Ship Scheduling with Time-Varying Draft Restrictions: A Case Study in Optimisation with Time-Varying Costs

Elena Kelareva

A thesis submitted for the degree of
Doctor of Philosophy
The Australian National University

June 2014

Except where otherwise indicated, this thesis is my own original work.

Elena Kelareva
2 June 2014

To my mother, the first Dr Kelareva.

Acknowledgments

Many people helped me with this PhD research, both academically and by providing support and advice along the way. First and foremost, I would like to thank my supervisors Sylvie Thiébaux, Philip Kilby and Mark Wallace for their invaluable advice, mentoring and support throughout my PhD.

Thanks also to Kevin Tierney, for collaborating with me on the paper which formed the basis of Chapter 7, as well as Sebastian Brand who came up with some of the ideas for CP model improvements in Chapter 4, and Tzameret Rubin, who helped me with some of the financial impact calculations in Chapter 5. I would also like to thank Charles Gretton and Julien Fischer for their assistance with some of the experiments in Chapter 7, and Olivia Smith for suggesting the idea of sequence variables discussed in Chapter 4.

Thanks also to the researchers in the maritime logistics research group at the Norwegian University of Science and Technology, whom I visited for a week and who gave me a great deal of useful feedback on my research, particularly Marielle Christiansen and Kjetil Fagerholt.

I would also like to thank my colleagues at OMC International, particularly Peter O'Brien and Giles Lesser, who were instrumental in getting the ship scheduling project started and without whose support this research would never have been possible, as well as Gregory Hibbert, Kalvin Ananda, Gordon Lindsay, Andreas Schlichting, Mihir Mone, Lisa Solazzo, Sam Bao, Doug Leng and Jason Ryan, all of whom contributed in various ways to the development of the DUKC® Optimiser ship scheduling system discussed in Chapter 5.

I am also deeply grateful to my family and friends for helping to keep me sane through such a long project, particularly my mother, whose experiences with her own PhD were always a source of inspiration, as well as my husband, Rob, without whose support I probably would never have made it to the end. I would also like to thank my fellow PhD students and the Monday night games group at NICTA for being very welcoming and supportive during my visits to Canberra – you made completing my PhD as an external student a much more social experience.

Finally, this research was supported by the Australian government through an Australian Postgraduate Award, as well as by NICTA. NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

Abstract

In the last few decades, optimisation problems in maritime transportation have received increased interest from researchers, since the huge size of the maritime transportation industry means that even small improvements in efficiency carry a high potential benefit.

One area of maritime transportation that has remained under-researched is the impact of draft restrictions at ports. Many ports have restrictions on ship draft (distance between the waterline and the keel) which vary over time due to variation in environmental conditions. However, existing optimisation problems in maritime transportation ignore time variation in draft restrictions, thus potentially missing out on opportunities to load more cargo at high tide when there is more water available for the ship to sail in, and more cargo can be loaded safely.

This thesis introduces time-varying restrictions on ship draft into several optimisation problems in the maritime industry. First, the Bulk Port Cargo Throughput Optimisation Problem is introduced. This is a novel problem that maximises the amount of cargo carried on a set of ships sailing from a draft-restricted bulk export port. A number of approaches to solving this problem are investigated, and a commercial system – DUKC® Optimiser – based on this research is discussed. The DUKC® Optimiser system won the Australia-wide NASSCOM Innovation Student Award for IT-Enabled Business Innovation in 2013. The system is now in use at Port Hedland, the world's largest bulk export port, after an investigation showed that it had the potential to increase export revenue at the port by $275 million per year.

The second major contribution of this thesis is to introduce time-varying restrictions on ship draft into several larger problems involving ship routing and scheduling with speed optimisation, starting from a problem involving optimising speeds for a single ship travelling along a fixed route, and extending this approach to a cargo routing and scheduling problem with time-varying draft restrictions and speed optimisation.

Both the Bulk Port Cargo Throughput Optimisation Problem and the speed optimisation research shows that incorporating time-varying draft restrictions into maritime transportation problems can significantly improve schedule quality, allowing more cargo to be carried on the same set of ships and reducing shipping costs.

Finally, this thesis also considers issues beyond time-varying draft restrictions in the maritime industry, and investigates approaches in the literature for solving optimisation problems with time-varying action costs. Several approaches are investigated for their potential to be generalisable between different applications, and faster, more efficient approaches are found for both the Bulk Port Cargo Throughput Optimisation problem, and another problem in maritime transportation – the Liner Shipping Fleet Repositioning Problem.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Maritime transportation is one of the most important forms of freight transportation, with over 8 billion tonnes of goods per year transported by sea [UNCTAD, 2011]. With the huge volumes of goods being transported, there is a high potential for cost optimisation in the maritime shipping industry. There has been significant interest in recent years on optimisation problems in maritime logistics, but this area is still under-researched compared to land and air transportation, as discussed in two recent reviews of the maritime transportation literature: Christiansen, Fagerholt, and Ronen [2004] and Christiansen et al. [2013]. Though maritime shipping has been traditionally a conservative industry, computerised decision support systems are beginning to be more frequently accepted as their benefits are becoming more clear [Fagerholt, 2004].

This thesis investigates several optimisation problems in the maritime transportation field, particularly focussing on scheduling and routing problems. Scheduling problems typically aim to select execution times for a set of tasks so as to optimise some cost or value function, subject to problem-specific constraints, particularly on the availability of resources such as machines. Traditional scheduling problems usually aim to minimise the makespan, or total time, of the resulting schedule, though more complex cost functions have also been investigated.

Routing problems have many similarities with scheduling – both have actions that need to be scheduled in time, both may have resource constraints and constraints on setup (waiting) time between actions, and both have some cost function that needs to be optimised. However, the pickup and delivery tasks being scheduled in a routing problem have durations that depend on the relative locations of pickup and delivery nodes in space, and thus vary with the order of actions, whereas in a pure scheduling problem, action durations are generally fixed and do not depend on order, though some scheduling problems do involve sequence-dependent setup times between actions.

Optimisation problems in the maritime transportation domain typically focus on minimising the cost or maximising the profit of some aspect of maritime industry operations, such as the cost required to transport a fixed amount of cargo, or the profit obtained by a shipping company. Alternative objectives may include maximising the utilisation of a costly resource such as berths at a port, minimising fuel usage and carbon dioxide emissions, or optimising customer satisfaction. These objectives are similar to those found in ground, rail and air transportation; however, the constraints involved in maritime shipping may differ substantially. The large value of even a small percentage reduction in global shipping costs makes maritime transportation problems an area of great potential for future research.

One consideration in maritime logistics which does not occur in other modes of transportation is that most ports have restrictions on the draft of ships that are able to safely enter and leave the port. Draft is the distance between the waterline and the bottom of the ship's keel, and is a function of the amount of cargo loaded onto the ship. Ships with a deep draft may risk running aground in shallow water, therefore most ports restrict ship sailing drafts using safety rules that estimate the under-keel clearance (UKC) – the depth of water under a ship's keel.

These safety rules are dependent on environmental conditions such as tide, waves and currents, and therefore in practice, the allowable sailing draft at most ports varies with time. Existing ship scheduling algorithms either ignore draft constraints entirely, for example Fagerholt [2004]; Norstad, Fagerholt, and Laporte [2011], or only consider draft constraints that are do not vary with time [Christiansen et al., 2011; Rakke et al., 2011]. This may result in suboptimal solutions, where ships sail with less cargo than they could have carried if the algorithm had considered the possibility of sailing with a higher draft at high tide.

This dissertation investigates the question: what is the impact of considering draft restrictions on ship scheduling?

The main thesis of this dissertation is that accurate consideration of time-varying draft restrictions in maritime transportation problems can result in reduced shipping costs or increased cargo throughput by taking advantage of the opportunity to load more cargo at high tide. We introduce time-varying draft restrictions into a number of maritime transportation problems, and show that this approach can significantly reduce shipping costs, as well as increase revenue for shippers by allowing more cargo to be transported on the same set of ships.

1.1 Contributions

There are three main areas of contribution consisting of three main novel problems investigated in this thesis:

1. Introducing the novel Bulk Port Cargo Throughput Optimisation Problem (BPC-TOP) with time-varying drafts, and investigating a number of approaches to solving it.

2. Adding time-varying draft restrictions to the ship routing and scheduling problem with speed optimisation.

3. A more general investigation of scheduling and routing problems with time-varying cost functions in related fields.

First, we introduce the novel problem of optimising cargo throughput at a bulk export port with time-varying draft restrictions – the Bulk Port Cargo Throughput

Optimisation Problem (BPCTOP). This problem is motivated by a real port – Port Hedland in Western Australia, the world's largest bulk export port, which is draft-restricted and has up to six metres difference between high and low tides, and can therefore benefit significantly from optimising ship schedules with draft restrictions taken into account.

The Bulk Port Cargo Throughput Optimisation Problem involves scheduling ship sailing times at only one port; however, even at one port, this is quite a challenging problem, since it involves complex resource constraints and interactions between ships. The high tidal variation means that the draft changes significantly even in as short a time as five minutes. Since the allowable sailing draft for a ship cannot be specified by any simple equation, the problem needs to be modelled using a very fine time-indexed formulation. Typical ship scheduling and routing problems can be modelled with a time resolution of hours or even days; however, introducing time-varying drafts requires increasing the resolution to intervals on the order of five minutes, resulting in large domains for the decision variables, and making the problem more difficult to solve.

We create a Constraint Programming model for the BPCTOP and investigate a number of approaches to modelling different aspects of the problem. One part of the problem that is particularly challenging to model is the constraints on the availability of tugs – small boats that are used to assist ships entering and leaving the port. Tug constraints are very complex, since the minimum duration of a tug's action in assisting a ship depends on the destination of the current ship, as well as the origin of the next ship, since travel time between these locations needs to be taken into account. We considered three approaches to modelling tugs, and were able to take advantage of features of the problem to simplify tug constraints and create a tractable model that was able to solve realistic sized problems within five minutes. The CP model is also compared against a Mixed Integer Programming model, with the finding that a CP solver with a good choice of search strategy is able to prove optimality faster for most problem instances.

We also investigate a number of user-defined search strategies for solving the BPCTOP CP model to optimality, and find that the choice of search strategy has a significant impact on the time taken to prove optimality for this problem. We also investigate a logic-based Benders decomposition approach for solving the BPCTOP, with two different approaches to Benders cuts considered. Finally, we investigate three improvements to the CP model, and find that some minor changes to the CP model result in significantly faster calculation times, due to taking advantage of implementation details of the CP solver which make some constraints more efficient to apply than others. Some improvements to the CP model also affect which search strategies are the fastest to solve the problem to optimality. These investigations of calculation speed use a constructed data set which allows greater control of problem characteristics, thus giving additional information on what types of problems are most difficult to solve using different approaches, and allowing us to test our approach on larger problems than the current record at the world's largest bulk export port.

While these results are quite solver- and problem-specific, the broader implication of these results is that Constraint Programming approaches for solving new problems should consider the choice of model, solver and user-defined search strategy as an interacting system, rather than in isolation, as the calculation speed is affected by the interaction between these three factors. The results of our investigation into CP model improvements and user-defined search strategies for the BPCTOP also suggests a number of improvements for the G12 finite domain CP solver which we use to solve the CP model for the BPCTOP.

The best CP model with the fastest user-defined search strategy has been incorporated into a commercial system for scheduling ships at a port – DUKC® Optimiser. This system is now in use at Port Hedland in Western Australia, the world's largest bulk export port, after comparisons of DUKC® Optimiser results against schedules produced by human schedulers at Port Hedland showed that the system had the potential to increase export revenue at the port by US$275 million per year. These comparisons of DUKC® Optimiser results against real schedules from Port Hedland are presented in Chapter 5, along with a few simple examples to illustrate why the system is able to find schedules which allow ships to carry more cargo compared to approaches typically used by human schedulers. The DUKC® Optimiser system was awarded the NASSCOM Innovation Student Award for IT-Enabled Business Innovation in 2013 [Consensus Group, 2013].

Chapter 6 presents the second main contribution of this thesis by introducing time-varying draft restrictions into a larger bulk ship routing and scheduling problem with speed optimisation, building on the work of Norstad, Fagerholt, and Laporte [2011], who investigated ship speed optimisation for a bulk ship routing and scheduling problem. Optimising ship sailing speeds has been widely investigated in recent years, since rising fuel prices have made fuel costs comprise a larger proportion of total shipping costs, and fuel usage is a function of ship speed. However, draft restrictions affect allowable sailing windows at ports and draft-constrained waypoints, which may affect the optimal speed for ships. It is therefore worthwhile to consider these two aspects of ship scheduling in combination.

