makes sure that the port arrival time (t_{k+1}) during the sailing leg $(k+1)$ is greater than the addition of the port arrival time (t_k) during leg k, the administrative time (T^A) during leg k (if the present port is not the dummy port), all the loading and unloading times for the cargoes picked up and dropped off during leg k, and the travel time time during leg (k+1). The constraints for allocating cargoes to compartments are as follows:

$$\sum_{h\in N^H} c_{0jh} \geq 1 \qquad \forall j \in N^O\backslash\{P_j^D = P^I\}, \tag{4.14}$$

$$\sum_{h\in N^H} c_{0jh} \geq l_{0j} \qquad \forall j \in N^U, \tag{4.15}$$

$$\sum_{h\in N^H} c_{kjh} \geq \sum_{h\in N^H} \frac{c_{(k-1)jh}}{|N^H|} - \sum_{p\in N^P} z_{kpP_j^D} \qquad \forall k \in K\backslash\{0\}, j \in N^O\backslash\{P_j^D = P^I\}, \tag{4.16}$$

$$\sum_{h\in N^H} c_{kjh} \geq \sum_{h\in N^H} \frac{c_{(k-1)jh}}{|N^H|} + l_{kj} - u_{kj} \qquad \forall k \in K\backslash\{0, |K|\}, j \in N^U, \tag{4.17}$$

$$\sum_{h \in N^H} c_{kjh} \leq |N^H| \sum_{k' \in K \setminus \{k' > k\}} l_{k'j} \qquad \forall k \in K, j \in N^U, \quad (4.18)$$

$$\sum_{k' \in K \setminus \{k' < k\}} \sum_{h \in N^H} c_{k'jh} \leq |K||N^H|(1 - \sum_{p \in N^P} z_{kpP_j^D}) \qquad \forall k \in K \setminus \{0\}, j \in N^G \setminus \{0\}, \quad (4.19)$$

Constraints (4.14) and (4.15) enforce the cargo allocations during leg 0. Constraint (4.14) makes sure that all the on-board cargoes are assigned to at least one compartment unless they are delivered at the immediate destination. In the same way, Constraint (4.15) makes sure that if an unassigned cargo is picked up during leg 0 then it is assigned to at least one compartment. The Constraints (4.16) and (4.17) together maintain the continuity of unassigned and on-board cargoes respectively. Constraints (4.16) and (4.17) are trivially satisfied if the cargoes are dropped off during the present leg. However, if the on-board cargoes remains loaded on the ship during the present leg, then the Constraint (4.16) makes sure that the cargo is assigned to at least one compartment. Similarly, Constraint (4.17) makes sure that the unassigned cargoes are assigned to at least one compartment till they are on the ship. Finally, Constraints (4.18) and (4.19) enforce the fact that cargoes cannot be assigned to compartments before they are picked up and after they are dropped off. Some additional cargo assignment constraints are as follows:

$$\sum_{j \in N^G} c_{kjh} = 1 \qquad \forall k \in K, h \in N^H, \quad (4.20)$$

$$c_{(k-1)jh} + b_{kjh} = c_{kjh} + r_{kjh} \qquad forall k \in K \setminus \{0\}, j \in N^G \setminus \{0\}, h \in N^H, \quad (4.21)$$

$$c_{kjh} + \sum_{j' \in N_j^I} c_{kj'h'} \leq 1 \qquad \forall k \in K \setminus \{|K|\}, j \in N^G \setminus \{0\}, h \in N^H, h' \in N_h^B, \quad (4.22)$$

Constraint (4.20) makes sure that either the compartment is empty or it has exactly one cargo in it. Constraint (4.21) help us keep track of changeovers (cargo swapping) within every compartment during consecutive legs. Recall that if a compartment is empty we assume it has cargo 0. Constraint (4.21) and a negative objective function co-efficient makes sure that the variable b_{kjh} equals 1 if and only if a compartment is filled with different cargoes in succeeding sailing legs, or an empty compartment is filled with new cargo. Moreover, the variable r_{kjh} ensures that the variable b_{kjh} takes on value 0, when $c_{(k-1)jh}$ equals 1 and c_{kjh} equals 0. Constraint (4.22) imposes the cargo-cargo compatibility criteria. The following constraints implement the cargo weight per compartment related restrictions.

$$w_{kjh} \leq V_h \rho_j c_{kjh} \qquad \forall k \in K \setminus \{|K|\}, j \in N^G \setminus \{0\}, h \in N^H, \quad (4.23)$$

$$\sum_{h\in N^H} w_{kjh} = W_j \sum_{k'\in K\setminus\{k'>k\}} (l_{k'j} - u_{k'j}) \qquad \forall k \in K\setminus\{|K|\}, j \in N^U, \quad (4.24)$$

$$\sum_{h\in N^H} w_{kjh} = W_j(1 - \sum_{k'\in K\setminus\{0,k'>k+1\}} \sum_{p\in N^P} z_{k'P_j^D p}) \qquad \forall k \in K\setminus\{|K|\}, j \in N^O, \quad (4.25)$$

$$\sum_{h\in N^H} \sum_{j\in N^G\setminus\{0\}} w_{kjh} \leq DC \qquad \forall k \in K, \quad (4.26)$$

$$-\alpha \leq \sum_{h\in N^H} \sum_{j\in N^G\setminus\{0\}} w_{kjh}\iota_h \leq \alpha \qquad \forall k \in K, \quad (4.27)$$

$$-\beta \leq \sum_{h\in N^H} \sum_{j\in N^G\setminus\{0\}} w_{kjh}\kappa_h \leq \beta \qquad \forall k \in K. \quad (4.28)$$

Constraint (4.23) ensures that the weight of the cargo assigned to the compartment can be at most equal to the maximum capacity of the compartment. Constraint (4.24) makes sure that the total weight of the unassigned cargo distributed in various compartments is equal the total weight of that cargo between pick-up and delivery. Constraint (4.25) forces the same condition on the on-board cargoes. Constraint (4.26) makes sure that the total weight allocated to the chemical tanker is less than the draft constant. Constraint (4.27) and (4.28) are ensure that the maximum allowable trim and heel moments are not exceeded.

4.2.2 Original MILP formulation [Jetlund and Karimi (2004), Neo et al. (2006)]

Sets:

K = Set of sailing legs,

N^G = Set of all cargoes,

N^P = Set of ports,

N^O = Set of cargoes already on board the ship s∈S at time zero,

N^U = Set of unassigned cargoes during the planning horizon,

N_h^X = Set of cargoes that cannot be stored in compartment l of ship s∈S,

N_j^I = Set of incompatible cargoes for cargo j∈ N^G.

N^H = Set of compartments of ship s∈S,

N_h^B = Set of neighbouring compartments for compartment h∈ N^H,

Parameters:

$\Delta_{pp'}$ = Distance (nautical miles) between ports p$\in$P and p$'\in$P,

P^I = Immediate destination,

P_j^D = Set of discharge port for cargo j$\in N^G$,

P_j^L = Set of loading port for cargo j$\in N^G$,

R_j^D = Discharge rate of cargo j$\in N^G$,

R_j^L = Loading rate of cargo j$\in N^G$,

R_j^S = Shipping rate or revenue for cargo j$\in N^G$(US \$),

C^F = Cost of fuel per unit distance,

C_p^P = Port cost for ship at port $p \in P$,

C^T = Time charter cost per unit time for the ship,

$C_{kjj'h}^C$ = Fixed changeover cost of changing the cargo from j$\in N^G$ in leg (k-1)$\in K$ to j$'\in N^G$ in compartment $h \in N^H$ of the ship during sailing leg $k \in K$,

T_j^E = Earliest pick-up time for cargo j$\in N^U$,

T_j^L = Latest pick-up time for cargo j$\in N^U$,

T^A = Administrative Time (Time for inspections, customs and surveys for each port visit),

V_j = Volume of cargo j$\in N^G$,

W_j = Weight of cargo j$\in N^G$,

V_h^L = Volume of compartment h$\in N^H$ in ship,

Ψ = Total carrying capacity or volume of ship s$\in$S in tonnes,

S^S = Sailing speed of the ship (nm/day),

κ^h, ι_h = Lateral and longitudinal distance from compartment h$\in N^H$ to the centre of ship,

α, β = Maximum absolute permissible moments causing trim and heel of the ship,

δ, θ = Maximum allowable draft and trim angle at the ports,

$\psi, \rho, \rho_j, \lambda, \pi$ = Weight of the empty tanker, density of Water, density of cargo j$\in N^G$, length and cross-sectional area of the ship respectively,

M = Some large number.

Decision Variables:

t_k = Time at which leg k$\in K$ ends and ship arrives at a port, (Continuous)

tt_k = Time required to travel during leg(k+1)$\in K$, (Continuous)

x_{pk} = 1 if port p$\in$P is visited at the end of leg k$\in K$ (Binary),

u_{kj} = 1 if cargo j∈N^G is unloaded at the end of leg k∈K (Continuous),

l_{kj} = 1 cargo j∈N^U is loaded at the end of leg k∈K (Continuous),

y_{kj} = 1 if ship s∈S carries cargo j∈N^G on-board during leg k∈K (Binary),

y_j = 1 if cargo j∈N^G is served by the ship (Binary),

$z_{kpp'}$ = 1 if the ship moves from port p∈P to p′∈P during leg (k+1)∈K (Continuous),

c_{kjh} = 1 if compartment $h \in N^H$ of the ship carries cargo $j \in N^G$ at the end of leg $k \in K$ (Binary),

w_{kjh} = Weight of the cargo j∈N^G that is loaded into compartment $h \in N^H$ of the ship during leg k∈ K,

$m_{kjj'h}$[1] = 1 if the compartment $h \in N^H$ of ship holds cargo j∈ N^G during leg (k-1)∈ K and j′ during leg k∈ K (Continuous).

Objective Function:

$$\textbf{Maximize } Z = \sum_{j \in N^G} R_j^S \times W_j \times y_j - \sum_{k \in K} S^S \times C^F \times tt_k - C^T \times (t_{|K|} + \sum_{j \in N^G} \frac{V_j \times u_{j|K|}}{R_j^D}) - \sum_{p \in N^P} \sum_{k \in K} C_p^P \times x_{pk} - \sum_{k \in K/\{0\}} \sum_{j \in N^G} \sum_{j' \in N^G} \sum_{h \in N^H} C_{kjj'h}^C \times m_{kjj'h}, \quad (4.29)$$

The objective function of the revised MILP formulation and the original formulation is same in terms of what it maximizes. The objective function (4.29) maximizes the difference of the total revenue (first term) earned by servicing multiple cargoes and the different costs. Similar to the objective function of the revised MILP formulation, the second term represents the fuel cost, while the third term represents the cost incurred due to time-related activities. The fourth and fifth terms incorporate the fixed port cost and the changeover cost into the objective function (4.29), respectively.

Subject to:

$$\sum_{p \in N^P} x_{pk} = 1 \qquad \forall k \in K, \quad (4.30)$$

$$\sum_{k \in K} x_{pk} \leq 1 \qquad \forall p \in N^P/\{0\}, \quad (4.31)$$

$$x_{P^I k} \leq x_{P^I (k+1)} \qquad \forall k \in K/\{|K|\}, \quad (4.32)$$

[1] $m_{kjj'h}$ is only defined for $k \in K/\{1\}$

$$\sum_{p' \in N^P} z_{kpp'} = x_{pk} \qquad \forall p \in N^P, k \in K/\{|K|\}, \tag{4.33}$$

$$\sum_{p \in N^P} z_{kpp'} = x_{p'(k+1)} \qquad \forall p' \in N^P, k \in K/\{|K|\}, \tag{4.34}$$

Constraint (4.30) ensures that a ship can visit only one port during each sailing leg, while Constraint (4.31) makes sure that a ship cannot visit a port more than once in its planning horizon. Constraint (4.32) ensures that if the chemical tanker is routed to a dummy port, then it remains there for rest of the planning horizon. Constraints (4.33) and (4.34) help define the routing variables ($z_{kpp'}$) and treat them as 0-1 continuous variables. These constraints ensure that $z_{kpp'}$ can be non-zero, if and only if the tanker was at port p during sailing leg k and at port p' during sailing leg $k+1$. Even though $z_{kpp'}$ is defined as a continuous variable between 0-1, Constraint (4.30) in combination with the Constraints (4.33) and (4.34) forces the it to takes values 0 or 1.

$$y_j = 1 \qquad \forall j \in N^O, \tag{4.35}$$

$$y_{kj} = 1 \qquad \forall j \in N^O, k = 0, \tag{4.36}$$

$$x_{pk} = 1 \qquad \forall k = 0, p \in P^I, \tag{4.37}$$

$$\sum_{k \in K} l_{kj} = y_j \qquad \forall j \in N^U, \tag{4.38}$$

$$l_{kj} \leq x_{P_j^L,k} \qquad \forall k \in K, j \in N^U, \tag{4.39}$$

$$\sum_{k \in K} u_{kj} = y_j \qquad \forall j \in N^G, \tag{4.40}$$

$$u_{kj} \leq x_{P_j^D,k} \qquad \forall k \in K, j \in N^G, \tag{4.41}$$

$$\sum_{k \in K} k \times (u_{kj} - l_{kj}) \geq y_j \qquad \forall j \in N^U, \tag{4.42}$$

$$y_{(k+1)j} = y_{kj} - u_{(k+1)j} \qquad \forall j \in N^O, k \in K/\{|K|\}, \tag{4.43}$$

$$y_{(k+1)j} = y_{kj} + l_{(k+1)j} - u_{(k+1)j} \qquad \forall j \in N^U, k \in K/\{|K|\}, \tag{4.44}$$

$$\sum_{j \in N^G} V_j \times y_{kj} \leq \Psi \qquad \forall k \in K, \tag{4.45}$$

Constraints (4.35) and (4.36) ensure that onboard cargoes (N^O) are already assigned to the chemical tanker, while Constraint (4.37) ensures that the chemical tanker starts its voyage form the immediate destination (P^I). Constraints (4.38) amd (4.39) define the pick-up variables (l_{kj}) for each cargo. Similarly, Constraints (4.40) and (4.41) define the discharge variables (u_{kj}). Constraints (4.38) and (4.40) enforces the fact that if a ship doesn't service a cargo then it cannot be picked up or discharged during the any of the sailing legs. Constraint (4.39) makes sure that a cargo can be picked up at most once during the planning horizon ($l_{kj} \leq 1$, if $x_{P^L_j,k}$ = 1), and it can be picked up if and only if the tanker is visiting the pick-up port of cargo $j \in N^U$ ($x_{P^L_j,k}$ = 1). Similarly, Constraint (4.41) states that a cargo $j \in N^G$ can be discharged at most once during the planning horizon, and can be discharged only when the tanker is visiting a cargo's discharge port.

Constraint (4.42) is a precedence type constraint which ensures that a cargo has be picked up before it is discharged if it is serviced by the chemical tanker during the planning horizon. Constraint (4.43) ensures that onboard cargoes are assigned to the ship during every sailing leg until they are discharged. Similarly, Constraint (4.44) makes sure that cargo $j \in N^U$ are assigned to every sailing leg between the cargoes pick-up leg and discharge leg. Constraint (4.45) is the ship capacity constraint which ensures that the the total volume of all cargoes assigned to the ship during a sailing leg k is at most equal to the ship's capacity.

$$tt_k = \sum_{p \in N^P} \sum_{p' \in N^P} \frac{\Delta_{pp'} \times z_{kpp'}}{24S^S} \qquad \forall k \in K/\{|K|\}, \tag{4.46}$$

$$t_k \leq (T^L_j - 0.5T^A) \times l_{kj} + M(1 - l_{kj}) \qquad \forall j \in N^U, k \in K/\{|K|\}, \tag{4.47}$$

$$t_{(k+1)} \geq (T^E_j + 0.5 \times T^A) \times l_{kj} + \frac{V_j \times l_{kj}}{R^L_j} + tt_k \qquad \forall j \in N^U, k \in K/\{|K|\}, \tag{4.48}$$

$$t_{(k+1)} \geq t_k + T^A \times (1 - x_{P^I k}) + \sum_{j \in N^G} \frac{V_j \times u_{kj}}{R^D_j} + \sum_{j \in N^U} \frac{V_j \times l_{kj}}{R^L_j} + tt_k \qquad \forall k \in K/\{|K|\}, \tag{4.49}$$

Constraints (4.46), (4.47), 4.48 and (4.49) model the temporal activities in the formulation. Constraint (4.46) defines the t_k variable as a function of distance over speed between ports p

and p' if $z_{kpp'}$ equals one. Constraints (4.47) and (4.48) define the pick-up time windows for every unassigned cargo $j \in N^U$ during every sailing leg $k \in K/\{|K|\}$. The final time-related constraint makes sure that the arrival time at port during sailing leg (k+1) is greater than or equal to the addition of the arrival time at port during sailing leg (k), administrative time during sailing leg (T^A), the total loading and unloading time of all cargoes during sailing leg (k) and the travel time (tt_k) during sailing leg (k).

$$\sum_{h \in N^H} c_{kjh} \geq y_{kj} \qquad \forall j \in N^G, k \in K, \tag{4.50}$$

$$\sum_{h \in N^H} c_{kjh} \leq y_{kj} \times |N^H| \qquad \forall j \in N^G, k \in K, \tag{4.51}$$

$$\sum_{h \in N^H} \sum_{k \in K} c_{kjh} \geq y_j \qquad \forall j \in N^G, \tag{4.52}$$

$$\sum_{h \in N^H} \sum_{k \in K} c_{kjh} \leq y_j \times |N^H| \qquad \forall j \in N^G, \tag{4.53}$$

$$\sum_{j \in N^G} c_{kjh} = 1 \qquad \forall k \in K, h \in N^H, \tag{4.54}$$

$$w_{kjh} \leq V_h \times \rho_j \times c_{kjh} \qquad \forall k \in K, h \in N^H, j \in N^G, \tag{4.55}$$

$$\sum_{h \in N^H} w_{kjh} = W_j \times y_{kj} \qquad \forall k \in K, j \in N^G, \tag{4.56}$$

$$-\alpha \leq \sum_{h \in N^H} \sum_{j \in N^G} w_{kjh} \times \iota_h \leq \alpha \qquad \forall k \in K, \tag{4.57}$$

$$-\beta \leq \sum_{h \in N^H} \sum_{j \in N^G} w_{kjh} \times \kappa_h \leq \beta \qquad \forall k \in K, \tag{4.58}$$

$$\frac{\sum_{h \in N^H} \sum_{j \in N^G} w_{kjh} + \psi}{\pi \times \rho} - (\lambda \times \tan(\theta/2)) \leq \delta \qquad \forall k \in K, \tag{4.59}$$

Constraints (4.50) - (4.63) model the cargo to compartment assignment activities. Constraint (4.50) states that if cargo j is on-board the ship during leg k then it has to be in at least one compartment. Constraint (4.51) states that the maximum number of compartments that can hold any on-board cargo should be less than or equal to the total number of compartments ($|N^H|$). Similarly, the Constraints (4.52) and (4.53) enforce the same conditions with respect to y_j. Any compartment of the ship should carry either one of the cargoes $j \in N^G$ or should be empty. This condition is modelled by Constraint (4.54). This formulation uses cargo $j = 0$ to model no cargo condition. Additionally, this constraint also enforced the at most one cargo per compartment condition.

Constraint (4.55) is the compartment capacity constraint. This enforces the condition that the weight of cargo $j \in N^G$ per compartment is less than or equal to the volume of that compartment $h \in N^H$ only if cargo j is assigned to the compartment ($c_{kjh} = 1$). Constraint (4.56) makes sure that the total weight of a cargo $j \in N^G$ assigned to all compartments during sailing leg $k \in K$, across all compartments of the ship should be equal to the total weight of the cargo j if that cargo is loaded on the tanker during the sailing leg k. Constraints (4.57) and (4.58) limit the maximum and minimum trim and heel causing moment within the minimum and maximum permissible limits $[-\alpha,\ \alpha]$ and $[-\beta,\ \beta]$, respectively. Constraint (4.59) makes sure that the cargo is distributed in such a way that across the ship that the total allowable draft of the ship is not exceeded.

$$c_{kjh} = 0 \qquad \forall k \in K, j \in N_h^X, h \in N^H, \tag{4.60}$$

$$c_{kjh} + \sum_{h' \in N_h^B} \sum_{j' \in N_j^I} c_{kj'h'} \leq 1 \qquad \forall k \in K, j \in N^G, h \in N^H, \tag{4.61}$$

$$\sum_{j' \in N^G} m_{kjj'h} = c_{(k-1)jh} \qquad \forall k \in K/\{0\}, j \in N^G, h \in N^H, \tag{4.62}$$

$$\sum_{j \in N^G} m_{kjj'h} = c_{kj'h} \qquad \forall k \in K/\{0\}, j' \in N^G, h \in N^H. \tag{4.63}$$

For every compartment $h \in N^H$, and every sailing leg $k \in K$, the cargo-compartment incompatibility is checked and enforced by Constraint (4.60). Cargo-cargo compatibility is enforced by Constraint (4.61). Constraints (4.62) and (4.63) state that $m_{kjj'h}$ is one if and only if during leg $(k-1)$ compartment $h \in N^H$ carried cargo $j \in N^G$, and during leg k compartment $h \in N^H$ carried cargo $j' \in N^G$. Both these constraints together define the changeover activity allowing flexibility to re-arrange cargoes in the compartments. In the next section, we will discuss some key differences between the revised formulation and the original formulation.

4.2.3 Key differences between s-PDP-TWTAC formulations: existing vs. revised

We now describe the key advantages of our model over other existing ones. First, several decision variables defined by Jetlund and Karimi (2004) and Neo et al. (2006) have been eliminated/fixed. Second, we capture the changeover (cargo swapping) activities in a new way, which reduces the complexity of the problem. Third, we propose a different approximation of the pick-up time windows for better modelling. The complete MILP formulation introduced by Jetlund and Karimi (2004) and extended by Neo et al. (2006) is presented in the Appendix (Section 4.2.2). Thus, our model is more realistic and at the same time, more tractable than the earlier ones. Next, we describe the improvements implemented by us.

4.2.3.1 Eliminating/fixing of decision variables from the existing model

A MILP solver performs advanced pre-processing automatically. However, they only look at the mathematical formulation, and have no knowledge about the application and the model. As a result, the solver sometimes can not do model level or application specific reformulations. We also see a substantial improvement in the running time with the proposed reformulations which shows that the solvers are unable to discover and deploy the proposed reformulation techniques. These modifications, even-though elementary, can be overlooked by the reader. Please refer to Section 4.2 for all the definitions. We eliminate the following decisions from the model.

1. The port 0 (immediate destination) of the ship is given. Therefore, during leg 1, we eliminate all arcs not originating from port 0.
   - $z_{1pp'} = 0 \qquad \forall p \in P \backslash \{P^I\}, p' \in P,$
2. If immediate destination of the ship is equal to the loading port of certain cargoes, then the cargo can only be picked up at the end of leg 0. Consequently, the cargo pick-up variable for these cargoes is eliminated for legs greater than 0.
   - $l_{kj} = 0 \qquad \forall k \in K \backslash \{0\}, j \in N^U \backslash \{j | P^I \neq P^L_j\},$
3. The cargo-compartment incompatibility states that incompatible cargoes cannot be stored within certain compartments. This restriction can easily be enforced by eliminating fol-

lowing variables from the model.

- $c_{kjh}, w_{kjh}, b_{kjh}, r_{kjh} = 0 \qquad \forall k \in K, j \in N_h^X, h \in N^H$,

4. At the end of the planning horizon all the cargoes need to be delivered. Thus, the following decisions can be fixed to zero.

   - $l_{|K|j}, u_{|K|j} = 0 \qquad \forall j \in N^U$,
   - $c_{|K|jh}, w_{|K|jh}, b_{|K|jh}, r_{|K|jh} = 0 \qquad \forall j \in N^G, h \in N^H$,

5. Eliminate all cargo-related decision variables if immediate destination of the ship is equal to the discharge port of these cargoes.

   - $l_{kj} = 0 \qquad \forall k \in K, j \in N^U \backslash \{j | P^I \neq P_j^D\}$,
   - $u_{kj} = 0 \qquad \forall k \in K \backslash \{0\}, j \in N^U \backslash \{j | P^I \neq P_j^D\}$,
   - $c_{kjh}, w_{kjh}, b_{kjh}, r_{kjh} = 0 \qquad \forall k \in K, j \in N^G \backslash \{0, j | P^I \neq P_j^D\}, h \in N^H$,

Additionally, the travel time decision variable t_k in the original formulation is also eliminated because it is can be represented by $constant \times z_{kpp'}$, where the constant is defined as the constant time taken to travel between ports p and p'. Further analysis of the s-PDP-TWTAC also enables us to understand that $z_{kpp'}$ can be used to eliminate variables x_{pk}, while rest of the assignment variables can be replaced with c_{kjh}. The discharge variable u_{kj} included in the original formulation is made redundant due to $z_{kpp'}$ because a cargo is always discharged when its discharge port is visited. As such, if $z_{kpP_j^D} = 1$, then it implies that $u_{kj} = 1$. Table 4.1 compares the decision variables defined in the original formulation with the ones proposed by us in the revised formulation.

4.2.3.2 Remodelling of the changeover decision variables

The original formulation captures the changeover activity using a four indexed decision variable as shown in Table 4.1. They define a changeover variable $m_{kjj'h} = 1$ if at the end of leg $k \in K$ cargo $j \in N^G$ is replaced with cargo $j' \in N^G$ in compartment $h \in N^H$. In contrast, we model the changeover activity using three indexed variables (b_{kjh} and r_{kjh}). The three indexed variables are an extension of the on/off variable idea that is presented in Schwindt et al. (2015). As a result, the changeover activity can be captured by a significantly reduced number of vari-

Table 4.1: This table gives a comparison between the decision variables defined in the revised formulation and the original formulation.

Category	**Decision Variables** **Original Formulation**	 **Revised Formulation**	**Comments**
Time Related	t_k, tt_k	t_k	Variable tt_k is eliminated by replacing it with a constant times $z_{kpp'}$
Routing	$z_{kpp'}$	$z_{kpp'}$	No changes
Assignment	x_{pk}, y_j, y_{kj}, c_{kjh}	c_{kjh}	All of the eliminated variables are redundant as they and all the related constraints can be re-designed using c_{kjh} or $z_{kpp'}$.
Pick-up, discharge of cargoes	l_{kj}, u_{kj}	l_{kj}	Analysis of the problem helps us understand that cargo will always be discharged when it discharge port is visited. As such, u_{kj} can be eliminated as $u_{kj} = 1$ is same as $z_{kpp'} = 1$, where $p' = P^D_j$
Quantity	w_{kjh}	w_{kjh}	No change
Changeovers	$m_{kjj'h}$	b_{kjh}, r_{kjh}	Decision variable defined in the original formulation models changeovers with more granularity. However, for a practical problem granularity provided by the revised decision variables is sufficient. Moreover, these new definitions drastically reduce the total number of variables and constraints. Also, they give a tighter LP relaxation for the revised formulation, compared to the LP relaxation of the original formulation.

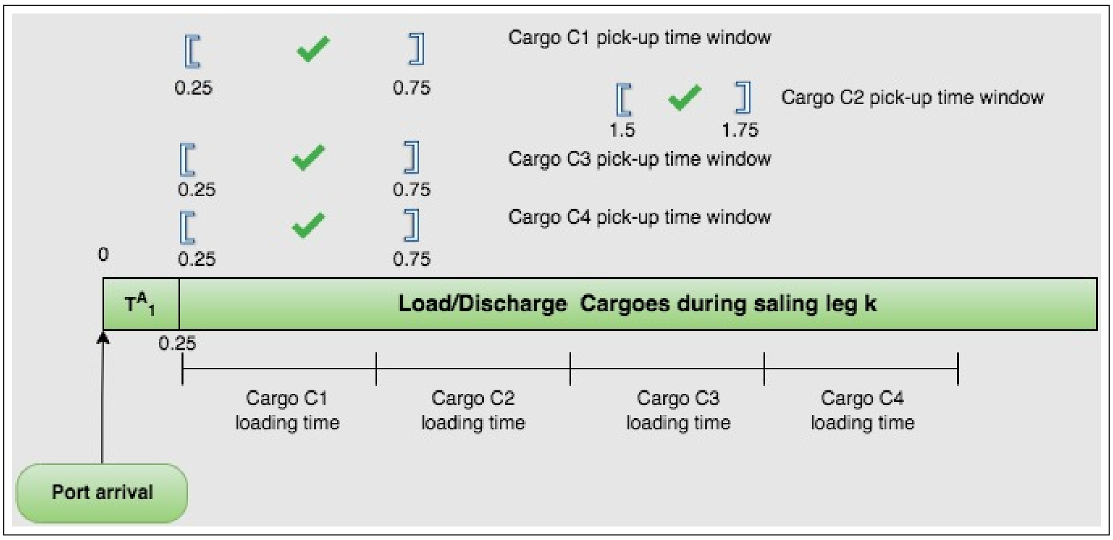

Figure 4.6: This figure represents the pick-up time window definition presented by Neo et al. (2006) and Jetlund and Karimi (2004)

ables. Moreover, empirical tests (Section 4.4)indicate that our formulation yields tighter linear relaxations than the existing formulation presented by Neo et al. (2006).

4.2.3.3 Generalising the definition of the pick-up time windows

The pick-up time windows, as defined by Jetlund and Karimi (2004), Neo et al. (2006) and Cóccola et al. (2015), had some practical limitations. Their definition stated that if a cargo is being picked up, its latest pick-up time should be greater than the port arrival time plus half of the administrative time. Additionally, the definition also stated that the earliest pick-up time should be less than the port departure time minus half of the administrative time and loading time of that cargo. Jetlund and Karimi (2004), Neo et al. (2006) and Cóccola et al. (2015) present the following constraints for the pick-up time windows:

$$t_{k+1} \geq (T_j^E + T_2^A + T_j^P) l_{kj} + \sum_{p \in P} \sum_{p' \in P} T_{pp'}^T z_{kpp'} \qquad \forall k \in K \backslash \{|K|\}, j \in N^U,$$

$$t_k \leq (T_j^L - T_1^A) l_{kj} + M(1 - l_{kj}) \qquad \forall k \in K \backslash \{|K|\}, j \in N^U.$$

We elaborate the need for our approximation with Figures 4.6 and 4.7. For simplicity, assume that there are four cargoes, and that the ship is empty when it arrives at the port. Figure 4.6

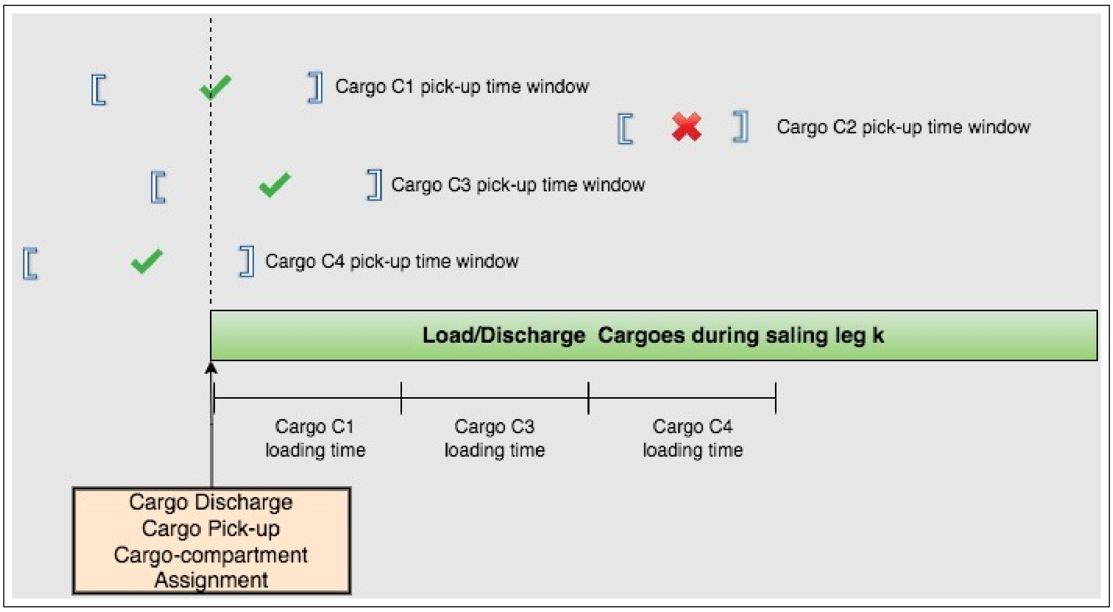

Figure 4.7: This figure represents the pick-up time window definition as defined in the s-PDP-TWTAC

shows the pick-up time windows and the length of these time windows for the four cargoes. Let cargoes one, three and four have the same pick-up time-windows, while Cargo two has a different time period.

If the existing definition of the time windows is considered, then all the cargoes can be picked up. Further, the assumption that the cargoes are loaded consecutively will lead to the scenario presented in Figure 4.6. It can be observed that the actual pickup of cargoes C2, C3 and C4 happens outside their corresponding pick-up time windows. Such a situation might frequently occur in practical instances.

As a result, we re-define the pick-up time-windows. Figure 4.7 shows a scenario describing the revised definitions of the pick-up time windows. According to the revised definition, cargoes can be picked up only if the cargo-assignment point (Figure 4.7) lies within its pick-up time window. This approximation captures more generalised real-world instances. Changes in the definition of the pick-up time windows make the s-PDP-TWTAC more realistic and improve the correctness of the formulation. No particular effect was observed on the total solution time or the quality of solutions that were discovered within the time limit.

Even-though the model presented in Section 4.2 is cleaner and smaller than the existing models, it is still difficult to solve even for medium-sized test instances. In order to find good feasible solutions faster, we propose a heuristic in the following section.

4.3 Instance generator

Benchmark data sets in maritime transportation research are scarce. Only a few researchers have presented reusable benchmark datasets. Brouer et al. (2011) present a benchmark dataset for liner shipping network design models. Their dataset is composed of data from the liner company, Maersk Line. Similarly, Papageorgiou et al. (2014) and Hemmati et al. (2014) present an extensive list of real-world benchmark data for maritime inventory routing problems and tramp scheduling problems. Hemmati et al. (2014) develop their data to represent various shipping segments based on factors like the deep sea or short sea and full-load or mixed-load problems. However, certain limitations restrict the use of their data to our problem. For example, they do not provide data related to the operational facets of our problem, such as the volume of compartments, compartment materials, compartment dimensions, cargo-cargo, and cargo-compartment compatibility. In order to overcome this limitation, we introduce an instance generator that is based on real-world data and parameters. The instance generator code and instances are publicly available online[2]. Our instance generator is built in the `R` programming language. It has three main components, the core data folder, the instance generation engine, and the input parameter file. The core data contains static data used by the instance generator to create the final problem-specific instances. The instance generator engine is the actual code responsible for producing problem instances by processing the core data based on the specifications from the user. Finally, the input parameter file allows the user to select different parameter settings for the instances being produced. Figure 4.8 outlines the structure of our instance generator.