We start by incorporating time-varying draft restrictions into a simple subproblem – speed optimisation for a single ship travelling along a fixed route. We extend the recursive smoothing algorithm introduced by Norstad, Fagerholt, and Laporte [2011] to be able to handle our more complex problem, with multiple time windows, one around each high tide, as well as a non-monotonic cost function that takes into account not only fuel usage, but also ship draft as well as the opportunity cost of not being able to use the ship for other voyages if the duration of the current voyage takes longer due to sailing at a slower speed. We compare our approach against an alternative way of handling draft restrictions – having the ship wait at draft-restricted waypoint until the tide rises high enough for it to sail – and find that considering time-varying draft restrictions in ship speed optimisation reduces costs by an average of 22%.

The speed optimisation problem for a single ship travelling along a fixed route with draft-restricted waypoints is then used as a subproblem in solving the larger bulk ship routing and scheduling problem with speed optimisation and time-varying draft restrictions. We compare four ways of solving this problem:

1. Considering both time-varying draft restrictions and resource conflicts between ships at waypoints in speed optimisation.

2. An approach that considers time-varying draft restrictions in the speed optimisation problem but ignores resource conflicts between ships, thus requiring ships to wait for each other to sail.

3. Ignoring draft restrictions in speed optimisation and having ships wait for the tide to rise if required, as well as waiting for other ships to sail.

4. Sailing at a fixed service speed and waiting at waypoints for the tide to rise as well as for other ships.

Considering time-varying draft restrictions in speed optimisation results in a 22% reduction in shipping cost on average for our problem instances compared to sailing at a fixed service speed. The method presented by Norstad, Fagerholt, and Laporte [2011] of optimising speeds without considering draft constraints and resource conflicts at waypoints results in a 15% reduction in shipping cost compared to sailing at service speed. Considering resource conflicts at waypoints resulted in a small additional reduction in cost, about 2% more than speed optimisation with time-varying draft restrictions alone. These results show that considering time-varying draft restrictions in maritime optimisation problems can result in a substantial benefit to industry, and we hope that more maritime logistics researchers will incorporate time-varying draft restrictions into their models in future.

The third main contribution of this thesis is to look beyond the specific problem of time-varying draft restrictions in maritime optimisation, to consider the more general problem of scheduling and routing with time-varying costs. The time-varying cost function is one of the main sources of complexity for the problems considered in this thesis. Other routing and scheduling problems in related fields have also considered time-varying action costs, so the final chapter in this thesis summarises how related problems in other fields have dealt with time-varying costs, and investigates the potential of generalising some of these approaches between different applications.

Traditional approaches to routing and scheduling problems with time-varying costs include Mixed Integer Programming and local search, as well as more complex solve-and-improve approaches. We present a Constraint Programming model for another maritime transportation problem with time-varying costs – the Liner Shipping Fleet Repositioning Problem (LSFRP), which involves finding cost-optimal ways to move ships between services in a liner shipping network – and find that the CP model outperforms earlier Mixed Integer Programming and automated planning models. This result, together with the earlier results for the BPCTOP, show that Constraint Programming is a useful technique that may be worth investigating for other problems which are traditionally solved using Mixed Integer Programming, even though, as discussed earlier, the performance of Constraint Programming approaches is highly dependent on the choice of model, solver and search strategy.

We also investigate the effectiveness of a new CP solver on both the LSFRP and BPCTOP. This solver uses Lazy Clause Generation, a recently introduced technique which allows a CP solver to learn areas of the search space where no good solutions have previously been found, and speed up the search by avoiding them in future. We find that the LCG solver outperforms traditional finite domain CP solvers on both the LSFRP and BPCTOP.

We also compare the CP model and LCG solver for our problems against a more typical approach to speeding up calculation of a problem with time-varying costs – a solve-and-improve method which first solves a simplified version of the problem without time-varying costs, then uses the real cost function to improve the solution. We find that, while the simplified models for both problems are faster to solve than the original approach, the speed improvement provided by the solve-and-improve method is lower than that obtained by the CP model for the LSFRP and the LCG solver for the BPCTOP.

Finally, we investigate a novel approach to solving the BPCTOP by modelling it as a Vehicle Routing Problem with Soft Time Windows (VRPSTW) and using an existing fast VRP solver to solve this problem. We find that modelling a scheduling problem as a VRP can be an efficient way to get good quality solutions, though the VRP solver has limitations on the kinds of constraints that can be modelled, which limits the types of scheduling problems that can be solved using this approach, and in the case of the BPCTOP, limits the accuracy of the resulting solutions.

These results show that there is scope for generalisation of approaches between scheduling and routing problems with time-varying costs in related fields, and furthermore, that some approaches such as Constraint Programming and Lazy Clause Generation can outperform methods traditionally used for such problems such as Mixed Integer Programming.

1.2 Thesis Outline

This thesis is formatted as follows.

- Chapter 2 introduces the reader to the maritime transportation industry and presents a review of optimisation problems in maritime logistics. This chapter also introduces a number of techniques used to solve transportation problems, such as Constraint Programming, Mixed Integer Programming and Benders Decomposition. This chapter also presents an overview of how draft restrictions are calculated at ports, as well as how draft constraints have been considered to date in maritime transportation research. This is by no means an exhaustive review of either the maritime transportation literature, or of optimisation techniques, but it is intended as an introduction for readers new to the maritime field, or unfamiliar with some of the methods discussed in this dissertation.

- Chapter 3 introduces the Bulk Port Cargo Throughput Optimisation Problem with time-varying draft restrictions. We present a Constraint Programming (CP) model for this problem, and discuss a number of approaches for modelling the most complex part of this problem – constraints on the availability of tugs.
- In Chapter 4, we investigate the scalability of our CP model for realistic problem sizes, and we compare a number of search strategies for solving this model. We also compare our CP model against a Mixed Integer Programming (MIP) model for this problem, as well as against a Benders Decomposition approach that breaks the model up into a master and subproblem, both of which can be implemented as either CP or MIP models. Finally, this chapter investigates a number of variations on our initial CP model aimed at improving scalability.
- In Chapter 5, we discuss DUKC® Optimiser, the commercial software system for scheduling ship sailing times that has been implemented based on our BPC-TOP models. Comparisons of DUKC® Optimiser results against human schedulers on real data are also presented in this chapter.
- Chapter 6 incorporates time-varying draft restrictions into two more problems in maritime logistics – the problems of ship speed optimisation for a single ship travelling along a fixed route, and the problem of ship scheduling and routing with variable speeds and time-varying draft restrictions. We compare our results for both problems against approaches that optimise speed while ignoring time-varying draft constraints, as well as against traditional approaches that ignore draft and use fixed ship speeds.
- In Chapter 7, we consider our work in the more general context of scheduling and routing problems with time-dependent cost functions, and compare several approaches that have been used to solve related problems in other fields. We also investigate the effectiveness of several of these approaches for a related problem with time-dependent costs – the Liner Shipping Fleet Repositioning Problem (LSFRP). The LSFRP is a problem first introduced by Kevin Tierney, and the sections on the LSFRP presented in this chapter are joint work with Kevin Tierney. Other chapters are primarily the author's work, with feedback and suggestions from numerous people as listed in the acknowledgements.
- Chapter 8 presents concluding remarks and several avenues for future work arising from this research.

Parts of this thesis have previously been published as:

- Kelareva et al. [2012b] – short conference paper; brief summary of Chapter 3 and parts of Chapter 4.
- Kelareva et al. [2012a] – Chapter 3 and parts of Chapter 4.
- Kelareva [2011] – parts of Chapter 5.

- Kelareva [2012] – parts of Chapter 5.
- Kelareva et al. [2013] – short conference paper; brief summary of parts of Chapter 6.
- Kelareva, Tierney, and Kilby [2013a] – parts of Chapter 7.
- Kelareva, Tierney, and Kilby [2013b] – Chapter 7.

Chapter 2

Background

This chapter presents an overview of optimisation problems in the maritime transportation industry, as well as several general techniques commonly used to solve these types of problems which are also used in this thesis. This chapter also discusses how draft restrictions at ports are calculated in practice, as well as how draft restrictions in maritime transportation problems have been considered in the existing literature.

This chapter is not an exhaustive survey of the maritime transportation literature; in particular, only a few problems in container shipping are discussed, as this thesis focusses primarily on problems in bulk cargo shipping. The section on general optimisation techniques written as an introduction for readers unfamiliar with any of the approaches used in this thesis, and is not intended as an exhaustive survey of the field of optimisation.

2.1 The Maritime Transportation Industry

Cargo and shipping service types

Commercial shipping services fall into three distinct types – *tramp*, *industrial* and *liner*.

A *liner* shipping service operates similarly to a bus timetable, with several vessels travelling along a fixed route with a fixed timetable. Liner services are typically used for containers and general cargo. Roll-on Roll-off (RoRo) vessels used to transport cars and other rolling equipment are also typically part of a liner shipping service.

Tramp ships operate similarly to a taxi company, with tramp ships taking up *contracts of affreightment* to transport a specified amount of cargo from an origin to a destination port within a fixed timeframe for a given price. Tramp ships are typically used for bulk cargo, and a tramp operator typically tries to maximise the profit produced by the fleet of ships.

Industrial ships are either owned by the cargo owner, or hired on a *time charter* for a fixed period of time. Industrial ships are typically used for high-volume bulk cargoes such as oil, coal and iron ore.

All three types of cargo ship operators are able to charter additional vessels if required to meet excess demand, or charter out their own vessels in the event of fleet capacity exceeding demand.

Non-commercial vessels such as naval vessels are not considered in this thesis, as they involve quite different problems compared to commercial cargo shipping.

Ships

Ships come in many different sizes, with differing speeds and fuel usages, as well as capacities of weight and volume of cargo. Ships are also limited in the types of cargo they can carry – *tankers* carry liquid bulk cargo; *bulk carriers* carry dry bulk such as wheat, timber or ore; *container* ships carry standard-sized containers used for transporting packaged goods; *Roll-on Roll-off* (RoRo) ships are used for transporting cars or other rolling equipment. Other types of ships include refrigerated vessels used to transport cargo that requires refrigeration such as meat or vegetables, and Liquefied Natural Gas (LNG) carriers, used to carry pressurised, refrigerated gas. Bulk ships may be able to carry either one type of cargo, or several types of cargo in separate compartments, possibly with constraints on the types of cargo that may be transported in each compartment, or the types of cargo that may be stored adjacent to each other.

Ship sizes vary significantly, with ships falling into several main size classes based on their dimensions and deadweight tonnage (DWT) – the total weight the ship can safely carry, including all extras such as fuel, ballast and crew. The most commonly used size categories for dry cargo as listed in the UNCTAD [2012] Review of Maritime Transport and Rodrigue, Comtois, and Slack [2013] are:

1. *Handysize* – small bulk carriers, mostly used for minor bulk products and small ports. 10,000 – 34,999 DWT.

2. *Handymax* – slightly larger than Handysize: 35,000 – 54,999 DWT.

3. *Panamax* – the largest ship size that can travel through the Panama canal: 55,000 – 84,999 DWT; breadth < 32.31m. This size class is used for both bulk carriers and tankers.

4. *Capesize* – large bulk cargo vessels that are too large to pass through the Suez and Panama canals and must thus travel around the Cape of Good Hope or Cape Horn to travel between oceans. These vessels are only able to serve ports with deepwater terminals, and are primarily used to transport raw materials such as iron ore and coal. The size range is defined slightly differently in different sources [UNCTAD, 2012; Rodrigue, Comtois, and Slack, 2013], but generally includes ships larger than Panamax, ie. 80,000+ DWT, breadth > 32.31m.

5. *Very Large Ore Carrier / Ultra Large Ore Carrier* – 200,000+ DWT and 300,000+ DWT respectively; used for transporting iron ore from Brazil. Only a few ports worldwide can accommodate vessels of this size.

The main size categories for tankers are:

1. *Aframax* – 80,000 – 124,999 DWT, larger than Panamax.
2. *Suezmax* – the largest ship size that can travel through the Suez canal, 125,000 – 199,999 DWT.
3. *Very Large Crude Carrier* – 200,000 – 320,000 DWT. Can dock at many terminals, and can be ballasted through the Suez Canal.
4. *Ultra Large Crude Carrier* – 320,000+ DWT; can only dock at a few terminals.

Several other ship size classes are the *Seawaymax* – the largest ships able to pass through the St Lawrence Seaway; *Q-Max* or *Qatar-max* – the largest LNG carriers able to dock at the Qatar LNG terminal; and *Malaccamax* – the largest ships able to sail through the Malacca Strait.

Different ship types and size classes may be subject to different port charges, as well as variations in safety rules at ports. For example, at Port Hedland in Western Australia, vessel berthing charges depend on the length of the vessel, and pilotage charges depend on the total deadweight tonnage [Port Hedland Port Authority, 2013a]. Different vessel size classes also have different safety requirements for the number of tugs that must assist the vessel in arriving or departing a berth, and in entering or leaving the port [Port Hedland Port Authority, 2013c].

Ports

Ports vary in their geography and environmental conditions, which may affect safety requirements for ships operating in and around the port, including restrictions on draft. Each port has a number of berths, with different dimensions and loading/unloading equipment, which may limit the size and type of ships that can use each berth. Different loading and unloading equipment may also have varying loading and unloading rates. Some ports are used for both pickup and delivery of cargo; many other ports, particularly in the bulk shipping industry, are used almost exclusively for one or the other.

Some ports or berths may be privately owned, and thus reserved for use by ships owned or chartered by one company; other berths are public, and may be shared by ships operated by many different companies, with the port charging fees for docking, refueling, and other services. The port typically aims to minimise the turnaround time of ships using the port, as cycling ships through faster increases utilisation of port facilities and thus port revenue. Ports may also aim to maximise the total cargo throughput, particularly if they charge fees per tonne of cargo shipped.

One issue that is of importance to ports is queueing rules used to determine the order in which ships may dock. Many ports use a first-come first-served queueing system, which is simple to administer, but which results in ships sailing quickly to reach the port, followed by waiting for up to several weeks in the queue outside the port. This is highly inefficient, as ships sit idle for weeks at a time, when they could have sailed slower and used less fuel.

Ports have large and expensive infrastructure, such as berths, tugs (small boats which may be required to assist ships in entering and leaving the port), land-side

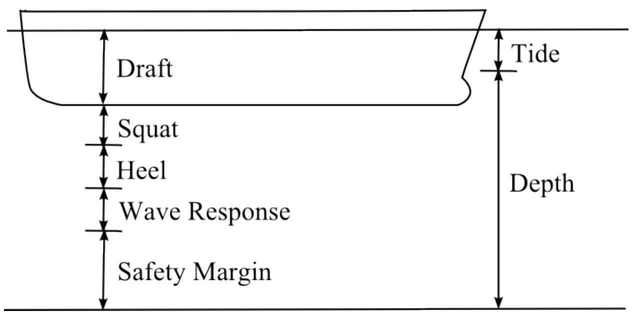


Figure 2.1: Factors affecting under-keel clearance.

transportation equipment such as ship loaders, cranes and container transport vehicles, cargo storage facilities, and other equipment such as pilot boats used to transport harbour pilots to inbound vessels, and lines boats used to assist in mooring ships at a berth. Capacity planning for ports is therefore an important and complex problem, since projects to extend the capacity of a port by extending the infrastructure, for example by building additional berths or dredging the channel to increase the allowable ship drafts, can cost billions of dollars [BHP Billiton, 2011]. Optimisation and other software systems that improve the utilisation of existing port capacity may also be installed to increase the extra port capacity provided by such projects.

2.1.1 Draft Restrictions at Ports

Draft is the distance between the waterline and the ship's keel. Most ports have safety restrictions on the draft of ships allowed to transit through the channel to reduce the risk of deep-draft ships running aground in shallow water. At draft-restricted ports, accurate modelling of draft constraints allows more cargo to be loaded onto each ship in good environmental conditions without compromising safety, which increases profit for shipping companies. In practice, draft constraints at ports are usually calculated by estimating the under-keel clearance of a ship – the amount of water under the ship's keel.

The equations used to calculate under-keel clearance calculations vary between ports. Some ports use simple static estimates of under-keel clearance, which do not consider any live measurements of environmental conditions, and which must therefore be very conservative to maintain safety. In recent years, some ports have begun to use more sophisticated systems which incorporate live environmental forecasts and measurements to produce more accurate estimates of under-keel clearance, and therefore can allow ships to sail with a higher draft on average, while maintaining safety. The Dynamic Under-Keel Clearance (DUKC®) software developed by OMC International, which is the only commercial software to estimate under-keel clearance using live environmental conditions, has allowed ships to sail with up to two metres of extra draft in some environmental conditions, producing large economic benefits for ports [O'Brien, 2002].

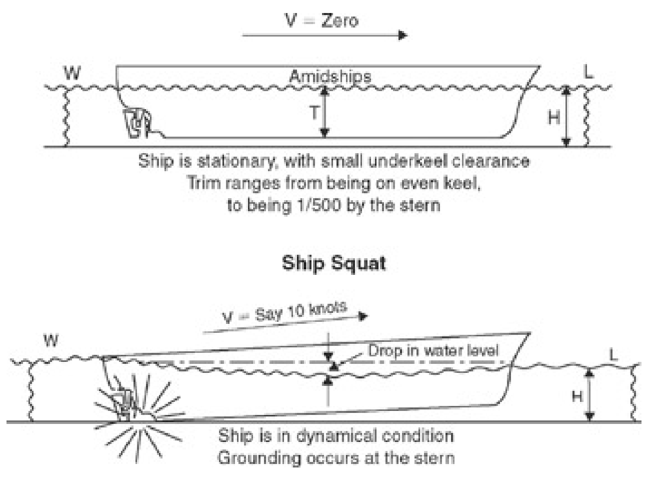

V = Zero
W
Amidships
T
H
L
Ship is stationary, with small underkeel clearance
Trim ranges from being on even keel,
to being 1/500 by the stern
Ship Squat
V = Say 10 knots
W
Drop in water level
L
H
Ship is in dynamical condition
Grounding occurs at the stern

Figure 2.2: Ship squat [Barrass, 2004]

Factors which affect the under-keel clearance of a ship (shown in Figure 2.1) include:

- the *depth* of water at each point along the channel.
- the predicted *tide height* at the time the ship will be transiting through the channel.
- the *draft* of the ship.
- *squat* – a phenomenon caused by the Bernoulli effect, which causes a ship travelling faster in shallow water to sit lower in the water, as illustrated in Figure 2.2 [Barrass, 2004].
- *heel* – the effect of a ship leaning to one side under the effect of centripetal force due to turning, or due to the force of wind. Heel causes one side of the ship to sit lower in the water, thus decreasing under-keel clearance, as illustrated in Figure 2.3.
- *wave response* – the vertical component of a ship's motion in response to waves, as illustrated in Figure 2.4.

Other factors may play a part at specific ports, for example, the changing density of water may affect under-keel clearance at estuary ports where the salinity, and therefore density, of the water varies at different locations along the channel, and at different times in the tide cycle. See O'Brien [2002] for a detailed overview of under-keel clearance at ports.

Equation (2.1) shows an example under-keel clearance constraint for a port. A vessel v will be allowed sail at time t if the constraint expressed in Equation (2.1) is met, ie. if the sum of the positive UKC factors (depth D and tide height T), minus

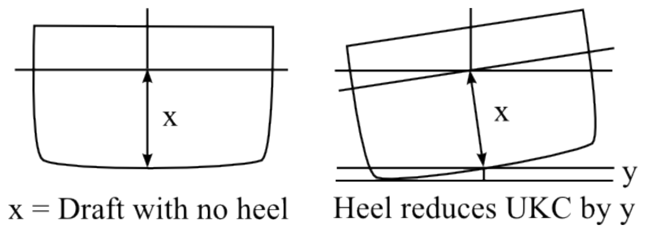


Figure 2.3: Effect of ship heeling on under-keel clearance

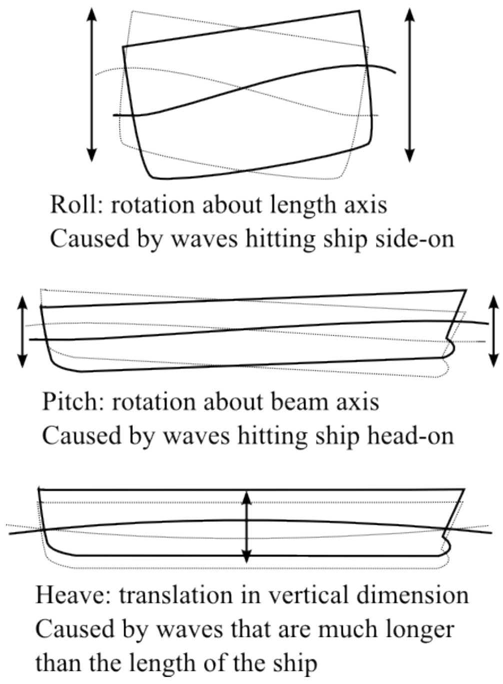

Roll: rotation about length axis
Caused by waves hitting ship side-on
Pitch: rotation about beam axis
Caused by waves hitting ship head-on
Heave: translation in vertical dimension
Caused by waves that are much longer
than the length of the ship

Figure 2.4: Vertical components of ship response to waves

the sum of the negative UKC factors (draft d, squat s, heel h, wave response w, etc) exceeds some safety factor F.

$$D(t) + T(t) - d(v) - s(v,t) - h(v,t) - w(v,t) \geq F \tag{2.1}$$

Ports may also have additional safety requirements that limit the times at which ships may sail, for example, some ports may have strong currents that restrict the times when large ships may transit the channel. Other ports do not allow high-risk ships such as oil tankers to transit outside of daylight hours.

2.1.2 Optimisation Problems in Maritime Transportation: Introduction

Optimisation problems in maritime transportation are traditionally divided into three levels: *strategic*, *tactical* and *operational*, based on the time-frame of decision-making. However, there is substantial overlap between these levels, partly because of unclear boundaries between them, and partly because even high-level problems like fleet size and mix or port capacity planning will depend on the details of the day-to-day demand for the resources in question.

Strategic planning problems include port capacity planning; optimisation of shipping fleet size and mix; liner shipping service and network design; and supply chain design.

Tactical planning problems include bulk cargo routing and scheduling; liner fleet deployment – the process of assigning ships to liner shipping routes; the Liner Shipping Fleet Repositioning Problem (LSFRP), which aims to minimise the cost of moving a ship between different liner routes; the Berth Allocation Problem; and the Maritime Inventory Routing (MIR) problem – the combined problem of coordinating inventory at various stages of a supply chain, as well as minimising the cost of routing cargo.

The boundary between tactical and operational planning problems is often unclear, but some problems clearly fall into the short-term operational planning horizon. These include environmental routing – the process of choosing routes that avoid floating ice or poor weather conditions, or take advantage of ocean currents to reduce fuel usage or travel times; and problems involving ship speed selection aimed to minimise fuel costs or emissions from shipping.

Section 2.2 provides an introduction to a number of general techniques that are used in this thesis, which are commonly used in maritime transportation problems as well as in related optimisation problems in other fields. Section 2.3 then discusses a few key optimisation problems in maritime transportation in more detail, particularly problems in bulk cargo ship routing and scheduling that are most closely related to the research presented in this thesis.

2.2 General Techniques

In this section we present an overview of optimisation techniques used in this thesis, which are either commonly used to solve problems in the field of maritime transportation, or techniques that have been used previously for related optimisation problems in other fields and which may be suitable for solving maritime transportation problems. These techniques include Mixed Integer Programming and Constraint Programming, with associated decomposition or simplification techniques such as Benders decomposition, Dantzig-Wolfe decomposition, set partitioning, Lagrangian Relaxation and column generation. The recently introduced technique of Lazy Clause Generation which combines Constraint Programming with propositional satisfiability is also discussed. Some maritime transportation problems are closely related to planning problems, so methods for solving planning problems are also briefly presented.

2.2.1 Mixed Integer Programming

Mixed Integer Programming involves minimising or maximising a linear function subject to linear constraints. Specifically, a Mixed Integer Program has the form:

$$\begin{aligned} \text{minimise } & c^T x \\ \text{subject to } & Ax = b \\ & l \leq x \leq u \\ & \text{some or all } x_j \text{ are integers} \end{aligned} \tag{2.2}$$

Mixed Integer Programs are typically solved using a *branch-and-bound* approach [Land and Doig, 1960]:

1. Continuous (Linear Programming) relaxation: allow all x to take non-integer values, and find the optimal real solution for both branches. If this solution gives integer values for all variables that are required to be integers, then we are done. Otherwise, do steps 2-4.

2. Branch: find a variable x_j that is required to be an integer, which has a non-integer value in the LP solution, say v_j. Create two new subproblems, one with the added constraint $x_j \leq \lfloor v_j \rfloor$, and the other with the added constraint $x_j \geq \lceil v_j \rceil$.