4.3.1 Instance format

A single instance generated by the instance generator consists of four files; namely, the ship data file, the onboard cargo data file, the unassigned cargo data file and the problem data file. The ship data file consists of all the ship-related data like ship number, ship name, ship structure,

[2] https://ladageanurag.shinyapps.io/s-PDP-TWTAC/

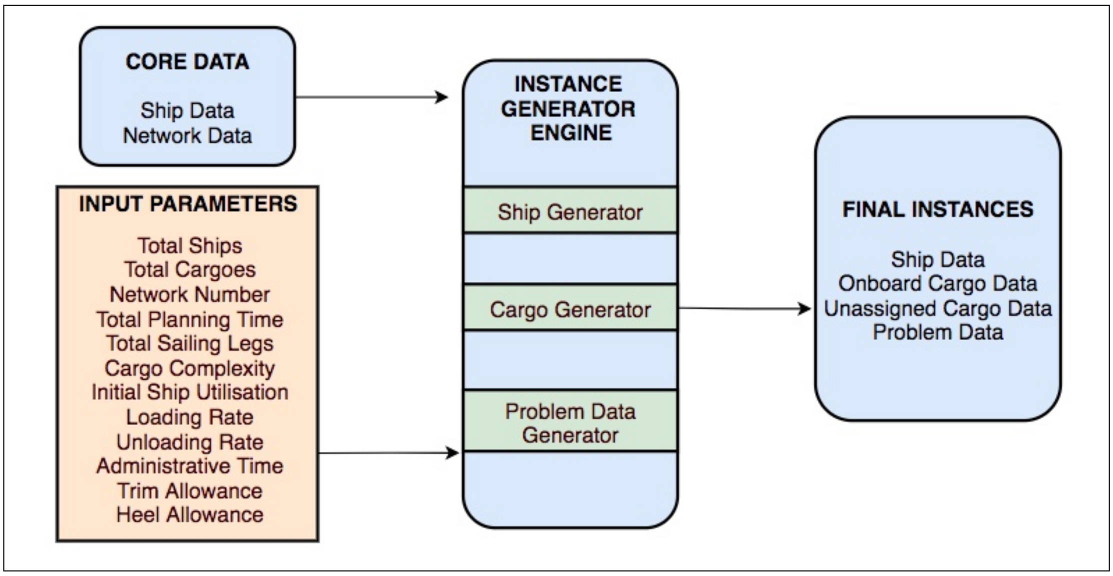

Figure 4.8: The figure details the principle components of the instance generator, namely, core data, input parameters, instance generator engine, and final instances.

port cost, time-chartered cost, and so on. The onboard cargo data file and the unassigned cargo data file consists of cargo data like cargo number, cargo weight, origin, destination, cargo-cargo compatibility restrictions, etc. Finally, the problem data file consists of miscellaneous problem-related data like the total number of ports, port names, port distances and administrative time. Figure 4.9 presents a complete list of data included in a single instance.

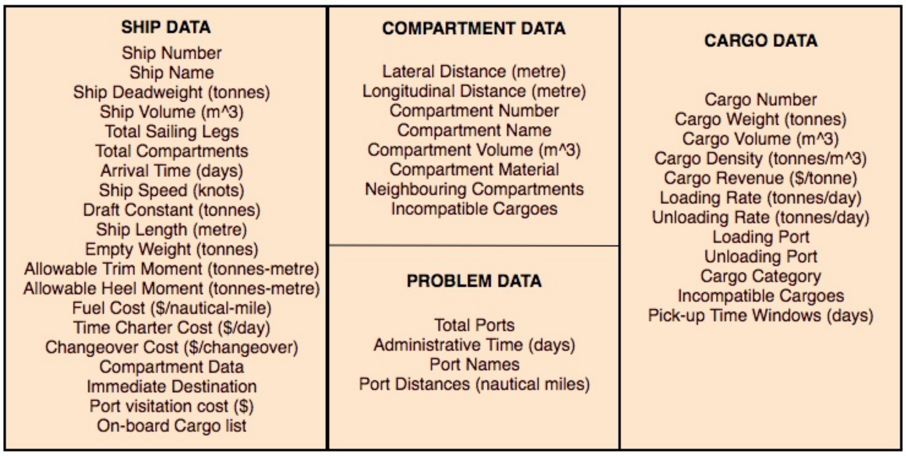

Figure 4.9: Complete list of data generated for a single instance

4.3.2 Core data for generator

The generator relies on the core data to create instances. The core data includes the data collected by us and can be enhanced by the user. The core data consists of 38 structurally different ships and four networks of ports. The designs of the chemical tankers are largely based on Odfjell's chemical tanker fleet[3]. The number of compartments on the chemical tanker ranges from 16 to 52. The compartment walls are made of stainless steel, zinc or epoxy. Network data consists of nautical distances between ports[4]. Network 1 is borrowed from Jetlund and Karimi (2004). Network 2 consists of the 98 busiest ports of 2015 as specified by American Association of Port Authorities[5]. Network 3 consists of the top 47 busy ports in the Asia Region, as published by the International Association of Ports and Harbour[6]. Finally, Network 4 consists of the busiest ports in the year 2016 for the NAFTA region[7]. The NAFTA region consists of ports in the USA, Mexico and Canada. A user can add additional chemical tankers and network-related data to this core data. The instance generator engine reads this core data and generates instances.

4.3.3 Input parameters file

Our model depends on many parameters. To keep our instance generator flexible, we have provided multiple value levels for each parameter. The list of all input parameters that can be specified by the user are as follows:

- total_ships: This parameter allows the user to specify the total number of instances that need to be generated. If a single value (n) is provided, then the generator engine randomly selects *n* different ship data files to create *n* single ship instances. If a list is supplied, then only those ships are used to create instances. When a list is provided, the number of single

[3] https://www.odfjell.com/tankers/our-fleet/

[4] http://ports.com/sea-route/

[5] http://aapa.files.cms-plus.com/Statistics/WORLD%20PORT%20RANKINGS%202015.xlsx

[6] http://www.iaphworldports.org/iaph/wp-content/uploads/WorldPortTraffic-Data_for_IAPH_using_LL_data_2017_Final.pdf

[7] http://aapa.files.cms-plus.com/Statistics/NAFTA%20REGION%20CONTAINER%20TRAFFIC%20PORT%20RANKING%202016_T3.pdf

ship instances generated is equivalent to the length of the list. By default, a maximum of 38 (the default number of different ships included in the core data) instances can be generated if all other input parameters are fixed.

- total_cargoes: This specifies the total numbers of cargoes (both onboard and unassigned) that need to be generated. The number of cargoes can be from one to infinity. Practically, values between 50 and 150 might be interesting, depending on market conditions.

- network_number: Enables the user to select any single network, for instance generation. The network number can be between 1 and 4. The four networks have 36, 98, 46 and 48 ports respectively.

- total_planning_time: The total short-term problem planning horizon. Specifying this parameter ensures that the pick-up time windows of all the unassigned cargoes start before the parameter value. The length of the pick-up time windows varies randomly from 3 days to 7 days. The maximum and minimum time-windows lengths are determined from the literature. We use the value 30 for our tests.

- cargo_complexity: This parameter can take values, 1, 5 or 10. Each value varies the percentage of cargoes that belongs to each category. Setting this parameter value to 1 makes sure that 20% of the cargoes belong to category 1, 30% of the cargoes belong to category 2, 20% of the cargoes belong to category 3, while the rest of the cargoes belong to category 4. Similarly, setting this parameter to 5 segregates the cargoes into four categories by {40%,30%,20%,10%} percentage split. Furthermore, setting this parameter to value 10 yields a cargo split that follows {70%,10%,10%,10%} percentage split. All four cargo categories are described in Section 4.3.4.

- totallegs: This number specifies the total number of legs per ship. If total_ships is a list, then the totallegs parameter also needs to be a list of the same size separated by spaces. If a single number is specified, then all the ships with as many numbers of legs are generated. MILP solution time grows exponentially as the number of sailing legs increase. The suggested range of values is from 7 to 15.

- ship_util_level: This parameter specifies the maximum allowable chemical tanker utilisation at the beginning of the time horizon. For example, setting the value to 0.5 assumes that a maximum of half of the chemical tanker can be filled up with onboard cargoes in the generated data. The minimum and maximum values for this parameter are 0 and 1.

- loading_rate: Specifies the loading rate for all the cargoes. The default tested value is 4800 tonnes/day, which is borrowed from Jetlund and Karimi (2004).
- unloading_rate: Specifies the unloading rate for all the cargoes. Default tested value is 4800 tonnes/day, which is borrowed from Jetlund and Karimi (2004).
- administrative_time: The default tested value is 0.25 days as stated by Jetlund and Karimi (2004).
- Alpha and beta: Absolute maximum allowable trim and heel moments in tonnes-metre. The tested value for both parameters in our experiments is 1 tonnes-metre.

4.3.4 Instance generator engine

The instance generator engine takes multiple input parameters that are provided by the user through a text file. A detailed description of all the input parameters is provided in Section 4.3.3. A high-level pseudo-code of the algorithm, which is used to generate the problem instances is presented using Algorithm 1. A single problem instance comprises of a chemical tanker data file, an on-board cargoes data file, an unassigned cargoes data file, and the problem data file. The chemical tanker data file and both the cargo data files store chemical tanker and cargo-related information, respectively. The problem data file stores port-related information that includes the list of ports in the network and distances (nautical miles) between them. The problem data file also includes the administrative time constant.

The *readInputFile()* processes the inputs that are provided by the user. Subsequently, the *generateShips()* and *generateCargoes()* functions generate the interim chemical tanker data file and the cargo data file. The *generateCargoes()* sub-routine is capable of generating many cargoes infinitely; it can generate as many as four different categories of cargoes. These categories are adopted as per cargo categories defined by Jetlund and Karimi (2004) and Neo et al. (2006). From a practical perspective, more cargo categories can be introduced. However, we believe that from an operations research perspective these categories should suffice for most type of experiments.

Algorithm 1: Instance Generator Engine

```
generate_instances(core_data_directory, output_directory){
    readInputFile();
    generateShips();
    generateCargoes() → Returns cargo_data_file;
    modifyToOnboardCargoes(cargo_data_file){
        solveWeightAssignmentsLP();
        assignCargoNumbersToWeights();
    };
    modifyToUnassignedCargoes(cargo_data_file);
    modifyShipData();
    generateProblemData();
}
```

The first category can be stored in any compartment and has no conflict with any other cargo category. Cargoes in category two have conflicts with cargoes of category three. Further, the cargoes in category three have conflicts with cargo categories two and four. Additionally, the cargoes in category three cannot be stored in epoxy-coated compartments. Finally, the cargoes in category four also have conflicts with cargo category three, and can only be stored in compartments that are made of stainless steel. The *generateCargoes()* sub-routine also generates other cargo related data shown in Figure 4.9. Loading port and unloading port of the cargoes are randomly selected from the port network, such that the travel time between both the ports is not more than the total planning horizon.

The cargo data file that is generated by *generateCargoes()* acts as an input to the *modifyToOnboardCargoes()* and *modifyToUnassignedCargoes()* functions. Both these functions generate the final instance files for on-board cargoes and unassigned cargoes. The final list of onboard cargoes has to be generated such that there is at least one cargo-compartment allocation by weight, which respects the chemical tanker stability requirements and the compartment capacities. One way to generate such an initial set of cargoes is to generate a set of cargoes, along with the cargo weights. For this, a cargo compartment assignment MILP can be solved repeated, by adding a new cargo during every iteration, and terminating when no new cargoes can be inserted. However, this is quite a time consuming process for even a single instance. Thus, this approach cannot be used to generate a large number of instances. Instead, we implement a bottom down approach.

For this purpose, the sub-routine *solveWeightAssignmentsLP()* solves a linear program (Equations (4.64) - (4.67)) below. This linear program tries to maximise the weight in each compartment (w_h) while satisfying the compartment capacity constraint (4.65) and chemical tanker stability constraints (4.66, 4.67). Parameter ρ_{min} equals the minimum density amongst all the cargo generated for that instance. Rest of the parameters used in the linear program are already defined in Section 4.2.

$$\textbf{Maximise:} \sum_{h \in N^H} w_h \tag{4.64}$$

$$\textbf{Subject to: } 0 \leq \quad w_h \quad \leq V_h \rho_{min} \quad \forall h \in N^H, \tag{4.65}$$

$$-\alpha \leq \sum_{h \in N^H} w_h \iota_h \leq \alpha, \tag{4.66}$$

$$-\beta \leq \sum_{h \in N^H} w_h \kappa_h \leq \beta. \tag{4.67}$$

Subsequently, cargo numbers are assigned to weights using the *assignCargoNumbersToWeights()* sub-routine. The *assignCargoNumbersToWeights()* sub-routine takes into consideration all the compatibility constraints to give a list of on-board cargoes with at least one feasible cargo-compartment assignment allocation.

The generator then modifies the chemical tanker data file to include the list of on-board cargoes, immediate destination and port costs through the *modifyShipData()* routine. The *modifyShipData()* routine completes the chemical tanker data instance file. Finally, it generates the problem instance file using the *generateProblemData()* function. In the next section, we discuss computational experiments on instances obtained from the generator.

4.4 Computational study

The computational study is divided into two main parts. First, we discuss the effects of improvements in the model formulation. Second, we present a secondary study discussing the sensitivity of performance parameters of the revised formulation to that of some important input parameters. We first present a brief discussion around the test instances generated for this study.

4.4.1 Description of test instances

We generated 200 test instances for our experiments in the following way. A default seed value of 10 and *total_ships* input parameter value of 38 were provided to our generator to obtain 1,672 (44 different instance sets and 38 chemical tankers) random instances. Instances with the same *Instance Set* number have identical input parameters. The input parameters include the total number of cargoes, the port network, the total planning time, the cargo complexity parameter value, the total number of sailing legs, the utilisation of the ship, the loading rate, the unloading rate, the administrative time, and the alpha and beta parameter values (defined in Section 4.3.3). However, every instance within the same instance set has different chemical tanker characteristics. Additionally, even though the input parameter value *total_cargoes* is same for an instance set it only defines the cardinality of the cargo set. Individual cargoes differ in terms of cargo characteristics like total weight, density, origin, destination and pick-up time-windows.

To keep the number of test instances reasonable, we selected a subset of 13 chemical tankers (Table A.1), with the most diverse characteristics. We narrowed down our test set by randomly selecting 200 test instances in such a way that there is at least one instance from each of the 44 instance sets, and at least one for each of the 13 ships. The instances are named *INST_SET_SHIP*. *SET* denotes the instance set number for a given instance. *SHIP* denotes the chemical tanker number that is used in that particular instance. For example, instance *INST_1_1* would belong to the instance set 1, and model ship 1 (BOW MEKKA) operations. Table A.2 lists the different input parameter values used to generate 44 instance sets. Tables A.2 and A.1 also tabulate some solution-related statistics.

The generator was run using R (version 3.6.1) and RStudio (version 1.1.383). All subsequent tests are carried out using the Cplex 12.7.1 MILP solver. Each instance was solved using 4 cores of the Xeon-E5-2667-v3 3.20 GHz CPU and 8 GB RAM. We used C++11 standard libraries and Cplex Concert Technology libraries to construct all the formulations.

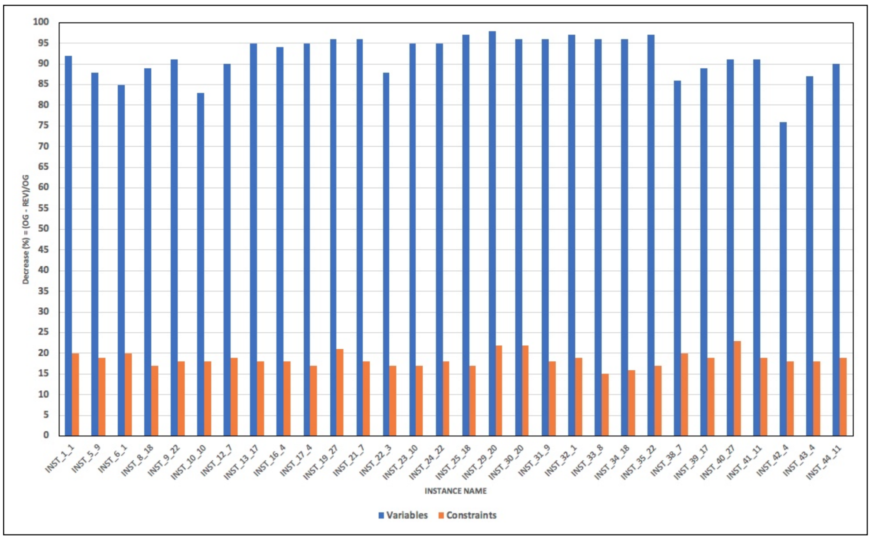

Figure 4.10: Percentage reduction in the number of variables and constraints - OG formulation vs. REV formulation.

We have uploaded along with the instance generator all the 200 test instances[8]. Logs and solution files for the Cplex run are available online. Uploaded Cplex run includes solving the REV formulation presented in Section 4.2 using the Cplex solver for a CPU time limit of 86,400 seconds, and the default MIP gap tolerance of 0.01 %.

For the 200 test instances, the total number of variables is between 23,115 and 4,97,749. The total number of constraints for the 200 test instances varies from 24,440 to 5,23,392.

4.4.2 Effects of improvements in the model formulation

We now compare our revised (REV) formulation and an existing (OG) formulation of the s-PDP-TWTAC. In order to make a fair comparison, we make use of the new approximation of time windows in both the formulations. We refer to the formulation presented by Neo et al. (2006), which is altered with our definition of pick-up time windows as the original formulation (OG). Further, we refer to the formulation presented in Section 4.2 as the revised formulation (REV). As the solution times are large we limit our comparative study to 30 instances selected from the above set of 200 instances. For this study, our overall time limit is 86400 seconds CPU time.

[8] https://ladageanurag.shinyapps.io/s-PDP-TWTAC/

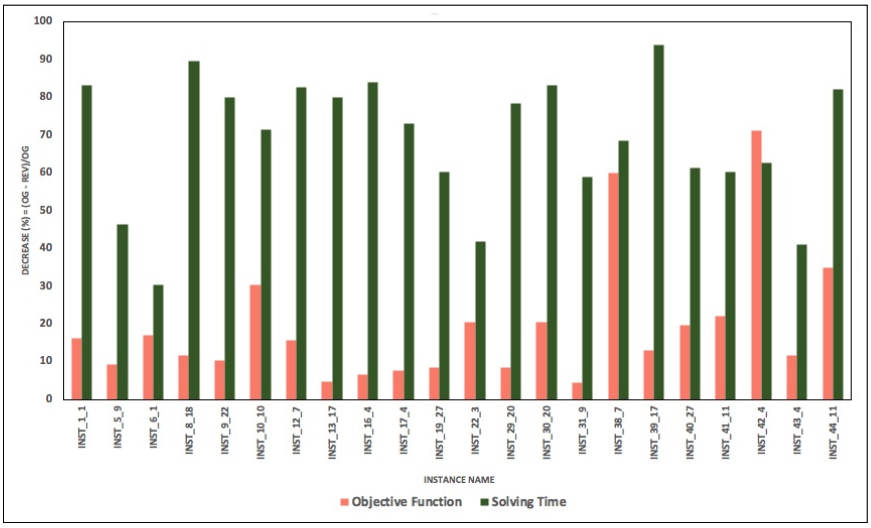

Figure 4.11: LP Relaxation - Objective value and solution time comparison - OG formulation vs. REV formulation.

The OG and the REV formulations are compared on problem size, and their linear relaxations. The LP relaxation provides an upper bound to the optimal value of the model. The lower the upper bound, the tighter is the LP relaxation giving a solution closer to the integer feasible points. Figure 4.10 reports the percentage reduction from OG to REV in the number of variables and constraints. It was observed that the problem size decreased drastically for all the instances when the REV formulation is used. 21 out of 30 instances show a reduction of at least 90 % in the total number of variables, and other instances show a reduction by at least 76 %. Further, the total constraints decrease by 15 % to 23 %. Changes described in Section 4.2.3.1 helped reduce the number of decision variables and constraints between 5 to 15 percent. These changes did not have any impact on the tightness of the LP relaxation. The most significant improvement was achieved by reformulating the changeover decision variables (presented in Section 4.2.3.2). On an average, this change lead to a 70 % to 95 % decrease in the total number of decision variables, and 10 % to 15 % decrease in the total number of constraints.

The LP relaxation of both the formulations could be solved within the time limit for 22 out of 30 instances. Figure 4.11 compares the LP relaxation value of OG and REV formulations on these 22 instances. It shows that both the solution time and the upper bound decreases for our revised (REV) formulation. Since, the LP relaxation gives a lower value, we obtain a tighter bound from the REV formulation before any cutting is done. The remodelled changeover decision variables are solely responsible for tightening of the LP relaxation of the revised formulation in comparison to the original formulation. In the remaining 8 instances, the LP relaxation of the OG formulation ran out of memory. In contrast, solver managed to solve the REV formulation without any memory issues.

Now we compare performance parameters such as the total solution time and the relative gap at time limit for the two MILP formulations (Table 4.2). The second column (*Cplex Status*) reports *OOM* if the MILP solver ran out of memory. The column that is labelled, *Best Objective*, lists the best lower bound obtained at termination. The total solution time at termination for all the 30 instances is reported in the succeeding column (*Total CPU Time*). Additionally, the *Relative Gap* column presents the relative gap (%) between the upper and lower bound on the optimal value reported by Cplex at termination.

Out of the 30 instances, 18 instances ran out of memory (8 GB) without discovering any feasible integer solution when solved using the OG formulation. The OG formulation has a weaker linear relaxation as compared to the REV formulation. Additionally, it has an exponential number of decision variables which causes the branch and bound tree to explode. We believe this is the reason for OOM status. Additionally, for the 18 instances that ran out of memory, the total solution time represents the time at which the problem was terminated due to memory limits. In contrast, the REV formulation stays within memory limits and finds at least one feasible integer solution for all 30 instances. Additionally, REV formulation finds the optimal solution in 15 instances (within the time limit), while the OG formulation terminates with optimality in only six instances. We observe that REV formulation is much faster (up to 15 times) and memory efficient as compared to the OG formulation. We now move our discussion towards the a short study on sensitivity of performance parameters with respect to some of the input data.

4.4.3 Sensitivity analysis of Cplex run performance parameters

We discuss the sensitivity of the performance parameters, namely, the Gap (%) and the total CPU time of the Cplex run with respect to the different input parameters. The Gap (%) is calculated with respect to the upper bound reported by Cplex. The primary input parameters considered for this study are the total number of cargoes, the total number of ports, the maximum number of sailing legs, and the number of discharge ports of the onboard cargoes. Moreover,

Table 4.2: Comparison of problem status, best integer solution, total solution time and relative gap between the original and the revised formulation.

Instances	Cplex Status		Best Objective		Total CPU Time		Relative Gap	
	OG	REV	OG	REV	OG (sec)	REV (sec)	OG (%)	REV (%)
INST_1_1	Optimal	Optimal	1904910	1904910	5651	412	0.01	0.01
INST_5_9	Optimal	Optimal	995262	995262	72102	31006	0.01	0.01
INST_6_1	Feasible	Optimal	1537990	1537990	86400	25274	54.563	0.01
INST_8_18	OOM	Optimal	No Sol	2269880	718	20858	Inf	0.01
INST_9_22	OOM	Feasible	No Sol	3239300	594	86400	Inf	53.261
INST_10_10	Feasible	Feasible	407682	372226	86400	86400	607.687	689.32
INST_12_7	OOM	Feasible	No Sol	1172500	779	86400	Inf	104.12
INST_13_17	OOM	Optimal	No Sol	2801950	1054	2288	Inf	0.01
INST_16_4	OOM	Optimal	No Sol	1512150	1538	15294	Inf	0.01
INST_17_4	OOM	Feasible	No Sol	1829870	834	86400	Inf	109.954
INST_19_27	Feasible	Feasible	3821410	4684360	86400	86400	82.4	37.607
INST_21_7	OOM	Feasible	No Sol	1531150	14	86400	Inf	314.955
INST_22_3	OOM	Feasible	No Sol	-281065	70	86400	Inf	1220.24
INST_23_10	OOM	Feasible	No Sol	2951890	30	86400	Inf	132.28
INST_24_22	OOM	Feasible	No Sol	793339	30	86400	Inf	695.712
INST_25_18	OOM	Optimal	No Sol	3769140	37	70888	Inf	0.01
INST_29_20	OOM	Optimal	No Sol	3798930	270	75667	Inf	0.01
INST_30_20	OOM	Optimal	No Sol	2219180	110	43172	Inf	0.01
INST_31_9	OOM	Feasible	No Sol	5863370	747	86400	Inf	15.953
INST_32_1	OOM	Feasible	No Sol	2450400	33	86400	Inf	109.332
INST_33_8	OOM	Feasible	No Sol	991297	33	86400	Inf	771.241
INST_34_18	OOM	Feasible	No Sol	581309	1	86400	Inf	1035.6
INST_35_22	OOM	Feasible	No Sol	-585672	34	86400	Inf	1740.83
INST_38_7	Optimal	Optimal	-1167910	-1167910	6477	5420	0.01	0.01
INST_39_17	Optimal	Optimal	1218010	1218010	16594	1014	0.01	0.01
INST_40_27	Optimal	Optimal	623656	623656	1640	750	0.01	0.01
INST_41_11	Feasible	Feasible	25338.7	200491	86400	86400	3081.64	0.05
INST_42_4	Feasible	Optimal	-811647	-781745	86400	32743	51.717	0.01
INST_43_4	Optimal	Optimal	2836360	2836360	76294	15602	0.01	0.01
INST_44_11	Feasible	Optimal	470923	505503	86400	36092	199.387	0.01

some secondary input parameters like the total number of compartments, the ship speed, the draft constant, the fuel cost, the time charter cost, and the average compartment volume were also considered during this study.

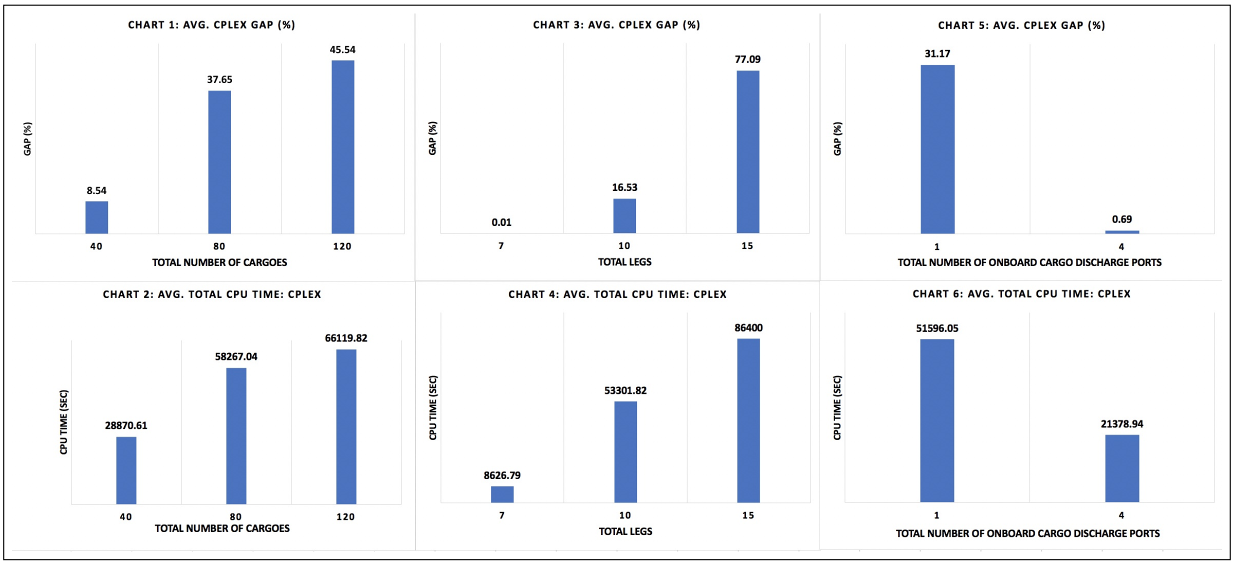

Figure 4.12: This figures presents the sensitivity analysis of the Cplex performance parameters with respect to the total number of cargoes, total legs, and the total number of onboard cargo discharge ports.

Analysis using multiple linear regression was performed to explore the effects of the input parameters on the performance parameters. Some of the primary input parameters significantly affect the performance parameters. Figure 4.12 helps us illustrate this claim. However, the performance parameters seem to be insensitive to the secondary input parameters. Figure 4.12 classifies the test instances into different categories based on the input parameters. The vertical axis in these figures presents the average of the performance parameters. For example, the first chart in Figure 4.12 differentiates the test instances based on the total number of cargoes on the horizontal axis. Similarly, the vertical axis presents the average Gap (%).

In Figure 4.12, Chart 3 and 4 show that both the average solution quality and the average total CPU time worsen with the increase in the maximum number of sailing legs. Both the Cplex performance parameters deteriorate with the increase in the maximum number of legs.

Charts 5 and 6 in Figure 4.12 illustrate that the increase in the total number of onboard cargo discharge ports significantly improves the Gap (%) and the total CPU time for the Cplex runs. This effect is correct because the total number of onboard cargo discharge ports reduces the flexibility of the route of the chemical tanker. As per the problem definition, all onboard cargoes must be delivered. Consequently, their corresponding discharge ports have to be visited. Thus, a higher number of different discharge ports of onboard cargoes reduces the number of new ports on the route of the ship. This reduces the feasible region of the problem. Additionally, our sensitivity study showed that the performance parameters were not affected by the number of ports or the number of compartments.

This chapter discussed the MILP formulation and numerical experiments related to the s-PDP-TWTAC. It also discussed the design and construction of the instance generator. However, even the improved MILP formulation is insufficient to be used in a real-world tactical setting. As such, the next chapter discusses different heuristics designed to solve the s-PDP-TWTAC.

This page was intentionally left blank.

Chapter 5

Neighbourhood search heuristics for the s-PDP-TWTAC

In this section, we present different neighbourhood-search heuristics. Each of them has two phases. Phase 1 of the heuristics generates one or more initial solutions for Phase 2. Subsequently, Phase 2 implements a search strategy to improve the initial solutions further. We implement different strategies for Phase 1 and Phase 2 of the heuristics.

Chapter 5 is structured as follows. We first describe the two important sub-routines, namely, *insertNewCargo* and *portRemove*. These two sub-routines form the core of the search strategy (Phase 2) used by all the heuristics. *insertNewCargo* can be thought of as a local search step, which helps generate different feasible solutions within the same neighbourhood, while *portRemove* helps change the neighbourhood. Subsequently, this chapter presents a detailed discussion presenting all the heuristics. The neighbourhood search heuristics are divided into two main categories. *LPNS* and *LPNS+* heuristics depend on the linear relaxation to generate initial solutions. As such, we classify them as *linear relaxation guided neighbourhood search heuristics*. On the other hand *Heuristic 1*, *Heuristic 2*, *Heuristic 3*, and *GRASP* heuristic depend on an integer programming relaxation to generate the initial solution during Phase 1. As

Table 5.1: This table presents a high-level summary of the heuristics discussed in this chapter

<table>
<tr><th></th><th>LPNS+</th><th>H1</th><th>H2</th><th>H3</th><th>GRASP</th></tr>
<tr><td>Core-subroutine 1: insertNewCargo</td><td colspan="5">This sub-routine tries to improve a given solution by inserting a new cargo into it. If phase 2 of all the heuristics is viewed as an iterated local search strategy, then this sub-routine is the local search component within Phase 2.</td></tr>
<tr><td>Core-subroutine 2: portRemove</td><td colspan="5">This sub-routine tries to randomise a given solution by removing a port from the route of the ship. This sub-routine primarily helps the heuristics jump out of local optimal solutions. Within the scope of an iterated local search, this sub-routine can be viewed as an perturbation step.</td></tr>
<tr><td colspan="6">The above sub-routines form the core on which the second phase of the heuristics are constructed.</td></tr>
<tr><td>Phase 1</td><td>LP is solved, followed by an MILP to generate a single initial solution.</td><td>An MILP is solved over a restricted set of ports to generate a single initial solution.</td><td>A relaxed MILP is solved to fix the route of the chemical tanker. The s-PDP-TWTAC MILP is solved for this fixed route to generate a single initial solution.</td><td>The discharge ports of the onboard cargoes will always be part of an feasible solution. As such, route permutations are generated from these ports. Solutions corresponding to each route permutation is an initial solution.</td><td>Multiple restricted MILPs of the s-PDP-TWTAC are solved to generate multiple initial solutions.</td></tr>
<tr><td>Phase 2</td><td>Improves the phase 1 solution using the iterated local search strategy.</td><td>Improves the phase 1 solution using the iterated local search strategy.</td><td>Improves the phase 1 solution using the iterated local search strategy.</td><td>Every initial solution is improved using the iterated local search strategy.</td><td>For all other heuristics, the insertNewCargo sub-routine inserts cargoes greedily, while for GRASP it follows a greedy random approach.</td></tr>
<tr><td colspan="6">Note: The differences in the implementations of the Phase 1 and Phase 2 of the heuristics are described in details in the succeeding sections.</td></tr>
</table>

a result, these heuristics are grouped together as *integer programming guided neighbourhood search heuristics*. Table 5.1 gives a high level summary of the heuristics discussed in this chapter. Next, we discuss the two core sub-routines.

5.1 Core sub-routines

Phase 2 of our heuristics primarily improves the initial solutions using two core sub-routines, namely, the *insertNewCargo* sub-routine and the *portRemove* sub-routine. The *insertNewCargo* sub-routine tries to improve the solution greedily based on profit, while the *portRemove* sub-routine tries to randomize the local search space. A single iteration of Phase 2 involves either the inserting of a new cargo or the removing of a port from the chemical tanker's route. We first discuss the two core sub-routines in the next two Sections 5.1.1 and 5.1.2.

5.1.1 Core sub-routine 1: insertNewCargo

The first sub-routine is the *insertNewCargo* sub-routine. In a single iteration, this sub-routine identifies a profitable cargo to service. Additionally, if a profitable cargo is identified, this sub-routine also updates the schedule of the chemical tanker to incorporate the pickup and delivery of the new cargo. It starts by classifying cargoes into four categories and generating a list of potential cargoes (N^{U1}). Set N^{U1} is a subset of unassigned cargoes (N^{U}) and excludes all cargoes that are already loaded on the chemical tanker in the present solution.

Category +PD cargoes are the ones for which the pickup port and the discharge port are not part of the tanker route in the current solution. For a Category +D cargo, the pickup port is already part of the current route but not its discharge port. Similarly, for a cargo in Category +P, its pick port has to be added to the route of the chemical tanker. Finally, a cargo included in Category +$\emptyset$ has both its pickup port and its discharge port on the route of the chemical tanker. As a result, no new port associated with the cargo needs to be added. For Category +$\emptyset$ cargoes, their pickup and discharge ports should satisfy the precedence constraint in order to be able to pick them up. The precedence constraint states that the pickup port has to be visited before the discharge port.

In our problem, the maximum number of sailing legs is fixed. The maximum number of ports that can be inserted into the present route of the chemical tanker is equal to the number of dummy ports (DP) on the route. A dummy port is an artificial port assigned to a sailing leg if a real port is yet to be assigned. As an artificial location, the dummy port is at zero nautical miles from all other ports and has no associated port cost when assigned to the route. As a result, set N^{U1} can include Category +PD cargoes only if the number of DP is greater than 1. Similarly, cargoes in categories +D and +P are included in set N^{U1} only if the number of DP is greater than 0.

Inserting a new cargo into the present solution entails the inserting of the corresponding pickup and discharge ports of that cargo. As such, for every cargo $j \in N^{U1}$, we generate a list of feasible positions. A position is represented as a tuple that consists of the cargo number, the pickup sailing leg, the discharge sailing leg, profit and the cargo category. The pickup sailing leg and the discharge sailing leg are the sailing legs during which the new cargo should be picked up and discharged. As a result, one cargo can have multiple feasible positions. Profit is defined as the change in the present solution's objective function value if the cargo is picked up during the pickup sailing leg and discharged during the discharge sailing leg. The final element of the position tuple indicates the category of the cargo.

A position is feasible if it passes through four different feasibility checks. First, we check for sufficient chemical tanker capacity at each port on the route of the chemical tanker. The capacity of the chemical tanker is calculated based on the volume of empty compartments. Second, we check whether the pickup port of the cargo to be inserted can be reached within the pickup time window of the cargo. Next, we ensure that inserting the position does not violate the pickup time windows of already loaded cargoes. Before performing the fourth feasibility check, the feasible positions are sorted into a descending order of profit, and all the positions with negative profit are discarded.