3. Bound: the solution to the LP relaxation provides a lower bound on the solution to the MIP problem. Any feasible integer solution found so far provides an upper bound, which is updated during the search as better solutions that satisfy integrality constraints are found. These upper and lower bounds can be used to prune branches during the search: if the lower bound for any branch is above the upper bound found so far, that branch cannot provide any better quality solutions, and can be pruned.

4. Solve subproblems: recursively solve the two subproblems created by branching using steps 1-4, until either a) the optimal solution to the continuous relaxation is infeasible, or b) the optimal solution to the continuous relaxation satisfies integrality constraints, or c) the branch can be pruned by step 3.

Other methods used for solving Mixed Integer Programs include:

1. The *cutting planes* method, which repeatedly finds optimal solutions under a continuous relaxation and adds additional inequalities or *cuts* to separate non-integer optimal solutions away from the feasible set.

2. *Branch-and-cut*, which combines a branch-and-bound search with cutting planes used to tighten the continuous relaxation at each step of the branch-and-bound.

3. *Column generation*, a method for solving large Mixed Integer Programs that would be infeasible to solve directly, by considering only a subset of variables at a time, repeatedly adding new variables (columns) that can improve the solution.

4. *Dantzig-Wolfe decomposition* [Dantzig and Wolfe, 1960] which splits a problem into several independent subproblems, and a master problem which enforces the coupling constraints between the subproblems.

See Wolsey [1998] and Schrijver [1986] for a more detailed introduction to Mixed Integer Programming.

MIP programs are usually solved using an existing solver, rather than by custom-building a new solver. Some existing MIP solvers include CPLEX, SCIP, Gurobi and CBC (Coin-OR).

2.2.2 Constraint Programming

Constraint Programming involves solving problems with more general constraints than Mixed Integer Programming – constraints and objectives can be non-linear, but on the other hand, in Constraint Satisfaction Problems (CSPs) and constrained optimisation problems, variables can only take discrete values ranging over a finite domain. Every MIP program with integer-only variables is also a constrained optimisation problem that can be solved using a finite domain solver; however MIP solvers may be able to solve the same set of constraints more efficiently due to being optimised for quickly solving a problem with linear constraints.

Solvers for CSPs and constrained optimisation problems typically work using *backtracking search*. The simplest form of backtracking search would exhaustively search the entire domain of every variable by assigning variables to values until either a solution is found or a constraint is violated, then backtracking and trying other values if an assignment results in some constraint being violated. This is similar to the *branch-and-bound* algorithm described above in Section 2.2.1, but backtracking search for CSPs typically uses depth-first search, where the domain of each variable is explored completely before exploring other variables.

However, exhaustively searching every combination of variable values is clearly extremely inefficient even for small variable domains. In practice, CP solvers combine backtracking search with *constraint propagation* – a method for using constraints to eliminate parts of the domains of each variable from the search space, thus avoiding having to explore all possible assignments of variables to values. Constraint propagation uses information about the domains of some variables in a constraint to eliminate infeasible values from the domains of other variables in the constraint, thus making the domains *consistent* with one another. If any variable's domain becomes empty, the CSP is unsatisfiable.

Constraint solvers differ in the ways they apply constraint propagation to reduce variable domains. Three basic constraint propagation rules are *node consistency*, *arc* *consistency* and *bounds consistency*. *Node consistency* applies constraints with one variable to reduce the domain of that variable, for example, the constraint $x > 3$ could be applied to reduce the domain of x from $\{1, 2, 3, 4, 5\}$ to $\{4, 5\}$. *Arc consistency* applies constraints with multiple variables to ensure that for every value of each variable, there exist combinations of values in the domains of the other variables that satisfy the constraint. For example, variables x, y with domains D_x, D_y are arc consistent with a constraint if for every value $u \in D_x$ there exists a value $v \in D_y$ such that $x = u, y = v$ satisfies the constraint.

Arc consistency becomes very time-consuming to apply for constraints with large numbers of variables, so a more common form of consistency used for constraints with many variables is *bounds consistency*, which applies constraints with multiple variables to remove extreme values from the range of a variable's domain, if these values are inconsistent with the extreme values of other variables in the constraint. *Bounds consistency* only needs to consider extreme values of all variables, instead of searching through all possible combinations of all variables in the constraint, which makes bounds consistency faster to evaluate than arc consistency for complex constraints.

Constraint solvers usually also implement specialised propagators for other constraints, which allows problems with those types of constraints to be solved faster by avoiding exhaustively searching all values. For example, most CP solvers implement an *alldifferent* constraint propagator, which can detect that a set of 8 integers each with the domain $\{1, 2, 3, 4, 5, 6, 7\}$ cannot be all different, without needing to exhaustively check every combination of assignments of variables.

Constraint solvers are more flexible than MIP solvers and able to model a wider range of problems, including those with non-linear constraints; however, as a result of this flexibility, constraint solvers may be slower than MIP solvers for problems that are naturally modelled with linear constraints. The efficiency of constraint solvers for a particular problem may also be highly dependent on the search strategy and constraint propagators used by the solver. Many solvers therefore allow the search strategy to be specified by the user so that the best search strategy can be chosen for the specific model. For example, the user may be able to specify the method used to select the order of variables to search on, as well as the method used for splitting the domain of each variable. See Rossi, Van Beek, and Walsh [2006] and Marriott and Stuckey [1998] for a more detailed introduction to Constraint Programming.

Some constraint programming solvers include Gecode, ECLiPSe, Choco, and the G12 system which integrates with many CP solvers and includes a number of built-in solvers such as a finite domain solver (G12-FD) and the CPX solver with Lazy Clause Generation (see below).

2.2.2.1 Lazy Clause Generation

Lazy Clause Generation (LCG) [Ohrimenko, Stuckey, and Codish, 2009] or Constraint Programming with learning is a recently introduced technique which has been found to be effective on a number of scheduling problems [Schutt et al., 2012; Feydy and

Stuckey, 2009; Chu et al., 2010]. LCG combines a finite domain CP solver with a propositional satisfiability (SAT) solver by mapping finite domain propagators to clauses in a SAT solver.

Mapping a complete CP problem to a SAT problem often results in a very large SAT problem which are intractable for even the best modern SAT solvers. LCG gets around this limitation by lazily adding clauses to the SAT solver as each finite domain propagator is executed, precisely at the point when the new clauses are able to trigger unit propagation. This approach benefits from efficient SAT solving techniques such as nogood learning and backjumping, while maintaining the flexible modelling of a CP solver and enabling efficient propagation of complex constraints [Ohrimenko, Stuckey, and Codish, 2009].

This approach is similar to *cutting planes* in Mixed Integer Programming, where additional constraints, or *cuts* are added to eliminate portions of the search space. However, in Lazy Clause Generation, the new constraints are automatically combined to form more powerful and more general constraints using methods from SAT. This allows the solver to learn areas of the search space where it previously failed, thus avoiding searching infeasible regions in future.

2.2.3 Planning

An automated planning problem involves several variables that can be in any of several states; a set of actions with prerequisites and effects on variable values; and an initial condition and goal state. Solving a planning problem requires finding a sequence of actions to transform the variable values from the initial state to the goal state, and may also involve optimisation on cost or number of actions.

Methods commonly used to solve planning problems include heuristic search in the state space, eg. [Bonet and Geffner, 2001]; using a planning graph [Blum and Furst, 1997] as the basis for heuristic search, eg. [Hoffmann, 2001]; converting the planning problem to a boolean satisfiability (SAT) problem and solving using one of the many efficient SAT solvers, eg. [Rintanen, 2010; Castellini, Giunchiglia, and Tacchella, 2003]; search in a space of partially ordered sets of actions, eg. [Coles et al., 2010a]; and decomposition approaches, eg. [Amir and Englehart, 2003; Brafman and Domshlak, 2006].

While some planning problems can be modelled as a Constraint Programing or Mixed Integer Programming problem, planning in general is a significantly different problem. CP and MIP are NP-complete problems, whereas Planning is PSPACE-complete, since Planning can involve an exponential number of actions in the solution, so the solution cannot be checked in polynomial time as is required for an NP-complete problem. The number of actions in the solution cannot be usefully bounded in general, whereas CP and MIP both have fixed numbers of variables, and thus can only be used to model a planning problem with a limited number of actions. On the other hand, planning problems all have a very specific structure, with actions, preconditions and effects, which limits the types of problems that planning solvers are able to solve.

However, for problems that do have this structure, a planning solver may be more efficient than a MIP or CP solver, since a planning solver may be able to take advantage of the tight constraints on preconditions and effects, which limit the actions can be taken and what actions are required to reach the goal, thus eliminating large areas of the search space. Tierney et al. [2012a] found that a planning solver was faster at solving the Liner Shipping Fleet Repositioning Problem than the MIP solvers which are traditionally used in the liner shipping domain, so planning may be worth investigating for other maritime problems.

See Ghallab, Nau, and Traverso [2004] for a more detailed introduction to automated planning.

2.2.4 Decomposition Techniques

Many large optimisation problems are too complex to be solved by standard solvers. However, decomposition techniques have been successfully used in many domains, including Mixed Integer Programming [Dantzig and Wolfe, 1960; Benders, 1962], Constraint Programming [Hooker and Ottosson, 2003] and Planning [Amir and Englehart, 2003; Brafman and Domshlak, 2006] to break problems into smaller, more tractable subproblems.

One method commonly used for solving large Linear Programming and Mixed Integer Programming problems is Benders decomposition [Benders, 1962], which decomposes the problem into a master and subproblem which are smaller and more tractable than the original problem. The two problems are solved iteratively, with the master problem providing a bound on optimal solution quality, and the subproblem providing *cuts* – new constraints – that are used to remove infeasible areas of the master problem domain from the search. Benders' decomposition involves adding new constraints to the master problem after each iteration of the subproblem, whereas by contrast, Dantzig-Wolfe decomposition uses column generation, which adds new variables to the master problem at each iteration.

While Benders decomposition was originally used only for linear and Mixed Integer Programming problems, in recent years it has been extended to a logic-based form that can be used for non-linear Constraint Programming problems [Hooker and Ottosson, 2003; Hooker, 2007]. Efficient logic-based Benders cuts for scheduling problems have been investigated for a number of objective functions, such as minimum makespan and minimum total tardiness.

Benders' decomposition and Dantzig-Wolfe decomposition are both effective techniques for solving large problems that have weakly-interacting variables and constraints, ie. problems that have some natural way of breaking them into a master and subproblem. However, Benders' decomposition is usable for both MIP and CP.

2.3 Optimisation Problems in Maritime Transportation

This section presents a more detailed review of several optimisation problems in maritime transportation, mostly routing and scheduling problems in bulk cargo shipping, which are introduced in Subsection 2.3.1. The rest of this section reviews a number of specific problems in more depth, particularly problems that are extended with time-varying draft restrictions or solved with new approaches in the rest of this dissertation.

Subsection 2.3.2 reviews existing research on the Vehicle Routing Problem with Soft Time Windows (VRPSTW), as well as ship scheduling and routing problems that include soft time windows. Soft time windows result in action costs that vary with time, which is closely related to the effect of adding time-varying draft restrictions to a maritime transportation problem.

Subsection 2.3.3 presents the literature on ship speed and cost optimisation, with fuel usage modelled as a function of speed. Some research considers ship speed optimisation in the context of a routing and scheduling problem, whereas other research investigates queueing strategies at ports, or optimising speeds for a ship travelling along a fixed route.

Subsection 2.3.4 presents the Liner Shipping Fleet Repositioning Problem (LSFRP), which deals with minimising the cost of moving container ships between services in a liner shipping network. This problem involves both routing and scheduling, as the best route and time for moving each ship must be chosen.

Subsection 2.3.5 presents an overview of problems in maritime transportation that currently consider draft restrictions, as a comparison of the current state-of-the-art, before we introduce our first maritime transportation problem with time-varying draft restrictions in the next chapter.

What we do not cover

Most problems in container shipping are not covered in depth, since optimisation problems in container shipping are generally focussed on network design, assignment of ships to routes, or land-side container terminal optimisation, which are less closely related to the problems investigated in this thesis. For a detailed review of the maritime side of container shipping problems, see Christiansen et al. [2013] and Christiansen et al. [2007]. For a review of land-side container terminal operations, see Stahlbock and Voss [2008].

We also generally do not consider problems involving the design and management of an individual ship; cargo stowage; naval logistics; and highly specialised problems in specific industries, such as offshore logistics for transporting cargo between oil and gas platforms and offshore depots. Many of these types of problems are discussed by Christiansen et al. [2013] and Christiansen et al. [2007]. Problems in port capacity planning and design are also omitted; see Agerschou et al. [2004] for an in-depth overview of this area.

2.3.1 Bulk Cargo Ship Routing and Scheduling

Ship routing problems involve determining the most cost-effective routes for a set of ships to transport a set of cargoes with fixed origins and destinations. Ship scheduling problems also involve selecting the best sailing times for those ships, subject to constraints. In ship scheduling and routing problems, constraints may include the amount and type of cargo that can be carried on each ship, allowable sailing times, and other more complex constraints that vary between specific problems.

In bulk cargo shipping, there are two main problem types that involve routing and scheduling decisions – the *cargo routing* problem and the *maritime inventory routing* problem [Al-Khayyal and Hwang, 2007]. The *cargo routing* problem involves allocating a fixed set of cargoes to ships, and is a variant of the well-known pickup and delivery problem. The *maritime inventory routing* problem involves finding a route and schedule of ships that maintains inventory quantities at a set of loading and discharge ports within specified limits – the number of cargoes and their pickup and delivery windows are not known in advance, and need to be determined as part of the solution.

Cargo Routing

In cargo routing, cargoes may have associated time windows for pickup and delivery, and there may be constraints on which ships may carry which cargoes. Some problems have considered soft time windows, where there is a penalty for arriving outside of the customer's preferred time windows [Fagerholt, 2000, 2001; Christiansen and Fagerholt, 2002; Gatica and Miranda, 2011]. This scenario matches real-world conditions more closely, and can allow lower-cost schedules to be found at a small inconvenience to some customers. Some approaches also consider multiple time windows, for example due to ports being closed overnight or on weekends [Christiansen and Fagerholt, 2002; Kobayashi and Kubo, 2010].

Some cargo routing problems only allow ships to carry one cargo at a time [Brown, Graves, and Ronen, 1987; Fisher and Rosenwein, 1989; Kim and Lee, 1997; Hwang, 2005; Gatica and Miranda, 2011]. However, most problems investigated in recent years allow multiple cargoes to be carried per ship, subject to capacity constraints and ship suitability constraints, eg. [Brønmo et al., 2007; Korsvik, Fagerholt, and Laporte, 2011; Norstad, Fagerholt, and Laporte, 2011].

Some problems may allow splitting a cargo between several ships [Fagerholt, 2004; Korsvik, Fagerholt, and Laporte, 2011], and some problems consider flexible cargo quantities, where each cargo has a range of quantities that can be transported, with the revenue being dependent on the actual amount of cargo carried [Brønmo, Christiansen, and Nygreen, 2007; Korsvik and Fagerholt, 2010; Brønmo, Nygreen, and Lysgaard, 2010].

If there are insufficient ships to carry all cargoes, the fleet operator may charter extra ships for a single voyage (spot charters) to transport any excess cargoes. Alternatively, if there is excess capacity in the fleet, optional extra cargoes (spot cargoes) can be carried to gain additional revenue. Some cargo routing problems also consider decisions on ship speed [Bausch, Brown, and Ronen, 1998; Norstad, Fagerholt, and Laporte, 2011; Gatica and Miranda, 2011; Psaraftis, 2012]. The approach of Fagerholt [2001] considered variation in ship speed in post-processing, even though speed was not a decision variable during the routing. To the author's knowledge, variable ship speed has not yet been investigated in the context of maritime inventory routing.

Maritime Inventory Routing

The maritime inventory routing problem is usually found in the manufacturing industry, where large quantities of bulk materials are regularly produced at one port, then shipped to another port for processing. Instead of a pre-determined set of cargoes, the level of inventory at ports is required to be maintained within given limits, which may arise from storage capacity constraints, as well as the need to avoid idle time on expensive processing equipment. The problem may involve only one type of product [Halvorsen-Weare and Fagerholt, 2013; Song and Furman, 2010], or multiple types with constraints on what ships can be used to carry which products [Hwang, 2005; Al-Khayyal and Hwang, 2007; Christiansen et al., 2011; Rakke et al., 2011]. Some problems with multiple products only allow ships to carry one product type at a time [Rakke et al., 2011]. Others allow ships to carry multiple cargoes [Hwang, 2005; Al-Khayyal and Hwang, 2007; Christiansen et al., 2011]. All cargoes have inherently flexible quantities and are split among several ships. As with the cargo routing problem, spot charters may be used to carry additional cargoes if the fleet capacity is not sufficient.

The most common solution approaches for the cargo routing problem include formulating the problem as a MIP model that is then solved using a branch-and-cut algorithm, column generation or Dantzig-Wolfe decomposition; set partitioning models; heuristic search approaches; and less commonly, genetic algorithms. The maritime inventory routing problem is typically formulated using a MIP model and solved using branch-and-cut or branch-and-price approaches, heuristic search, rolling horizon heuristics, or genetic algorithms.

Other Problems

Several routing and scheduling problems in bulk cargo shipping have also been investigated which do not fall into either of the above problem types.

Rakke et al. [2012] considered a Traveling Salesman Problem (TSP) with draft limits at ports limiting the order in which ports can be visited. This problem involved routing decisions; however, no scheduling decisions or time windows were considered. The draft constraints were modelled as constant, rather than varying over time.

On the other hand, the problem of optimising ship speeds along a fixed route, investigated by Ronen [1982]; Fagerholt, Laporte, and Norstad [2010]; Meng and Wang [2011] involves no routing decisions, since the route is provided as input to the problem, but does involve scheduling decisions on the time when each ship arrives and departs from each port.

Another problem which involves scheduling but no routing decisions is the Berth Allocation Problem, which involves scheduling ship arrival times at a single port, as well as assigning ships to berths for loading and unloading. This problem is most commonly considered in the context of container shipping, for example by Cordeau et al. [2005]; Lee and Chen [2009]; Du et al. [2011]. However this has also been researched in the context of bulk shipping, for example by Kao and Lee [1996]; Álvarez, Longva, and Engebrethsen [2010].

A typical objective for the Berth Allocation Problem is minimising the waiting time for ships, as well as maximising the utilisation of berths, as well as other expensive infrastructure. Kao and Lee [1996] investigated the Berth Allocation Problem with the objective of minimising demurrage (fees paid to ship owners for tardiness) by scheduling arrival times of ships at the port to minimise congestion and reduce ship waiting times. They were able to increase the utilisation rate of berths and reduce the amount of time ships spend waiting for a berth to become available. Lee and Chen [2009] considered a Berth Allocation Problem where the berths are continuous rather than discrete segments. Cordeau et al. [2005] considered both discrete and continuous berths, and incorporated quay crane allocation. Their objective was minimising the service time for ships – the sum of the waiting time and the loading time.

Du et al. [2011] and Álvarez, Longva, and Engebrethsen [2010] considered the Berth Allocation Problem with an objective of minimising fuel costs, by considering variable ship speed, scheduling ships to sail more slowly, and investigating alternative queueing rules from the traditional first-come first-served rule at ports. These approaches maximised berth utilisation while also minimising fuel costs. Du et al. [2011] also minimised emissions along with fuel usage.

2.3.2 Routing and Scheduling with Soft Time Windows

Ship routing and scheduling with soft time windows is a special case of the Vehicle Routing Problem with Soft Time Windows (VRPSTW), though the constraints involved in a maritime problem differ somewhat from the constraints involved in vehicle routing.

In vehicle routing, time windows allow the modelling of real-world constraints on delivery deadlines. However, hard time windows often result in tightly constrained problems which require a larger fleet of vehicles in order to meet deadlines. The VRPSTW specifies penalties for early and late arrival outside each customer's preferred time window instead of hard time window constraints. Soft time windows allow better utilisation of vehicles, thus reducing transportation costs, while still servicing most customers within their preferred time windows. However, soft time windows result in a more complex objective function, making the problem significantly more difficult to solve [Qureshi, Taniguchi, and Yamada, 2009].

Approaches to solving the VRPSTW differ in terms of the objectives being optimised for at various stages of the calculation. Some approaches optimise first for the number of vehicles (routes), then for minimal travel time and distance, then for minimal cost only as a final objective with time window penalties included. One such approach was presented by Figliozzi [2010], who used an iterative route construction and improvement algorithm where the improvement step initially attempted to merge routes with underutilised vehicles to reduce fleet size, and then attempted to reduce the total route duration and time window penalties. This approach was further extended by Figliozzi [2012] to take into account time-dependent travel times, which result in much more complex constraints between tasks, but which allow the routing solutions to take into account time-varying speed limits, or travel times being affected by peak hour traffic.

Other VRPSTW approaches optimise for a combined weighted objective that includes all of the above costs, such as the work of Qureshi, Taniguchi, and Yamada [2009], who investigated an exact solution approach for the VRPSTW based on column generation, and compared results of different levels of time window relaxation against solutions with hard time windows.

VRPSTW approaches also differ in the types of penalties modelled. For example, Qureshi, Taniguchi, and Yamada [2009] only applied penalties for lateness, with no penalties for arriving early, whereas other work such as that of Figliozzi [2010] and Fan et al. [2011] includes penalties for both late and early arrival.

VRP problems also vary widely in the types of side constraints considered in each problem. The soft time window constraints themselves are one example of side constraints that are not considered by all VRP problems. Many problems also consider other additional side constraints, such as the time-dependent travel times investigated by Figliozzi [2012]. One of the most general approaches in recent years was presented by Kilby and Verden [2011], who combined large neighbourhood search [Shaw, 1997] with Constraint Programming to solve vehicle routing problems. Their Indigo vehicle routing solver implements some constraints natively, for faster calculation, but also integrates with a CP solver to allow flexible side constraints to be added as required, such as a "blood bank" constraint requiring that deliveries be made within 20 minutes of pickup. Indigo implements time window constraints natively, and can consider soft time windows with linear early and late penalties. The CP system uses backtracking search with Indigo acting as the variable/value choice heuristic – if a variable/value assignment made by Indigo leads to a CP constraint being violated, this information gets propagated back to Indigo, to prevent the Large Neighbourhood Search from making the same assignment in future. The Large Neighbourhood Search destroys part of the solution in each iteration, and uses insertion heuristics to repair the partial solution.

Ship routing and scheduling problems are very similar to the VRP; however, soft time windows have rarely been considered in maritime transportation. Perhaps because of this, many approaches to ship routing and scheduling with soft time windows compare the impact of different penalty functions on costs or other aspects of the solution. This includes the work of Fagerholt [2001] and Fagerholt [2000], who compared several penalty functions for deliveries outside the preferred window which could be used for different situations, e.g. when a customer gets a fixed discount for a delivery outside the preferred time, or when the discount depends on the lateness of the delivery. They were able to achieve in a 15% reduction in shipping costs by shipping 12% of cargo outside the preferred delivery times, showing that relaxing time windows can produce substantial benefits at a small inconvenience to customers. Another related approach was presented by Karlaftis and Kepaptsoglou [2009], who investigated the effect of varying time window penalties on the average delay for a container ship routing problem, finding that a 5% penalty for delay resulted in 3 times longer average delays compared to a 100% penalty for delay.

An alternative penalty function was investigated by Halvorsen-Weare and Fagerholt [2013], who considered a maritime inventory routing problem with a requirement that deliveries be evenly spread over the scheduling period. This constraint was enforced by breaking the time range up into a set of cargoes each with an inner and outer time window, where the inner time windows represent target delivery dates that satisfy the "evenly spread" requirement. Penalties were applied for early and late delivery outside the inner time window, and the outer time windows were considered hard constraints in the optimisation model, to enforce a maximum departure from the target delivery date.

Another interesting time window penalty function was presented by Christiansen and Fagerholt [2002], who investigated penalties for ships arriving at port close to weekends, to avoid the risk of long delays caused by some ports closing on weekends. This problem included multiple soft time windows – a situation rarely explored in the literature, with the exception of a few vehicle routing problems. The penalty function peaked before weekends, and dropped to 0 at the start of each week. The objective function was a weighted sum of the transportation costs and the penalty costs for "risky" arrival times.

Our work on introducing time-varying draft constraints into maritime transportation problems could also be considered as a problem with soft time windows, where the penalty function is the opportunity cost of not being able to carry extra cargo if the ship sails outside the time period around the high tide when it is able to sail with maximum cargo. This penalty function is similar to that for a soft time window with late and early arrival penalties; however, for any problem where the time range covers multiple high tides, there will also be multiple soft time windows, one for each high tide.