The final feasibility check is conducted to ensure that the position does not violate the compartment-related constraints. To simplify each iteration, we assume that all the pre-existing cargo-compartment allocations are fixed. Thus, we solve the mixed-integer program (MIP) only for decision variables that are indexed on the cargo to be inserted. The decision to fix the pre-existing allocations is primarily to reduce the time complexity of the mixed integer program solved to generate these allocations. Some initial experiments were performed to generate an entirely new cargo-compartment allocation. However, this substantially added to the overall solving time of each MIP.

Readers can also observe that the MIP (Equation (5.1) - (5.10)) is solved to accept the first feasible cargo-compartment allocation by setting the objective function to zero. However, in order to generate better quality allocations, two alternate objective functions were also used in initial experimentations. These objectives tried to minimize the total number of changeovers (b_{kjh}) or the total number of compartment utilization (c_{kjh}). Both these objectives drastically increased the solution time to even generate a single feasible cargo-compartment allocation. Moreover, we did not notice any improvement in the overall quality of the solutions generated by the heuristics. However, as the alternative objectives don't affect feasibility, they could be used as a form of *solution polishing* to provide an equivalent solution to the best found at the end of the main algorithm in which cargo plans have been optimised according to secondary criteria relating to stability, handling cost, or other considerations.

We solve the following MIP to generate a feasible cargo-compartment allocation plan. The following MIP is a special case of the s-PDP-TWTAC formulation. Let J represent the cargo number of the input cargo. Let K' be the set of sailing legs between the pickup leg and the discharge leg of the new cargo. Additionally, let K be the set of all sailing legs. N^C and N^H be the set of all cargoes and set of cargo holds/compartments of the chemical tanker, respectively. Set N^C has a special cargo indexed as 0 with weight 0 and no compatibility restrictions.

The MIP (Equation (5.1) - (5.10)) fixes all variables of the s-PDP-TWTAC formulation except c_{kjh}, w_{kjh}, b_{kjh} and r_{kjh}. c_{kjh} is binary and takes the value 1 if cargo j is stored in cargo hold h at the end of sailing leg k. w_{kjh} is real-valued, and denotes the actual weight of cargo j stored in cargo hold h at the end of leg k. b_{kjh} and r_{kjh} are both binary. b_{kjh} is 1 if cargo j at the end of leg k replaces any other cargo in compartment h. Similarly, r_{kjh} takes the value 1 if at the end of leg k cargo j is removed from the cargo hold h.

Objective function:

$$\textbf{Minimise} \;\; z = 0 \tag{5.1}$$

Subject to:

$$\sum_{h \in N^H} c_{kJh} \geq 1 \qquad \forall k \in K', \tag{5.2}$$

$$\sum_{j \in N^C} c_{kjh} = 1 \qquad \forall k \in K, h \in N^H, \tag{5.3}$$

$$c_{(k-1)Jh} + b_{kJh} = c_{kJh} + r_{kJh} \qquad \forall k \in K' \setminus \{0\}, h \in N^H, \tag{5.4}$$

$$w_{kJh} \leq V_h^H \rho_J c_{kJh} \qquad \forall k \in K', h \in N^H, \tag{5.5}$$

$$\sum_{h \in N^H} w_{kJh} = W_J \qquad \forall k \in K', \tag{5.6}$$

$$-\alpha \leq \sum_{h \in N^H} \sum_{j \in N^C \setminus \{0\}} w_{kjh} \iota_h \leq \alpha \qquad \forall k \in K', \tag{5.7}$$

$$-\beta \leq \sum_{h \in N^H} \sum_{j \in N^C \setminus \{0\}} w_{kjh} \kappa_h \leq \beta \qquad \forall k \in K', \tag{5.8}$$

$$b_{kJh} \leq c_{kJh} \qquad \forall k \in K' \setminus \{0\}, h \in N^H, \tag{5.9}$$

$$r_{kJh} \leq c_{(k-1)Jh} \qquad \forall k \in K', h \in N^H. \tag{5.10}$$

Constraint (5.2) states that the input cargo J has to be assigned to at least one compartment between its pickup and discharge legs. Constraint (5.3) makes sure that every compartment is either empty or has, at the most, one cargo assigned to it. Constraint (5.4) models changeovers. It tracks the swapping of cargoes within the chemical tanker's compartments. Constraint (5.5) ensures that the compartment capacity is not exceeded. Constraint (5.6) ensures that the quantity of a cargo that is distributed in multiple compartments during a sailing leg is equal to the total weight of the cargo. Constraints (5.7) and (5.8) ensure that the maximum allowable trim and heel moments are not exceeded. Constraints (5.9) and (5.10) ensure that b_{kjh} and r_{kjh} are not free variables. Constraint (5.9) ensures that b_{kjh} is zero if cargo J was not assigned to the compartment in the present leg. Constraint (5.10) ensures that the r_{kjh} can take value 1 if and only if cargo J was assigned to the compartment during the last leg. The cargo-compartment and cargo-cargo incompatibilities are enforced in pre-processing by fixing the bounds of the c_{kjh} variables. The *insertNewCargo* sub-routine is presented Algorithm 2.

Algorithm 2: insertNewCargo(data, s)

```
cargo_by_category := generatePotentialCargoes;
Declare: position_list;
for cargo in cargo_by_category do
    createPositionTuple(cargo);
    populate(position_list);
if isNotEmpty(position_list) then
    sortByProfit(position_list);
    for position in position_list do
        if profit > 0 and compartmentAllocationExists then
            generateNewSchedule(position, s);
else
    return s;
```

We further explain the *insertNewCargo* sub-routine with the help of an example. Figure 5.1 presents a toy-sized example and an initial solution. The port network is a complete bidirectional graph that consists of seven ports. The chemical tanker is at port Shanghai at the beginning of the time horizon. A maximum of four ports can be inserted into the chemical tanker's route. Let N^O consist of Cargo 5, while N^U equals $\{1,2,3,4\}$. Relevant cargo-related information is presented in Figure 5.1. For simplicity, we consider a chemical tanker with five stainless steel compartments with volumes, as shown in Figure 5.1.

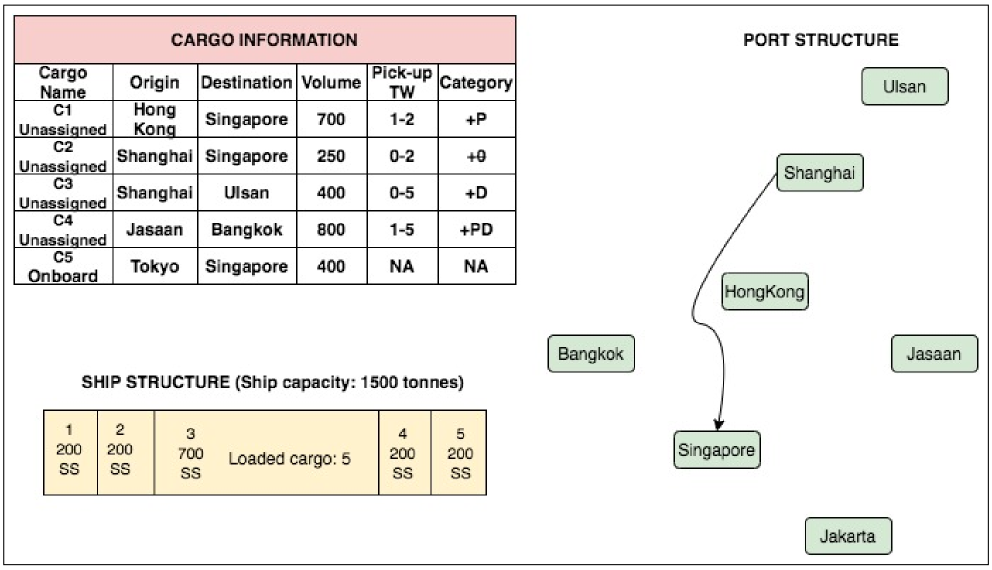

Figure 5.1: This figure represents a toy sized instance of the s-PDP-TWTAC along with an initial solution s^0 for that instance.

Phase 1 of the Heuristic *H1* generates the initial solution, s^0, as shown in Figure 5.1. The chemical tanker starts from Shanghai and travels to Singapore during sailing leg 1. Moreover, Cargo 5 is delivered at its discharge port, Singapore. During the first sailing leg, Cargo 5 (volume 400) is stored in Compartment 3. The objective function of s^0 is 1658.33 units. The chemical tanker capacity available at leg zero is 800 units (equal to the volume of empty compartments).

Based on s^0, Phase 2 of the heuristic assigns Cargo 1 in Category +P, Cargo 2 in Category +∅, Cargo 3 in Category +D, and Cargo 4 in Category +PD. Further, position(s) are created for each of the four cargoes. As mentioned earlier, a position is represented as a tuple that consists of the cargo number, the pickup leg, the discharge leg, the profit and the cargo category. Figure 5.2 presents all seven feasible positions for the four feasible cargoes. Figure 5.2 illustrates the updated solutions when cargoes from each of the categories are inserted into s^0. The middle row presents the initial solution, s^0. The top and bottom rows show the updated solutions obtained by inserting the most profitable position for each cargo.

For example, the most profitable position for Cargo 1 is (1, 1, 1, 6846, +P). Position (1, 1, 1, 6846, +P) states that Cargo 1 should be picked up and discharged during the first sailing leg. The profit that is obtained by inserting Position (1, 1, 1, 6846, +P) in s^0 is 6846 units. Finally, it

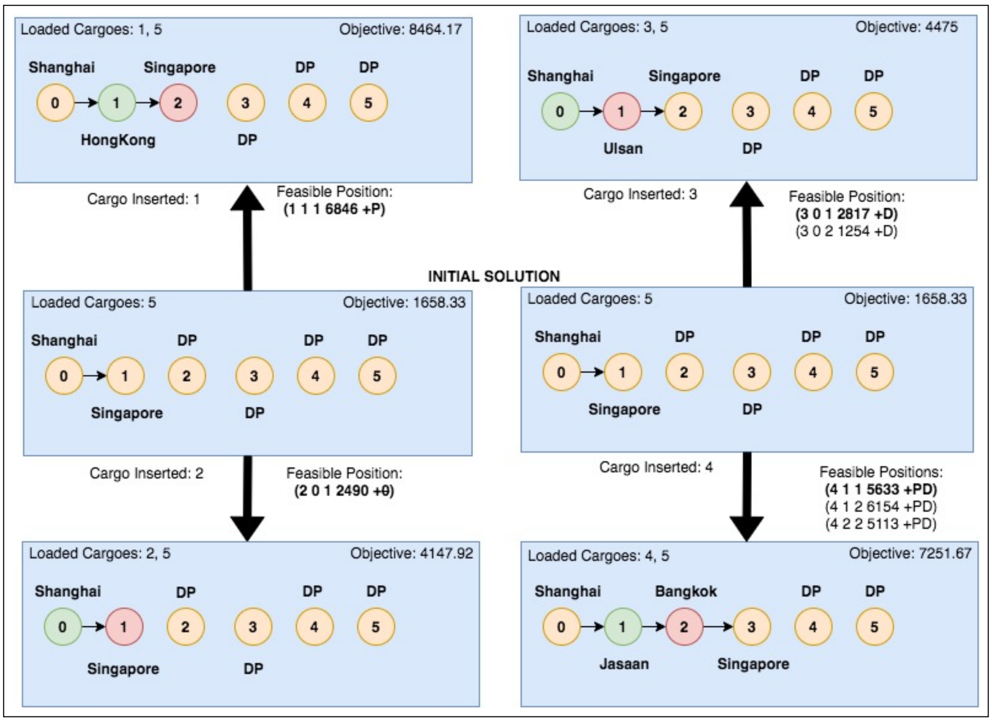

Figure 5.2: Core sub-routine 1: insertNewCargo is illustrated in this figure, which constitutes the local search strategy incorporated in Heuristics LPNS+, H1, H2, H3, and GRASP.

also states that Cargo 1 belongs to Category +P. The block in the first row and the first column represents the updated solution that results from the inserting Position (1, 1, 1, 6846, +P) into s^0. Port Hong Kong (pickup port of Cargo 1) is inserted during the initial solution's first sailing leg. Additionally, keeping in mind the precedence constraint, the pickup port is placed before the discharge port. The discharge port of Cargo 3 (Singapore) is already on the route of the chemical tanker. In the updated schedule, the chemical tanker travels from Shanghai to Hong Kong and ends its voyage in Singapore. Cargo 3 is picked up in Hong Kong, and both Cargo 5 and Cargo 3 are delivered in Singapore. Further, the objective function of the updated solution is 8464.17 units.

5.1.2 Core sub-routine 2: portRemove

The second core sub-routine is the *portRemove* sub-routine that modifies the existing solution by removing one port from the present route of the chemical tanker. There are two conditions under which the *portRemove* sub-routine is invoked. First, the number of iterations since the last port removal should be equal to the number of pre-defined iterations, and second, the *insertNewCargo* sub-routine is unable to generate any new feasible positions for the present solution.

The port to be removed from the route of the chemical tanker is randomly selected using uniform distribution. However, it cannot be a discharge port of an on-board cargo. Once a port is removed, the updated route of the chemical tanker is generated without altering the sequence of other ports. Removing a port from the present route of the chemical tanker affects the arrival times of other ports, as well as the cargo pickups and drop-offs that are associated with it. As such, all unassigned cargoes that are already allocated to the chemical tanker have to be re-assigned. As a result, we construct a new solution with the updated route of the chemical tanker and with only on-board cargoes being delivered. The construction of the updated route of the chemical tanker still involves the generating of a feasible cargo-compartment allocation for the on-board cargoes. A simple way to generate an allocation plan for the on-board cargoes is by solving the MILP defined by Equations (5.1) - (5.10).

Since solving an MILP may be time-consuming, we employ an alternative strategy to generate a feasible cargo-compartment allocation plan for the on-board cargoes N^O. For heuristics 1 and 3, the original sequence of ports from s^0 remains unchanged throughout the heuristic. Consequently, for every new route that is generated by eliminating a port, the cargo-compartment allocations for the on-board cargoes can be calculated directly from s^0. This alternative strategy enables us to generate cargo-compartment allocations at a much faster rate.

Let the set of discharge ports of on-board cargoes be P, and its compliment be P^c. The ports that appear on the updated route of the chemical tanker belong either to P or P^c. For all ports, $p \in P$, the cargo-compartment allocation is equal to the allocation at port p, in the initial solution s^0. For all the ports, $p' \in P^c$, the cargo allocation is calculated as follows. Let port $p \in P$ be the last visited discharge port of on-board cargo before port p' in the updated route of the chemical tanker. Then, the cargo-compartment allocation at the port, p', in the updated solution, is equal to the cargo-compartment allocations at the port, p, in the initial solution, s^0.

Figure 5.3 illustrates the *portRemove* sub-routine. Given the present solution (s), we first identify the ports that can be removed. Excluding the immediate destination (P^I) of the chemical tanker, and the discharge port of Cargo 5, we are left with ports Jasaan and Bangkok. Port Bangkok is eliminated randomly from the route of the chemical tanker along with unassigned Cargo 4. Subsequently, the updated solution (s') is constructed as shown in Figure 5.3. The

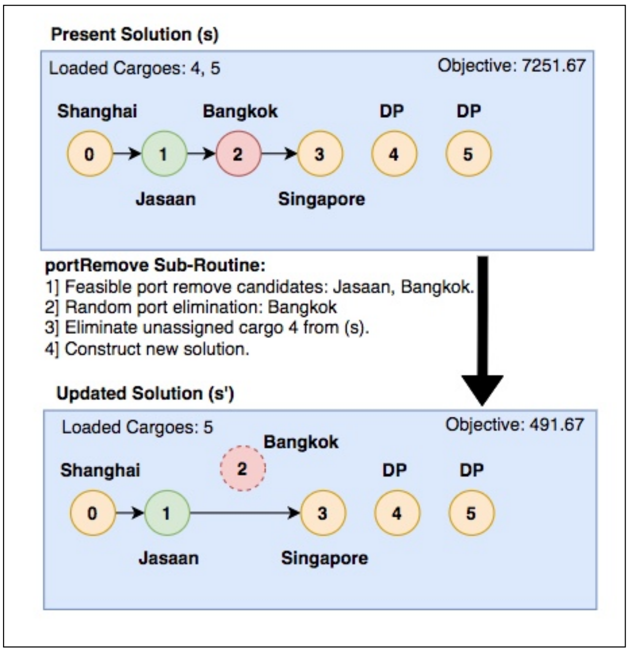

Figure 5.3: Core sub-routine 2: portRemove is illustrated in this figure, which is a part of the local search strategy incorporated in Heuristics LPNS+, H1, H2, H3, and GRASP.

objective function of s' is 491.67 units. The set P includes ports Shanghai and Singapore, while set P^c includes port Jasaan. Thus, corresponding to the cargo-compartment arrangement in the initial solution, Cargo 5 is stored in the third compartment at port Shanghai, while the chemical tanker is empty at port Singapore. Additionally, at port Jasaan, the cargo-compartment arrangement is the same as that at port Shanghai. Thus, Cargo 5 remains in Compartment 3 throughout the entire voyage.

The *insertNewCargo* sub-routine constructs an updated solution, s', by identifying and adding new positions to the present solution, s. The *portRemove* sub-routine identifies a port to be removed from the present solution, s. It then constructs an updated solution, s', by eliminating the selected port and all the loaded unassigned cargoes from the present solution, s. Having understood the two core sub-routines, we now present our heuristics. The heuristics can be broadly divided into linear relaxation guided neighbourhood search heuristics, and integer relaxation guided neighbourhood search heuristics.

5.2 Linear relaxation guided neighbourhood search heuristics

5.2.1 *LPNS* heuristic

Linear programming based neighbourhood search (LPNS) heuristic is a preliminary construction heuristic proposed to find a good feasible solution. Our heuristic first solves a linear programming (LP) relaxation of the MILP. Solving an LP is usually much faster than MILP. If the LP relaxation is infeasible, MILP is also infeasible. Otherwise, we fix a large number of variables and solve a much smaller MILP. LPNS is a modification of the *Relaxation Enforced Neighbourhood Search* (RENS) heuristic introduced by Berthold (2007), and the *Relax and Fix* (RaF) heuristic implemented by Rodrigues et al. (2016) and Giavarina dos Santos et al. (2020).

According to Giavarina dos Santos et al. (2020), the RaF heuristic is effective on problems that can be divided into n sets of integer variables. For example, Rodrigues et al. (2016) define the sets of variables (to relax) based on time intervals. On the other hand, Giavarina dos Santos et al. (2020) divide the variables based on heirarchy of decisions like routing variables, cargo assignment variables and so on. The n sets of variables are disjoint sets. In every iteration, the RaF heuristic solves the MIP formulation by relaxing a set of integer variables. Solutions generated in the previous iterations are provided as initial solution to the solver. According to Giavarina dos Santos et al. (2020), some constraints are also relaxed for every iteration to reduce the problem complexity. However, as constraints are relaxed a heuristic is required to repair any infeasible solution that is generated on solving the sub-problems.

There are some similarities between the RaF heuristic implementations and LPNS because all of them solve a relaxation to provide insight for the overall problem. Additionally, similar to Giavarina dos Santos et al. (2020), we divide the decision variables into different sets based on a heirarchy of decisions. Unlike Rodrigues et al. (2016) and Giavarina dos Santos et al. (2020) we solve the linear programming (LP) relaxation, which relaxes all of the integer variables and includes all problem constraints. Additionally, instead of relaxing, we fix a subset of integer variables to reduce the complexity of the MIP formulation.

Berthold (2007) also implement a heuristic which uses the LP relaxation to reduce the problem complexity of the MIP formulation. Based on the LP solution, the MIP is solved over this restricted feasible region to generate a local optimum. Consequently, we introduce an adaption of the RENS heuristic, which solves the revised formulation over a restricted feasible region. The feasible region is restricted by updating the bounds of a subset of integer variables based on the LP solution.

The LPNS bound update rule is derived from the structural analysis of the problem. Specifically, the update rule eliminates ports (except the dummy port ($|N^P|$)) that are not visited by the chemical tanker in the LP optimal solution. Let $(\bar{l},\bar{u},\bar{z},\bar{c},\bar{w},\bar{b},\bar{r})$ be the LP relaxation of the revised formulation presented in Section 4.2. Let, $U_{kpp'} = 1$ be the upper bound on $z_{kpp'}$. Mathematically, the upper bounds are updated as follows:

Bound update Rule:

$$If \sum_{k' \in K \setminus \{0\}} \sum_{p' \in N^P \setminus \{|N^P|\}} (\bar{z}_{k'pp'} + \bar{z}_{k'p'p}) = 0,$$

$$Then \sum_{k' \in K \setminus \{0\}} \sum_{p' \in N^P \setminus \{|N^P|\}} (U_{k'pp'} + U_{k'p'p}) = 0 \qquad \forall p \in N^P \setminus \{|N^P|\}.$$

Constraint (4.5) ensures that any feasible solution of the LP relaxation will always contain discharge ports of the onboard cargoes. As a result, LPNS will terminate with at least one feasible solution if the optimal solution of the LP relaxation is found.

There are multiple reasons that make this heuristic a viable option for our problem. The primary reason being that the revised formulation(4.2) has a tighter linear relaxation than the previous formulation presented in the literature. Since its LP relaxation is closer to the convex hull of MILP feasible region, its neighbourhood should provide a reasonable starting solution.

A unique feature of the LPNS heuristic is the bound update rule that is based on the problem structure. As part of the structural analysis, we tried to fix various groups of decision variables. Once certain groups of variables were fixed, we analysed their effect on parameters like the total solution time, the number of nodes explored, and the initial relative gap. We carried out certain experiments that fixed the ports to visit (not the order in which these ports should be visited), or the entire ship route was fixed, or the cargoes to be served were fixed. Restricting

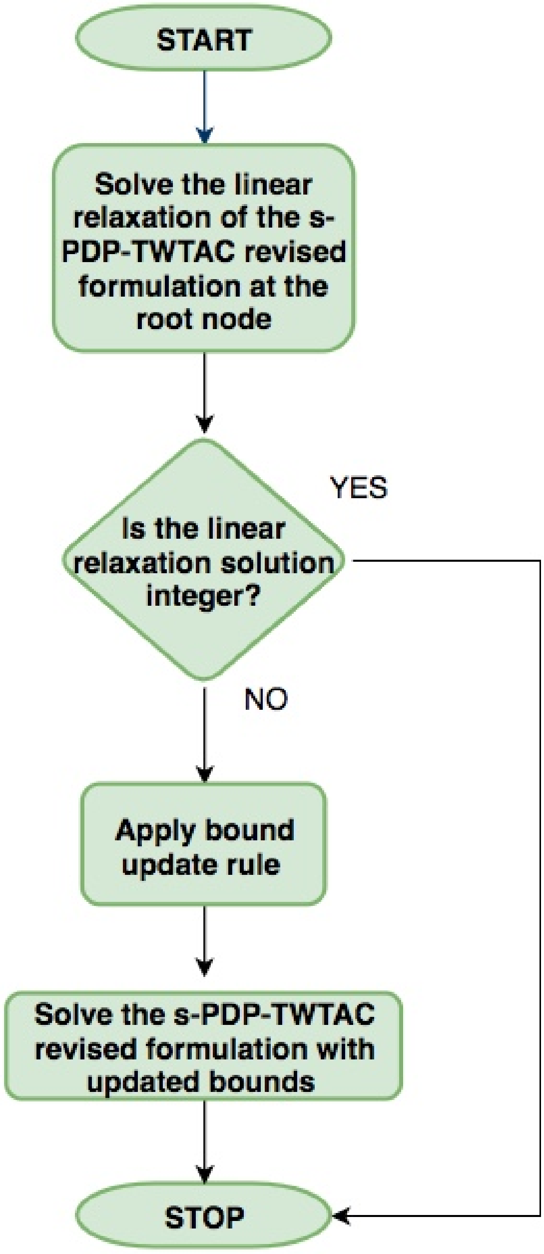

Figure 5.4: Flowchart for the important steps of the LPNS heuristic.

other decision did not lead to a substantially simplified MILP. Out of the three decision sets, fixing either the cargoes or the ship's route made the ship's moment extremely restrictive in the temporal plane. Additionally, we observed that a significant number of route defining constraints (Constraints 4.2 - 4.7) were active in the optimal basis of the linear relaxation in all of the benchmark instances.

Furthermore, we observed that restricting the feasible set of ports (not the sequence in which these ports should be visited) significantly reduced the MILP termination time. Additionally, letting MILP decide the sequence of ports increased the feasible region of the problem substantially when compared to fixing the exact route of the ship. Figure 5.4 presents the flowchart for the LPNS heuristic.

5.2.2 *LPNS*+ heuristic

The *LPNS*+ heuristic, which is an extension of the *LPNS* heuristic. The first phase of *LPNS*+ generates an initial solution using the *LPNS* heuristic. The second phase of the *LPNS*+ heuristic tries to improve the Phase 1 initial solution by using a local search strategy. Phase 2 of the *LPNS*+ heuristic is exactly same as the Phase 2 of the Heuristic, *H2*, which is described in Section 5.3.2. We will discuss the Phase 2 of both, the LPNS+ and the Heuristic, *H2* later in Section 5.3.2.

5.3 Integer relaxation guided neighbourhood search heuristics

5.3.1 Heuristic *H1*

Phase 1 of Heuristic *H1* generates an initial seed solution by solving the MILP formulation of the s-PDP-TWTAC over a restricted set of ports. Phase 2 of the heuristic tries to improve the initial solution using a local search strategy, which performs cargo insertions and port removals.

The Heuristic *H1* terminates when Phase 2 reaches the maximum number of iterations or if no improvement in the objective function value is observed for a fixed number of iterations.

The primary goal of Phase 1 is to generate a good feasible solution for the revised formulation of the s-PDP-TWTAC as quickly as possible. Our past experiments showed that fixing the routing variables ($z_{kpp'}$) significantly reduces the total solution time of the problem. In the light of this and by considering the problem definition, Phase 1 employs a simple strategy to build an initial solution. By definition, all the on-board cargoes have to be compulsorily delivered within the planning horizon. As a result, a feasible route can always be created from the discharge ports of on-board cargoes, N^O.

The initial seed solution is generated such that only the on-board cargoes are delivered. As such, in the initial seed solution we only allow cargo-compartment assignments for the onboard cargoes to be non-zero, while all the decision variables related to un-assigned cargoes are set to zero. Phase 1 of the heuristic solves the s-PDP-TWTAC MILP formulation. The optimal solution generated at the end of the first phase acts as an initial seed solution (s^0) for Phase 2.

Let s^0 be the initial solution that is generated by Phase 1. The second phase of the heuristic begins by initializing the maximum number of iterations (*max_iter*) and the present iteration ($iter$) = 1, temporary solution (s') = s^0, the best solution (s^*) = s^0 and the no-improvement counter (*no_improv*) = 0. The no-improvement counter keeps a track of the successive number of iterations during which the heuristic fails to improve s^*, where s^* is the best solution discovered by the heuristic so far. The interval between two successive calls of the *portRemove* sub-routine is defined as $port\_remove\_interval$. The temporary solution, s', is updated during each iteration, while s^* is updated only if an improved solution is found. We then invoke the *insertNewCargo* sub-routine, which tries to improve s'. However, if the *insertNewCargo* sub-routine fails to improve s', then the *portRemove* sub-routine is executed. Both, the *insertNewCargo* sub-routine and the *portRemove* sub-routines are presented in Section 5.1.1 and 5.1.2, respectively.

The Heuristic *H1* terminates if the maximum iteration limit is reached. It also terminates if *no_improv* equals 5, or if the heuristic cycles back to s^0. We present the flow chart of Heuristic *H1* in Figure 5.5. We present the experimental results that pertain to Heuristic *H1* in Section 5.4. In the next section, we discusses the second heuristic.

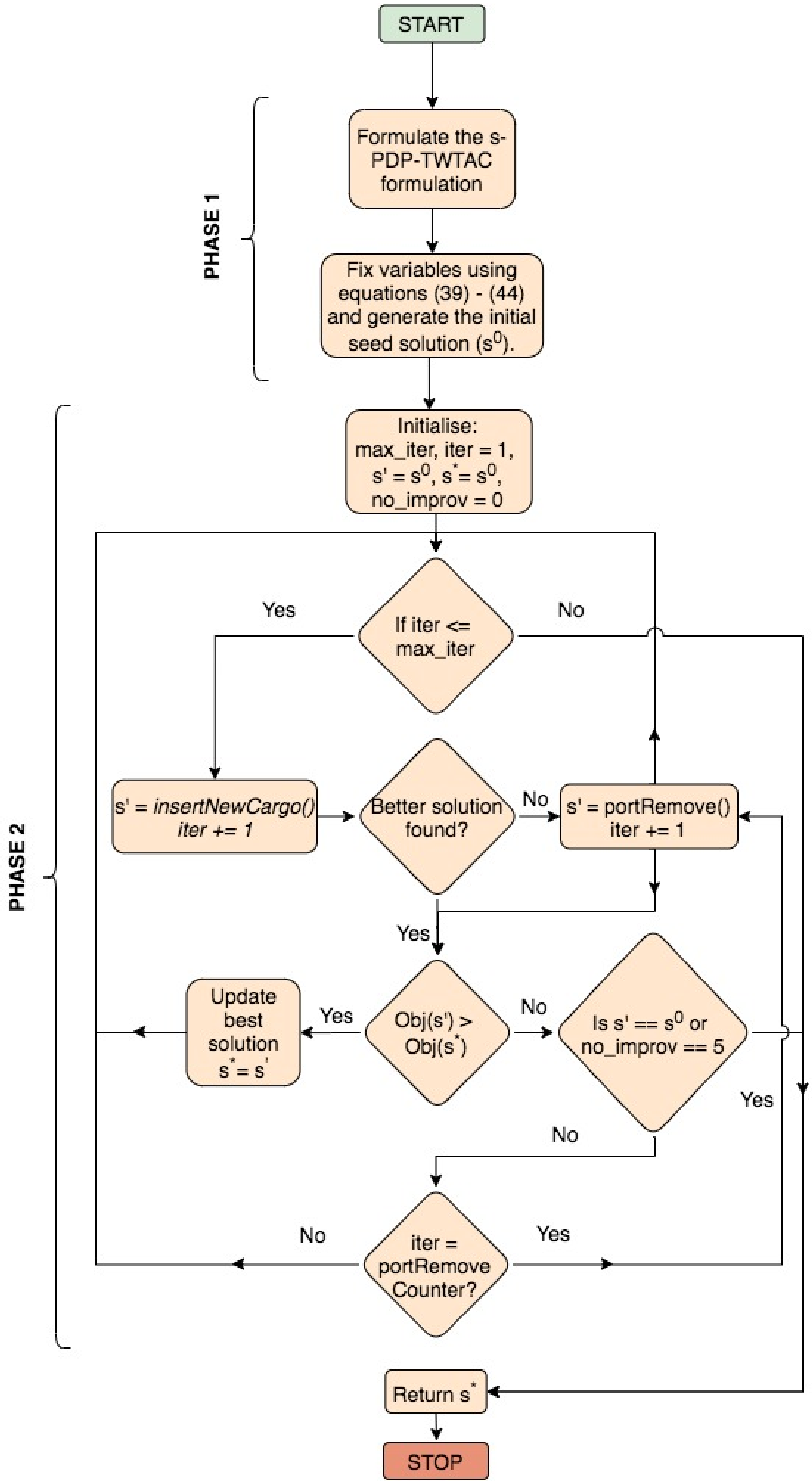

Figure 5.5: This figure illustrates the flowchart for Heuristic *H1*.

5.3.2 Heuristic *H2*

The first phase of Heuristic *H2* generates an initial solution (s^0), while the second phase tries to improve s^0 further. Recall that Phase 1 of the Heuristic *H1* uses a simple strategy to generate an initial solution. It restricts the set of feasible ports to discharge ports of the on-board cargoes, and disallows picking up of any unassigned cargo. A significantly different strategy is implemented in Heuristic *H2* to generate s^0. There are certain other secondary differences in Phase 2 of both the heuristics, which will be discussed later in this section.

Phase 1 of the Heuristic *H2* is motivated by the heuristic presented by Ladage et al. (2021). The LPNS heuristic that is proposed by Ladage et al. (2021) is a neighbourhood-search method guided by the LP relaxation of the s-PDP-TWTAC problem. The *LPNS* heuristic generates feasible solutions to the MILP formulation by solving the MILP on a restricted neighbourhood. The rule to define the neighbourhood eliminates ports (except the dummy port ($|N^P|$)) that are not visited by the chemical tanker in the LP optimal solution. Let $(\bar{l},\bar{u},\bar{z},\bar{c},\bar{w},\bar{b},\bar{r})$ be the LP relaxation of the s-PDP-TWTAC MILP formulation. Let, $U_{kpp'} = 1$ be the upper bound on $z_{kpp'}$. Mathematically, the upper bounds are updated as follows:

LPNS Heuristic update Rule:

$$\begin{aligned}
&For\ \ p \in N^P \backslash \{|N^P|\}, \\
&\quad If\ \ \bar{z}_{k'pp'} = \bar{z}_{k'p'p} = 0 \qquad \forall k' \in K\backslash\{0\}, p' \in N^P\backslash\{|N^P|\} \\
&\quad then\ \ U_{k'pp'} = U_{k'p'p} = 0 \quad \forall k' \in K\backslash\{0\}, p' \in N^P\backslash\{|N^P|\}
\end{aligned}$$

Phase 1 of the Heuristic *H2* starts by solving an MIP relaxation (not an LP relaxation) of the original MILP formulation. After several experiments, we arrived at a relaxation by dropping the chemical tanker balancing constraints, the draft constraints, the cargo-cargo incompatibility constraints, the one-cargo-per-compartment constraints, the compartment capacity constraints, and the changeover constraints. We refer to this relaxation as the relaxed MIP. The relaxed MIP is terminated after a few minutes, and the MIP feasible solution is then used to enforce the following bound update rule.

The relaxed MIP bound update rule: Let $z_{kpp'} \in \{0,1\}$ be the routing variable that takes value 1 if at the end of leg k the chemical tanker travelled from port p to p'. Similarly, let $\bar{z}_{kpp'}$

be the corresponding values obtained by solving the relaxed MIP. Let, $U_{kpp'}$ be the upper bound on $z_{kpp'}$. Mathematically, each $U_{kpp'}$ is updated as follows:

$$U_{kpp'} = \bar{z}_{kpp'} \qquad\qquad \forall k \in K \backslash \{0\}, p \in P \backslash \{|P|\}, p' \in P \backslash \{|P|\}.$$

The bound update rule uses the relaxed MIP to fix the chemical tanker's route. This significantly reduces the number of decision variables in the MILP formulation of the s-PDP-TWTAC problem. Subsequently, s^0 is generated by solving the MILP formulation for the fixed route. However, if the relaxed MIP fails to generate at least one feasible solution within the stipulated CPU time limit, we use Phase 1 of the Heuristic *H1* to generate an initial solution (s^0).

The initial solution (s^0), generated at the end of Phase 1 is improved in the second phase. The only difference between the second phase of both the heuristics is the design of the *portRemove* sub-routine. Recall that the *portRemove* sub-routine presented in Section 5.1.2 calculates the cargo-compartment arrangement based on the initial solution. This is only possible because of the absence of unassigned cargoes in s^0.

Unlike Heuristic *H1*, Heuristic *H2* Phase 1 can potentially generate an initial solution that delivers unassigned cargoes. Consequently, the *portRemove* sub-routine has to generate a feasible cargo-compartment allocation plan for the on-board cargoes which is independent of s^0. This plan is generated by solving the MILP that is defined by Equations (5.1) - (5.10) such that $j = J$ is replaced by $j \in N^O$. Additionally, K' is replaced by K'_j, where K'_j is the set of sailing legs between leg 0 and the discharge leg of cargo $j \in N^O$. In conclusion, Heuristic *H2* tries to generate an initial solution of a better quality during Phase 1. Figure 5.6 presents a flowchart of the first phase of the second heuristic. Subsequently, Phase 2 of the second heuristic tries to improve the initial solution with a local search strategy.