2.3.3 Ship Speed Optimisation

Rising fuel prices in recent years have led to increased interest in problems that involve optimising shipping speeds to minimise fuel consumption [Christiansen et al., 2007, 2013; Psaraftis and Kontovas, 2013]. Fuel costs can be on the order of US$100,000 per day, and can form up to 75% of the operating costs for a large cargo ship [Ronen, 2011]. Fuel consumption over time can be approximated as a cubic function of ship speed [Ronen, 1982], thus optimising speeds to minimise shipping costs can produce large reductions in both shipping costs and CO2 emissions.

Some of the earliest papers looking at bulk cargo ship routing and scheduling with variable ship speeds include Ronen [1982], who analysed the trade-off between savings in fuel cost caused by sailing slower against the increased costs caused by a longer voyage duration. Brown, Graves, and Ronen [1987] and Bausch, Brown, and Ronen [1998] considered ship schedule optimisation problems with variable ship speeds and hard time windows for delivery of each cargo. In the research by Fagerholt [2001] on ship routing and scheduling with soft time windows, the soft time windows also allowed cost reduction via better allocation of routes or better optimisation of sailing speeds, with a small inconvenience to customers.

In liner shipping, Ronen [2011] looked at the effect of oil price on the trade-off between reducing sailing speed and increasing fleet size for container ships, under the assumption that ship speeds are constant for entire voyage rather than varying per leg. Løfstedt et al. [2010] presented a set of benchmarks for liner shipping network design problems, with fuel costs taken into account. Meng and Wang [2011] optimised the service frequency, fleet deployment plan and sailing speeds for a liner service route, including variable sailing speeds at different legs of the route. They used a Mixed Integer Non-Linear Programming model, approximated by a MILP model to give a lower bound. Branch-and-bound was then used to find a solution for the MINLP within a given error tolerance. The Liner Shipping Fleet Repositioning Problem [Tierney et al., 2012a], discussed in more detail in Section 2.3.4 also involves decisions on the ship speed in each leg of the voyage. Notteboom and Vernimmen [2009] presented an overview of how liner service operators have adapted their service for high fuel prices in practice: reducing speed; adding new ships to maintain service frequency; assigning the largest vessels to the longest routes with the largest ratio of travel time to time spent in ports; and adapting the number of port calls on each route based on a tradeoff of demand vs cost.

Other problems that consider variable ship speed include Brown et al. [2007], who looked at varying ship engine modes to optimise the fuel consumption of a single vessel. Du et al. [2011] considered the problem of berth allocation with speed optimisation to reduce fuel usage and minimise congestion at ports. They found that ships sometimes sail below the most efficient speeds in some circumstances, and that costs could be improved by speeding up in those cases. The most efficient approach to berth allocation was found to be one that spread out ship arrival times at the port constantly over time, as this approach reduced congestion and minimised unused resources.

Ship speed optimisation has also been considered in the context of policy changes that can be put in place by governments and ports to encourage reduction of CO2 emissions. Corbett, Wang, and Winebrake [2009] investigated the effect on CO2 emissions of increasing fuel prices by means of a fuel tax, assuming that shipping companies would use speed optimisation to minimise shipping costs. Linstad, Asbjørnslett, and Strømman [2011] investigated the effect of speed reduction on CO2 emissions and costs and compared the estimated effectiveness of several policy changes (fuel tax, emissions trading and speed limits) aimed at reducing emissions. They found that substantial CO2 reductions can be achieved, but the world fleet would need to grow to reach full potential, due to more ships being required to meet transportation demand when ships sail slower. Speed limits were found to be the most effective way of reducing emissions, since shippers are willing to pay more for shorter transportation times, so an increase in fuel price would have to be very high to significantly affect emissions.

Álvarez, Longva, and Engebrethsen [2010] investigated improvements to port berthing policies and shipping contracts to reduce the incentive to sail fast. Many ports have first-come-first-served berthing policies, which contributes to port congestion thus introducing safety risks, and also encourages ships to sail at full speed rather than reducing costs and emissions by sailing slower. Álvarez, Longva, and Engebrethsen [2010] compared FCFS against two new berthing policies, and found that both alternative policies resulted in significant improvements in shipping costs and port congestion. Improving port berthing policies to reduce congestion and shipping costs has been trialled in practice – the port of Newcastle has recently introduced a "virtual arrival" system to reduce fuel costs and CO2 emissions, as well as improve ship queueing times [Corbett, 2009; Newcastle Port Corporation, 2010].

Returning to speed optimisation in the context of bulk cargo routing and scheduling, the two papers involving ship speed optimisation that are most closely related to the work presented in this thesis are Fagerholt, Laporte, and Norstad [2010] and Norstad, Fagerholt, and Laporte [2011]. Fagerholt, Laporte, and Norstad [2010] optimised the ship speed on each leg of a fixed liner shipping route, with time windows for arrival at each waypoint. Norstad, Fagerholt, and Laporte [2011] investigated a ship routing and scheduling problem with speed optimisation on each leg of each route. The approach of Fagerholt, Laporte, and Norstad [2010] was used to determine the profit contribution of each candidate route by optimising speeds for a single ship on each candidate route. The routing and scheduling problem was solved by generating a number of initial solutions and using a multi-start local search heuristic to improve the best solutions. Chapter 6 discusses both of these methods in more detail, and extends them to consider time-varying restrictions on ship draft at each waypoint.

Other approaches to bulk cargo routing and scheduling with ship speed optimisation consider slightly different cost functions and side constraints. One similar approach to Fagerholt, Laporte, and Norstad [2010] was investigated by Gatica and Miranda [2011], whose network-based modeling approach to the bulk cargo ship routing and scheduling problem also used discretisation of the time windows for pickup and delivery of cargoes in order to consider variable ship speeds. While their approach only considered fuel costs and fixed port charges, the network-based model can be easily extended to include soft time windows and port costs that vary with arrival time. The optimal navigation speed for each feasible arc between discretised pickup and delivery times was calculated by an external model, and the results estimated a potential cost reduction of around 21% on average compared to a fixed-speed continuous model.

Most earlier work on ship speed optimisation, including Fagerholt, Laporte, and Norstad [2010], Norstad, Fagerholt, and Laporte [2011] and Gatica and Miranda [2011], ignored costs that were not affected by speed, such as the charter freight rate of ships. One exception is the work of Psaraftis [2012], who presented a ship pickup-and-delivery problem with variable speeds which considered both the fuel cost and the charter freight rate of ships. Considering these costs together results in a more realistic model that matches the observed behaviour of shipping companies in response to changes in fuel cost and freight rate. As demand for shipping grows and freight rate increases, ships sail faster; on the other hand, during economic downturns, when the demand for shipping and the freight rates are both lower, ships sail slowly to minimise fuel consumption. Freight rate is paid per day, so travelling

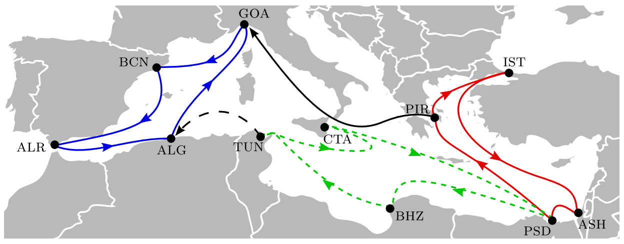


Figure 2.5: An example repositioning scenario with two vessels (solid and dashed black lines) reposition from their initial services to the goal service.

slower reduces fuel cost, but increases the freight rate. Psaraftis [2012] also considered a "delay cost" of the cargo – the cost of storage of the cargo prior to pickup, as well as the opportunity cost of lost revenue due to delayed delivery.

Finally, Vilhelmsen, Lusby, and Larsen [2013] considered a bulk cargo ship routing and scheduling problem with decisions on refuelling locations and amounts, to take into account variability in the price of fuel between times and locations, as well as limitations on the combined quantity of fuel and cargo that can be loaded onto a ship simultaneously. The results showed a small increase in profit – around 0.5% – by considering these additional variables in the routing decisions; however, in an industry with such huge numbers as bulk shipping, even a small increase in profit can still be significant.

It is clear that ship speed optimisation decisions affect many different aspects of shipping cost. Our work on bulk cargo routing and scheduling with speed optimisation and time-varying draft restrictions considers both the fuel cost and charter freight rate, similarly to Psaraftis [2012]; however, we also consider the cost impact of ship draft, as well as the impact of time-varying draft restrictions on allowable sailing times at waypoints and the effect this has on waiting times. Our approach builds on the algorithms presented by Fagerholt, Laporte, and Norstad [2010] and Norstad, Fagerholt, and Laporte [2011], which are described in more detail in Chapter 6.

For a more detailed review and taxonomy of maritime transportation problems involving variable speed, see Psaraftis and Kontovas [2013].

2.3.4 The Liner Shipping Fleet Repositioning Problem

The Liner Shipping Fleet Repositioning Problem (LSFRP) [Tierney et al., 2012a] is one example of a problem in container shipping that involves decisions on the speed of each ship on each leg of the voyage. It is also similar to problems with time-varying draft restrictions in that the action costs vary with time. We discuss it in more detail in this thesis, since it is investigated further in Chapter 7 to test how well the techniques investigated in this thesis can be generalised to other maritime shipping problems.

Liner shipping networks must be regularly updated in order to adjust them to seasonal demands and shifts in the world economy. In order to do this, shipping lines *reposition* ships between *services* in the network. A service is a cyclical route

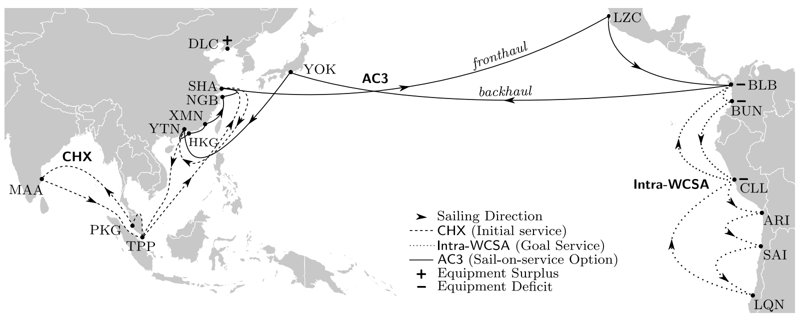


Figure 2.6: A subset of a real-world repositioning scenario, from Tierney et al. [2012a].

consisting of a sequence of ports with fixed visitation times. Services can be thought of like bus routes, in that each port is visited on a regular (i.e., periodic) frequency. Many shipping lines set this frequency to be one or two weeks, meaning each port on a service is visited by a vessel every week or every other week. This regularity allows liner shippers, who are the customers of shipping lines, to more easily integrate cargo shipments into their international supply chains.

Given a set of vessels and a goal service, where each vessel is assigned an initial service, the aim of the LSFRP is to reposition each vessel to the goal service within a given time period at minimal cost. Figure 2.5 shows an example repositioning at an abstract level. Two vessels (black lines) are repositioned to their goal service (blue, left).

A vessel begins repositioning when it *phases out* from its original service, and ends repositioning when it *phases in* to its new service. The total cost of the repositioning depends on the fuel usage from sailing the vessel between ports, and a fixed hourly cost, called the *hotel cost*, that is paid for the time between the phase-out and phase-in. The cost function for a sailing action decreases as the duration of a sailing increases because less fuel is needed to sail at slow speeds than at fast speeds. We use a linear approximation of vessel fuel consumption in this work. The cost of sailing also depends on whether the vessel sails empty or carries empty containers. Carrying empty containers is called a *sail-equipment* (SE) opportunity. Another option for repositioning vessels is to replace another vessel in a regular service, which is called a *sail-on-service* (SoS) opportunity. SoS opportunities significantly reduce the cost of a repositioning, but may increase sailing duration, due to the delay in loading and unloading cargo. The times at which these sailings may be performed may also be limited. Our model also includes *cabotage restrictions*, which prevent ships from violating laws in certain countries that protect domestic shipping routes from international competition. The LSFRP is not a pure scheduling problem, as there is a choice of actions and action durations.

Figure 2.6 shows a subset of a repositioning scenario in which the new Intra-WCSA service requires three vessels that must be repositioned from their services in South-East Asia. One of the vessels was originally on the CHX service, and the other two were on other services not shown in this figure. The cost of repositioning in this scenario can be reduced by carrying equipment from China to South America (e.g. DLC to BLB), or using the AC3 service as an SoS opportunity.

2.3.5 Draft Restrictions in Maritime Transportation Problems

The majority of ship routing and scheduling algorithms either ignore draft entirely, or mention draft restrictions only in passing as one of the possible reasons for a ship being incompatible with some ports. Some papers specifically list draft restrictions as one of the issues that are left to human schedulers to consider, for example Fagerholt [2004].