Although Heuristic *H1* and Heuristic *H2* perform reasonably well, they get stuck at the local optimum for many instances. A widely accepted methodology to avoid the local optimum is to generate multiple initial seed solutions. However, generating initial solutions that are of a good quality for the s-PDP-TWTAC problem is a difficult task. Our final heuristic tries to generate different initial seed solutions.

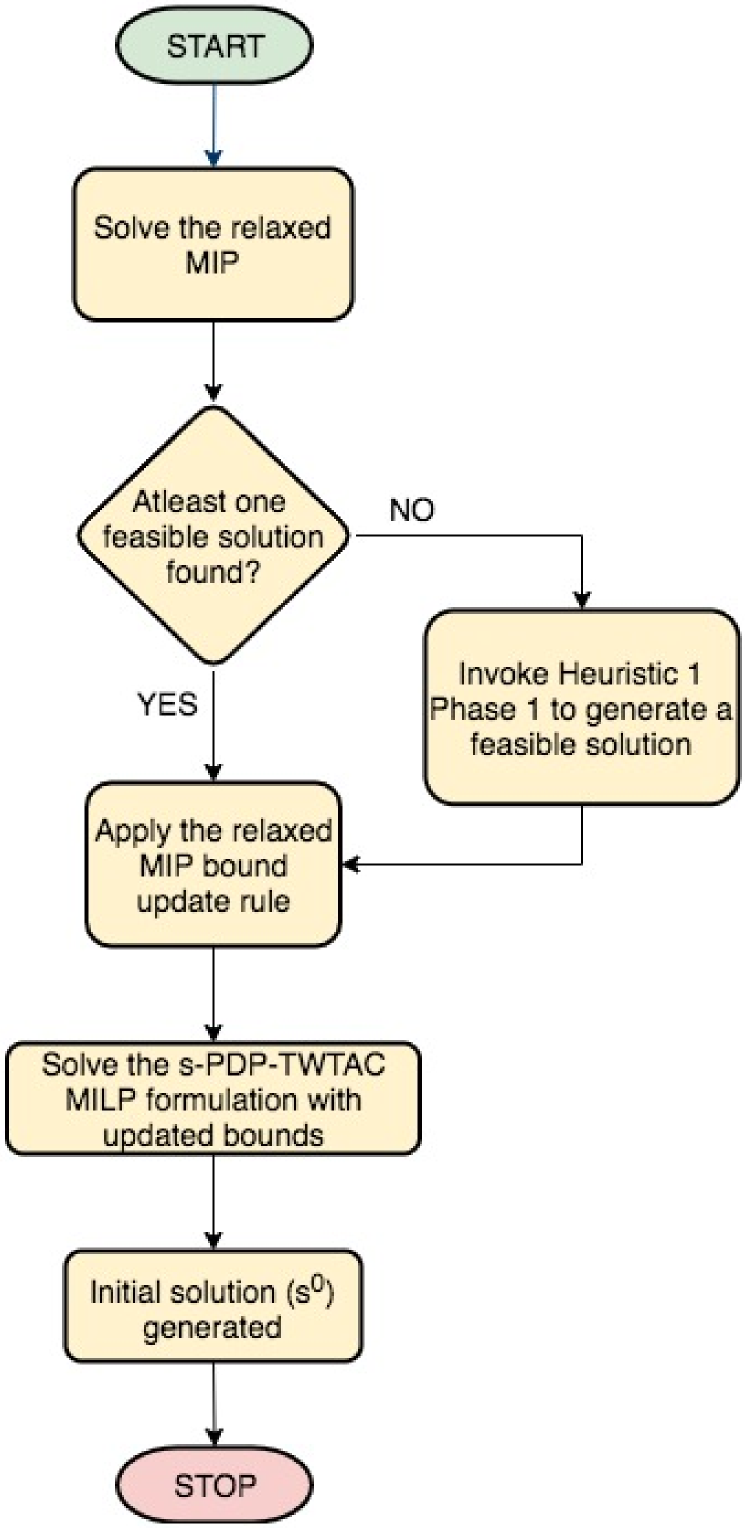

Figure 5.6: This figure illustrates the flowchart for the first phase of Heuristic *H2*. Phase 1 generates an initial solution for the heuristic.

5.3.3 Heuristic *H3*

Phase 1 of Heuristic *H3* is responsible for generating the initial solutions, while Phase 2 tries to improve it. Let the set of all initial solutions be S^0. However, unlike Heuristic *H1* and Heuristic *H2*, our third heuristic can generate multiple initial solutions, which enables the heuristic to explore a larger feasible region.

The strategy introduced to generate S^0 stems from the fact that all on-board cargoes have to be compulsorily delivered. Hence, any feasible route for the s-PDP-TWTAC problem will always include the discharge ports of the on-board cargoes (N_1^P). Therefore, we construct S^0 using $n!$ different routes (permutations) that are generated from the set, N_1^P. One promising strategy to select initial route from the set N_1^P would be to generate solutions for each initial routes, and then sort them ascending order of cost. However, this strategy is computationally too expensive. As such, to reduce complexity, we randomly select routes to generate initial solutions till the termination criteria is reached.

For a given route of the chemical tanker, we construct an initial solution, $s \in S^0$, as follows. For a given route, the discharge ports and by extension the discharge legs of onboard cargoes are already known. To simplify the generation of the solution, we enforce the condition that no unassigned cargo can be picked up. Finally, to complete the solution we generate cargo-compartment allocation plans at the end of every sailing leg by solving the MILP defined by Equations (5.1) - (5.10) for on-board cargoes.

Following the construction of S^0, we move on to Phase 2 of the heuristic. Phase 2 of the Heuristic *H3* and the Heuristic *H1* are similar. The *insertNewCargo* and the *portRemove* core sub-routines implemented in Phase 2 of the third heuristic are as described in Sections 5.1.1 and 5.1.2, respectively. However, minor modifications to these sub-routines enable the heuristic to iterate through S^0.

We begin by setting the number of iterations between two successive calls to the *portRemove* sub-routine within the solve function. As explained earlier, the heuristic starts by generating $n!$ initial routes. Let R be the set of all the initial routes. We construct an initial solution (s^0) from the first route $r \in R$. We initialise a temporary solution (s') and the best solution (s^*) with s^0.

The primary while loop terminates when there are no initial routes to construct a new s^0. The *de-*

Algorithm 3: H3(port_remove_interval)

```
generateInitialRoutes();
s^0 = constructFirstInitialSolution;
Initialise: s', s* = s^0, next_port_removal := port_remove_interval, iter := 1;
while withinTimeLimit or allInitialRoutesNotExplored do
    call destroyRepair;
    if betterObjFound then
        update Solution;
    else
        if allInitialRoutesExplored then
            return s*;
        else
            construct NewInitialSolution;
            reset ImprovementCounter;
            goto whileLoopStart;
        increment NoImprovementCounter;
    if noImprovement for 5 iters then
        if allInitialRoutesExplored then
            return s*;
        else
            construct NewInitialSolution;
            reset ImprovementCounter;
            goto whileLoopStart;
        return s*;
    if iter == next_port_removal then
        call portRemove;
        update portRemoveCounter;
return s*;
```

Algorithm 4: destroyRepair(s', iter, next_port_removal, port_remove_interval)

call insertNewCargo;

if *sameSolutionFound* **then**

  call PortRemove;

  update portRemoveCounter;

else

  update Solution;

return newSolution;

stroyRepair sub-routine updates the temporary solution s' with the help of the core sub-routines that are described in Section 5.1.1 and 5.1.2. The solution is updated only if the new solution improves the objective function, else we perform some additional checks: First, if the heuristic has cycled back to the most recent s^0 and if the set R is empty, then the heuristic terminates. If the heuristic cycles back to the most recent value of s^0 and the set $R \neq \emptyset$, we update the value of s^0 and s' and go back to the start of the main while loop. Algorithms 3 and 4 present the pseudo codes for the third heuristic and the *destroyRepair* sub-routine, respectively.

Second, the heuristic terminates if a better solution has not been discovered for five iterations and the set $R = \emptyset$. However, if the set R is non-empty, then a new initial solution is constructed and the *destroyRepair* sub-routine is applied to it. After a fixed number of iterations the solution is randomized by invoking the *portRemove* sub-routine. The heuristic terminates if there are no new initial routes to be explored.

5.3.4 GRASP heuristic

Algorithm 5: GRASPNewCargo(data, s)

```
cargo_by_category := generatePotentialCargoes;
for cargo in cargo_by_category do
    createPositionTuple(cargo);
    populate(position_list);
if isNotEmpty(position_list) then
    flipBiasedCoin with p(H)=customProb;
    if HEADS then
        sortByCustomPDF(position_list);
    else
        sortByProfit(position_list);
    for position in position_list do
        if cargoCompartmentAllocationExists then
            s′ = generateNewSchedule(position, s);
            return s′;
else
    return s;
```

In this section, we introduce a greedy random adaptive search procedure (GRASP) heuristic, which is a modification of the Heuristic, *H3*. This modification enables GRASP to scan the search space more effectively. Recall that the improvements between the solutions of Phase 1 and Phase 2 for all the heuristics mentioned above is dependent on the sequence in which new cargo positions are inserted into the existing solution. The *insertNewCargo* sub-routine that is employed by Heuristics *H1*, *H2*, *H3*, and *LPNS+* inserts the cargo positions in a greedy deterministic manner into the solution of Phase 1; this enables quick solutions that are also of a high quality. However, the completely greedy deterministic method of inserting cargo positions restricts the solution search space that is explored by the heuristics. Additionally, a greedy deterministic sequence of cargo additions has a low probability of identifying the optimal solution.

Keeping this in mind, we propose a GRASP heuristic that determines the sequence of cargo insertions in a greedy random manner. This ensures that even for a single Phase 1 solution, the heuristic can discover difference sequences in which to insert different cargo positions. Additionally, the GRASP heuristic has a non-zero probability of discovering the optimal solution for any instance. The main difference between GRASP and other heuristics described above lies in the implementation of the *insertNewCargo* sub-routine. To avoid ambiguity, we refer to the cargo insertion function of the GRASP heuristic as the *GRASPNewCargo* sub-routine.

Similar to the *insertNewCargo* sub-routine, we generate all the feasible cargo positions in the *GRASPNewCargo* sub-routine as well. However, a biased coin flip decides whether the cargo positions are sorted in a greedy deterministic way or by using a custom probability distribution function (PDF). The custom PDF is defined such that a higher profit cargo position has a higher probability of being placed first in the sorted list of cargo positions. The *GRASPNewCargo* sub-routine ends by inserting the first cargo position from the sorted list, which generates a feasible cargo-compartment plan. Additionally, let p(H) be the probability of the biased coin turning heads. Setting p(H) = 0 is equivalent to Heuristic, *H3*, while p(H) = 1 tries to insert every new cargo in a greedy random way. Algorithm 5 outlines the pseudocode of the *GRASPNewCargo* sub-routine.

Heuristic, H3, terminates if all the initial solutions (or initial routes) are explored exactly once because each initial route leads to the same final feasible solution. On the other hand, the GRASP heuristic continues to cycle through the initial routes until the time limit is reached.

Table 5.2 lists the similarities and differences between heuristics *LPNS+*, *H1*, *H2*, and *H3*. Column *Criteria* lists the different parameters that are used to compare the four different heuristics. The first criterion specifies the maximum number of solutions that can be generated and passed to the second phase of the heuristic. The next criterion *Methodology to obtain initial solution* specifies the optimization problem solved to obtain the initial solution(s).

The rows, three to seven, specify the different parameters that are related to Phase 2. The criteria *Cargo-compartment plan generation* presents the method that is used to obtain cargo-compartment plans during Phase 2. Heuristics *LPNS+* and *H2* solve an MILP, while *H1* and *H3* generate allocations from the Phase 1 solution. The succeeding criteria specify the number of iterations between two successive calls to the *portRemove* sub-routine, the number of ports removed during every *portRemove* sub-routine call, and the maximum number of feasible solutions for the s-PDP-TWTAC that can be generated during the second phase of the heuristic.

Table 5.2: Comparison table highlighting the major similarities and differences between the LPNS+, H1, H2, H3, and GRASP heuristics.

	Comparison table summarising the similarities and differences between different heuristics					
		LPNS+	**H1**	**H2**	**H3**	**GRASP**
Phase 1 (P1)	**Possible solutions generated at end of Phase 1**	1	1	1	>= 1	>= 1
	Methodology to generate the initial solution	LP followed by restricted MILP of the s-PDP-TWTAC	Restricted MILP of the s-PDP-TWTAC	Relaxed MILP followed by restricted MILP of the s-PDP-TWTAC	Multiple restricted MILP of the s-PDP-TWTAC	Multiple restricted MILP of the s-PDP-TWTAC
Phase 2 (P2)	**Cargo-compartment plan generation**	MILP	Generated using P1 solution	MILP	Generated using P1 solution	Generated using P1 solution
	Cargo added per iteration	1	1	1	1	1
	Cargo addition methodology	Deterministic	Deterministic	Deterministic	Deterministic	Greedy Random
	Port remove interval	Equal across heuristics				
	Port removed per interval	1	1	1	1	1
	Port removal methodology	Random	Random	Random	Random	Random
	Feasible solution generated for the s-PDP-TWTAC	>= 1	>= 1	>= 1	>= 1	>= 1

All the heuristics that are discussed in this section start by generating an initial solution(s). The heuristics then try to improve it with a local search strategy (Phase 2). Phase 2 tries to improve the initial solution by either inserting a new cargo or removing a port from the chemical tanker's route. In the next section, we present the empirical results of our experiments.

5.4 Computational results

5.4.1 Empirical study: Cplex vs. LPNS heuristic

All the 200 test instances were used to perform this empirical computational study. The primary goal of this experiment is to present the results of solving the s-PDP-TWTAC revised formulation with the LPNS heuristic. The LPNS heuristic is run for a CPU time of 86,400 seconds. LPNS heuristic also terminates if the Cplex reported relative gap (%) is less than 0.01

%. We also make some preliminary comparisons between the Cplex run and the heuristic run. Further, we discuss the sensitivity of some of the performance parameters with respect to the input parameters used to generate the test instances. The performance parameters include the Gap (%) and the total CPU time (sec) of both the Cplex run and the LPNS heuristic run. The comparison is aimed at understanding why the MILP solver takes a long time. A solver might be slow because it is not able to find a good solution early on. It might also be slow because it is unable to prove that the solution is optimal. Even though the LPNS heuristic finds better quality solutions faster for several instances, the Cplex run can solve the problem exactly for some others. Further, the Cplex run also generates an upper bound for the overall problem that the LPNS heuristic does not.

We report some of the solution statistics in Tables A.2 and A.1. As explained earlier, the first four columns of Tables A.2 and A.1 give the instance set characteristics and the chemical tanker characteristics, respectively. In Table A.2, Column *Instance per set* gives the number of test instances (out of 200) that belong to each instance set. Similarly, in Table A.1, Column *instance per ship* gives the number of test instances (out of 200) for every chemical tanker. In both tables, Columns *Avg. variables* to *Avg. Heur CPU time* tabulate the corresponding solution statistics. Table A.2 presents average solution statistics for every instance set. Likewise, Table A.1 report solution statistics averaged for each of the chemical tankers.

We record four performance parameters, the Heur Gap (%), the Cplex Gap (%), the Cplex CPU time (sec) and the Heur CPU time (sec). The *Heur Gap (%)* presents the percentage difference between the upper bound obtained during the Cplex run, and the lower bound obtained from the heuristic run. Moreover, the *Cplex Gap (%)* is the percentage difference between the Cplex upper and lower bounds. Both the gaps (%) are with respect to the absolute values of the upper bounds generated by Cplex. Lower the gap better is the performance of the run. The *Cplex Gap (%)* and the *Heur Gap (%)* for a given instance are calculated as follows:

$$\text{Cplex Gap } (\%) = \frac{\text{Cplex upper bound - Cplex best objective}}{\text{Abs(Cplex upper bound)}} \times 100$$

$$\text{Heur Gap } (\%) = \frac{\text{Cplex upper bound - LPNS heuristic best objective}}{\text{Abs(Cplex upper bound)}} \times 100$$

Subsequently, the average performance measures reported in Tables A.2 and A.1 are calculated as follows.

$$\text{Avg. Cplex Gap (\%)} = \frac{\text{Total Cplex Gap (\%) for } n \text{ instances}}{n}$$

$$\text{Avg. Heur Gap (\%)} = \frac{\text{Total Heur Gap (\%) for } n \text{ instances}}{n}$$

$$\text{Avg. Cplex CPU time (sec)} = \frac{\text{Total Cplex run CPU time (sec) for } n \text{ instances}}{n}$$

$$\text{Avg. Heur CPU time (sec)} = \frac{\text{LPNS heuristic CPU time (sec) for } n \text{ instances}}{n}$$

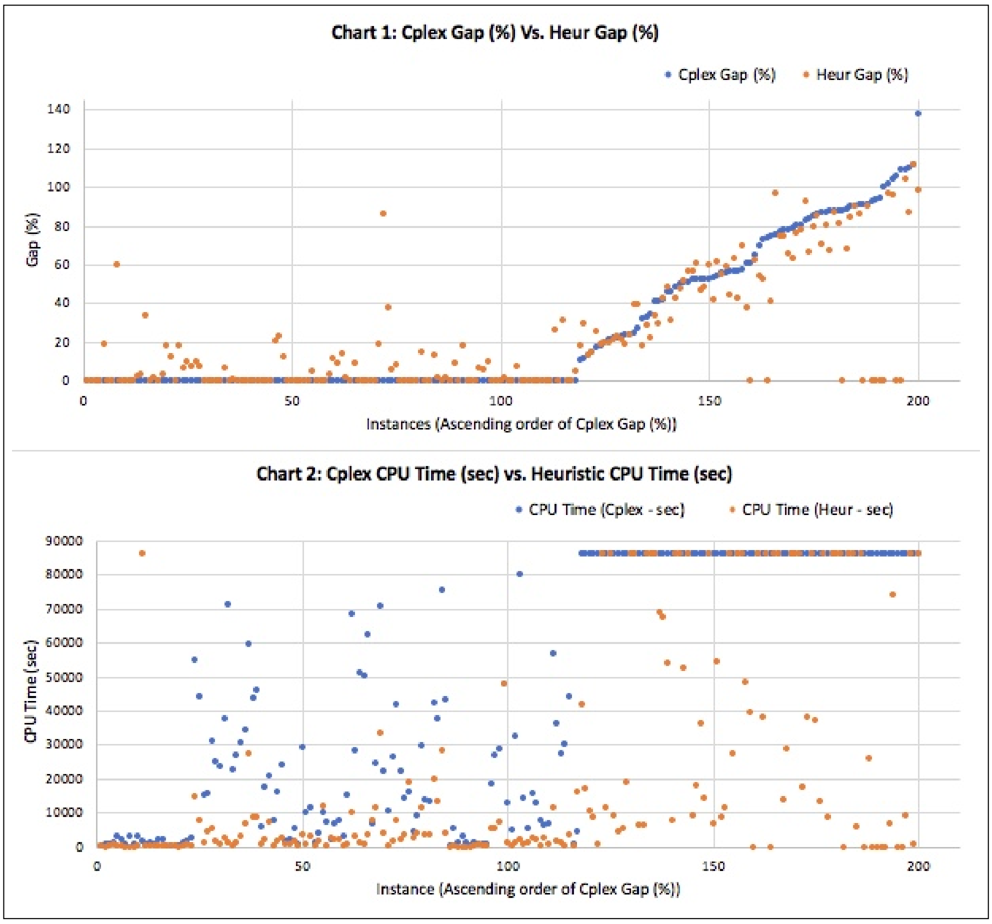

Figure 5.7: Comparison of the Gap (%) and the CPU time (sec) of the Cplex and LPNS heuristic runs as the problem difficulty (Cplex Gap (%)) increases.

Table A.2 reports n in Column *Instances per set*, while Table A.1 reports it in Column *Instances per ship*. The heuristic run terminates with either no integer solution (Phase I OOM), local optimal solution (Local Optimal) or a feasible integer heuristic solution (Phase II OOM or time limit). The local optimal solution is the best possible solution that can be generated by the heuristic without hitting the time limit or the memory limit. It may not be the optimal solution of the REV formulation.

Out of the 200 instances, our heuristic terminated due to local optimality in 157 instances, due to feasibility and time limit (Phase II time limit or OOM) in 33 instances, and due to Phase I OOM issue in 10 instances. A run terminates with *Phase I OOM* if memory is exhausted while solving the linear relaxation of the problem. On the other hand, the Phase II time limit or OOM termination occurs if the Phase II MILP does not terminate within the time limit or the solver runs out of memory, respectively.

Figure 5.7 presents the Gap (%) and CPU time (sec) of both the runs. The horizontal axis plots instances in increasing order of Cplex Gap (%). The Gap (%) of both the runs are plotted in Chart 1, while the second chart plots the CPU time (sec) for both runs. We classify the 200 test instances into two sets. Set I (118 instances) includes all the instances with Cplex Gap (%) less than 1 %, while Set II (82 instances) includes all the instances with Cplex Gap (%) greater than 1 %.

The heuristic terminated with a solution equivalent to the Cplex optimal solution for 63 instances, which are a subset of Set I. For these 63 instances, the total solution time reduced by 71.62 %. Within Set I, the Cplex run terminated with a lower Gap (%) when compared to the heuristic run for 54 instances. For the instances in Set I, the total solution time of the heuristic run increased for 6 instances with an average of 786.67 %, while it decreased for 112 instances with an average of 79.87 %, when compared to the Cplex run.

Within Set II, the heuristic could not find a solution for 10 instances. For rest of the 72 instances (within Set II), compared to the Cplex run, the heuristic run terminated with a better, same and worst lower bound for 49, 3 and 20 instances, respectively. For Set II, an average Cplex gap of 59.95 % was observed. In comparison, the heuristic run resulted in an average gap (%) of 55.74 %. Moreover, the CPU time (sec) for the heuristic run decreased by 41.75 % when compared to the Cplex run for the instances in Set II. In summary, Figure 5.7 shows that for instances in Set I, Cplex finds better quality solutions than the heuristic. However, as the Cplex Gap (%) increases the heuristic consistently finds better quality solutions compared to Cplex. Moreover, for majority of the test instances, the heuristic run terminates faster than the Cplex run.

Further, we discuss the sensitivity of the performance parameters, namely; the Gap (%) and the total CPU time of the LPNS heuristic runs related to the different input parameters. We

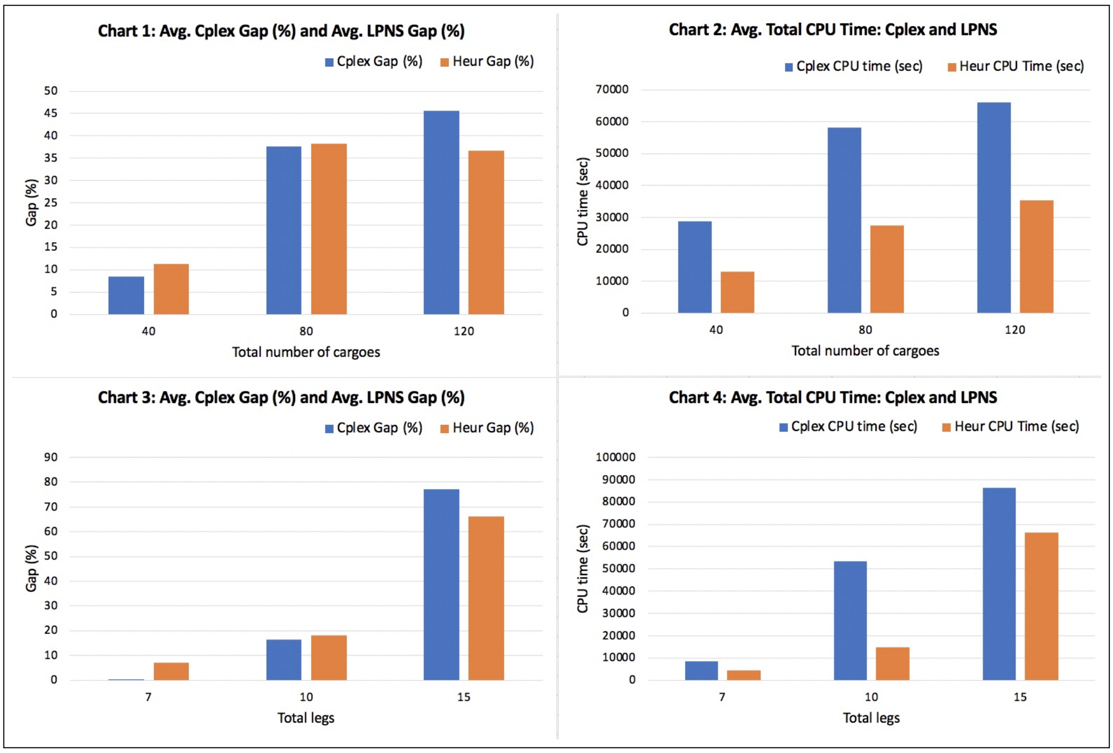

Figure 5.8: The effect of the total number of cargoes and the maximum number of sailing legs on the average performance parameters of the Cplex and LPNS heuristic runs.

also include the sensitivity results from Chapter 4 for comparison purposes. The Gap (%) of both the runs is calculated with respect to the upper bound reported by Cplex. The primary input parameters considered for this study are the total number of cargoes, the total number of ports, the maximum number of sailing legs, and the number of discharge ports of the onboard cargoes. Moreover, some secondary input parameters like the total number of compartments, the ship speed, the draft constant, the fuel cost, the time charter cost, and the average compartment volume were also considered during this study.

Analysis using multiple linear regression was performed to explore the effects of the input parameters on the performance parameters. Some of the primary input parameters significantly affect the performance parameters. Figures 5.8 and 5.9 help us illustrate this claim. However, the performance parameters seem to be insensitive to the secondary input parameters. Figures 5.8 and 5.9 classify the test instances into different categories based on the input parameters. The vertical axis in these figures presents the average of the performance parameters. For example, the first chart in Figure 5.8 differentiates the test instances based on the total number of cargoes on the horizontal axis. Similarly, the vertical axis presents the average Gap (%).

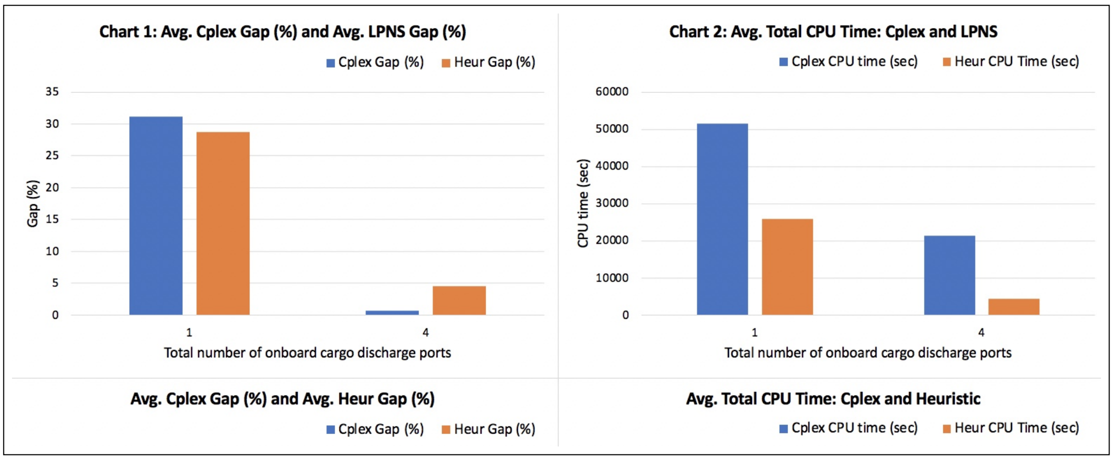

Figure 5.9: The effect of the total number of onboard discharge ports on the average performance parameters of the Cplex and LPNS heuristic runs.

In Figure 5.8, Chart 3 and 4 show that both the average solution quality and the average total CPU time worsen with the increase in the maximum number of sailing legs. Similarly, both the Cplex performance parameters deteriorate with the increase in the maximum number of legs. Additionally, even though the average total CPU time for the heuristic run worsens with the total number of cargoes, the solution quality does not.

Charts 1 and 2 in Figure 5.9 illustrate that the increase in the total number of onboard cargo discharge ports significantly improves the Gap (%) and the total CPU time related to both the runs. This effect is correct because the total number of onboard cargo discharge ports reduces the flexibility of the route of the chemical tanker. As per the problem definition, all onboard cargoes must be delivered. Consequently, their corresponding discharge ports have to be visited. Thus, a higher number of different discharge ports of onboard cargoes reduces the number of new ports on the route of the ship. This reduces the feasible region of the problem. Additionally, our preliminary sensitivity analysis shows that the performance parameters were not affected by the number of ports or the number of compartments. We discuss the experimental results related to Heuristic *H1* (H1), Heuristic *H2* (H2), and Heuristic *H3* (H3) in Section 5.4.2.

5.4.2 Empirical study of Heuristic *H1*, *H2* and *H3*

We now discuss experiments related to the Heuristic, *H1* (H1); Heuristic, *H2* (H2); and Heuristic, *H3* (H3). We focus our discussions on the 173/200 instances for which an MILP solver

Table 5.3: This table presents the termination status, Avg. Obj diff (%), average computation time, average CPU time for phase 1 and phase 2, and improvements in solution quality related to *H1*, *H2* and *H3*

	Parameters	Phase 1 status	Phase 2 status	Avg. Obj Diff (%)	Avg. total CPU time (sec)	CPU time (Phase 1) (%)	CPU time (Phase 2) (%)	Avg. Total Improvements	Min Total Improvements	Max Total Improvements
Number of Instances	**H1**	14	159	-330.36	8.97	72.64	27.36	4	1	9
	H2	94	79	-324.72	184.56	96.7	3.3	2	1	10
	H3	0	173	-333.42	28.24	9.11	90.89	4	1	14

(CPLEX) did not find an optimal solution. For these 173 instances, Table 5.3 tabulates some key observations related to the three heuristics. Columns *Phase 1 status* and *Phase 2 status* denote the number of instances that terminate with *P1* and *P2* status, respectively. Columns *Avg. Obj Diff (%)* and *Avg. total CPU time* tabulate average objective difference (%) and the total average CPU time for each of the heuristic. The *Avg. Obj Diff (%)* is calculated as follows.

$$\text{Avg. Obj Diff (\%)} = Average(\frac{\text{(CPLEX\_Obj - Heur\_Obj)}}{ABS(\text{CPLEX\_Obj})} \times 100)$$

The next column presents the average total solution CPU time in seconds for each heuristic. We also classify the instances into different CPU time buckets, as shown in Figure 5.10. The next columns, *CPU time (Phase 1) (%)* and *CPU time (Phase 2) (%)* give the percentage of time spent in each phase during each of the heuristic runs. The last three columns show the average, minimum, and maximum total improvements, successively.

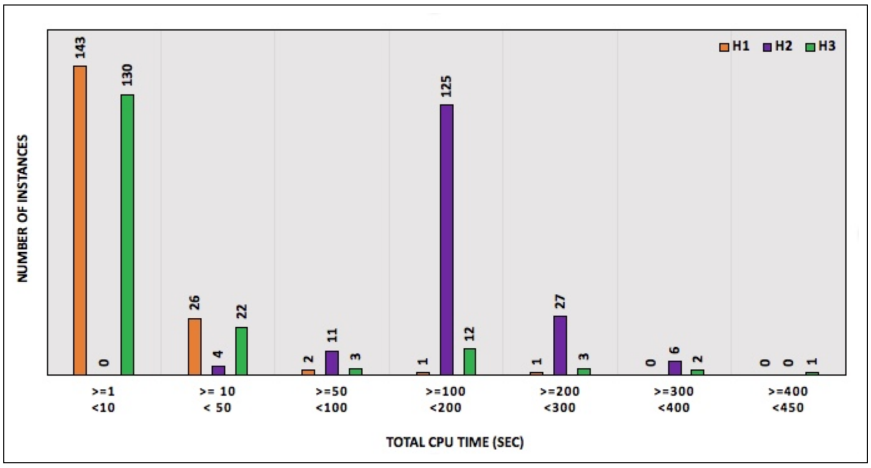

Figure 5.10: This figure shows the total CPU time required by Heuristics H1, H2, and H3 to solve the test instances.

We observed that Phase 2 of the first Heuristic was able to improve Phase 1 solution for 159/173 instances. *H2* found the best solution during Phase 1 for 94/173 instances. Similarly, Phase 2 of

H3 improves Phase 1 solution for all 173 instances. Figure 5.10 and Table 5.3 report that *H1* is the quickest with an average CPU time of 8.97 seconds. Moreover, *H2* and *H3* terminate with an average CPU time of 184.56 seconds and 28.24 seconds, respectively. *H3* spends the least total CPU time (%) to generate the initial solutions, while *H2* spends 96.7 % of its total CPU time generating the initial solution. All the heuristics find at least 1 feasible solution. Thus, the minimum total improvement is 1. Moreover, on average, *H2* required the least number of improvements (2) to reach the best solution.

Further analysing Figure 5.11 shows that the first phase of *H2* generates better quality initial solutions for 129/173 instances when compared to Phase 1 of both *H1* and *H3*. However, Phase 1 of *H2* is computationally much more expensive than Phase 1 of *H1* and *H3*. As *H3* generates multiple feasible solutions, we consider the first initial solution generated by *H3* for comparison. Figure 5.11 also shows the improvement between Phase 1 and Phase 2 of each of the Heuristic *H1*, *H2*, and *H3*. Figure 5.11 indicates that the solution quality of *H1* and *H3* is similar. As such, we performed the two sample Wilcoxon paired test, which empirically proves that *H3* generates statistically significant (p-value = 0.0045) better quality solution than *H1*.

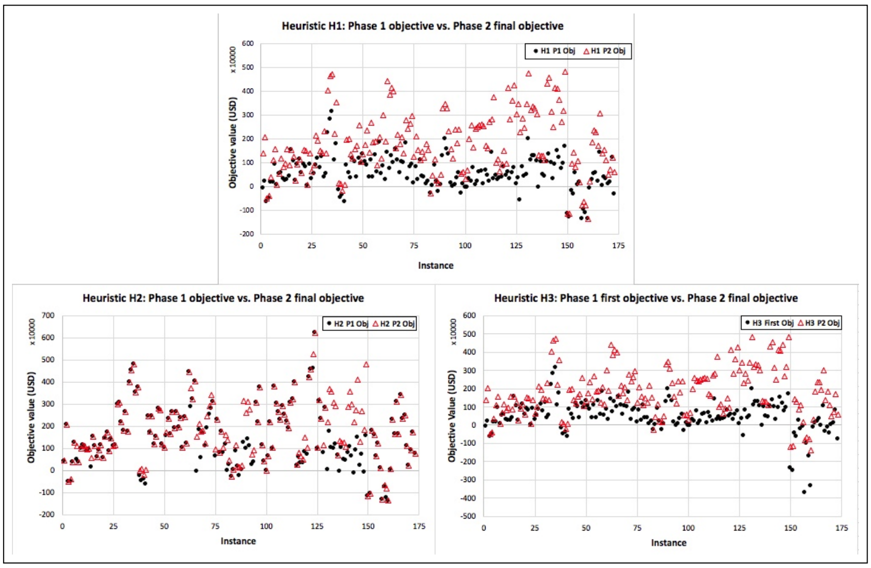

Figure 5.11: This figure shows the solution improvement between Phase 1 and Phase 2 of Heuristics H1, H2 and H3

Table A.3 presented in appendix tabulates all the experimental findings on *H1*, *H2*, and *H3* runs for each of the 200 test instances. It shows different heuristic related parameters like Phase 1 CPU time, total CPU time, objective function value of the first solution found, best solution objective value, and total solution improvements achieved by each heuristic for each of the 200 instances. In the next section, we compare the three heuristics (*H1*, *H2*, *H3*) to an MILP solver, the *GRASP* heuristic and the *LPNS+* heuristic run.

5.4.3 Comparison of *H1*, *H2*, *H3* and *GRASP* to an MILP solver and *LPNS+* Heuristic runs

We combine the results obtained by sequentially solving the three heuristics (*H1*, *H2*, *H3*), and refer to this combination heuristic as *H1_H2_H3*. We present a study comparing the outcomes of *H1*, *H2*, *H3*, *H1_H2_H3* and GRASP with MILP solver (IBM CPLEX) run and the *LPNS+* heuristic run. The CPLEX runs results are obtained by solving the s-PDP-TWTAC MILP formulation using CPLEX 12.7.1. The CPLEX run, the *GRASP* heuristic and the *LPNS+* heuristic run have a CPU time limit of 1800 secs.

Table A.4 summarizes the results for the CPLEX run, the *LPNS+* heuristic run, the *GRASP* heuristic run and the *H1_H2_H3*. Column *CPLEX Status* indicates the problem status reported by CPLEX with a CPU time limit of 1800 seconds. The succeeding three columns, *CPLEX CPU Time*, *CPLEX Obj*, and *CPLEX Gap (%)* contain the total CPU time, the objective function value, and the relative gap (%) reported by CPLEX. Columns *LPNS+ Heur Obj* and *LPNS+ Heur CPU Time* tabulate the objective function value and the total CPU time for the *LPNS+* heuristic. Similarly, the succeeding two columns present the GRASP objective function value and the GRASP objective difference (%). The objective difference (%) for a heuristic is calculated as follows:

$$\text{Obj Diff (\%)} = \frac{\text{(CPLEX\_Obj - Heur\_Obj)}}{ABS\text{(CPLEX\_Obj)}} \times 100$$

The next column, *H1_H2_H3 Status*, highlights the contribution of each of the heuristic for the combination heuristic *H1_H2_H3*. For example, the *H1_H2_H3 Status* for Instance 1 is H2 Phase 1. This denotes that after sequentially running all three heuristics, the best solution was the one obtained at the end of *H2* Phase 1. The values in Column *H1_H2_H3 CPU Time* are calculated by adding the total CPU times of *H1*, *H2* and *H3*. The following column, *H1_H2_H3 Obj* reports the maximum of the best solution objectives obtained from Heuristics *H1*, *H2* and *H3*. The final column, *H1_H2_H3 Obj Diff (%)* gives a comparison between the best solutions obtained from the CPLEX run and *H1_H2_H3* run.

After 1800 secs of CPU time, CPLEX terminated with an optimal solution for 27 instances and a feasible solution for 156 instances. Moreover, for 17 instances, CPLEX could not find a feasible solution within the allotted time limit. CPLEX reported an average relative gap of 8028 % for the 183 instances for which it found at least one feasible solution. For these 183 instances, CPLEX generated a solution within 1 % and within 100 % relative gap for 30 and 79 instances, respectively.

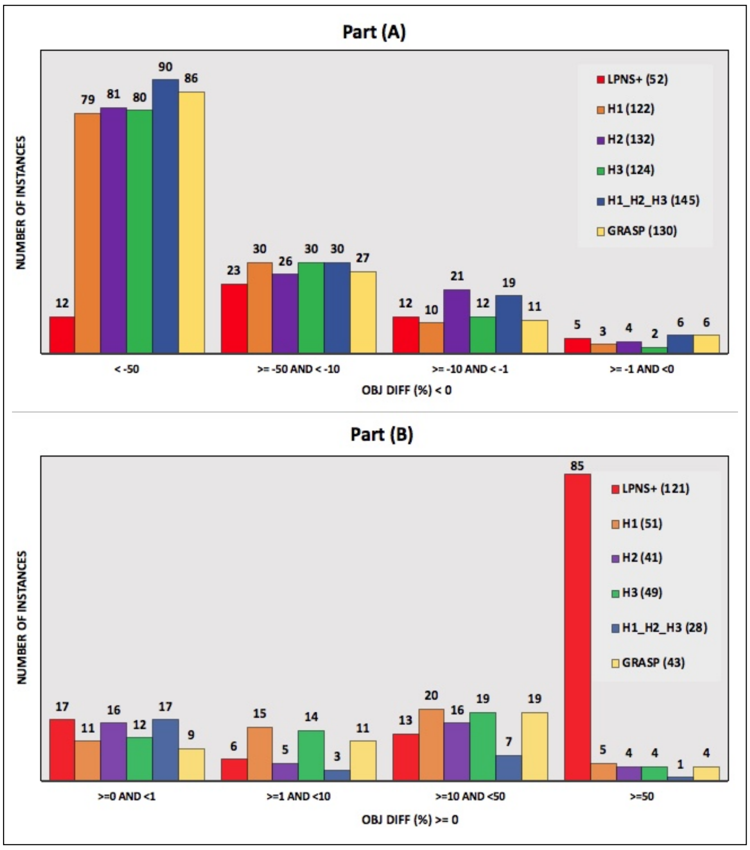

Figure 5.12: Number of instances sorted by Obj Diff (%). A negative *Obj Diff (%)* signifies that a better objective value was obtained when compared to the CPLEX runs.

Next, we compare the solutions generated by *H1*, *H2*, *H3*, *H1_H2_H3*, *LPNS+* and *GRASP* heuristics. To establish a standard scale, we make comparisons with respect to the best solution generated by CPLEX. Additionally, we focus this study on the same 173 instances for which CPLEX couldn't find the optimal solution within the time limit.

Figure 5.12 classifies the 173 instances in two parts based on Obj Diff (%). The Obj Diff (%) calculates the difference (%) between the best solution generated by CPLEX and each of the heuristics. For a given instance, an Obj Diff (%) of less than zero indicates that a heuristic found a better feasible solution than CPLEX. Part (A) presents all the instances with Obj Diff (%) $<$ 0, while part (B) presents all the instances with Obj Diff (%) ≥ 0. The numbers in the legend denote the total number of instances where a heuristic found a better (Figure 5.12A) or worse (Figure 5.12B) solution.

Heuristic *H1_H2_H3* finds better quality feasible solutions than CPLEX in 145 instances, which is more than any other heuristics. Additionally, our experiments using the two sample Wilcoxon paired test show that *H1_H2_H3* generate statistically significant better quality solutions than the CPLEX solver as well as the other heuristics discussed in this article. Heuristics *H1*, *H2* and *H3* also perform reasonably well by finding a better quality solution than CPLEX in 122, 132, and 124 instances, respectively. Moreover, the GRASP heuristic finds a better solution than CPLEX in 130 instances.

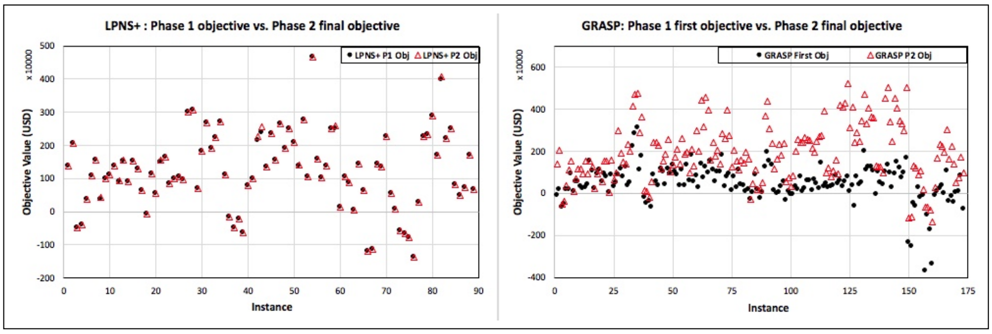

Figure 5.13: Effect of Phase 2 on the Phase 1 solutions of the LPNS+ and GRASP heuristics

The GRASP heuristic finds better quality solutions than the *H1_H2_H3* heuristic for 54/200 instances. As shown in Figure 5.12, the *LPNS+* heuristic is the worst performing heuristic for the assumed time limit. We also observe in Figure 5.13 that the Phase 2 local search does not significantly improve the Phase 1 LPNS+ solution. However, the Phase 2 of the GRASP heuristic improves the Phase 1 solution significantly.

The CPLEX run and the *LPNS+* heuristic terminate with an average total solution time of 1664 seconds and 1384 seconds. The GRASP heuristic continues to explore the search space till the given time limit of 1800 seconds. The average total solution time for Heuristic *H1* is the lowest with 8 seconds. Similarly, Heuristic *H2* and *H3* terminate with an average total solution time of 166 seconds and 30 seconds, respectively.

We now study the effects of different input parameters on the performance parameters of the

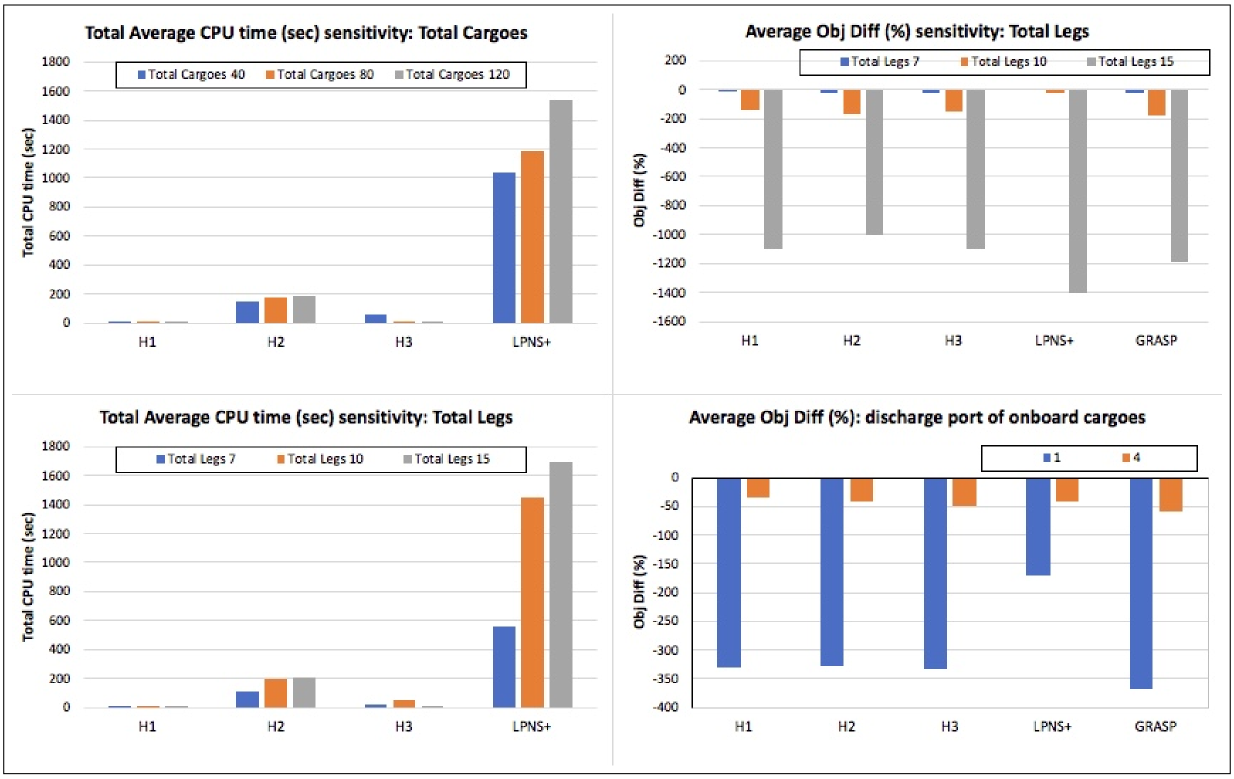

Figure 5.14: Effect of input parameters on the average total CPU time and the average Obj Diff (%) of different heuristics

heuristics, namely, the average total CPU time (sec) and the average Obj Diff (%). A sensitivity analysis was carried out with respect to different input parameters such as total cargoes, total sailing legs, total discharge ports of on-board cargoes, and total compartments. We observe that the average total CPU time of the *LPNS+* heuristic is significantly affected by the total cargoes and the total sailing legs. The average total CPU time of Heuristic *H2* also increases with the increase in the total number of cargoes. However, the average total CPU time of the Heuristics, *H1* and *H3*, does not show any observable change upon increase in the total cargoes and the total sailing legs.

Further, we see that the average Obj Diff (%) of all neighbourhood search heuristics decreases with the increase in the total sailing legs, and increases with the increase in the number of discharge ports of on-board cargoes. We should note that the size of the feasible region of the MILP increases with the increase in the total number of sailing legs, and the decrease in the discharge ports of on-board cargoes. In the light of this, a possible conclusion is that, as the size of the feasible region increases, it is difficult for an MILP solver to produce solutions that are of a good quality, while the heuristics consistently find solutions of a good quality. Other input parameters did not show any observable effect on the performance parameters.

This chapter presented six heuristics developed to tackle the s-PDP-TWTAC problem. Detailed discussed around the methodology, and construction of these heuristics was also presented. Moreover, a empirical study was performed to verify the performance of these heuristics. The next chapter will divert the discussion towards the multi-period cargo assignment problem (mp-CAP), and discuss two frameworks adopted to solve the mp-CAP.

This page was intentionally left blank.

Chapter 6

The multi-period cargo assignment problem (mp-CAP)

6.1 Problem description

The presence of multiple compartments provides added flexibility to transport multiple chemicals. However, it also makes the problem more complicated. In the literature, this specific problem of assigning cargoes to compartments during a single sailing leg is called the cargo-assignment problem (CAP). CAP generates a feasible plan for a given set of cargoes and compartments. The cargo-compartment plan must adhere to compatibility, stability, and compartment capacity constraints. Some other modelling constraints like at most one cargo can be assigned to a compartment, cargo has to be assigned to at least one compartment, and the total weight of cargo distributed in all compartments should be equal to the total available weight of the cargo, need to be respected. Some recent publications that discuss the chemical tanker scheduling problem while addressing the difficulties originating due to the cargo-assignment as-

[0] Contents of Chapter 6 have been submitted to, and are under review by Springer as a book chapter.

pect of the problem are Neo et al. (2006), Hvattum et al. (2009), Wang et al. (2018), Giavarina dos Santos et al. (2020), and Ladage et al. (2021).

Here is a small example illustrating the mp-CAP. Assume that the chemical tanker (with four compartments) visits three ports on its route and transports four chemicals during its voyage. Cargo 1 remains on the tanker for all three ports. Cargo 2 is loaded at Port 1 and dropped off at Port 3, while Cargo 3 is loaded at Port 2 and remains on the tanker at Port 3. Cargo 4 is loaded at Port 2 and delivered at Port 3. Additionally, Cargo 1 and Cargo 4 are incompatible with each other, while Cargo 2 and Cargo 3 are incompatible. Assume that incompatible cargoes cannot be stored in neighbouring (vertically or horizontally adjacent) compartments.

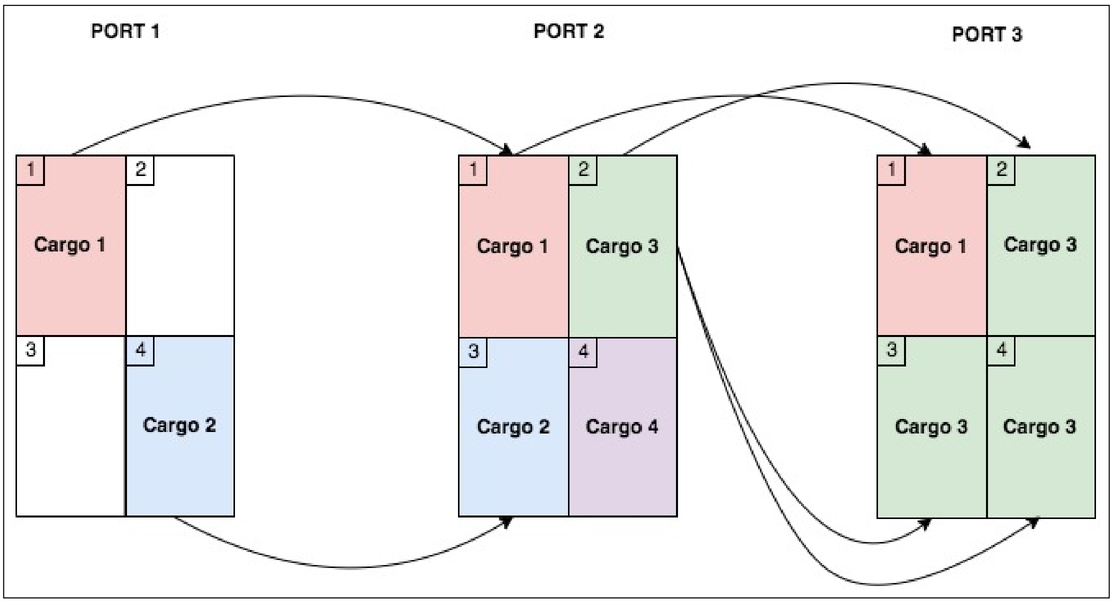

Figure 6.1: A toy size instance, and a feasible solution of the multi-period cargo-assignment problem (mp-CAP).

Figure (6.1) portrays a feasible solution generated by the mp-CAP. Cargo 1 remains assigned to Compartment 1 through the tanker's voyage. At the first port, Cargo 2 was stored in Compartment 4. Considering the incompatibilities among the cargoes, at Port 2, Cargo 2 had to be reassigned to Compartment 3, while Cargo 3 and 4 were placed in Compartments 2 and 4, respectively. At Port 3, to maintain the balance of the tanker, Cargo 3 was reassigned into Compartments 2, 3, and 4. The directed arrows in Figure (6.1) show the flow of cargoes from one port to another. The readers should be able to see that the solution instance presented in Figure (6.1) results in seven cargo swaps, namely, two at Port 1, three at Port 2, and two at Port 3. Note that for completeness, for the loading leg of the cargo, the number of changeovers is equal to the number of cargo-compartment assignments.

The mp-CAP is defined to generate a cargo-compartment assignment plan at each port on the tanker's voyage (route) while minimizing the total number of changeovers or cargo swaps. The following section elaborates the mp-CAP further by presenting its mixed-integer linear programming (MILP) formulation.

6.2 MILP formulation for the mp-CAP

The *multi-period cargo-assignment problem* (mp-CAP) minimizes the total number of cargo changeovers throughout the voyage of the chemical tanker. A changeover is expensive as it incurs tank cleaning costs, time, and labour costs throughout the voyage. The mp-CAP simultaneously generates a feasible cargo-compartment allocation plan for the chemical tanker's entire voyage. This would generate an assignment plan with the least number of changeovers, significantly reducing the associated costs while providing the necessary flexibility to the model for future cargo assignments. Further, a mathematical formulation[1] for the mp-CAP is presented by the authors.

Sets:

K = Set of sailing legs,

N^G = Set of all cargoes/goods assigned to the chemical tanker during the voyage,

N_k^O = Set of cargoes/goods loaded on the chemical tanker at the end of leg $k \in K$,

K_j = Set of sailing legs for cargo $j \in N^G$ is onboard the chemical tanker,

N_j^I = Set of cargoes incompatible with cargo $j \in N^G, N_j^I \subset N^G$,

N^H = Set of compartments (cargo holds) in the ship,

N_h^B = Set of neighbouring/bordering compartments for compartment $h \in N^H, N_h^B \subset N^H$,

N_h^X = Set of cargoes that cannot be stored in compartment $h \in N^H, N_h^X \subset N^G$.

Indices:

k = Index for sailing leg (Index 0 indicates that the chemical tanker is at its starting port),

[1]The formulation in this section is heavily derived from Ladage et al. (2021)

$j =$ Index for cargo,

$h =$ Index for compartment (cargo hold).

Decision variables:

$c_{kjh} =$ 1 if the chemical tanker at the end of leg k$\in K_j$ carries cargo j$\in N^G$ in compartment h$\in N^H$ (Binary),

$w_{kjh} =$ Weight of cargo j$\in N^G$ assigned to compartment h$\in N^H$ of chemical tanker at end of leg k$\in K_j$ (Continuous),

$b_{kjh} =$ 1 if the chemical tanker at the end of leg k$\in$ K_j replaces any cargo j$\in N^G$ (other than itself) with cargo j$\in N^G \setminus \{0\}$ in compartment h$\in N^H$ (Binary).

Parameters:

$K_j^L =$ Loading leg for cargo $j \in N^G$,

$K_j^D =$ Unloading leg for cargo $j \in N^G$,

$C^S =$ Cost per changeover/swap including cleaning, labour, etc. related to swapping cargoes within compartments. Also represents the cost incurred if a cargo j$\in N^G \setminus \{0\}$ is filled in an empty compartment,

$W_j =$ Weight of the cargo j $\in N^G$,

$\rho_j =$ Density of the cargo j $\in N^G$,

$V_h =$ Volume of compartment h$\in$ N^H,

$\kappa_h =$ Lateral distance from compartment h$\in N^H$ to the centre of the chemical tanker,

$\iota_h =$ Longitudinal distance from compartment h$\in N^H$ to the centre of the chemical tanker,

$\alpha =$ Maximum absolute permissible trim causing moment of the chemical tanker,

$\beta =$ Maximum absolute permissible heel causing moment of the chemical tanker,

Objective Function:

$$\textbf{Minimise} \quad \mathbf{z} = C^S \sum_{k \in K} \sum_{j \in N^G \setminus \{0\}} \sum_{h \in N^H} b_{kjh} \tag{6.1}$$

The objective function (6.1) minimizes the total changeover cost (C^S) throughout the voyage of the chemical tanker.

Set of constraints:

$$\sum_{h \in N^H} c_{kjh} \geq 1 \qquad \forall j \in N^G, k \in K_j, \tag{6.2}$$

$$\sum_{j \in N^O_k} c_{kjh} \leq 1 \qquad \forall k \in K, h \in N^H, \tag{6.3}$$

$$b_{kjh} \geq c_{kjh} - c_{(k-1)jh} \qquad \forall j \in N^G, k \in K_j \setminus \{K^L_j\}, h \in N^H, \tag{6.4}$$

$$b_{K^L_j jh} = c_{K^L_j jh} \qquad \forall j \in N^G, h \in N^H, \tag{6.5}$$

$$c_{kjh} + \sum_{j' \in \{N^I_j \cap N^O_k\}} c_{kj'h'} \leq 1 \qquad \forall k \in K, j \in N^O_k, h \in N^H, h' \in N^B_h, \tag{6.6}$$

$$w_{kjh} \leq V^H_h \rho_j c_{kjh} \qquad \forall j \in N^G, k \in K_j, h \in N^H, \tag{6.7}$$

$$\sum_{h \in N^H} w_{kjh} = W_j \qquad \forall j \in N^G, k \in K_j, \tag{6.8}$$

$$-\alpha \leq \sum_{h \in N^H} \sum_{j \in N^O_k} w_{kjh} \iota_h \leq \alpha \qquad k \in K, \tag{6.9}$$

$$-\beta \leq \sum_{h \in N^H} \sum_{j \in N^O_k} w_{kjh} \kappa_h \leq \beta \qquad \forall k \in K. \tag{6.10}$$

Constraint (6.2) states that for every cargo and for every leg it is on board the ship, the summation of assignment variables (c_{kjh}) should at least be greater than equal to one. Essentially, it means that cargo $j \in N^G$ has to be assigned to at least one compartment between its pickup and discharge legs. K is the set of sailing legs, while N^H is the set of compartments. Thus, Constraint (6.3) ensures that during every sailing leg k $\in K$, for each compartment h $\in N^H$ at most one assignment variable c_{kjh} can be one. This ensures that a compartment can either be empty or hold at most one cargo.

Constraint (6.4) models changeovers. It tracks the swapping of cargoes within the chemical tanker's compartments. For two consecutive sailing legs k and $k-1$, for every cargo and for every compartment, Constraint (6.4) states that the changeover variable (b_{kjh}) is 1 if cargo j is assigned to compartment h in leg k ($c_{kjh} = 1$) and was not assigned ($c_{(k-1)jh} = 0$) to compartment h in leg $k-1$. Readers can observe that for any other combination of c_{kjh} and $c_{(k-1)jh}$, variable b_{kjh} becomes a free variable. However, the objective function forces it to zero. Constraint (6.5) defines the changeover variables for each cargo j and each compartment h. It is defined for the loading leg of each cargo j.

Constraint (6.7) ensures that the compartment capacity is not exceeded. Essentially, it makes sure that if a cargo j is assigned to a compartment h ($c_{kjh} = 1$), then the quantity of that cargo (w_{kjh}) assigned to the compartment cannot exceed the compartment capacity ($V_h^H \rho_j$). Constraint (6.8) ensures that the quantity of a cargo that is distributed in multiple compartments during a sailing leg is equal to the total weight of the cargo. Constraints (6.9) and (6.10) ensure that the maximum allowable trim and heel moments are not exceeded. Constraints (6.9) and (6.10) are essential for the stability of the ship. These constraints ensure proper (stable) distribution of cargoes different compartments. The cargo-compartment are enforced in pre-processing by fixing the bounds of the c_{kjh} variables. If a cargo j cannot be stored in compartment h, then for all sailing legs $k \in K$ the upper bounds on variables c_{kjh} are set to 0.

At this point, the readers are suggested to pause and study the MILP formulation carefully. Remember the earlier statement about the presence of *special structure* as a motivation towards employing DW decomposition and reformulation? Can the readers identify such structures in the MILP formulation of the mp-CAP? The next section presents a DW reformulation and column generation framework applied to the *mp-CAP*, which will help the readers address the above questions.

6.3 Dantzig-Wolfe reformulation and column generation framework for the mp-CAP

This section discusses the Dantzig Wolfe (DW) decomposition and reformulation of the mixed-integer linear program (MILP) presented in Section 6.2. The unique structures in this problem provide two advantages. First, the constraint set can be divided into *nice* and *complicated* constraints. This is used to decompose the problem with a DW reformulation. This results in a formulation with fewer constraints but an exponentially large number of decision variables. The decomposed reformulation is solved within a column generation framework, also presented in this section. A unique secondary structure helps the authors generate heuristic MILP solutions at any iteration of the column generation procedure, without the Branch & Bound algorithm.

Observe the formulation presented in Section 6.2. The reader might observe that Constraint (6.4) is the only constraint linking decisions from subsequent sailing legs. Ignoring this *complicated* or linking constraint decomposes the problem into multiple sub-problems. Nonetheless, this is not the only decomposition possible for the mp-CAP MILP formulation.

Another decomposition can be obtained by retaining Constraints (6.3), (6.6), (6.9) and (6.10) in the master problem. Coincidentally, even this decomposition results in multiple sub-problems. Similarly, there exist multiple decompositions based on the MILP formulation presented in Section 6.2. However, readers will understand the multiple advantages of the problem decomposition presented in this section as the discussion progresses. Interestingly, there even exists a Bender's decomposition for the mp-CAP MILP formulation. However, that discussion is beyond the scope of this chapter. Interested readers should refer to Chapter 8 of [Conforti et al. (2014)], which provides an excellent overview of reformulations and relaxations in the context of integer programming.

In this section, we present the decomposition that retains Constraints (6.4) and (6.5) in master problem. The rest of the problem disintegrates into as many sub-problems as the number of ports visited by the tanker on its voyage. Each sub-problem deals with generating a cargoes-compartment assignment plan at each port. Such a special structure is known as a block diagonal structure. This decomposition allows us to leverage this structure inherently present in the MILP formulation. Any problem with a block diagonal structure can be decomposed into one master problem and multiple sub-problems. The DW reformulation is presented in Sections 6.3.1 and 6.3.2.

6.3.1 Restricted master problem

This section presents the restricted master problem (RMP) of the DW reformulation for the mp-CAP MILP formulation (6.2). Variable b_{kjh} and λ^v are both continuous variables bounded between zero and one. Constraints (6.4) and (6.5) are retained in the master problem. Let $\{V_k | k \in K\}$ be the set of all corner points of the feasible region (polytope) of the k^{th} sub-problem. Let $V'_k \subseteq V_k$ be the set of corner points present in the restricted master problem at any iteration. $\{f^v_{jh} | k \in K, j \in N^O_k, h \in H, v \in V_k\}$ is used to denote a corner point. A corner point or a column in the RMP is a cargo-compartment allocation plan for a port on the route of the tanker. During any iteration, the maximum number of columns that can be added to the RMP is equal to the number of sub-problems. As the readers can imagine, there are an exponential number of feasible allocations or columns, which necessitates the use of column generation to solve this reformulation.

λ^v are the multipliers dictating the convex combinations of the corner points. Every unique vector λ defines a unique cargo-compartment allocation plan. Based on Theorem (3.1), the readers must observe that the authors substitute $c_{kjh} = \sum_{v \in V'_k} f^v_{jh} \lambda^v$ while formulating the below master problem. The mathematical formulation of the RMP (linear) is as follows.

Objective Function:

$$\textbf{Minimise} \quad \mathbf{z_{mp}} = C^S \sum_{j \in N^G} \sum_{k \in K_j} \sum_{h \in N^H} b_{kjh} \tag{6.11}$$

Set of constraints:

$$\sum_{v' \in V'_{(k-1)}} f^{v'}_{jh} \lambda^{v'} - \sum_{v \in V'_k} f^v_{jh} \lambda^v \geq -b_{kjh} \qquad \forall j \in N^G, k \in K_j \backslash \{K^L_j\}, h \in N^H, \tag{6.12}$$

$$\sum_{v \in V'_{K^L_j}} f^v_{jh} \lambda^v = b_{K^L_j jh} \qquad \forall j \in N^G, h \in N^H, \tag{6.13}$$

$$\sum_{v \in V_k} \lambda^v = 1 \qquad \forall k \in K, \tag{6.14}$$

$$\lambda^v \geq 0 \qquad \forall k \in K, v \in V_k. \tag{6.15}$$

Objective function (6.11) is same as that of the original problem. It minimizes the total number of changeovers throughout the voyage of the ship. Constraints (6.12) and (6.13) are the reformulated constraints corresponding to Constraints (6.4) and (6.5), respectively. Constraint (6.12) defines the changeover activity throughout the voyage of the tanker. For the loading leg (K^L_j) of the cargo $j \in N^G$, Constraint (6.13) states that the number of changeovers ($b_{K^L_j jh}$) is equal to the convex combination of compartment assignments of cargo $j \in N^G$. Constraint (6.14) is the convexity constraint, which forces the sum of λ's for every $k \in K$ to be equal to 1.

At every iteration, new decision variables λ^v's along with its corresponding new columns/corner points $\{f^v | v \in V_k, k \in K\}$ are added to the RMP. If new columns were added, the RMP is re-optimized for decision variables b_{kjh} and λ^v. The updated λ vector is passed to the sub-problem. The formulation and implementation of these sub-problems are presented in the next section.

6.3.2 Dantzig-Wolfe sub-problem

The multi-period cargo-assignment problem (mp-CAP) has a block diagonal structure while considering the proposed decomposition. This results in multiple sub-problems. Contrary to traditional implementations of DW-CG frameworks, we solve integer sub-problems. Each sub-problem generates partial columns during every iteration of the column generation procedure. These partial columns are aggregated together to form columns entering the master problem.

Due to the exponential number of columns, the traditional DW relaxation and column generation often leads to slow convergence. As a result, some form of guidance towards generating good quality columns is often introduced to accelerate the algorithm's convergence. Some such customization for the mp-CAP are also discussed further in this section.

Let $\bar{\lambda}^v$ (continuous, bounded between 0 and 1) the optimal solution to the master problem and $\bar{\mu}^1_{kjh}, \bar{\mu}^2_{K^L_j jh}, \bar{\mu}^3_k$ be the corresponding optimal dual values for Constraints (6.12), (6.13) and (6.14), respectively. Let μ^{LP}_{kjh} be the dual values corresponding to Constraints (6.4) and (6.5) obtained by solving the LP relaxation of the TAP formulation (Equations (6.1) - (6.10)). Similarly, let μ^{3LP}_k be the dual value associated with Equation (6.2). Decision variables c_{kjh} are binary variables, and take the value 1 if cargo j is assigned to compartment h during leg k. In the below sub-problems N^O denotes the onboard cargoes on the ship during that specific leg. The sub-problem can be decomposed on the number of sailing legs. Thus, for a fixed $k \in K$ the sub-problem is as follows.

Objective Function:

$$\textbf{Minimise} \quad \mathbf{z_k} = \sum_{j \in N^G} \sum_{h \in H} \Delta_{kjh} c_{kjh} \tag{6.16}$$

$$\Delta_{kjh} = \alpha \bar{\mu}_{kjh} + (1-\alpha)\mu_{kjh}^{LP} + c_{kjh}^{heur} \qquad \forall k \in K, j \in N^G, h \in N^H, \alpha \in [0,1].$$

$$\mu_k^3 = \alpha \bar{\mu}_k^3 + (1-\alpha)\mu_k^{3LP}$$

$$\bar{\mu}_{kjh} = \begin{cases} \bar{\mu}^2_{K_j^L jh} - \bar{\mu}^1_{(k+1)jh}, & \text{if } |K_j| > 1 \text{ and } k = K_j^L \\ \bar{\mu}^2_{K_j^L jh}, & \text{if } |K_j| = 1 \text{ and } k = K_j^L \\ \bar{\mu}^1_{kjh}, & \text{if } |K_j| > 1 \text{ and } k = |K_j| \\ \bar{\mu}^1_{kjh} - \bar{\mu}^1_{(k+1)jh}, & \text{otherwise.} \end{cases}$$

$$c_{kjh}^{heur} = \begin{cases} 0, & \text{if } k = K_j^L \text{ or } c_{(k-1)jh} = 1 \\ \text{Penalty Constant}, & \text{otherwise.} \end{cases}$$

Note: $|K_j| = 1$ signifies that the cargo was picked up and immediately dropped off at the next port. Additionally, columns are accepted into the master problem only if $z_k < \mu_k^3$.

Set of constraints:

$$\sum_{h \in N^H} c_{kjh} \geq 1 \qquad \forall j \in N^G, \tag{6.17}$$

$$\sum_{j \in \{N_k^O \cup 0\}} c_{kjh} = 1 \qquad \forall h \in N^H, \tag{6.18}$$

$$c_{kjh} + \sum_{j' \in \{N_j^I \cap N_k^O\}} c_{kj'h'} \leq 1 \qquad \forall j \in N_k^O, h \in N^H, h' \in N_h^B, \tag{6.19}$$

$$w_{jh} \leq V_h^H \rho_j \bar{c}_{jh} \qquad \forall j \in N^G, h \in N^H, \tag{6.20}$$

$$\sum_{h \in N^H} w_{jh} = W_j \qquad \forall j \in N^G, \tag{6.21}$$

$$\sum_{h \in N^H} \sum_{j \in N_k^O} \iota_h w_{jh} \geq -\alpha, \tag{6.22}$$

$$\sum_{h \in N^H} \sum_{j \in N_k^O} \iota_h w_{jh} \leq \alpha, \tag{6.23}$$

$$\sum_{h \in N^H} \sum_{j \in N_k^O} \kappa_h w_{jh} \geq -\beta, \tag{6.24}$$

$$\sum_{h \in N^H} \sum_{j \in N_k^O} \kappa_h w_{jh} \leq \beta. \tag{6.25}$$

Traditionally, the column generation framework is used to solve linear programs, which means that both the master problem and the sub-problems are linear. However, in the proposed framework, the master problem is linear, while the sub-problems are MILPs. These MILP sub-problems generate only integer columns that enter into the master problem. As the procedure progresses, only columns fulfilling the reduced cost criteria are accepted into the master problem, ensuring improvement in the objective function and termination of the algorithm.

Column generation theory also dictates that the sub-problem objective should minimize the reduced cost equation to find columns with negative reduced cost. However, in the proposed framework, the sub-problem objective function uses a customized cost co-efficient (Δ) that improves the convergence of the overall framework. The non-standard objective function (stabilization and heuristic) is only used temporarily during some of the intermediate iterations. Other iterations are solved using the original/true objective to ensure convergence. Thus, in theory, if we let the algorithm run long enough, it will converge to the LP optimal solution.