The few maritime scheduling papers that do consider draft restrictions at ports usually only model them as constant limits that do not vary with time, for example, assuming that a given port will always allow ships with a draft of 13 metres or less, and never allow ships with deeper drafts to enter. Examples of papers with constant draft limits include Christiansen et al. [2011]; Korsvik, Fagerholt, and Laporte [2010]; Bausch, Brown, and Ronen [1998]; Fisher and Rosenwein [1989]; Brown, Graves, and Ronen [1987].

Berth allocation problems such as Cordeau et al. [2005]; Lee and Chen [2009] also sometimes consider draft, since draft restrictions may limit which ports are compatible with which berths. However, these approaches have only considered draft as a static constant limit. In the context of a berth allocation problem, this is a reasonable simplification, since the main factor affecting berth draft limits is the water depth at low tide, rather than the depth of water at a range of times.

Another interesting problem involving draft restrictions was introduced by Rakke et al. [2012], who investigated a Travelling Salesman Problem involving a ship delivering cargo to a set of ports, each of which has an associated draft limit. The draft of the ship depends on the amount of cargo loaded, so the ports must be visited in an order that conforms to the draft limits. Again, only constant draft limits were considered, but in this problem, ships can enter and leave each port close to high tide, so extending this problem to model the draft restrictions as varying over time may allow lower-cost tours to be found in future.

To the author's knowledge, the only existing ship routing and scheduling approach which has considered time variation in draft restrictions is Song and Furman [2010]. This problem includes draft limits at ports that vary with the seasons, with one constant draft being modelled per day. Variation between high and low tide is not yet included, but the model could be extended to consider tidal variation by increasing the time resolution from one-day time slots to a fine resolution that could capture the tidal variation in draft restrictions. This would increase the size of the problem, making it harder to solve, but may result in schedules that allow ships to load more cargo. Song and Furman [2010] don't compare their model against one which approximates the time-varying draft restrictions with a constant limit, but this would also be interesting to investigate, to see how much time-varying draft restrictions improve the solutions.

2.3.6 Summary

Table 2.1 summarises the bulk cargo ship routing and scheduling problems presented in this chapter, including both problems in *cargo routing* and *maritime inventory routing* (MIR).

Time windows for each problem are specified as either hard or soft, with multiple time windows and quantity-dependent time windows also indicated. All MIR problems have inventory-dependent implicit time windows; however, Halvorsen-Weare and Fagerholt [2013] also consider soft time windows with penalties.

This table also lists whether each paper considers optional cargoes (ie. cargoes that can be transported by a spot charter, or optional extra cargoes that can be carried in the event of excess capacity in the fleet), as well as flexible cargo amounts and multiple cargoes per ship. Variable ship speed is considered in some papers, with Fagerholt [2001] using post-processing to select ship speeds, and Vilhelmsen, Lusby, and Larsen [2013] assuming that all ships sail at load-dependent optimal speeds. All papers that consider draft restrictions are also indicated, as either constant, manual, or time-dependent.

Table 2.2 summarises all the other problems discussed in this chapter, including the Liner Shipping Fleet Repositioning Problem (LSFRP); the Travelling Salesman Problem (TSP) with draft limits investigated by Rakke et al. [2012]; ship speed optimisation problems for ships travelling along one route, some of which also consider the problem of determining the optimal fleet size for that route, as extra ships may be required to maintain the desired frequency of service with slower ship speeds. This table also lists berth allocation problems and vehicle routing problems with soft time windows (VRPSTW).

For each problem type, this table lists whether the problem deals with container or bulk cargo (except for the VRPSTW); whether the problem involves routing decisions; whether there are no time windows, or whether they are hard, soft (with penalties at edges), or there is only a requirement for service frequency. For problems that involve any decisions on the routing of cargoes, we also specify whether the cargo amounts are flexible, and whether ships/vehicles can carry multiple cargoes simultaneously. The table lists whether each problem considers draft restrictions and variable ship speed – Kao and Lee [1996] consider variable ship speeds implicitly by varying ship arrival times, though speed is not explicitly considered as a decision variable, and fuel costs are not considered. We also specify whether the problem involves optimisation at one port, or multiple ports.

Several problems consider draft restrictions, but the impact of time-varying draft restrictions on scheduling has not yet been investigated. The most basic requirement for ship routing and scheduling is that schedules must satisfy safety rules on maximum sailing draft at every port. However, since many ports around the world are both tidal and draft-restricted, scheduling approaches that use overly simple approximations to draft constraints may miss the opportunity to load a ship to a deeper draft at high tide, or when waves are low, leading to suboptimal schedules. Since many ports around the world are draft-restricted, there may be many problems where con-

Paper	Problem	Time Windows	Optional Cargoes	Flexible Cargo Amounts	Multiple Cargoes	Split Loads	Variable Speed	Draft
Brown, Graves, and Ronen [1987]	CargoRouting	Hard	Yes	No	No	No	Yes	Const
Fisher and Rosenwein [1989]	CargoRouting	Hard	Yes	No	No	No	No	Const
Kim and Lee [1997]	CargoRouting	Hard	Yes	No	No	No	No	No
Bausch, Brown, and Ronen [1998]	CargoRouting	Hard	Yes	No	Yes	No	Yes	Const
Fagerholt [2000]	CargoRouting	Soft	Yes	No	Yes	No	No	No
Fagerholt and Christiansen [2000]	CargoRouting	Hard	Yes	No	Yes	No	No	No
Fagerholt [2001]	CargoRouting	Soft	Yes	No	Yes	No	PostProc	No
Christiansen and Fagerholt [2002]	CargoRouting	Soft, multiple	Yes	No	Yes	No	No	No
Fagerholt [2004]	CargoRouting	Hard	Yes	No	Yes	Yes	No	Manual
Brønmo et al. [2007]	CargoRouting	Hard	Yes	No	Yes	No	No	No
Brønmo, Christiansen, and Nygreen [2007]	CargoRouting	Hard	Yes	Yes	Yes	No	No	No
Hwang, Visoldilokpun, and Rosenberger [2008]	CargoRouting	Hard	Yes	No	No	No	No	No
Brønmo, Nygreen, and Lysgaard [2010]	CargoRouting	Hard, quantity dep.	Yes	Yes	Yes	No	No	No
Kobayashi and Kubo [2010]	CargoRouting	Hard, multiple	Yes	No	Yes	No	No	No
Korsvik and Fagerholt [2010]	CargoRouting	Hard	Yes	Yes	Yes	No	No	No
Korsvik, Fagerholt, and Laporte [2010]	CargoRouting	Hard	Yes	No	Yes	No	No	Const
Gatica and Miranda [2011]	CargoRouting	Soft	No	No	No	No	Yes	No
Korsvik, Fagerholt, and Laporte [2011]	CargoRouting	Hard	Yes	No	Yes	Yes	No	No
Norstad, Fagerholt, and Laporte [2011]	CargoRouting	Hard	Yes	No	Yes	No	Yes	No
Psaraftis [2012]	CargoRouting	None	No	No	Yes	No	Yes	No
Vilhelmsen, Lusby, and Larsen [2013]	CargoRouting	Hard	Yes	No	Yes	No	LoadDep	No
Hwang [2005]	MIR	InvDep	Yes	Yes	Yes	Yes	No	No
Al-Khayyal and Hwang [2007]	MIR	InvDep	Yes	Yes	Yes	Yes	No	No
Song and Furman [2010]	MIR	InvDep	No	Yes	No	Yes	No	TimeDep
Christiansen et al. [2011]	MIR	InvDep	Yes	Yes	Yes	Yes	No	Const
Rakke et al. [2011]	MIR	InvDep	Yes	No	No	Yes	No	No
Halvorsen-Weare and Fagerholt [2013]	MIR	InvDep, penalties	Yes	No	No	Yes	No	No

Table 2.1: Routing and scheduling problems in bulk shipping. All involve routing between multiple ports.

Paper	Cargo Type	Problem	Routing	Time Windows	Opt. Cargo/ Ship	Flexible Cargo Amounts	Multiple Cargoes	Variable Speed	Single Port	Draft
Tierney et al. [2012a]	Container	LSFRP	Yes	Hard	Yes	Yes	No	Yes	No	No
Tierney and Jensen [2012]	Container	LSFRP	Yes	Hard	Yes	Yes	No	Yes	No	No
Tierney et al. [2013]	Container	LSFRP	Yes	Hard	Yes	Yes	No	Yes	No	No
Rakke et al. [2012]	Bulk	TSP	Yes	None	No	No	Yes	No	No	Const
Ronen [1982]	Bulk	SpeedOpt, 1Route	No	Soft	No	-	-	Yes	No	No
Fagerholt, Laporte, and Norstad [2010]	Bulk	SpeedOpt, 1Route	No	Hard	No	-	-	Yes	No	No
Meng and Wang [2011]	Container	FleetSize, SpeedOpt, 1Route	No	Freq	No	-	-	Yes	No	No
Ronen [2011]	Container	FleetSize, SpeedOpt 1Route	No	Freq	No	-	-	Yes	No	No
Kao and Lee [1996]	Bulk	BerthAlloc	No	Soft	No	-	-	Implicit	Yes	No
Cordeau et al. [2005]	Container	BerthAlloc	No	Hard	No	-	-	No	Yes	Const
Lee and Chen [2009]	Container	BerthAlloc	No	Hard	No	-	-	No	Yes	Const
Álvarez, Longva, and Engebrethsen [2010]	Bulk	BerthAlloc	No	Soft	No	-	-	Yes	Yes	No
Du et al. [2011]	Container	BerthAlloc	No	Soft	No	-	-	Yes	Yes	No
Qureshi, Taniguchi, and Yamada [2009]	-	VRPSTW	Yes	Soft	No	No	No	No	No	No
Figliozzi [2010]	-	VRPSTW	Yes	Soft	No	No	No	No	No	No
Fan et al. [2011]	-	VRPSTW	Yes	Soft	No	No	No	No	No	No
Kilby and Verden [2011]	-	VRPSTW	Yes	Soft	Yes	Yes	Yes	No	No	No
Figliozzi [2012]	-	VRPSTW	Yes	Soft	No	No	No	No	No	No

Table 2.2: Other maritime optimisation and vehicle routing problems

sideration of draft restrictions would improve cargo throughput and reduce shipping costs.

The next chapter begins the main contribution of this thesis by introducing a novel problem in maritime transportation – maximising cargo throughput at a bulk export port with time-varying draft restrictions. Later chapters of the thesis investigate a number of approaches to the bulk port cargo throughput optimisation problem, as well as introduce time-varying draft restrictions into a cargo routing problem with ship speed optimisation, optional cargoes, split loads and flexible cargo sizes. The final chapter looks beyond these two problems to other problems in related fields which have time-varying action costs, and considers the possibility of generalising approaches between these problem types.

Chapter 3

Optimising Cargo Throughput at a Bulk Export Port

3.1 Introduction

This chapter introduces the problem of optimising cargo throughput for a bulk export port with time-varying draft restrictions. This begins the main contribution of the thesis: extending maritime transportation problems to be able to deal with time-varying draft restrictions, and investigating the impact of time-varying draft restrictions on schedule quality.

The problem arises from a real-world situation at Port Hedland in Western Australia – the world's largest bulk export port, exporting 238 million tonnes of iron ore in the 2011-12 financial year [Port Hedland Port Authority, 2012]. Port Hedland is draft-restricted, with a very high tidal variation – there can be up to 6 metres difference between high and low tide. The biggest high tides may allow Capesize bulk carriers to sail with up to 18.5m draft, whereas at low tide, only very small or empty (inbound) ships are able to sail.

Maximising cargo throughput is an important problem for ports exporting major bulk cargoes such as iron ore and coal. There is typically more cargo available for shipping than can be loaded onto the set of ships currently at the port, so if the schedule of ship sailing times can be optimised to increase capacity on a ship, this extra capacity can be used to carry extra cargo. This reduces future shipping costs, since that cargo will not need to be transported on a later ship.

3.1.1 How Ship Sailing Times are Scheduled in Practice

In practice, scheduling ship sailing times and drafts at an individual port is done manually at present, or using simple tools such as Microsoft Excel. Until recently, most ports used simple rules to calculate when ships can sail and how much cargo they can carry. These rules have the benefit of being easy to calculate, and easier to use for scheduling, since the draft limits for each ship do not change as the time of sailing approaches. However, such rules are quite conservative, as they must make allowances for uncertainty in environmental conditions.