The cost coefficient vector (Δ) is generated as a linear combination of three sets of parameters. The first set of parameters includes the duals ($\bar{\mu}$) generated by solving the LP master problem at every iteration. These are the reduced cost coefficients. The second set of parameters contains the dual values (μ^{LP}) generated from the LP relaxation of the TAP formulation. Vector (μ^{LP}) corresponds to Constraints (6.5), (6.4) and (6.3). μ^{LP} help warm start the algorithm and guide the search during the initial iterations. These also ensure that the lower bound (minimization problems) generated by the DW relaxation is at least as good as the bound generated by the LP relaxation. The idea of the stabilising initial process of column generation using the duals of the primal master problem is similar to the one discussed by Amor et al. (2009). However, as the column generation algorithm progresses, the effect of μ^{LP} is reduced to let $\bar{\mu}$ dictate the search direction.

The final set of parameters (c^{heur}) are generated using a simple repair heuristic. These are introduced to help guide the column generation process by transmitting information between sub-problems. The columns generated from each of the sub-problems are passed into the master problem. Traditionally, the sub-problems are solved without sharing any information between them. Even though this enables solving these sub-problems in parallel, it does lead to slower convergence.

For example, consider a small example. Remember that the overall objective is to minimize the total number of changeovers. For simplicity, ignore the ship balancing criteria, and assume that only one cargo needs to be assigned to a tanker (with two compartments) during two consecutive sailing legs. Recall that a column generated by the sub-problems is nothing but different cargo-compartment allocations for the tanker's route. Consider two different columns that can be generated. The first column gives an allocation that assigns the cargo to compartment one during both the sailing legs. Alternatively, the second column defines an allocation that assigns the cargo to compartment one during the first sailing leg and compartment two during the second sailing leg.

The readers should be able to recognize that one of these columns is superior. Even though both columns can enter the master problem, the first column is superior as it results in zero changeovers. The cargo remains assigned to the same compartment during both legs. The c^{heur} parameters enable us to transmit this information between consecutive sub-problems.

Even though c^{heur} helps transmit useful information across the sub-problems, it does have some disadvantages. The c^{heur} parameter can only be used if the sub-problems are solved sequentially. Additionally, including the c^{heur} within the objective function no longer guarantees convergence for that iteration as the objective no longer represents the reduced cost. Thus, the repair heuristic is only applied intermittently to guide the solution search. It is worth noting that as the heuristic approach is only used intermittently, we are guaranteed to converge to at least the LP optimum of the master problem. Additionally, the iterations for which the repair heuristic is applied need not fulfil the reduced cost criteria.

The DW relaxation (RMP and SP) is solved until a termination criteria is met. The solution obtained at any iteration or even the optimal solution to the DW relaxation is not feasible for the MILP formulation. This is because, for the original formulation, the decision variables b_{kjh} are still fractional. To generate integer solutions at the end of any iteration, one would need to solve the Branch & Bound (B&B) tree by branching on the b_{kjh} variables. This would generate a feasible solution for the MILP from the corresponding DW relaxation solution. However, a unique structure exists in the master problem, eliminating the need to solve the B&B tree. The following section explores this unique structure of the master problem.

6.3.3 Master problem - shortest path problem representation

The master problem is solved as a linear program (LP) within the proposed solution framework. Consequently, solving the restricted master problem to optimality produces a fractional solution that is not feasible for the MILP formulation (Equations (6.1)-(6.10)) of the TAP. In this section, we show that the restricted master problem (Section 6.3.1) can be represented as the shortest path problem. This enables us to quickly generate a heuristic MILP solution for the TAP from master problem columns at any iteration without applying the Branch & Bound (B&B) procedure.

Let $\{V_k | k \in K\}$ and $\{f^{v_k} | v_k \in V_k\}$ denote the set of corner points and a corner point for the k^{th} sub-problem, respectively. Let each corner point be represented by a vertex or node on the directed graph. Directed edges can only exist between corner points of two consecutive sub-problems. Moreover, let the edge weights represent the total changeovers ($B_{v_k,v_k^{'}}$) between any two vertices. Simple arithmetic gives us the total number of changeovers ($B_{v_k,v_k^{'}}$) for any pair of vertices (v_k,$v_k^{'}$). An artificial origin node and a destination node are introduced to complete the shortest path representation. The problem then becomes finding the shortest path (least number of changeovers) between the origin node and the destination node.

Figure (6.2) presents a small example. Assume, that three ports are visited giving rise to three sub-problems. At the present iteration, we have three corner points for each of the sub-problems. Figure (6.2), vertex $f^{1,1}$ represents corner point f_{1_1}. Vertices O and D are the artificial origin and destination nodes. The solution to the instance represented in Figure (6.2) is as follows. Among the set of corner points, corner points (allocations) f_{2_1} -> f_{1_2} - > f_{2_3} result in the least

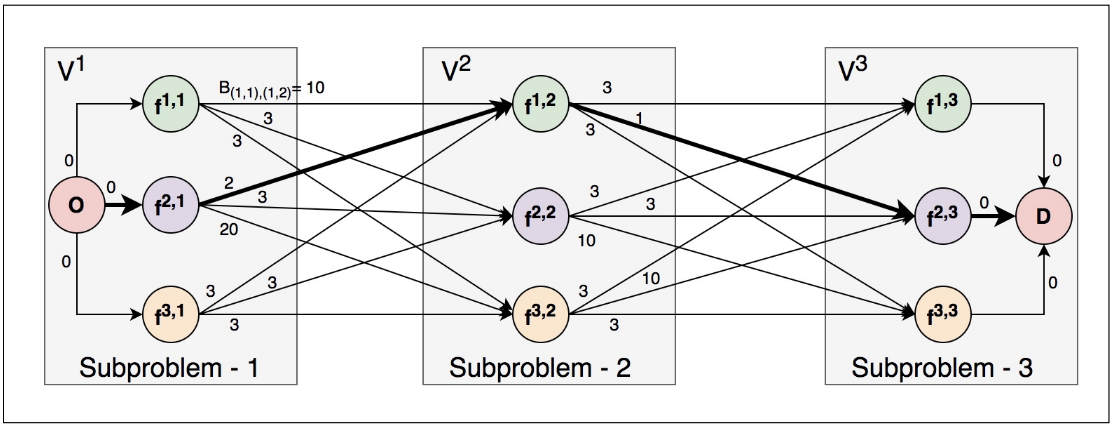

Figure 6.2: The shortest path representation of the master problem presented in Section 6.3.1

number of changeovers i.e two. Thus, at any iteration in the column generation framework we can generate a MILP solution for the TAP from a set of master problem columns.

The authors hope that the readers were able to improve their understanding of the DW reformulation and the column generation algorithm with the help of a real-world example. The readers must also realize that the practical implementation of these advanced optimization techniques requires consideration of some additional parameters. Some of these parameters have already been discussed in Section 3.2. In the next section, we will discuss some computational results related to the mp-CAP.

6.4 Computational results

Computational experiments based on the above formulations are now described. Since these problems require several inputs, this section starts by describing synthetically generated test instances of the mp-CAP and other related chemical scheduling problems. While some of the inputs are generated randomly, the range of random parameters are close to realistic values seen in practice.

6.4.1 Test instances for numerical experimentation

The test instances and an instance generator presented in Chapter 4 are the starting points for computational work described in this section. As discussed in the previous chapters, the test instances are divided into 44 instance sets depending on the different input characteristics. Inputs include the total number of cargoes, the port network number, the maximum number of sailing legs, the total planning time, and the cargo characteristics like total weight, density, loading port, unloading port, pickup time windows, and compatibility criteria. The total number of compartments in a tanker ranges from 16 to 52. For each instance, the total number of cargoes is 40, 80, or 120. Every instance is named according to the instance set to which it belongs and the chemical tanker that is used to generate the instance.

Since mp-CAP models a smaller set of decisions as compared to the general scheduling and routing problem, the instance set and the solution sets together are used for inputs to mp-CAP. The solution files for the S-PDP-TWTAC provide a list of cargo loaded on the tanker at every port. The mp-CAP tries to assign these loaded cargoes to different compartments of the tanker. Additionally, the chemical tanker scheduling problem is defined in such a way that the list of cargoes used as an input for the mp-CAP has at least one feasible cargo-compartment allocation plan for the mp-CAP formulations described in Sections 6.2 and its reformulation. The study presented in the following section discusses fifteen instances selected from the above set.

6.4.2 Performance analysis of the MILP model and column generation framework

This section discusses an empirical study conducted to test the mixed integer linear programming (MILP) formulation and the Dantzig-Wolfe (DW) reformulation solved using the customized column generation (CG) framework. Ideally, the CG framework would be embedded in a Branch & Price-Cut tree, with the CG framework being used at every node of the tree. Implementation of a user-defined Branch & Price-Cut tree requires knowledge of branching rules, cutting planes, and tree pruning strategies, which are out of the scope of this chapter. Hence, the CG framework presented here only solves the root node LP and does not solve the mp-CAP entirely.

The MILP and DW formulations are built using Julia 1.6 and JuMP. Julia was introduced by Bezanson et al. (2017) and is a programming language suitable for scientific and engineering applications. JuMP is a modelling language designed by Dunning et al. (2017) for mathematical optimization in Julia. It enables easy construction of optimization models and allows one to call optimization solvers, like Gurobi-9.1 used here from a Julia program. Source files and the fifteen test instances have been provided[2] as a GitHub repository. Interested readers can implement these to further improve their understanding of the framework and recreate the results. The 15 test instances are solved in two ways:

- Using the Gurobi 9.1 MILP solver taking the mp-CAP formulation (Section 6.2) as an input. The formulation is modelled in JuMP. In this chapter, we refer to these runs as MILP runs.
- The root node of the CG framework (Section 6.3) implemented using JuMP. The master problem and sub-problems are solved using Gurobi 9.1 MILP solver. In this chapter, we refer to these runs as CG runs.

The MILP formulation and its LP relaxation are solved using the Gurobi Optimization (2022) commercial solver. Tables 6.1 and 6.2 record important problem statistics and performance parameters for the MILP runs, and CG runs. The total time limit for all instances is five minutes wall clock time.

Table 6.1 presents the problem sizes and solving times for the MILP and CG runs. Initial two columns give the *Instance set* and *Ship number* of the instances. Both these parameters aggregate specific problem characteristics as explained in Section 6.4.1. The following two columns present the total number of variables and constraints for the MILP runs. Columns *CG master vars* and *CG master constrs* show the number of variables and constraints for the master problem of the CG framework. Similarly, the following two columns tabulate the total number of variables and constraints for all sub-problems. Column *Sub-problems per iter* gives the number of sub-problems solved during each iteration of the CG runs, while the final two columns report the wall clock time (in seconds) for the MILP runs, and the CG runs. The last row of Table 6.1 gives the average value of every parameter for the 15 instances.

[2]https://github.com/anuragladage/mp-CAP.git

Table 6.1: Problem size and solving time comparison for MILP, and CG framework solved at root node

Instance set	Ship no	MILP vars	MILP constrs	CG master vars	CG master constrs	Sub-problem vars	Sub-problem constrs.	Sub-problem per iter	MILP time (sec)	CG time (sec)
2	9	2304	6004	106	208	294	1036	8	2	2
22	3	2304	5930	65	128	147	758	16	1	2
10	4	6336	15848	149	296	402	1616	16	2	3
14	17	4032	10492	177	352	510	1780	8	3	4
18	3	2640	7082	160	310	441	1534	11	91	6
6	1	12012	30618	376	750	1099	4274	11	4	8
2	1	11232	29194	478	952	1413	4894	8	54	22
13	18	21216	55310	1003	1992	2983	9834	8	81	22
15	9	3072	8498	573	496	735	2374	8	10	24
8	9	4224	11402	609	566	833	2742	11	5	32
23	18	24960	61956	537	1072	1570	5836	16	2	32
35	11	15360	38280	337	672	968	3896	16	22	35
33	10	24816	61492	535	1066	1562	5700	16	209	36
1	3	3072	8420	836	464	686	2204	8	19	37
3	7	12480	33516	784	128	147	758	16	5	51
Average		**10004**	**25603**	**448**	**630**	**919**	**3282**	**12**	**34**	**21**

Table 6.2: Comparison of upper and lower bounds generated by MILP runs and CG runs. Bound diff (CG) gives the difference between upper and lower bounds generated during CG runs

Instance set	Ship no	Lower bounds CG root	Lower bounds LP value	Lower bounds Diff	Upper bounds CG root	Upper bounds MILP Obj	Upper bounds Diff	Bound diff (CG)
2	9	450	350	100	450	450	0	0
22	3	200	200	0	200	200	0	0
10	4	350	350	0	350	350	0	0
14	17	300	300	0	350	300	50	50
18	3	300	250	50	350	350	0	50
6	1	400	350	50	400	400	0	0
2	1	450	450	0	450	450	0	0
13	18	950	850	100	950	950	0	0
15	9	650	400	250	850	650	200	200
8	9	650	400	250	850	650	200	200
23	18	500	500	0	500	500	0	0
35	11	400	400	0	400	400	0	0
33	10	550	550	0	550	550	0	0
1	3	400	200	200	650	400	250	250
3	7	600	500	100	650	650	0	50

Table 6.1 shows that the average number of variables and constraints are significantly less for the CG runs compared to the MILP runs. Remember that the CG framework uses a delayed column generation procedure, which only generates columns as and when required. This results in a significant reduction in the number of variables. The reduction in the number of constraints is a property of the DW reformulation. Table 6.1 also shows that the average times taken by MILP runs and CG runs are 34 seconds and 21 seconds, respectively.

Table 6.2 presents the upper and lower bounds generated during the MILP runs, and the CG runs. CG and MILP runs terminate optimally within the time limit for all 15 instances. Like Table 6.1, the first and second columns tabulate the instance set number and the ship number. Following two columns, tabulate the lower bounds generated during the CG and MILP runs. Column *LB diff* shows the difference between these two lower bounds. A positive value (eight instances) in Column *LB diff* gives instances for which the CG runs found a tighter lower bound than the LP relaxation.

The upper bounds of the CG and MILP runs, and the difference between these upper bounds are tabulated in the following three columns. Zero value in the Column *UB diff* signifies that the CG framework could find the optimal solution for the mp-CAP MILP formulation for eleven instances. The CG framework discovered these optimal solutions at the root node. Column *Bound diff (CG)* presents the difference between the upper and lower bounds generated during the CG runs. Recall that the CG upper bounds are generated from the shortest path representation of the master problem, while the CG lower bounds are the optimal master problem objective value. Observe the six instances with positive values in this column. For all of these six instances, to prove optimality, the CG framework has to be embedded into a Branch & Price-Cut tree.

These instances can be bifurcated into two sets. Four out of six of these instances also have a positive value in Column *UB diff*. For these four instances, optimality could not be proved because at the root node, the upper bound is higher than the optimal solution (MILP Obj). This is because the CG lower bounds are constructed from a fractional combination of optimal columns (fractional b_{kjh} variable values). In contrast, the CG upper bounds are constructed from optimal integer columns (binary b_{kjh} variable values). On the other hand, for Instances INST_18_3 and INST_3_7, the optimality could not be proved as the lower bounds are lower than the optimal solution (MILP Obj).

This chapter presented the mp-CAP problem, which is often solved as part of various scheduling problems faced in the chemical tanker industry. We discussed two different frameworks for solving the mp-CAP, namely, a MILP framework, and a DW-CG framework. Advantages, disadvantages, and insights obtained by using both the frameworks was also outlined in this chapter. In the next chapter, we will summarise the research carried out as part of our Ph.D. thesis, and also present important conclusions that will help other researchers continue the research in this field.

This page was intentionally left blank.

Chapter 7

Summary and Conclusions

We present an improved formulation for the single-ship chemical tanker scheduling problem. Our revised formulation substantially reduces the number of decision variables and constraints. The revised formulation performs better than the existing formulation that is available in the literature in terms of memory requirements, solution quality and Cplex solution time. The linear relaxation of the revised formulation is also tighter than the one presented in the literature. We hope other researchers will use our instance generator and the library of instances to improve the models and solution techniques. In its entirety, Cplex found the s-PDP-TWTAC MILP formulation considerably challenging due to the large search space of the problem. As a result, we moved our focus to neighbourhood search heuristics, which could find good quality neighbourhoods, and reduce the Cplex search tree.

The heuristics presented by us can be classified into two groups: linear programming guided neighbourhood search heuristics (LPNS, LPNS+) and integer programming guided neighbourhood search heuristics (H1, H2, H3, GRASP). LPNS heuristic tries to define an initial search neighbourhood from the LP relaxation of the problem. We fortify the LPNS heuristic with a local search strategy to improve the quality of the solutions. We call this heuristic LPNS+. LPNS and LPNS+ heuristics reduce the total solution time of the problem compared to the MILP formulation. We show that the LPNS heuristic consistently finds better quality lower bounds than the Cplex run as the problem complexity (Cplex gap (%)) increases. However, for medium to large instances, the LP relaxation is the bottleneck restricting the overall performance of the heuristic.

To further explore the problem structure and complexity, we performed a sensitivity analysis on the performance parameters obtained from Cplex and the LPNS heuristics. The sensitivity analysis helps us conclude that the Cplex and LPNS heuristic performances deteriorate with the increase in the number of sailing legs and the total number of cargoes. Cplex and the LPNS heuristic performance also improves with the number of different discharge ports for onboard cargoes. A preliminary study showed that all other inputs to the instance generator did not significantly affect the gaps and total solution time (both runs). However, an additional study using a significantly higher number of test instances is required to discover better inputs to performance relationships. Solving a higher number of test instances will enable us to perform a robust and unbiased statistical analysis that might help us discover compounded effects of different input parameters affecting the solution time and quality.

Heuristic *H1* generates a naive initial solution (s^0). Subsequently, the second phase of *H1* improves s^0 by either inserting new cargoes or modifying the chemical tanker's route. The second heuristic uses a more sophisticated relax-and-fix strategy to generate s^0. Phase 1 of the second heuristic finds better quality initial solutions than Phase 1 of the first heuristic. Heuristic *H3* increases the size of the feasible region that can be explored by generating multiple initial solutions. Readers might recall, that initial routes are randomly selected from a set of all possible initial routes. We believe that significant improvement can be achieved if some useful strategy is implemented to select more promising routes instead of an random approach. This would significantly improve the quality of initial solution and help in quicker overall convergence.

Finally, we propose a combination of the three heuristics, which maximises the overall performance of the heuristics. For 188/200 test instances, Heuristic *H1_H2_H3* finds a solution within one percent of the solutions generated during the Cplex run. Compared to the Cplex run, there is a significant decrease in the average total solution time of Heuristic *H1_H2_H3*.

We also discuss the *multi-period cargo-assignment problem* (mp-CAP), its MILP formulation and its Dantzig Wolfe (DW) reformulation in this thesis. The presence of *easy* and *complicated* constraints, along with the shortest path type unique structure, motivates applying the DW reformulation to the mixed-integer linear programming (MILP) formulation. The DW reformulation represents the problem defined by the original objective function and *complicated* constraints in terms of the corner points or vertices of the feasible region defined by the *easy* set of constraints. Such a reformulation has multiple advantages.

First and foremost, it reduces the total number of constraints in the problem. The DW reformulation helps us break the symmetry in the problem. This symmetry occurs due to similar allocations with the same objective value. The DW reformulation also helps us decompose the problem into master and multiple sub-problems. These multiple sub-problems disintegrate into separate problems, thus allowing us to solve them in parallel. The master problem also helps us exploit a unique shortest path structure within the mp-CAP. This eliminates the need to solve the Branch and Bound tree to get a MILP solution from a given set of master problem columns.

However, the Dantzig Wolfe reformulation generally results in an exponential number of decision variables or columns. This necessitates using delayed column generation, which generates columns as and when required. The experiments showed that the CG framework could find good MILP solutions using fewer variables. Moreover, the DW formulation solved using the CG framework found tighter lower bounds than the LP relaxation.

For bigger instances, the CG framework can converge extremely slowly. In practice, the CG framework needs to be embedded within a Branch & Price-Cut tree. This would require a very fast convergence of the CG framework. Thus, fast heuristic approaches to generate good quality solutions to the sub-problems or tighter sub-problem formulations are a couple of ways to improve this framework further. Additionally, a problem-specific Branch & Price-Cut tree implementation will help leverage all the advantages of the DW formulation.

A further study of the symmetric solutions could be a promising research avenue to discover stronger cuts. The problem's symmetry increases with the increase in the number of compartments. The right cuts would enable the CG framework to avoid symmetric solutions and significantly boost its convergence. Another promising research avenue could be to introduce stronger cover cuts for the mp-CAP MILP formulation at each node of the Branch & Price-Cut tree. A study related to the ship's compartment structure and cargo characteristics could help discover cover cuts that can limit the total number of cargoes assigned to the ship as a function of empty compartments. Other avenues include generating reasonable heuristic solutions for sub-problems and an improved sub-problem objective function that can improve the quality of columns entering the master problem.

Interested researchers could also study some extensions of the mp-CAP. A critical extension of the mp-CAP arises by including a more realistic version of the compatibility constraints. For example, the current definition of mp-CAP considers cargo-compartment incompatibility as a function of compartment material. However, cargo-compartment incompatibility also results from cargo stored in a compartment in the preceding sailing legs. Incorporating this into the model will change the structure of the problem and might impact the DW-CG framework.

It should be noted that even though the mp-CAP is individually is very interesting problem, the primary motivation for studying the mp-CAP is to better understand its problem structure. This would help us improve the integration of the mp-CAP into the s-PDP-TWTAC. While designing the neighbourhood search heuristics for the s-PDP-TWTAC, it was noticed the simple greedy heuristics for the mp-CAP sub-problem were ineffective. However, given its special structure, if future researchers are able to improve the convergence rate and solution time of the DW-CG framework, it could be used as part of the neighbourhood search heuristics. It would not only help solve multiple sub-problems in parallel, but also help effectively tackle the symmetry problem hindering the heuristics designed for the s-PDP-TWTAC.

Further, researchers could also solve the s-PDP-TWTAC as a two level optimisation problem. A solution can be generated without the cargo-compartment plan (adjusted s-PDP-TWTAC formulation). This solution can be used as an input to the mp-CAP DW-CG framework. If the input solution is infeasible, a cover cut making the the present combination of cargoes infeasible can be added to adjusted s-PDP-TWTAC formulation. If at least one feasible solution is obtained then the DW-CG framework can be solved to optimality to generate a solution for the s-PDP-TWTAC. This framework could also be used to generate good initial solutions for the s-PDP-TWTAC.

An important practical extension of the s-PDP-TWTAC is to include time windows for both pickup as well as deliveries. This is generally the case in the real world. In general, introducing time-windows on discharge of cargoes should make the problem more difficult. However, one could also tighten the formulation by introducing precedence constraints between cargoes resulting from their discharge time-windows. As such, further investigation is needed to understand the effect of introducing different types of cargo-related time windows.

The instance generator presented in Chapter 4 can also be improved in multiple ways improve its usage, accuracy and practicality. Statistical analysis and experiments can be employed to detect any unwanted biases that be occurring within the data. Presently, the instance generator partially relies on randomly generated data. These could potentially be replaced by real data leading to better overall accuracy of the optimization algorithms. Moreover, the data set can also be extended to include different types of bulk cargoes as well as different type of tankers. This would make the instance generator more usable.

Chemicals play an important role in everyone's lives. Many household and commercial products like sanitisers, petrol, diesel, and medicines are made of chemicals. Notably, during the pandemic crisis, these products' timely transportation and availability was critical. These chemicals are transported across ports with the help of chemical tankers. Our research aimed to improve the scheduling of these chemical tankers. We presented a MILP formulation and some heuristics for the s-PDP-TWTAC.

One important simplifying assumption has been to simplify the in-port operations of the chemical tankers. Including the berthing operations in the problem definition would make the problems more realistic. Also, the industry enforces softer time-window constraints on pick-ups and deliveries and a penalty cost for missing these windows. As such, comparing the effects of hard and soft time-window constraints on the complexity of s-PDP-TWTAC is another avenue for future research.

We also present a multi-period cargo-to-compartment assignment problem (mp-CAP). The mp-CAP forms an essential part of a chemical tanker's scheduling operations. The presence of domain-specific constraints, a multi-period approach, and flexibility to re-arrange cargoes in compartments makes this assignment problem unique. We discussed a decomposition approach for this problem. It is important to remember that different problems have different decompositions and implementation hurdles than those discussed in this chapter. It is often impossible to predict whether one approach will be faster or yield better results than others. Dantzig-Wolfe reformulation is one promising approach, but it must be used carefully to be effective.

Even though there are multiple assumptions simplifying the research problems, we believe these, along with the related heuristics, can be used as part of a decision support system to generate real-time schedules for chemical tankers. Additionally, tanker operators often have to generate schedules for their entire fleet, which requires solving a multi-ship version of the s-PDP-TWTAC. A critical constraint that makes the multi-ship solution different is that cargo has to be optimally assigned to one or many chemical tankers. Although, in most cases, cargo is not split between multiple tankers. This restricts the multi-ship scheduling problem from disintegrating into multiple single ships scheduling problems. Our heuristics could enable future researchers to improve the tractability of the multi-ship chemical tanker scheduling problem. We hope our research can be improved further by future researchers and proves an important milestone in automating the scheduling process of chemical tankers.

Appendix A

Appendix

A.1 Computation Tables

Table A.1: Ship data combined with average (per ship) statistics related to Cplex and heuristic runs of the revised formulation

Ship numbers	Ship name	Draft constant (tonnes)	Total compartments	Ship speed (knots)	Instances per ship	Avg. variables	Avg. constraints	Avg. Cplex gap (%)	Avg. heur gap (%)	Avg. Cplex CPU time (sec)	Avg. Heur CPU time (sec)
1	BOW MEKKA	11176	52	14.3	16	148233	205209	20.78	19.18	41222.12	25090.25
3	BOW HECTOR	8154	16	14.2	15	82946	57522	17.43	26.97	41849	9133.67
4	BOW SANTOS	5027	22	14.1	18	103885	87223	29.21	34.03	49803.89	21144.22
7	BOW FAGUS	11176	52	14.3	18	160705	211473	22.03	21.48	46607.94	27238.88
8	BOW ATLANTIC	4541	24	13.6	16	105609	112466	34.74	37.67	52575.31	18861.38
9	BOW KISO	8793	16	13.8	16	187126	235564	21.15	23.54	39526.56	10960.44
10	BOW FORTUNE	10915	47	14.3	17	78678	63175	22.09	16.16	42259.41	16540.25
11	BOW SAGA	10262	40	14.6	17	146181	177033	22.02	22.13	46645.71	22842.81
17	BOW ARCHITECT	7691	28	14.5	13	125817	169780	29.04	30.09	47945.46	27024.69
18	BOW CEDAR	11176	52	14.3	15	204315	237503	31.08	16.82	50513	28112.42
20	BOW LIND	13416	29	13.7	11	108811	151759	24.6	20.99	50484.27	34533.27
22	BOW FIRDA	10915	47	14.3	19	120954	121219	43.95	27.94	59761.16	37577.53
27	BOW HERON	9450	31	14.5	9	96451	100068	13.05	15.52	26950.22	10811.44

Table A.2: Instance set data combined with the average (per instance set) statistics related to Cplex and heuristic runs of the revised formulation

Instance set	Total cargoes	Network number	Total legs	Onboard cargoes discharge port	Instances per set	Avg. variables	Avg. constraints	Avg. Cplex gap (%)	Avg. heur gap (%)	Avg. Cplex CPU time (sec)	Avg. heur CPU time (sec)
1	40	1	7	1	6	42024	55806	0.01	3.25	1448.83	366.83
2	40	2	7	1	6	92933	58744	0.01	10.16	1681.83	14530.67
3	40	3	7	1	8	43871	54630	0.01	7.92	970.25	127.88
4	40	4	7	1	4	45443	50718	0.01	7.66	2600	4358.5
5	40	1	10	1	5	48653	58368	0.01	8.47	32214.4	6831.6
6	40	2	10	1	6	134869	81700	3.71	3.67	44494.67	1866
7	40	3	10	1	6	59822	61954	0.01	1.42	40243.83	9357
8	40	4	10	1	6	70401	82240	0.01	0.01	15380.5	2397.83
9	40	1	15	1	5	99074	130647	24.8	20.79	86400	70942.6
10	40	2	15	1	4	193238	92981	101.22	96.37	86400	22569.25
11	40	3	15	1	3	113211	134061	45.25	30.72	86400	86400
12	40	4	15	1	2	112603	131483	46.19	40.92	86400	77056.5
13	80	1	7	1	4	80353	115966	0.01	13.99	2498	738.75
14	80	2	7	1	4	133313	121179	0.01	0.01	13012.75	1903
15	80	3	7	1	5	84426	117185	0.01	1.04	6140.8	3676.4
16	80	4	7	1	3	70428	92405	0.01	8.04	8695.67	1076
17	80	1	10	1	6	96019	131734	17.5	20.21	73734.83	12582.5
18	80	2	10	1	6	184214	157150	35.43	37.19	80524.83	14011.67
19	80	3	10	1	3	137923	191019	43.48	45.35	86400	48502.33
20	80	4	10	1	4	109692	144237	26.04	31.32	71406.75	5128.75
21	80	1	15	1	4	163157	235329	83.68	81.54	86400	86400
22	80	2	15	1	6	262573	198540	100.18	90	86400	37930
23	80	3	15	1	5	193277	261004	68.95	61.75	86400	76789
24	80	4	15	1	1	230100	312487	87.43	86.67	86400	86400
25	120	1	7	1	3	145451	217521	0.01	0.01	34131	17464.33
26	120	2	7	1	4	158509	165233	0.01	35.7	25290.25	3549.5
27	120	3	7	1	4	98466	134518	0.01	3.53	14367.5	6825.5
28	120	4	7	1	6	142529	206488	0.01	2.57	24387.83	9310.83
29	120	1	10	1	3	166199	285232	27.11	28.46	82822.33	67004.67
30	120	2	10	1	6	253056	275936	61.66	53.16	79195.33	20739
31	120	3	10	1	6	144424	203485	27.54	31.04	86400	34733
32	120	4	10	1	3	187376	279155	52.74	53.07	86400	36389.67
33	120	1	15	1	6	234597	337592	82.46	66.87	86400	73790.67
34	120	2	15	1	4	370090	395842	90.65	82.87	86400	86400
35	120	3	15	1	5	271205	390792	91.68	60.31	86400	86400
36	120	4	15	1	5	241076	324114	82.79	78.08	86400	53841.33
37	40	1	7	4	1	51325	81826	0.01	0.01	503	68
38	40	2	7	4	5	90619	57397	0.01	5.9	2208.4	127.8
39	40	3	7	4	2	55011	61616	0.01	0.01	1000	574.5
40	40	4	7	4	2	53026	64121	0.01	3.45	659.5	172.5
41	40	1	10	4	5	60526	81055	0.02	4.36	41816.4	21566.6
42	40	2	10	4	5	131062	76950	0.01	1.99	28926	1101
43	40	3	10	4	7	65225	77953	3.2	3.37	20168.57	2177.43
44	40	4	10	4	6	67204	83903	0.01	9.69	32624.83	2976.33

Table A.3: Results obtained from H1, H2 and H3 computational runs

Instance	Instance Name	Status			Phase 1 CPU Time (sec)			Total CPU Time (sec)			First Sol Objective (USD)			Best Sol Objective (USD)			Total Improvements		
		H1	H2	H3	H1	H2	H3	H1	H2	H3	H1	H2	H3	H1	H2	H3	H1	H2	H3
1	INST_1_1	P2	P1	P2	1	19	0	2	19	2	27816.8	1904910	27816.8	1904110	1904910	1904360	4	1	4
2	INST_1_3	P2	P1	P2	0	24	0	0	24	1	291092	2044990	291092	1975100	2044990	1975100	5	1	5
3	INST_1_4	P2	P1	P2	0	20	0	1	20	1	232705	2311570	232705	2188270	2311570	2188270	4	1	4
4	INST_1_7	P2	P1	P2	1	28	0	2	29	2	621564	2501350	621564	2500750	2501350	2500550	4	1	4
5	INST_1_8	P2	P2	P2	1	31	0	2	83	2	-87950.9	402773	-87950.9	1390110	432056	1390210	3	2	3
6	INST_1_22	P2	P1	P2	1	29	0	1	30	1	188470	2047180	188470	2046530	2047180	2046680	4	1	4
7	INST_2_1	P1	P1	P2	2	80	0	2	80	2	1034470	1034470	1034470	1034470	1034470	1034470	1	1	1
8	INST_2_3	P2	P1	P2	1	181	7	1	279	31	-639184	-496982	-639184	-497081	-496982	-497081	2	1	2
9	INST_2_9	P2	P1	P2	1	86	0	1	86	1	355220	449231	355220	449231	449231	449231	2	1	2
10	INST_2_10	P2	P2	P2	1	181	0	2	181	2	-507001	-507000	-507001	-416145	-400331	-400280	2	3	3
11	INST_2_11	P2	P1	P2	1	108	0	2	108	2	165042	375249	165042	364185	375249	364235	5	1	5
12	INST_2_18	P2	P1	P2	2	48	0	2	48	2	312440	398917	312440	398716	398917	398616	2	1	2
13	INST_3_4	P2	P1	P2	0	27	0	1	28	1	647718	1693030	647718	1908630	1693030	1908530	4	1	4
14	INST_3_7	P2	P1	P2	1	21	0	2	21	2	198926	1789680	198926	1704240	1789680	1704240	5	1	5
15	INST_3_8	P2	P1	P2	1	59	0	1	60	1	158911	1255950	158911	1082680	1255950	1082680	3	1	3
16	INST_3_9	P2	P2	P2	0	53	0	1	53	1	933798	484189	933798	1552090	1115640	1552090	3	5	3
17	INST_3_10	P2	P1	P2	1	18	0	1	18	1	445499	1830600	445499	1796980	1830600	1796930	4	1	4
18	INST_3_11	P2	P1	P2	1	18	0	1	19	1	1064760	3130260	1064760	2003370	3130260	2003420	5	1	5
19	INST_3_20	P2	P1	P2	1	27	0	3	27	3	1382970	2024440	1382970	1920780	2024440	1920780	3	1	3
20	INST_3_22	P2	P1	P2	1	20	0	1	21	1	812570	2604940	812570	1967110	2604940	1967110	5	1	5
21	INST_4_3	P2	P1	P2	0	64	0	1	65	1	35165.2	960599	35165.2	538691	960599	538841	3	1	3
22	INST_4_10	P2	P1	P2	1	39	0	1	40	1	749486	809402	749486	809302	809402	809302	2	1	2
23	INST_4_11	P1	P1	P2	1	68	0	1	69	1	72803.7	357419	72803.7	72803.7	357419	72803.7	1	1	1
24	INST_4_17	P1	P1	P2	1	63	0	1	64	1	535150	995349	535150	535150	995349	678455	1	1	4
25	INST_5_3	P2	P1	P2	0	183	5	1	183	22	565841	1158380	565841	1156290	1158380	1156240	5	1	5
26	INST_5_4	P2	P1	P2	1	184	0	1	184	1	310879	1094640	310879	816153	1094640	816253	2	1	2
27	INST_5_7	P2	P2	P2	1	183	0	2	184	2	238379	878770	238379	946176	940882	946126	5	3	5
28	INST_5_8	P2	P2	P2	1	182	0	1	183	1	275491	957024	275491	1556530	987876	1556580	5	3	5
29	INST_5_9	P2	P2	P2	0	181	0	1	182	1	413883	156535	413883	875248	552316	875198	3	5	3
30	INST_6_1	P1	P2	P2	2	181	0	2	182	2	1537990	1516510	1537990	1537990	1537990	1537990	1	2	1
31	INST_6_3	P2	P1	P2	1	182	9	1	183	31	1051200	1091800	1051200	1234530	1091800	1234530	2	1	2
32	INST_6_4	P1	P1	P2	1	187	0	1	188	2	231178	600615	231178	231178	600615	231178	1	1	1
33	INST_6_7	P1	P1	P2	3	182	0	3	182	3	918408	918408	918408	918408	918408	918408	1	1	1
34	INST_6_22	P1	P1	P2	2	181	0	2	182	3	1136620	1136620	1136620	1136620	1136620	1136620	1	1	1
35	INST_6_27	P1	P1	P2	2	181	0	2	181	2	558365	558365	558365	558365	558365	558365	1	1	1
36	INST_7_3	P2	P2	P2	1	180	0	1	181	1	915953	1438070	915953	1512550	1512600	1512550	4	2	4
37	INST_7_4	P2	P1	P2	1	184	0	1	185	1	803895	1733290	803895	1473740	1733290	1473740	5	1	5
38	INST_7_9	P2	P1	P2	1	181	0	1	181	1	25562.6	884124	25562.6	49356.6	884124	49306.6	2	1	2
39	INST_7_10	P2	P2	P2	2	216	0	3	217	3	900702	1408390	900702	1371930	1453030	1371930	5	3	5
40	INST_7_11	P2	P1	P2	1	190	0	2	191	2	306110	1085730	306110	643162	1085730	643162	5	1	5
41	INST_7_17	P2	P1	P2	1	184	0	2	185	2	522785	1136150	522785	883794	1136150	1010190	5	1	7
42	INST_8_9	P2	P1	P2	1	181	5	1	300	22	874483	2977610	874483	2123850	2977610	2123850	4	1	4

Table A.3 continued from previous page

Instance	Instance Name	Status			Phase 1 CPU Time (sec)			Total CPU Time (sec)			First Sol Objective (USD)			Best Sol Objective (USD)			Total Improvements		
		H1	H2	H3	H1	H2	H3	H1	H2	H3	H1	H2	H3	H1	H2	H3	H1	H2	H3
43	INST_8_10	P2	P1	P2	1	182	0	2	182	2	1149810	3064160	1149810	1920860	3064160	1921060	3	1	3
44	INST_8_18	P2	P1	P2	1	183	0	2	184	2	766519	2195940	766519	1473800	2195940	1473800	3	1	3
45	INST_8_20	P2	P1	P2	1	181	0	1	182	2	1243620	1793960	1243620	1410380	1793960	1410380	3	1	3
46	INST_8_22	P2	P1	P2	1	184	1	2	185	2	385564	2660010	385564	2320450	2660010	2320350	4	1	4
47	INST_8_27	P2	P1	P2	1	183	0	1	183	1	510847	1758740	510847	1393910	1758740	1393910	3	1	3
48	INST_9_11	P2	P1	P2	1	181	1	2	182	3	2229840	3974400	2229840	4027080	3974400	4027080	5	1	5
49	INST_9_17	P2	P1	P2	1	188	0	2	188	2	2819510	4532440	2819510	4641690	4532440	4641690	5	1	5
50	INST_9_18	P2	P1	P2	2	194	1	3	195	4	3127430	4807560	3127430	4706640	4807560	4706940	4	1	4
51	INST_9_20	P2	P1	P2	1	183	0	2	184	3	1105860	3508920	1105860	2217130	3508920	2217130	4	1	4
52	INST_9_22	P2	P1	P2	2	202	1	3	203	4	1763120	3745750	1763120	3539430	3745750	3539430	5	1	5
53	INST_10_4	P2	P2	P2	2	181	0	2	182	2	-168298	-238864	-168298	143141	7379.57	143141	2	2	2
54	INST_10_8	P2	P2	P2	2	181	0	3	182	3	-488536	-488537	-488536	78248.1	78247.6	78148.1	2	2	2
55	INST_10_9	P2	P2	P2	2	181	0	2	182	2	-384618	-384617	-384618	-206519	-206419	-206519	2	2	2
56	INST_10_10	P2	P2	P2	3	182	0	4	183	4	-641762	-641762	-641762	46214.1	45963.7	46114.1	3	3	3
57	INST_11_1	P2	P1	P2	3	313	0	4	314	4	881360	2452580	881360	1936510	2452580	1936360	3	1	3
58	INST_11_4	P2	P2	P2	1	181	0	2	182	2	574565	1728430	574565	1986650	1748950	1986650	5	2	5
59	INST_11_22	P2	P1	P2	2	227	0	3	228	3	349300	2455990	349300	1348410	2455990	1348410	3	1	3
60	INST_12_7	P1	P1	P2	3	183	0	3	183	4	1172500	1172500	1172500	1172500	1172500	1172500	1	1	1
61	INST_12_8	P2	P1	P2	1	196	0	2	197	2	1008720	1545780	1008720	1547330	1545780	1547330	4	1	4
62	INST_13_11	P2	P1	P2	1	42	0	2	43	3	1125700	3560080	1125700	2279480	3560080	2279480	5	1	5
63	INST_13_17	P2	P1	P2	1	27	0	2	27	2	398968	2801900	398968	1699260	2801900	1699260	5	1	5
64	INST_13_18	P2	P1	P2	2	63	0	3	64	3	1176260	2728980	1176260	2564190	2728980	2564190	5	1	5
65	INST_13_27	P2	P1	P2	1	23	0	2	24	2	301883	2829210	301883	2657470	2829210	2657470	5	1	5
66	INST_14_10	P2	P1	P2	3	203	1	3	205	4	946874	1183290	946874	1036090	1183290	1036040	3	1	3
67	INST_14_17	P1	P1	P2	2	182	0	2	183	2	1339480	1014410	1339480	1339480	1014410	1339480	1	1	1
68	INST_14_18	P2	P2	P2	2	182	0	3	184	3	1030070	1557740	1030070	1742350	1727530	1742500	4	4	4
69	INST_14_20	P2	P1	P2	2	122	0	2	122	2	473192	1334810	473192	705726	1334810	473192	4	1	1
70	INST_15_7	P2	P1	P2	2	41	0	3	43	3	865861	2348320	865861	2347620	2348320	2347620	4	1	4
71	INST_15_8	P2	P1	P2	1	127	1	1	128	1	376490	1647270	376490	1409240	1647270	1409240	3	1	3
72	INST_15_9	P2	P1	P2	1	31	0	1	32	1	1101260	2660630	1101260	2660130	2660630	2660130	5	1	5
73	INST_15_10	P2	P1	P2	1	53	0	2	54	2	376737	1909440	376737	1847950	1909440	1847950	4	1	4
74	INST_15_22	P2	P1	P2	2	81	0	3	82	3	1324310	2650910	1324310	1895250	2650910	1895350	5	1	5
75	INST_16_1	P2	P1	P2	2	92	0	3	93	3	598058	1942460	598058	884296	1942460	1231280	4	1	4
76	INST_16_3	P2	P1	P2	1	182	8	1	241	27	1828360	2360040	1828360	2073280	2360040	2073280	2	1	2
77	INST_16_4	P2	P1	P2	1	183	1	1	184	1	537889	1043050	537889	1156990	1043050	1156940	2	1	2
78	INST_17_3	P2	P1	P2	1	183	1	1	184	2	842428	2429840	842428	2983160	2429840	2983110	4	1	4
79	INST_17_4	P2	P1	P2	1	185	0	2	186	2	283566	1207800	283566	1865080	1207800	1865080	5	1	5
80	INST_17_7	P2	P2	P2	3	190	0	4	190	5	1404640	4434100	1404640	4421780	4459870	4421680	5	2	5
81	INST_17_8	P2	P2	P2	1	186	1	2	187	2	728576	2869200	728576	3850350	3713820	3850350	5	2	5
82	INST_17_9	P2	P1	P2	1	181	0	1	182	2	1259500	3200970	1259500	4122910	3200970	4122910	5	1	5
83	INST_17_10	P2	P1	P2	2	198	0	4	200	4	980699	4036640	980699	3971120	4036640	3971120	5	1	5
84	INST_18_3	P2	P2	P2	2	181	7	2	320	29	1542680	-55018.3	1542680	1584170	1542680	1584170	2	10	2
85	INST_18_11	P2	P2	P2	3	182	1	4	183	4	1098300	1821090	1098300	2167000	2107300	2167000	3	2	3

Table A.3 continued from previous page

Instance	Instance Name	Status			Phase 1 CPU Time (sec)			Total CPU Time (sec)			First Sol Objective (USD)			Best Sol Objective (USD)			Total Improvements		
		H1	H2	H3	H1	H2	H3	H1	H2	H3	H1	H2	H3	H1	H2	H3	H1	H2	H3
86	INST_18_17	P2	P2	P2	2	182	0	3	183	3	564794	564793	564794	1761070	1761070	1761020	4	4	4
87	INST_18_18	P2	P2	P2	4	184	1	5	188	6	1028520	1661930	1028520	2049360	1805870	2049360	5	4	5
88	INST_18_20	P2	P2	P2	3	185	0	3	189	3	997316	1104220	997316	1371000	1230670	1371000	2	3	2
89	INST_18_22	P2	P2	P2	4	197	1	5	200	5	1804230	1930230	1804230	2790120	2439700	2790120	4	4	4
90	INST_19_18	P2	P2	P2	3	204	1	4	207	5	312736	2516450	312736	2405320	2908200	2405170	4	3	4
91	INST_19_22	P2	P2	P2	3	202	1	4	204	5	805615	2784930	805615	2625540	3412740	2625540	5	3	5
92	INST_19_27	P2	P2	P2	2	182	0	3	183	3	914161	3090040	914161	2960940	3105150	2960940	5	3	5
93	INST_20_8	P2	P2	P2	1	182	1	2	183	2	143723	659309	143723	1242030	958721	1242130	3	4	3
94	INST_20_9	P2	P1	P2	1	182	0	2	196	2	801170	2300810	801170	2080170	2300810	2080120	4	1	4
95	INST_20_10	P2	P1	P2	2	186	1	3	187	4	260210	806529	260210	1537910	806529	1537910	4	1	4
96	INST_20_11	P1	P1	P2	2	185	0	2	186	2	1105800	1764550	1105800	1105800	1764550	1105800	1	1	1
97	INST_21_1	P2	P2	P2	4	385	1	7	387	8	401488	799847	401488	1445970	1011660	1445970	5	3	5
98	INST_21_3	P2	P2	P2	1	186	0	3	189	3	372018	1174330	372018	1211650	1608970	1211650	4	5	4
99	INST_21_4	P2	P2	P2	2	181	0	5	186	4	83134.5	5325.68	83134.5	1776880	1359710	2144440	7	8	7
100	INST_21_7	P2	P2	P2	4	348	1	7	355	8	200597	242945	200597	1515880	456186	1515680	5	4	5
101	INST_22_3	P1	P2	P2	3	182	0	3	182	3	-281065	-281065	-281065	-281065	-281065	-281065	1	2	1
102	INST_22_4	P2	P1	P2	3	183	0	4	183	4	18641.4	18641.5	18641.4	456472	18641.5	456422	2	1	2
103	INST_22_7	P2	P2	P2	7	185	0	8	188	8	884086	884087	884086	1132560	1132560	1132560	3	3	3
104	INST_22_8	P2	P2	P2	4	183	0	4	184	4	187709	187708	187709	235231	187709	187709	2	2	1
105	INST_22_9	P2	P2	P2	3	182	0	5	184	5	-236626	-236626	-236626	149051	148951	148901	3	3	3
106	INST_22_22	P2	P2	P2	6	185	1	7	189	8	43738.9	43739.6	43738.9	503482	237974	503482	3	2	3
107	INST_23_10	P2	P2	P2	3	183	0	7	185	8	1264520	1249800	1264520	3273630	3143980	3333040	6	5	7
108	INST_23_11	P2	P2	P2	3	183	0	5	188	6	1972740	998861	1972740	3448570	2610230	3448570	5	4	5
109	INST_23_17	P2	P2	P2	2	182	0	7	186	5	1555200	1423770	1555200	3284210	3118420	3087570	9	9	7
110	INST_23_18	P2	P2	P2	5	185	0	7	189	9	1338780	1103440	1338780	2320470	2762670	2320470	5	5	5
111	INST_23_20	P2	P2	P2	2	188	0	8	199	8	146925	258610	146925	1542080	720529	1542030	5	2	5
112	INST_24_22	P2	P2	P2	5	237	1	6	246	7	-26263.4	381811	-26263.4	1174020	908608	1174020	3	6	3
113	INST_25_10	P2	P1	P2	3	87	0	4	89	5	57155	3063670	57155	2391080	3063670	2391080	5	1	5
114	INST_25_11	P2	P1	P2	2	182	0	3	183	3	342154	2196220	342154	1805080	2196220	1805080	4	1	4
115	INST_25_18	P2	P1	P2	3	204	1	5	208	5	558820	3768240	558820	2373930	3768240	2373930	4	1	4
116	INST_26_1	P2	P1	P2	4	189	0	7	191	7	-310712	1151120	-310712	534773	1151120	534623	3	1	3
117	INST_26_3	P2	P1	P2	2	181	0	2	182	2	97291.1	418869	97291.1	597366	418869	597366	3	1	3
118	INST_26_4	P2	P2	P2	2	184	0	2	184	3	-32027.8	-19807.8	-32027.8	304092	-608.168	304242	4	2	4
119	INST_26_7	P2	P1	P2	4	183	1	6	186	5	-40802.9	629816	-40802.9	666506	629816	666556	2	1	2
120	INST_27_3	P2	P1	P2	1	182	8	2	249	30	305338	2200070	305338	1981320	2200070	1981320	5	1	5
121	INST_27_4	P2	P2	P2	1	131	0	2	135	3	190892	989487	190892	2566370	1016180	2566370	5	2	5
122	INST_27_7	P2	P1	P2	4	107	1	7	111	7	765546	3794670	765546	1617370	3794670	2424440	5	1	5
123	INST_27_8	P2	P1	P2	1	81	0	3	82	3	64923.6	3016620	64923.6	2413350	3016620	2413350	5	1	5
124	INST_28_1	P2	P1	P2	3	172	0	5	173	6	214015	2656510	214015	2486220	2656510	2486220	5	1	5
125	INST_28_10	P2	P1	P2	3	186	0	5	187	6	581943	2293760	581943	2546540	2293760	2546540	5	1	5
126	INST_28_11	P2	P1	P2	2	181	0	4	182	4	622658	2902840	622658	2582420	2902840	2582420	5	1	5
127	INST_28_17	P2	P1	P2	2	181	0	3	182	3	112475	2518210	112475	2517610	2518210	2517610	5	1	5
128	INST_28_18	P2	P1	P2	3	189	1	5	190	6	655030	2206160	655030	1696910	2206160	1696910	4	1	4

Table A.3 continued from previous page

Instance	Instance Name	Status			Phase 1 CPU Time (sec)			Total CPU Time (sec)			First Sol Objective (USD)			Best Sol Objective (USD)			Total Improvements		
		H1	H2	H3	H1	H2	H3	H1	H2	H3	H1	H2	H3	H1	H2	H3	H1	H2	H3
129	INST_28_22	P2	P1	P2	3	273	0	5	275	5	443632	1911850	443632	1601740	1911850	1601690	4	1	4
130	INST_29_1	P2	P1	P2	4	247	1	7	250	8	207156	3061130	207156	2717970	3061130	2717770	5	1	5
131	INST_29_20	P2	P1	P2	2	182	0	16	184	16	1431050	3206750	1431050	2809990	3206750	2809990	5	1	5
132	INST_29_22	P2	P1	P2	4	233	0	6	234	7	314870	3984000	314870	3754420	3984000	3754270	5	1	5
133	INST_30_1	P2	P2	P2	6	191	0	7	192	8	442284	226355	442284	982316	249591	982316	2	2	2
134	INST_30_10	P2	P2	P2	5	183	1	6	186	7	277834	608530	277834	1125880	787528	1125780	3	3	3
135	INST_30_11	P2	P2	P2	5	183	0	6	184	6	329722	329723	329722	647196	647346	647246	2	2	2
136	INST_30_17	P2	P2	P2	3	184	0	4	186	4	382658	335440	382658	906216	474367	906216	4	3	4
137	INST_30_18	P2	P2	P2	5	184	1	7	186	9	819613	819613	819613	1482630	1482630	1482630	3	3	3
138	INST_30_20	P2	P2	P2	3	183	0	5	192	5	469123	706907	469123	935620	1021850	935570	2	2	2
139	INST_31_1	P2	P1	P2	5	229	1	9	231	10	594876	4209650	594876	4144670	4209650	4144670	6	1	6
140	INST_31_3	P2	P1	P2	1	185	0	2	186	2	302876	4552600	302876	2810310	4552600	2810310	4	1	4
141	INST_31_4	P2	P2	P2	2	183	0	3	184	3	574374	4615680	574374	3596660	5262960	3596660	5	2	5
142	INST_31_7	P2	P1	P2	5	200	1	7	202	8	787571	6215470	787571	4238980	6215470	4238980	5	1	5
143	INST_31_8	P2	P2	P2	2	208	0	5	210	5	78936.9	975898	78936.9	3022880	1031620	3022930	4	2	4
144	INST_31_9	P2	P2	P2	1	192	0	4	194	3	362730	3129630	362730	3445800	3166080	3399680	8	2	6
145	INST_32_1	P2	P1	P2	4	211	1	7	213	7	-591496	2348340	-591496	2281840	2348340	2282090	5	1	5
146	INST_32_4	P2	P2	P2	2	190	1	3	192	4	797581	778434	797581	2830380	1124060	2830380	5	2	5
147	INST_32_7	P2	P1	P2	5	193	1	7	195	7	403077	2851050	403077	2444780	2851050	2444630	5	1	5
148	INST_33_8	P2	P2	P2	3	182	1	5	185	6	487014	16767.6	487014	3442600	2196430	3442700	5	6	5
149	INST_33_9	P2	P2	P2	2	182	6	3	188	28	1982310	1762740	1982310	4730440	3674710	4808250	5	5	5
150	INST_33_10	P2	P2	P2	6	185	1	10	193	10	1096290	1010460	1096290	3208890	3583190	3208890	5	6	5
151	INST_33_11	P2	P2	P2	5	183	0	8	187	9	1270160	999337	1270160	3330830	3052370	3330830	5	5	5
152	INST_33_17	P2	P2	P2	3	183	0	5	185	6	1259970	1180680	1259970	3290010	2892910	3290010	5	5	5
153	INST_33_22	P2	P2	P2	6	252	0	10	255	11	1049160	672234	1049160	3008080	682222	3008080	5	2	5
154	INST_34_18	P2	P2	P2	10	186	1	12	189	13	-46952.6	-46952.8	-46952.6	1315880	1315730	1315780	3	3	3
155	INST_34_20	P2	P2	P2	6	184	0	15	195	17	1016620	1016620	1016620	1257600	1257600	1257550	3	3	3
156	INST_34_22	P2	P2	P2	8	186	1	10	189	10	498676	498676	498676	1258830	1258830	1258830	3	3	3
157	INST_34_27	P2	P2	P2	6	182	1	8	186	9	434630	434629	434630	1081880	1081830	1081830	3	3	3
158	INST_35_11	P2	P2	P2	5	185	0	9	204	9	1031060	790292	1031060	4309370	2860430	4309370	5	5	5
159	INST_35_17	P2	P2	P2	3	183	0	6	190	6	1350200	1054830	1350200	4562780	3552100	4562780	5	5	5
160	INST_35_18	P2	P2	P2	7	185	1	11	190	12	991927	660439	991927	3147030	2979500	3147030	5	5	5
161	INST_35_20	P2	P2	P2	4	184	1	25	195	23	295438	-133999	295438	2499200	2193450	2499200	5	4	5
162	INST_35_22	P2	P2	P2	6	186	1	10	195	11	934061	901181	934061	4115810	2721460	4115810	5	4	5
163	INST_36_4	P2	P2	P2	3	181	0	5	184	5	1491290	1491290	1491290	4093030	4093030	4093030	5	5	5
164	INST_36_7	P2	P2	P2	8	194	1	17	199	19	1201350	227403	1201350	3642830	1307320	4381280	5	5	7
165	INST_36_8	P2	P2	P2	3	183	0	5	185	5	752528	721752	752528	2694340	2683420	2694440	3	3	3
166	INST_36_9	P2	P2	P2	2	182	0	5	192	5	958893	-92074.1	958893	3170960	1854000	3170710	4	10	4
167	INST_36_22	P2	P2	P2	7	189	1	11	196	11	1679350	1552210	1679350	4813040	4777380	4813040	5	6	5
168	INST_37_1	P2	P1	P2	7	21	14	8	21	67	148258	1050390	148108	1032260	1050390	1032410	3	1	3
169	INST_38_7	P1	P1	P2	80	171	17	80	174	129	-1169140	-1167910	-2355170	-1169140	-1167910	-1186610	1	1	4
170	INST_38_8	P2	P1	P2	9	183	40	10	258	134	-399686	-139291	-1639370	-338086	-139291	-139841	2	1	6
171	INST_38_9	P2	P1	P2	5	181	12	5	181	46	-1298300	-1037770	-2517640	-1141050	-1037770	-1140900	2	1	4

Table A.3 continued from previous page

Instance	Instance Name	Status			Phase 1 CPU Time (sec)			Total CPU Time (sec)			First Sol Objective (USD)			Best Sol Objective (USD)			Total Improvements		
		H1	H2	H3	H1	H2	H3	H1	H2	H3	H1	H2	H3	H1	H2	H3	H1	H2	H3
172	INST_38_10	P2	P1	P2	10	76	24	10	77	127	-950941	-835398	-1293270	-836898	-835398	-836798	4	1	9
173	INST_38_11	P2	P1	P2	14	74	41	14	75	176	-395754	-334712	-436240	-335863	-334712	-335613	2	1	3
174	INST_39_17	P2	P1	P2	2	30	19	2	55	84	263963	1218010	197729	1158140	1218010	1153130	3	1	4
175	INST_39_18	P2	P1	P2	18	28	33	18	35	120	35151.7	983150	-33224.3	933641	983150	934241	3	1	5
176	INST_40_18	P2	P1	P2	15	22	22	16	26	90	-109640	749710	-152504	646374	749710	658663	3	1	3
177	INST_40_27	P2	P1	P2	3	31	21	3	31	79	-228471	623607	-228971	494082	623607	494082	3	1	3
178	INST_41_7	P2	P1	P2	24	195	42	25	196	223	-175890	1806540	-448257	941310	1806540	1395720	4	1	10
179	INST_41_8	P2	P1	P2	32	148	81	33	149	434	-332989	1589510	-610235	1424880	1589510	1291850	5	1	12
180	INST_41_9	P2	P1	P2	12	182	54	14	205	247	542101	658734	260141	792936	658734	857170	3	1	8
181	INST_41_10	P2	P1	P2	11	187	7	11	193	41	72856.8	1211140	-190525	1062440	1211140	1078090	3	1	6
182	INST_41_11	P2	P1	P2	31	264	33	31	265	172	164335	98323.6	-92239.9	171688	98323.6	171688	2	1	7
183	INST_42_1	P2	P1	P2	48	185	24	49	200	132	-1361370	-1326110	-3686330	-813193	-1326110	-797390	4	1	14
184	INST_42_3	P2	P2	P2	6	181	36	7	260	147	-987528	-743510	-1012110	-658783	-659033	-658733	2	2	3
185	INST_42_4	P2	P2	P2	19	181	63	19	181	259	-1134990	-1227020	-1718010	-782295	-837644	-782095	2	4	5
186	INST_42_7	P1	P2	P2	113	182	16	113	193	162	-1366870	-1406350	-3357040	-1366870	-1367070	-1375440	1	3	6
187	INST_42_27	P2	P2	P2	9	182	47	9	183	179	-85406.3	38620.4	-107878	213014	76079.9	287393	2	2	3
188	INST_43_1	P2	P1	P2	20	132	29	21	132	144	273589	2278900	91439.2	1842150	2278900	1842350	5	1	14
189	INST_43_4	P2	P1	P2	10	183	30	10	183	150	540412	1632370	229237	2351740	1632370	2352040	3	1	8
190	INST_43_7	P2	P1	P2	27	104	29	28	110	144	569615	3103460	344677	2266330	3103460	2331400	5	1	7
191	INST_43_8	P2	P1	P2	17	181	64	19	182	302	194720	1645880	-22384.8	1690230	1645880	1910890	5	1	10
192	INST_43_9	P2	P1	P2	33	181	26	33	182	149	1399060	3408130	1055700	3046340	3408130	2988300	4	1	13
193	INST_43_22	P2	P1	P2	12	183	12	13	184	71	17286.4	2330230	-357600	1379450	2330230	1379500	4	1	7
194	INST_43_27	P2	P1	P2	16	181	55	16	181	190	368163	2503570	-132317	1525780	2503570	1823470	3	1	12
195	INST_44_1	P2	P1	P2	301	187	21	301	187	364	324026	1105890	-444642	1105790	1105890	1106090	3	1	7
196	INST_44_11	P2	P1	P2	2	185	0	2	186	2	102717	231915	40935.9	505303	231915	505303	2	1	3
197	INST_44_17	P2	P1	P2	4	359	20	4	359	82	157508	901716	110254	687377	901716	730186	2	1	7
198	INST_44_20	P1	P2	P2	21	181	26	22	235	144	1207710	1710010	813072	1207710	1711500	1704130	1	2	7
199	INST_44_22	P2	P1	P2	56	194	10	56	195	88	-349823	751151	-757983	590690	751151	590740	2	1	5
200	INST_44_27	P1	P1	P2	6	132	38	6	133	163	-632559	820885	-633059	-632559	820885	782557	1	1	7

Table A.4: Heuristic H1_H2_H3 comparison with the CPLEX run the GRASP heuristic and the LPNS heuristic

Instance	CPLEX Status	CPLEX CPU Time (sec)	CPLEX Obj	CPLEX Gap (%)	LPNS+ Obj	LPNS+ CPU Time (sec)	GRASP Obj	GRASP Obj Diff (%)	H1_H2_H3 Status	H1_H2_H3 CPU Time (sec)	H1_H2_H3 Obj	H1_H2_H3 Obj Diff (%)
1	Optimal	388	1904910	0	1904910	372	1904360	0.0289	H2 Phase 1	23	1904910	0
2	Optimal	577	2044990	0.01	2043390	30	1969210	3.7056	H2 Phase 1	25	2044990	0
3	Optimal	907	2311570	0.01	2311570	167	2188270	5.334	H2 Phase 1	22	2311570	0
4	Optimal	1371	2501350	0	2501350	895	2348750	6.1007	H2 Phase 1	33	2501350	0
5	Feasible	1800	1380550	57.01	1390410	392	1390210	-0.6997	H3 Phase 2	87	1390210	-0.6997
6	Feasible	1800	1588510	33.72	2047180	633	2046680	-28.8428	H2 Phase 1	32	2047180	-28.8742
7	Optimal	730	1034470	0	1034470	65	1034470	0	H1 Phase 1	84	1034470	0

Table A.4 continued from previous page

Instance	CPLEX Status	CPLEX CPU Time (sec)	CPLEX Obj	CPLEX Gap (%)	LPNS+ Obj	LPNS+ CPU Time (sec)	GRASP Obj	GRASP Obj Diff (%)	H1_H2_H3 Status	H1_H2_H3 CPU Time (sec)	H1_H2_H3 Obj	H1_H2_H3 Obj Diff (%)
8	Feasible	1800	-496981	246.14	-496981	186	-497081	0.0201	H2 Phase 1	311	-496982	0.0002
9	Optimal	719	449231	0.01	449231	58	449231	0	H1 Phase 2	88	449231	0
10	Feasible	1800	-399880	197.13	-399730	181	-400280	0.1	H3 Phase 2	185	-400280	0.1
11	Feasible	1800	375250	0.02	374473	1800	364235	2.9354	H2 Phase 1	112	375249	0.0003
12	Optimal	647	398916	0	398916	468	398616	0.0752	H2 Phase 1	52	398916	0
13	Optimal	960	2542040	0.01	2475070	118	2474670	2.6502	H1 Phase 2	30	1908630	24.9174
14	Optimal	744	1789680	0.01	1727460	181	1602960	10.4332	H2 Phase 1	25	1789680	0
15	Feasible	1800	1590210	28.3	1083080	296	1109640	30.2205	H2 Phase 1	62	1255950	21.0199
16	Feasible	1800	1540130	18.67	1552240	94	1552090	-0.7766	H1 Phase 2	55	1552090	-0.7766
17	Optimal	383	1830600	0.01	1801200	143	1796930	1.8393	H2 Phase 1	20	1830600	0
18	Optimal	524	3130260	0	3130210	83	2254910	27.9641	H2 Phase 1	21	3130260	0
19	Optimal	341	2024440	0.01	2024340	97	1920780	5.1204	H2 Phase 1	33	2024440	0
20	Optimal	491	2604940	0.01	2207470	76	1967110	24.4854	H2 Phase 1	23	2604940	0
21	Optimal	1267	960599	0.01	841207	134	538841	43.9057	H2 Phase 1	67	960599	0
22	Optimal	1452	809402	0	809402	708	809302	0.0124	H2 Phase 1	42	809402	0
23	Feasible	1800	438153	0.02	437903	555	72803.7	83.384	H2 Phase 1	71	357419	18.426
24	Feasible	1800	995348	0.03	995348	1800	678455	31.8374	H2 Phase 1	66	995349	-0.0001
25	Feasible	1800	1009490	169.12	1115760	1800	1170160	-15.916	H2 Phase 1	206	1158380	-14.749
26	Feasible	1800	1396690	139.93	1386210	1800	1180980	15.4444	H2 Phase 1	186	1094640	21.6261
27	Feasible	1800	947076	135.64	944126	1150	946126	0.1003	H1 Phase 2	188	946176	0.095
28	Feasible	1800	1641750	92.03	1559440	1800	1555880	5.2304	H3 Phase 2	185	1556580	5.1878
29	Feasible	1800	800062	295.37	920038	1800	919938	-14.9833	H1 Phase 2	184	875248	-9.3975
30	Feasible	1800	1537990	61.75	1537990	1680	1537990	0	H1 Phase 1	186	1537990	0
31	Feasible	1800	1051200	151.05	1291070	1800	1234530	-17.4401	H1 Phase 2	215	1234530	-17.4401
32	Feasible	1800	559717	202.89	637565	872	231178	58.6973	H2 Phase 1	191	600615	-7.3069
33	Feasible	1800	918408	107.84	-85457.5	1803	918408	0	H1 Phase 1	188	918408	0
34	Feasible	1800	1136620	127.53	1136620	1150	1136620	0	H1 Phase 1	187	1136620	0
35	Feasible	1800	558365	186.6	558365	418	558365	0	H1 Phase 1	185	558365	0
36	Feasible	1800	1372960	41.16	1562260	1512	1512550	-10.1671	H2 Phase 2	183	1512600	-10.1707
37	Feasible	1800	1498180	45.6	1659850	1800	1515810	-1.1768	H2 Phase 1	187	1733290	-15.693
38	Feasible	1800	743440	101.37	831957	1800	49306.6	93.3678	H2 Phase 1	183	884124	-18.9234
39	Feasible	1800	1221910	65.77	989611	1800	1567220	-28.2599	H2 Phase 2	223	1453030	-18.9147
40	Feasible	1800	625505	173.65	1063840	1800	628664	-0.505	H2 Phase 1	195	1085730	-73.5766
41	Feasible	1800	880645	107.11	972015	1800	970539	-10.2077	H2 Phase 1	189	1136150	-29.0134
42	Feasible	1800	1762630	126.22	3000330	795	2997330	-70.0487	H2 Phase 1	323	2977610	-68.93
43	Feasible	1800	2636650	52.7	3064160	1800	1921060	27.1401	H2 Phase 1	186	3064160	-16.2141
44	Feasible	1800	1385620	144.15	707783	1800	1473850	-6.3675	H2 Phase 1	188	2195940	-58.4807
45	Feasible	1800	1751950	66.52	1811590	489	1410380	19.4966	H2 Phase 1	185	1793960	-2.3979
46	Feasible	1800	1715380	109.76	2674590	1800	2320350	-35.2674	H2 Phase 1	189	2660010	-55.0683
47	Feasible	1800	1965690	84.34	1909250	1800	1997110	-1.5984	H2 Phase 1	185	1758740	10.5281
48	Feasible	1800	15862.4	33408.5	2229840	1800	4130450	-25939.2501	H1 Phase 2	187	4027080	-25287.5832
49	Feasible	1800	2819510	107.28	2719020	1800	4711300	-67.0964	H1 Phase 2	192	4641690	-64.6275
50	Feasible	1800	3127430	93.02	No Sol	NA	4739840	-51.557	H2 Phase 1	202	4807560	-53.7224

Table A.4 continued from previous page

Instance	CPLEX Status	CPLEX CPU Time (sec)	CPLEX Obj	CPLEX Gap (%)	LPNS+ Obj	LPNS+ CPU Time (sec)	GRASP Obj	GRASP Obj Diff (%)	H1_H2_H3 Status	H1_H2_H3 CPU Time (sec)	H1_H2_H3 Obj	H1_H2_H3 Obj Diff (%)
51	Feasible	1800	1105860	366.93	1105860	1800	2873140	-159.8105	H2 Phase 1	189	3508920	-217.3024
52	Feasible	1800	1763120	186.27	No Sol	NA	3616750	-105.1335	H2 Phase 1	210	3745750	-112.4501
53	Feasible	1800	-168298	2480.64	-168298	1800	143141	-185.0521	H1 Phase 2	186	143141	-185.0521
54	Feasible	1800	-777712	611.5	-488536	1800	78148.1	-110.0485	H1 Phase 2	188	78248.1	-110.0613
55	Feasible	1800	-384618	806.44	-206369	765	-206519	-46.3054	H2 Phase 2	186	-206419	-46.3314
56	Feasible	1800	-935979	431.93	-641762	1800	560027	-159.8333	H1 Phase 2	191	46214.1	-104.9375
57	Feasible	1800	343183	1079.25	No Sol	NA	2406520	-601.2352	H2 Phase 1	322	2452580	-614.6566
58	Feasible	1800	1011610	427.5	793403	1800	2412360	-138.4674	H1 Phase 2	186	1986650	-96.385
59	Feasible	1800	349300	1477.99	No Sol	NA	2258490	-546.576	H2 Phase 1	234	2455990	-603.1177
60	Feasible	1800	1172500	107.16	No Sol	NA	1172500	0	H1 Phase 1	190	1172500	0
61	Feasible	1800	1008720	201.82	1008720	1800	1552960	-53.9535	H1 Phase 2	201	1547330	-53.3954
62	Optimal	1659	3560140	0	3017060	591	2471470	30.5794	H2 Phase 1	48	3560080	0.0017
63	Feasible	1800	2760940	14.15	2270060	586	1848750	33.0391	H2 Phase 1	31	2801900	-1.4836
64	Feasible	1800	2729130	15.93	2566540	1785	2564390	6.0364	H2 Phase 1	70	2728980	0.0055
65	Optimal	513	2829210	0.01	2829210	121	2692880	4.8187	H2 Phase 1	28	2829210	0
66	Feasible	1800	911419	141.12	No Sol	NA	1036090	-13.6788	H2 Phase 1	212	1183290	-29.8294
67	Feasible	1800	1339480	74.23	1339480	654	1339480	0	H1 Phase 1	187	1339480	0
68	Feasible	1800	1372190	83.92	No Sol	NA	1740640	-26.8512	H3 Phase 2	190	1742500	-26.9868
69	Optimal	898	1334810	0	1334760	147	473192	64.5499	H2 Phase 1	126	1334810	0
70	Feasible	1800	2348220	10.44	2348270	1800	2100980	10.5288	H2 Phase 1	49	2348320	-0.0043
71	Feasible	1800	1539390	58.46	1551110	1800	1409240	8.4546	H2 Phase 1	130	1647270	-7.008
72	Feasible	1800	2121590	38.58	2660680	229	2622290	-23.6002	H2 Phase 1	34	2660630	-25.4074
73	Feasible	1800	1909490	12.17	1909390	1800	1847950	3.2229	H2 Phase 1	58	1909440	0.0026
74	Feasible	1800	2454910	29.98	2493250	1800	1962520	20.0574	H2 Phase 1	88	2650910	-7.984
75	Feasible	1800	1803340	45.91	No Sol	NA	953172	47.1441	H2 Phase 1	99	1942460	-7.7146
76	Feasible	1800	2349230	61.88	2097960	168	2073280	11.7464	H2 Phase 1	269	2360040	-0.4602
77	Feasible	1800	1340410	93.06	1417780	764	1277030	4.7284	H1 Phase 2	186	1156990	13.6839
78	Feasible	1800	2905890	83.76	2775360	1800	2983210	-2.6608	H1 Phase 2	187	2983160	-2.6591
79	Feasible	1800	833438	443.16	1039490	1800	1995230	-139.3975	H1 Phase 2	190	1865080	-123.7815
80	Feasible	1800	3304800	91.29	No Sol	NA	4421680	-33.7957	H2 Phase 2	199	4459870	-34.9513
81	Feasible	1800	1077950	449.53	No Sol	NA	3156140	-192.7909	H1 Phase 2	191	3850350	-257.1919
82	Feasible	1800	3152950	105.42	4664760	1800	4554510	-44.4523	H1 Phase 2	185	4122910	-30.7636
83	Feasible	1800	4056620	45.36	No Sol	NA	3971120	2.1077	H2 Phase 1	208	4036640	0.4925
84	Feasible	1800	1542680	219.43	1584270	888	1584170	-2.6895	H1 Phase 2	351	1584170	-2.6895
85	Feasible	1800	1098300	375.87	No Sol	NA	2396910	-118.2382	H1 Phase 2	191	2167000	-97.3049
86	Feasible	1800	1250730	243.32	1014250	1800	1762980	-40.9561	H1 Phase 2	189	1761070	-40.8034
87	Feasible	1800	1028520	371.49	No Sol	NA	2049360	-99.2533	H1 Phase 2	199	2049360	-99.2533
88	Feasible	1800	1074100	242.05	1387890	1163	1371000	-27.6417	H1 Phase 2	195	1371000	-27.6417
89	Feasible	1800	2384260	123.68	No Sol	NA	2858790	-19.9026	H1 Phase 2	210	2790120	-17.0225
90	Feasible	1800	1651450	298.15	No Sol	NA	2584470	-56.497	H2 Phase 2	216	2908200	-76.0998
91	Feasible	1800	1599940	360.64	No Sol	NA	3955260	-147.213	H2 Phase 2	213	3412740	-113.3042
92	Feasible	1800	3109360	122.36	2506510	1800	2766700	11.0203	H2 Phase 2	189	3105150	0.1354
93	Feasible	1800	1342410	235	No Sol	NA	1262570	5.9475	H3 Phase 2	187	1242130	7.4701

Table A.4 continued from previous page

Instance	CPLEX Status	CPLEX CPU Time (sec)	CPLEX Obj	CPLEX Gap (%)	LPNS+ Obj	LPNS+ CPU Time (sec)	GRASP Obj	GRASP Obj Diff (%)	H1_H2_H3 Status	H1_H2_H3 CPU Time (sec)	H1_H2_H3 Obj	H1_H2_H3 Obj Diff (%)
94	Feasible	1800	1935140	158.17	2592510	976	2065020	-6.7117	H2 Phase 1	200	2300810	-18.8963
95	Feasible	1800	464859	835.71	No Sol	NA	1537910	-230.8337	H1 Phase 2	194	1537910	-230.8337
96	Feasible	1800	1635200	160.33	No Sol	NA	1105800	32.3752	H2 Phase 1	190	1764550	-7.9103
97	Feasible	1800	-630808	19198.1	No Sol	NA	1587770	-351.7042	H1 Phase 2	402	1445970	-329.2251
98	No Sol	1800	NA	NA	No Sol	NA	1440190	Inf	H2 Phase 2	195	1608970	Inf
99	Feasible	1800	-626875	1101.2	No Sol	NA	1946780	-410.5531	H3 Phase 2	195	2144440	-442.0841
100	Feasible	1800	200597	70019.7	No Sol	NA	1625090	-710.1268	H1 Phase 2	370	1515880	-655.6843
101	No Sol	1800	NA	NA	No Sol	NA	-281065	Inf	H1 Phase 1	188	-281065	Inf
102	Feasible	1800	-1128010	13985.4	No Sol	NA	456422	-140.4626	H1 Phase 2	191	456472	-140.467
103	No Sol	1800	NA	NA	No Sol	NA	1132560	Inf	H1 Phase 2	204	1132560	Inf
104	Feasible	1800	-822230	18625.3	No Sol	NA	187709	-122.8293	H1 Phase 2	192	235231	-128.6089
105	Feasible	1800	-1285040	12322.2	No Sol	NA	203696	-115.8513	H1 Phase 2	194	149051	-111.5989
106	Feasible	1800	-1233860	11198.1	No Sol	NA	503482	-140.8054	H1 Phase 2	204	503482	-140.8054
107	Feasible	1800	1264520	12535.9	No Sol	NA	3696170	-192.2983	H3 Phase 2	200	3333040	-163.5814
108	No Sol	1800	NA	NA	No Sol	NA	4382480	Inf	H1 Phase 2	199	3448570	Inf
109	No Sol	1800	NA	NA	No Sol	NA	3070260	Inf	H1 Phase 2	198	3284210	Inf
110	Feasible	1800	1338780	10509.1	No Sol	NA	2373100	-77.2584	H2 Phase 2	205	2762670	-106.3573
111	Feasible	1800	146925	2662.97	146925	1800	1169120	-695.7257	H1 Phase 2	215	1542080	-949.5695
112	Feasible	1800	-26263.4	616263	No Sol	NA	1603160	-6204.1602	H1 Phase 2	259	1174020	-4570.1752
113	Feasible	1800	2905960	34.41	No Sol	NA	2324540	20.0078	H2 Phase 1	98	3063670	-5.4271
114	Feasible	1800	2108860	81.31	1064810	1800	1795930	14.8388	H2 Phase 1	189	2196220	-4.1425
115	Feasible	1800	1958750	157.08	No Sol	NA	2373930	-21.1962	H2 Phase 1	218	3768240	-92.3798
116	Feasible	1800	-376931	759.78	No Sol	NA	534673	-241.849	H2 Phase 1	205	1151120	-405.3928
117	Feasible	1800	445145	436.4	878299	464	747135	-67.8408	H1 Phase 2	186	597366	-34.1958
118	Feasible	1800	43489	4622.9	43489	1800	304242	-599.5838	H3 Phase 2	189	304242	-599.5838
119	Feasible	1800	666656	299.54	No Sol	NA	834192	-25.1308	H3 Phase 2	197	666556	0.015
120	Feasible	1800	2051540	53.49	No Sol	NA	1981320	3.4228	H2 Phase 1	281	2200070	-7.2399
121	Feasible	1800	1652410	141.24	No Sol	NA	2538420	-53.6193	H1 Phase 2	140	2566370	-55.3107
122	Feasible	1800	435201	883.03	No Sol	NA	2424440	-457.0851	H2 Phase 1	125	3794670	-771.935
123	Feasible	1800	3016570	9.63	No Sol	NA	2435980	19.2467	H2 Phase 1	88	3016620	-0.0017
124	Feasible	1800	2596450	44.21	No Sol	NA	2655410	-2.2708	H2 Phase 1	184	2656510	-2.3132
125	Feasible	1800	847292	334.28	No Sol	NA	2546340	-200.5269	H1 Phase 2	198	2546540	-200.5505
126	Feasible	1800	1511550	163.09	No Sol	NA	2529440	-67.3408	H2 Phase 1	190	2902840	-92.0439
127	Feasible	1800	1662840	97	No Sol	NA	2129590	-28.0694	H2 Phase 1	188	2518210	-51.4403
128	Feasible	1800	1214570	207.1	No Sol	NA	1953690	-60.8545	H2 Phase 1	201	2206160	-81.6412
129	Feasible	1800	1357050	180.59	No Sol	NA	2466310	-81.7405	H2 Phase 1	285	1911850	-40.8828
130	Feasible	1800	-1463270	546.77	No Sol	NA	2684440	-283.4549	H2 Phase 1	265	3061130	-309.1979
131	Feasible	1800	2739540	81.3	1431050	1800	2754570	-0.5486	H2 Phase 1	216	3206750	-17.0543
132	Feasible	1800	-1214980	13930.3	No Sol	NA	3929570	-423.4267	H2 Phase 1	247	3984000	-427.9066
133	Feasible	1800	442284	819.73	No Sol	NA	984465	-122.5866	H1 Phase 2	207	982316	-122.1007
134	Feasible	1800	277834	1400.49	No Sol	NA	1225630	-341.1375	H1 Phase 2	199	1125880	-305.2348
135	No Sol	1800	NA	NA	No Sol	NA	770858	Inf	H2 Phase 2	196	647346	Inf
136	No Sol	1800	NA	NA	No Sol	NA	1054700	Inf	H1 Phase 2	194	906216	Inf

Table A.4 continued from previous page

Instance	CPLEX Status	CPLEX CPU Time (sec)	CPLEX Obj	CPLEX Gap (%)	LPNS+ Obj	LPNS+ CPU Time (sec)	GRASP Obj	GRASP Obj Diff (%)	H1_H2_H3 Status	H1_H2_H3 CPU Time (sec)	H1_H2_H3 Obj	H1_H2_H3 Obj Diff (%)
137	Feasible	1800	819613	18506.4	No Sol	NA	1594950	-94.5979	H1 Phase 2	202	1482630	-80.8939
138	Feasible	1800	1135600	214.63	637085	1800	935570	17.6145	H2 Phase 2	202	1021850	10.0167
139	Feasible	1800	594876	1194.2	No Sol	NA	4177150	-602.1884	H2 Phase 1	250	4209650	-607.6517
140	Feasible	1800	977618	653.14	No Sol	NA	4084410	-317.792	H2 Phase 1	190	4552600	-365.6829
141	Feasible	1800	2839780	152.89	No Sol	NA	4290460	-51.0842	H2 Phase 2	190	5262960	-85.3298
142	Feasible	1800	72150.2	268204	No Sol	NA	5220430	-7135.5032	H2 Phase 1	217	6215470	-8514.6262
143	Feasible	1800	775975	839.84	No Sol	NA	3132100	-303.6341	H3 Phase 2	220	3022930	-289.5654
144	Feasible	1800	362730	1848.38	No Sol	NA	4077050	-1023.9903	H1 Phase 2	201	3445800	-849.9628
145	Feasible	1800	2071230	162.81	No Sol	NA	2399110	-15.8302	H2 Phase 1	227	2348340	-13.379
146	Feasible	1800	1269260	458.18	No Sol	NA	2866080	-125.8072	H1 Phase 2	199	2830380	-122.9945
147	Feasible	1800	-1247660	15045.7	No Sol	NA	2444630	-295.9372	H2 Phase 1	209	2851050	-328.5118
148	No Sol	1800	NA	NA	No Sol	NA	3461100	Inf	H3 Phase 2	196	3442700	Inf
149	Feasible	1800	1982310	12115.1	No Sol	NA	4709760	-137.5895	H3 Phase 2	219	4808250	-142.5579
150	Feasible	1800	-1107590	21982.1	No Sol	NA	3325550	-400.251	H2 Phase 2	213	3583190	-423.5123
151	Feasible	1800	641618	38220.1	No Sol	NA	4029260	-527.9843	H1 Phase 2	204	3330830	-419.1298
152	No Sol	1800	NA	NA	No Sol	NA	3637780	Inf	H1 Phase 2	196	3290010	Inf
153	Feasible	1800	-473234	55944.5	No Sol	NA	3588190	-858.2274	H1 Phase 2	276	3008080	-735.6433
154	No Sol	1800	NA	NA	No Sol	NA	1315780	Inf	H1 Phase 2	214	1315880	Inf
155	No Sol	1800	NA	NA	No Sol	NA	1257550	Inf	H1 Phase 2	227	1257600	Inf
156	No Sol	1800	NA	NA	No Sol	NA	1258830	Inf	H1 Phase 2	209	1258830	Inf
157	No Sol	1800	NA	NA	No Sol	NA	970715	Inf	H1 Phase 2	203	1081880	Inf
158	Feasible	1800	-946642	28317.9	No Sol	NA	4519120	-577.3843	H1 Phase 2	222	4309370	-555.227
159	Feasible	1800	1350200	19893.8	No Sol	NA	5006040	-270.7628	H1 Phase 2	202	4562780	-237.9336
160	No Sol	1800	NA	NA	No Sol	NA	3412440	Inf	H1 Phase 2	213	3147030	Inf
161	No Sol	1800	NA	NA	No Sol	NA	2444380	Inf	H1 Phase 2	243	2499200	Inf
162	Feasible	1800	-585672	44015.3	No Sol	NA	4496880	-867.8154	H1 Phase 2	216	4115810	-802.75
163	Feasible	1800	1491290	18475.3	No Sol	NA	4049910	-171.5709	H1 Phase 2	194	4093030	-174.4624
164	Feasible	1800	1201350	21785.3	No Sol	NA	3453510	-187.4691	H3 Phase 2	235	4381280	-264.6964
165	No Sol	1800	NA	NA	No Sol	NA	3318120	Inf	H3 Phase 2	195	2694440	Inf
166	Feasible	1800	958893	27860.9	No Sol	NA	2999000	-212.7565	H1 Phase 2	202	3170960	-230.6897
167	No Sol	1800	NA	NA	No Sol	NA	5027930	Inf	H1 Phase 2	218	4813040	Inf
168	Optimal	536	1050390	0.01	1050390	72	1038670	1.1158	H2 Phase 1	96	1050390	0
169	Feasible	1800	-1506970	107.94	-1188860	450	-1186610	-21.2586	H2 Phase 1	383	-1167910	-22.4995
170	Optimal	1050	-139291	0.01	-139291	99	-139841	0.3949	H2 Phase 1	402	-139291	0
171	Feasible	1800	-1037770	85.77	-1133860	40	-1140900	9.9377	H2 Phase 1	232	-1037770	0
172	Optimal	1074	-835348	0	-836649	67	-836798	0.1736	H2 Phase 1	214	-835398	0.006
173	Optimal	371	-334713	0	-335813	32	-335613	0.2689	H2 Phase 1	265	-334713	0
174	Optimal	964	1218010	0.01	1218010	352	1174460	3.5755	H2 Phase 1	141	1218010	0
175	Optimal	886	983149	0	983149	787	907711	7.6731	H2 Phase 1	173	983149	0
176	Optimal	537	749711	0	749661	253	658663	12.1444	H2 Phase 1	132	749711	0
177	Optimal	691	623656	0	580702	93	494082	20.7765	H2 Phase 1	113	623607	0.0079
178	Feasible	1800	1837280	30.88	1424910	1800	1310690	28.6614	H2 Phase 1	444	1806540	1.6731
179	Feasible	1800	1223400	72.08	1344020	1800	1229670	-0.5125	H2 Phase 1	616	1589510	-29.9256

Table A.4 continued from previous page

Instance	CPLEX Status	CPLEX CPU Time (sec)	CPLEX Obj	CPLEX Gap (%)	LPNS+ Obj	LPNS+ CPU Time (sec)	GRASP Obj	GRASP Obj Diff (%)	H1_H2_H3 Status	H1_H2_H3 CPU Time (sec)	H1_H2_H3 Obj	H1_H2_H3 Obj Diff (%)
180	Feasible	1800	2025430	58.2	2257820	1800	857170	57.6796	H3 Phase 2	466	857170	57.6796
181	Feasible	1800	1103200	79.75	557813	1800	1078090	2.2761	H2 Phase 1	245	1211140	-9.7843
182	Feasible	1800	-27097.4	3630.82	71763	1800	171688	-733.5958	H1 Phase 2	468	171688	-733.5958
183	Feasible	1800	-1505380	90.82	-575912	1297	-654078	-56.5506	H3 Phase 2	381	-797390	-47.0307
184	Feasible	1800	-658483	133.01	-658483	227	-656040	-0.371	H3 Phase 2	414	-658733	0.038
185	Feasible	1800	-859602	110.64	-781745	1323	-782095	-9.0166	H3 Phase 2	459	-782095	-9.0166
186	Feasible	1800	-1475460	82.46	-1366870	1800	-1375440	-6.7789	H1 Phase 1	468	-1366870	-7.3597
187	Feasible	1800	167856	537.69	287943	753	287393	-71.214	H3 Phase 2	371	287393	-71.214
188	Feasible	1800	1385190	99.91	2279000	1035	1684890	-21.636	H2 Phase 1	297	2278900	-64.5189
189	Feasible	1800	1650410	99.46	2331610	1800	2344330	-42.0453	H3 Phase 2	343	2352040	-42.5125
190	Feasible	1800	1998970	71.5	2891190	1800	2242210	-12.1683	H2 Phase 1	282	3103460	-55.253
191	Feasible	1800	1570660	101.04	1701370	1800	1960160	-24.7985	H3 Phase 2	503	1910890	-21.6616
192	Feasible	1800	3794790	17.9	4074500	463	3039620	19.9002	H2 Phase 1	364	3408130	10.1892
193	Feasible	1800	2170220	42.05	2214380	1800	1637380	24.5523	H2 Phase 1	268	2330230	-7.373
194	Feasible	1800	1939060	55.02	2503620	575	2221140	-14.5473	H2 Phase 1	387	2503570	-29.1126
195	Feasible	1800	187014	1282.24	804014	1800	1391960	-644.3079	H3 Phase 2	852	1106090	-491.4477
196	Feasible	1800	257320	482.82	505503	1366	505303	-96.3714	H1 Phase 2	190	505303	-96.3714
197	Feasible	1800	699417	196.66	730336	1139	730186	-4.3992	H2 Phase 1	445	901716	-28.9239
198	Feasible	1800	1676770	25.94	1719000	361	1704130	-1.6317	H2 Phase 2	401	1711500	-2.0712
199	Feasible	1800	200486	1237.09	666576	1800	957317	-377.4982	H2 Phase 1	339	751151	-274.6651
200	Optimal	733	820885	0	820885	117	799981	2.5465	H2 Phase 1	302	820885	0

A.2 Thesis contributions

This thesis makes theoretical and practical contributions to the field of chemical tanker scheduling problems. It introduces a MILP formulation, an instance generator, a set of benchmark instances, and multiple neighbourhood search heuristics for the s-PDP-TWTAC. It also discusses the mp-CAP and a DW-CG framework for the mp-CAP. Research published as part of this thesis is as follows:

1. Research Paper: *A revised formulation, library and heuristic for a chemical tanker scheduling problem*. Authors: Anurag Ladage, Davaatseren Baatar, Mohan Krishnamoorthy, Ashutosh Mahajan. Published in 2021 by *Computers & Operations Research* journal.

2. Springer book chapter: *Optimization in the chemical tanker industry: A multi-period cargo-assignment problem*. Authors: Anurag Ladage, Ashutosh Mahajan, Andreas Ernst, Mohan Krishnamoorthy. Accepted in 2023.

References

Amor, H. M. B., Desrosiers, J., Frangioni, A., 2009. On the choice of explicit stabilizing terms in column generation. Discrete Applied Mathematics 157 (6), 1167–1184.

Andersson, H., Hoff, A., Christiansen, M., Hasle, G., Lokketangen, A., 2010. Industrial aspects and literature survey: Combined inventory management and routing. Computers & Operations Research 37 (9), 1515–1536.

Barbucha, D., 2004. Three approximation algorithms for solving the generalized segregated storage problem. European Journal of Operational Research 156 (1), 54–72.

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W., Vance, P. H., 1998. Branch-and-price: Column generation for solving huge integer programs. Operations research 46 (3), 316–329.

Basso, S., Ceselli, A., 2022. Distributed asynchronous column generation. Computers & Operations Research, 105894.

Bausch, D. O., Brown, G. G., Ronen, D., 1998. Scheduling short-term marine transport of bulk products. Maritime Policy & Management 25 (4), 335–348.

Berthold, T., 2007. Rens-relaxation enforced neighborhood search.

Bezanson, J., Edelman, A., Karpinski, S., Shah, V. B., 2017. Julia: A fresh approach to numerical computing. SIAM Review 59 (1), 65–98.
URL https://epubs.siam.org/doi/10.1137/141000671

Bronmo, G., Christiansen, M., Fagerholt, K., Nygreen, B., 2007. A multi-start local search heuristic for ship scheduling—a computational study. Computers & Operations Research 34 (3), 900–917.

Bronmo, G., Nygreen, B., Lysgaard, J., 2010. Column generation approaches to ship scheduling with flexible cargo sizes. European Journal of Operational Research 200 (1), 139–150.

Brouer, B. D., Pisinger, D., Spoorendonk, S., 2011. Liner Shipping Cargo Allocation with Repositioning of Empty Containers. INFOR: Information Systems and Operational Research 49 (2), 109–124.
URL http://www.utpjournals.press/doi/abs/10.3138/infor.49.2.109

Castillo Villar, K. K., Gonzalez-Ramirez, R. G., Miranda Gonzalez, P., Smith, N. R., 2014. A heuristic procedure for a ship routing and scheduling problem with variable speed and discretized time windows. Mathematical Problems in Engineering 2014.

Chan, F. T., Shekhar, P., Tiwari, M., Dec. 2014. Dynamic scheduling of oil tankers with splitting of cargo at pickup and delivery locations: a Multi-objective Ant Colony-based approach. International Journal of Production Research 52 (24), 7436–7453.
URL http://www.tandfonline.com/doi/abs/10.1080/00207543.2014.932932

Chew, E. P., Christiansen, M., Günther, H.-O., Kim, K. H., Kopfer, H., Mar. 2015. Logistics and maritime systems. Flexible Services and Manufacturing Journal.
URL http://link.springer.com/10.1007/s10696-015-9218-2

Christiansen, M., Fagerholt, K., Nygreen, B., Ronen, D., 2007. Chapter 4 Maritime Transportation. In: Laporte, C. B. a. G. (Ed.), Handbooks in Operations Research and Management Science. Vol. 14 of Transportation. Elsevier, pp. 189–284.
URL http://www.sciencedirect.com/science/article/pii/S0927050706140049

Christiansen, M., Fagerholt, K., Nygreen, B., Ronen, D., Aug. 2013. Ship routing and scheduling in the new millennium. European Journal of Operational Research 228 (3), 467–483.
URL http://www.sciencedirect.com/science/article/pii/S0377221712009125

Christiansen, M., Fagerholt, K., Ronen, D., Feb. 2004. Ship Routing and Scheduling: Status and Perspectives. Transportation Science 38 (1), 1–18.
URL http://pubsonline.informs.org/doi/abs/10.1287/trsc.1030.0036

Cóccola, M. E., Dondo, R., Méndez, C. A., Jan. 2015. A MILP-based column generation strategy for managing large-scale maritime distribution problems. Computers & Chemical Engineering 72, 350–362.
URL http://linkinghub.elsevier.com/retrieve/pii/S0098135414001252

Cóccola, M. E., Méndez, C. A., 2015. An iterative milp-based approach for the maritime logistics and transportation of multi-parcel chemical tankers. Computers & Industrial Engineering 89, 88–107.

Conforti, M., Cornuéjols, G., Zambelli, G., 2014. Integer programming models. In: Integer Programming. Springer, pp. 45–84.

Dantzig, G. B., Wolfe, P., 1960. Decomposition principle for linear programs. Operations research 8 (1), 101–111.

Desaulniers, G., Desrosiers, J., Solomon, M. M., 2006. Column generation. Vol. 5. Springer Science & Business Media.

Desrosiers, J., Lübbecke, M. E., 2005. A primer in column generation. In: Column generation. Springer, pp. 1–32.

Dimitrakiev, D., Gunes, E., 2019. Recent developments and trends in the chemical tanker market. International Journal of Scientific and Technology Research 8, 2075–2079.

Dunning, I., Huchette, J., Lubin, M., 2017. Jump: A modeling language for mathematical optimization. SIAM Review 59 (2), 295–320.

Elgesem, A. S., Skogen, E. S., Wang, X., Fagerholt, K., 2018. A traveling salesman problem with pickups and deliveries and stochastic travel times: An application from chemical shipping. European Journal of Operational Research 269 (3), 844–859.

Fagerholt, K., Christiansen, M., 2000a. A combined ship scheduling and allocation problem. Journal of the Operational Research Society, 834–842.

Fagerholt, K., Christiansen, M., 2000b. A travelling salesman problem with allocation, time window and precedence constraints—an application to ship scheduling. International Transactions in Operational Research 7 (3), 231–244.

Fagerholt, K., Ronen, D., 2013. Bulk ship routing and scheduling: solving practical problems may provide better results. Maritime Policy & Management 40 (1), 48–64.

Gatica, R. A., Miranda, P. A., 2011. Special issue on latin-american research: A time based discretization approach for ship routing and scheduling with variable speed. Networks and Spatial Economics 11 (3), 465–485.

Giavarina dos Santos, P. T., Kretschmann, E., Borenstein, D., Guedes, P. C., 2020. Cargo routing and scheduling problem in deep-sea transportation: Case study from a fertilizer company. Computers & Operations Research 119, 104934.
URL https://www.sciencedirect.com/science/article/pii/S0305054820300514

Gilmore, P. C., Gomory, R. E., 1961. A linear programming approach to the cutting-stock problem. Operations research 9 (6), 849–859.

Gurobi Optimization, 2022. Gurobi Optimizer Reference Manual.
URL https://www.gurobi.com

Hellsten, E. O., Sacramento, D., Pisinger, D., 2022. A branch-and-price algorithm for solving the single-hub feeder network design problem. European Journal of Operational Research 300 (3), 902–916.

Hemmati, A., Hvattum, L. M., Fagerholt, K., Norstad, I., 2014. Benchmark suite for industrial and tramp ship routing and scheduling problems. INFOR: Information Systems and Operational Research 52 (1), 28–38.

Hennig, F., Nygreen, B., Christiansen, M., Fagerholt, K., Furman, K., Song, J., Kocis, G., Warrick, P., May 2012. Maritime crude oil transportation – A split pickup and split delivery problem. European Journal of Operational Research 218 (3), 764–774.
URL http://linkinghub.elsevier.com/retrieve/pii/S0377221711008964

Hennig, F., Nygreen, B., Furman, K., Song, J., May 2015. Alternative approaches to the crude oil tanker routing and scheduling problem with split pickup and split delivery. European Journal of Operational Research 243 (1), 41–51.
URL http://linkinghub.elsevier.com/retrieve/pii/S0377221714009461

Homsi, G., Martinelli, R., Vidal, T., Fagerholt, K., 2020. Industrial and tramp ship routing problems: Closing the gap for real-scale instances. European Journal of Operational Research 283 (3), 972–990.

Hvattum, L. M., Fagerholt, K., Armentano, V. A., Nov. 2009. Tank allocation problems in maritime bulk shipping. Computers & Operations Research 36 (11), 3051–3060.
URL http://linkinghub.elsevier.com/retrieve/pii/S0305054809000422

Jetlund, A. S., Karimi, I., Jul. 2004. Improving the logistics of multi-compartment chemical tankers. Computers & Chemical Engineering 28 (8), 1267–1283.
URL http://linkinghub.elsevier.com/retrieve/pii/S0098135403002102

Kjeldsen, K. H., 2011. Classification of ship routing and scheduling problems in liner shipping. INFOR: Information Systems and Operational Research 49 (2), 139–152.
URL http://www.utpjournals.press/doi/abs/10.3138/infor.49.2.139

Korsvik, J. E., Fagerholt, K., 2010. A tabu search heuristic for ship routing and scheduling with flexible cargo quantities. Journal of Heuristics 16 (2), 117–137.

Korsvik, J. E., Fagerholt, K., Laporte, G., Feb. 2011. A large neighbourhood search heuristic for ship routing and scheduling with split loads. Computers & Operations Research 38 (2), 474–483.
URL http://linkinghub.elsevier.com/retrieve/pii/S030505481000136X

Ladage, A., Baatar, D., Krishnamoorthy, M., Mahajan, A., 2021. A revised formulation, library and heuristic for a chemical tanker scheduling problem. Computers & Operations Research 133, 105345.
URL https://www.sciencedirect.com/science/article/pii/S0305054821001258

Lin, D.-Y., Liu, H.-Y., 2011. Combined ship allocation, routing and freight assignment in tramp shipping. Transportation Research Part E: Logistics and Transportation Review 47 (4), 414–431.

Lubbecke, M. E., Desrosiers, J., 2005. Selected topics in column generation. Operations research 53 (6), 1007–1023.

Menezes, G. C., Mateus, G. R., Ravetti, M. G., 2017. A branch and price algorithm to solve the integrated production planning and scheduling in bulk ports. European Journal of Operational Research 258 (3), 926–937.

Meng, Q., Wang, S., Lee, C.-Y., 2015. A tailored branch-and-price approach for a joint tramp ship routing and bunkering problem. Transportation Research Part B: Methodological 72, 1–19.

Neo, K.-H., Oh, H.-C., Karimi, I., 2006. Routing and cargo allocation planning of a parcel tanker. Computer Aided Chemical Engineering 21, 1985–1990.

Nishi, T., Izuno, T., Jan. 2014. Column generation heuristics for ship routing and scheduling problems in crude oil transportation with split deliveries. Computers & Chemical Engineering 60, 329–338.
URL http://linkinghub.elsevier.com/retrieve/pii/S0098135413003001

Norstad, I., Fagerholt, K., Laporte, G., 2011. Tramp ship routing and scheduling with speed optimization. Transportation Research Part C: Emerging Technologies 19 (5), 853–865.

Ostermeier, M., Henke, T., Hübner, A., Wäscher, G., 2021. Multi-compartment vehicle routing problems: State-of-the-art, modeling framework and future directions. European Journal of Operational Research 292 (3), 799–817.

Pache, H., Grafelmann, M., Schwientek, A. K., Jahn, C., 2020. Tactical planning in tramp shipping-a literature review. In: Data Science in Maritime and City Logistics: Data-driven Solutions for Logistics and Sustainability. Proceedings of the Hamburg International Conference of Logistics (HICL), Vol. 30. Berlin: epubli GmbH, pp. 281–308.

Pache, H., Kastner, M., Jahn, C., 2019. Current state and trends in tramp ship routing and scheduling. In: Digital Transformation in Maritime and City Logistics: Smart Solutions for Logistics. Proceedings of the Hamburg International Conference of Logistics (HICL), Vol. 28. Berlin: epubli GmbH, pp. 369–394.

Papageorgiou, D. J., Nemhauser, G. L., Sokol, J., Cheon, M.-S., Keha, A. B., 2014. Mirplib–a library of maritime inventory routing problem instances: Survey, core model, and benchmark results. European Journal of Operational Research 235 (2), 350–366.

Ralphs, T. K., Galati, M. V., 2010. Decomposition methods for integer programming. Wiley encyclopedia of operations research and management science.

Rodrigues, V. P., Morabito, R., Yamashita, D., Da Silva, B. J., Ribas, P. C., 2016. Ship routing with pickup and delivery for a maritime oil transportation system: Mip model and heuristics. Systems 4 (3), 31.

Ronen, D., 1983. Cargo ships routing and scheduling: Survey of models and problems. European Journal of Operational Research 12 (2), 119–126.
URL https://www.sciencedirect.com/science/article/pii/0377221783902151

Ronen, D., 1993. Ship scheduling: The last decade. European Journal of Operational Research 71 (3), 325–333.

Schwindt, C., Zimmermann, J., et al., 2015. Handbook on project management and scheduling vol. 1. Cham: Springer International Publishing.

Scott, J., 1995. A transportation model, its development and application to a ship scheduling problem. Asia-pacific journal of operational research 12 (2), 111–128.

Sherali, H. D., Al-Yakoob, S. M., Hassan, M. M., 1999. Fleet management models and algorithms for an oil-tanker routing and scheduling problem. IIE transactions 31 (5), 395–406.

Sieminski, A., 2016. Short-term energy outlook. Tech. rep., USDOE Energy Information Administration, Washington, DC. Short-Term Analysis Div.

Stålhane, M., Andersson, H., Christiansen, M., Cordeau, J.-F., Desaulniers, G., Dec. 2012. A branch-price-and-cut method for a ship routing and scheduling problem with split loads. Computers & Operations Research 39 (12), 3361–3375.
URL http://linkinghub.elsevier.com/retrieve/pii/S0305054812001013

Sun, P., Veelenturf, L. P., Hewitt, M., Van Woensel, T., 2018. The time-dependent pickup and delivery problem with time windows. Transportation Research Part B: Methodological 116, 1–24.

Tenold, S., Murphy, H., Jan. 2007. Strategy and hegemony in chemical tanker shipping, 1960-1985. Working paper, Norwegian School of Economics and Business Administration. Department of Economics.
URL http://brage.bibsys.no/xmlui/handle/11250/163046

Tran, N. K., Haasis, H. D., Sep. 2015. Literature survey of network optimization in container liner shipping. Flexible Services and Manufacturing Journal 27 (2-3), 139–179.
URL http://link.springer.com/10.1007/s10696-013-9179-2

United Nations, 2015. Review of maritime transport 2015. United Nations, UNCTAD 2015.

United Nations, 2019. Review of maritime transport 2019. United Nations, UNCTAD 2019.
URL https://unctad.org/en/PublicationsLibrary/rmt2019_en.pdf

United Nations, 2021. Review of maritime transport 2021. United Nations, UNCTAD 2021.
URL https://unctad.org/system/files/official-document/rmt2021_en_0.pdf

Vaclavik, R., Novak, A., Suucha, P., Hanzalek, Z., 2018. Accelerating the branch-and-price algorithm using machine learning. European Journal of Operational Research 271 (3), 1055–1069.

Vanderbeck, F., 2000. On dantzig-wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. Operations Research 48 (1), 111–128.

Vilhelmsen, C., Larsen, J., Lusby, R., 2016. A heuristic and hybrid method for the tank allocation problem in maritime bulk shipping. 4OR 14 (4), 417–444.

Vilhelmsen, C., Larsen, J., Lusby, R. M., 2015. Tramp ship routing and scheduling-models, methods and opportunities. Tech. rep., DTU Management Engineering.
URL https://core.ac.uk/reader/43253760

Vilhelmsen, C., Lusby, R., Larsen, J., 2014. Tramp ship routing and scheduling with integrated bunker optimization. EURO Journal on Transportation and Logistics 3 (2), 143–175.

Wang, X., Arnesen, M. J., Fagerholt, K., Gjestvang, M., Thun, K., 2018. A two-phase heuristic for an in-port ship routing problem with tank allocation. Computers & Operations Research 91, 37–47.
URL https://www.sciencedirect.com/science/article/pii/S0305054817302812

Wen, M., Ropke, S., Petersen, H. L., Larsen, R., Madsen, O. B., 2016. Full-shipload tramp ship routing and scheduling with variable speeds. Computers & Operations Research 70, 1–8.

Yildiz, B., Boland, N., Savelsbergh, M., 2022. Decomposition branching for mixed integer programming. Operations Research.

Acknowledgements

First and foremost, I would like to acknowledge the guidance and support of all my PhD supervisors (Dr Ashutosh Mahajan, Prof. Andreas Ernst, Prof. Mohan Krishnamoorthy, and Dr Davaatseren Baatar). Thank you to Dr Mahajan and Dr Krishnamoorthy for inviting me into the PhD program. Without them, this journey would never have started. Their support, guidance, experience, and valuable feedback have significantly shaped and improved my research experience and output. I want to extend my gratitude towards Dr Baatar, who was instrumental in guiding my research and helping me finalise the methodology published in my first journal paper. His vision and knowledge are greatly appreciated. I am also grateful to Dr Andreas Ernst for his ideas and support. They were extremely valuable for the completion of my research. It is an understatement to say that this journey would have been insurmountable without my supervisors.

I would also like to thank my RPC committee consisting of Dr Siddartha Sengupta, Dr Vishnu Narayanan and Dr Pierre Le Bodic, for their valuable inputs, which were essential to my research. I appreciate the financial support and research facilities provided by IITB-Monash Research Academy, IIT-Bombay, Monash University, and Tata consultancy services. A special mention to all my PhD friends who had my back through all my ups and downs. A special thanks to the administrative staff at IIT-Bombay and Monash university, who have helped me throughout my PhD journey. Finally, I would like to recognise the motivation given by my entire family, especially my parents (Anant Ladage and Savita Ladage) and my wife (Janhavi Snehi).