In recent years, some ports have begun to use computerised systems that use live measurements and forecasts for tides, currents and waves to calculate under-keel clearance – and therefore allowable sailing draft – more precisely. These more accurate calculations of under-keel clearance allow more cargo to be loaded safely in good weather conditions, as well as improving safety in poor weather conditions, compared to simpler static rules that don't take live environmental conditions into account.

Ports that switch from a static under-keel clearance rule to using the DUKC® software report being able to ship millions of tonnes more cargo each year due to more accurate predictions of under-keel clearance, resulting in less allowance for uncertainty being required. This can translate to hundreds of millions of dollars in increased revenue per year for shippers [OMC International, 2013]. However, this makes the problem of scheduling ship sailing times at the port more complex, since the allowable sailing times vary for each ship, and may be updated as real-time environmental conditions and forecasts change. This software is currently used by 22 ports worldwide, which makes it the most widely used under-keel clearance calculation software in the world [OMC International, 2013], and the number of ports using such automated software is likely to continue to increase in future.

Port Hedland uses the DUKC® to calculate allowable sailing drafts, which has allowed over 1 million tonnes more cargo to be shipped from the port per year [Port Hedland Port Authority, 2013b]. Port Hedland is also rapidly growing in export capacity, and the large numbers of ships sailing on each tide, together with the complexity of having a different range of allowable sailing times for each ship, make the problem of scheduling ship sailing times on each tide too complex for human schedulers to solve optimally. This makes an automated system for scheduling ship sailing times and drafts particularly valuable for Port Hedland.

3.1.2 The Bulk Port Cargo Throughput Optimisation Problem

We define the Bulk Port Cargo Throughput Optimisation Problem (BPCTOP) as the problem of choosing sailing times and drafts for a set of ships arriving at or departing from a single port so as to maximise the total cargo carried on the departing ships, while also meeting all safety rules in place at the port. The draft and cargo load of arriving ships is fixed, though extensions of this problem could consider routing between multiple ports, with the load of each ship being selected at the origin port so as to maximise cargo while meeting draft restrictions at the destination port. For an export port such as Port Hedland, almost all arriving ships are empty, so the draft and amount of cargo is only relevant for departing ships.

The problem involves a choice of objective functions. This thesis generally considers a simple objective function that maximises the total cargo carried on all ships. However, an alternative variation not investigated in depth in this thesis is a more complex objective function with multiple weighted objectives, including priority weightings between ships to take into account any queueing rules in effect at the port, and allowing shipping agents to set target drafts for each ship if the ship is contracted to carry a certain minimum amount of cargo.

The allowable sailing times for a ship arriving or departing a port depend on the draft and other dimensions of the ship, the times when other ships are scheduled to sail, and a complex set of factors affecting the under-keel clearance (amount of water under the ship's keel). Factors affecting under-keel clearance have been discussed in detail in Chapter 2; however, for the rest of this thesis, we simplify these calculations down to a function that specifies the maximum allowable sailing draft for each ship at a range of discretised times.

The major constraints involved in this problem are as follows:

- Ports may have safety restrictions on **minimum separation times between ships**, as ships sailing too close together in a narrow channel may pose a safety risk. Minimum separation times between ships may also depend on which berths the ships sail from, and the order in which they sail. If a ship departing from a berth at the back of the port sails first, a second ship sailing from a berth close to the channel exit will have to wait for it to pass. If the ships sail in the opposite order, however, they may be far enough apart at their starting locations to be able to start sailing simultaneously.

- The problem also needs to include constraints on the **earliest sailing time for each ship**, which for an outgoing ship would depend on the projected time when the ship will be fully loaded, refuelled, with all crew and supplies on board, and ready to set sail. For an incoming ship, the earliest sailing time would be the time by which the ship will arrive at the port and be ready to start sailing through the channel towards the berth. If both incoming and outgoing ships are included in the scheduling model, incoming ships would also have restrictions on their sailing times to ensure that any outgoing ship departing from the same berth has cleared the berth before the incoming ship arrives.

- Some ports may also have **resource constraints on the availability of tugs** – small boats used to assist large ships in entering and leaving a port. At Port Hedland, tugs are required to assist every ship over a given size, which includes all draft-restricted ships. When there are many ships sailing on a single tide, the number of tugs available may form a bottleneck in the ship scheduling problem, so tug constraints need to be included in the model. Tug constraints are complex and depend on the order in which ships sail, the origins and destinations of successive ships, and the safety rules in place at the port specifying the number of tugs required for each type of ship, as detailed in the Port User Guidelines and Procedures [Port Hedland Port Authority, 2013c]. Different approaches to modelling tug constraints are investigated in Section 3.3.

At tidal ports such as Port Hedland, ship sailing times are scheduled by first assigning ships to high tides, then choosing sailing times for the set of ships assigned to each high tide. This thesis uses a similar approach – the Bulk Port Cargo Throughput Optimisation Problem presented in this chapter optimises sailing times for a set

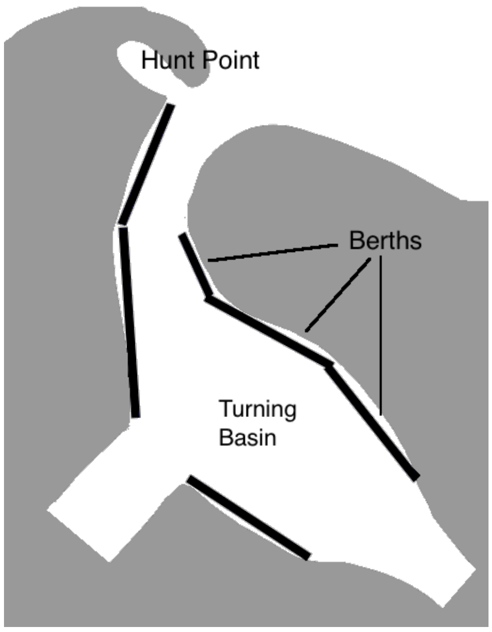


Figure 3.1: Port Hedland Layout

of ships scheduled to sail on one high tide, since this allows us to easily compare our results against real schedules for the same high tides produced by human schedulers in Chapter 5. The planning horizon is thus the time range around one high tide – approximately 12 hours for a port with semi-diurnal tides such as Port Hedland, which has two high tides per day. Note that some ports have diurnal tides – one high tide per day – so the planning horizon would need to be around 24 hours.

For the problem of scheduling ship sailing times at a single port, the model presented in this chapter can easily consider a larger planning horizon by simply extending the number of time steps considered in the problem. However, the models presented here assume that there is at most one ship arriving at and departing from each berth, since the single-tide planning horizon is too short to allow enough time for a ship to arrive, load, and depart again. A problem with a larger planning horizon may need to extend these models to consider ships arriving, loading and departing again. One approach to dealing with this problem could be implemented by first assigning a tide to each ship arrival and departure, then optimising ship schedules independently for each high tide, then attempting to move ships to an adjacent tide to see if that improves the schedule. This thesis presents a much larger ship routing and scheduling problem with multiple ports in Chapter 6.

3.1.3 BPCTOP Example

Figure 3.1 shows the layout of Port Hedland. The grey areas are land, white areas are water, and black lines represent locations of berths. The port has a single channel leading northwards out to sea, with berths on either side of the channel, as well as a

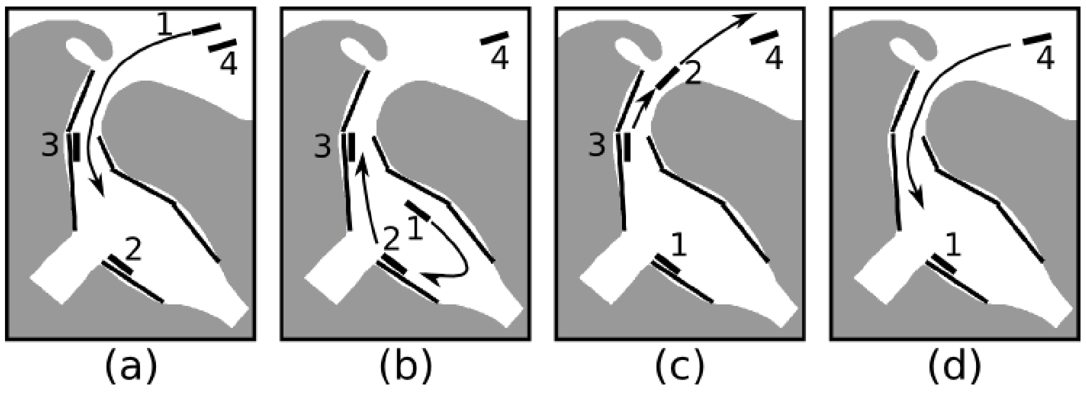


Figure 3.2: Example Bulk Port Cargo Throughput Optimisation Problem

turning basin at the south end of the port that is used by incoming ships to ensure that all berthed ships face out to sea for faster departure. The shallowest point along the channel is at Hunt Point, the exit from the inner harbour, so large ships need to start sailing ahead of the high tide so as to pass Hunt Point at the peak of the high tide when there is the maximum amount of water available.

Figure 3.2 shows an illustrated example of a ship schedule for a single high tide at Port Hedland. There are initially two ships at berth ready to sail, and two empty incoming ships waiting at sea. Ship 1 is scheduled to arrive at the same berth that ship 2 is departing from; ship 4 is scheduled to arrive at a currently empty berth.

The optimal schedule for these ships has the two outgoing (loaded) ships sailing so as to pass Hunt Point around the peak of the high tide, thus maximising the amount of cargo they can carry, with the empty incoming ships scheduled around them.

In Figure 3.2(a), ship 1 sails through the channel and arrives at the turning basin, clearing the channel for ship 2 to sail. In Figure 3.2(b), ship 2 leaves the berth, allowing ship 1 to dock. Ship 3 must wait for ship 2 to pass before it can start sailing. In Figure 3.2(c), ship 2 sails through the channel and out to sea. Ship 3 can start sailing as soon as there is sufficient time between them to meet port safety rules. In Figure 3.2(d) ship 4 can finally sail in to port after ships 2 and 3 have both cleared the channel.

Note that this example is quite simple with only 4 ships. The record at Port Hedland is 6 outgoing draft-restricted ships sailing on a single tide [Port Hedland Port Authority, 2013d], together with several incoming ships, which results in a much more complex problem.

3.1.4 Problem Features and Challenges

There are several key features of the Bulk Port Cargo Throughput Optimisation Problem (BPCTOP) which cause the problem to be computationally difficult. These are discussed below, including how they affect the problem formulation and the types of approaches used to solve the problem. Some of these issues would also apply to other ship scheduling and routing problems that involve time-varying draft restrictions, so in order to solve larger maritime transportation problems with time-varying draft, we may need to decompose them into simpler subproblems.

Optimality

At a draft-constrained port, such as Port Hedland in Western Australia, even a single centimetre of extra draft can allow 130 extra tonnes of cargo to be carried on an average-sized bulk carrier [Port Hedland Port Authority, 2011b]. Port Hedland is a large bulk export port, with around 1300 bulk carriers departing from the port in a year [Port Hedland Port Authority, 2011a], so even a 1 centimetre improvement in the average draft of ships sailing from the port would result in around 170,000 tonnes of extra cargo being shipped without the need to charter additional ships. Due to the huge scales involved, even a 1% reduction in schedule quality is undesirable. For this reason, all the solution approaches we consider in this chapter aim to find the optimal solution for our throughput optimisation problem. The problem is undersubscribed in general, as there is more than enough time for all ships to sail, but it is oversubscribed at the peak of the high tide, since all ships prefer to sail at the time when they can carry the most cargo. Attempting to solve the problem to optimality makes this problem much more computationally difficult.

Time-Indexed Formulation

The draft function for each ship depends on a variety of environmental conditions, including tide, wave and current speed, so the resulting safe sailing draft may vary even in as short a time as five minutes. This function is very complex and varies for each high tide, so it cannot be specified by any simple equation. Taking time-varying draft restrictions into account therefore requires the draft to be modelled as a time-indexed function with a very fine time resolution.

In this thesis, five-minute time increments are used to allow us to model the draft with sufficient accuracy. There are two main reasons for choosing this time resolution:

1. When the tide varies by 6 metres over 6 hours, which can happen at Port Hedland, the allowable draft will vary by 8cm or more in some 5-minute intervals, so making the time resolution coarser than 5 minutes would significantly reduce the quality of our schedules.

2. A finer time resolution would not provide any additional value, as it's impractical to expect large ships to sail exactly at a given minute. Scheduled sailing times for ships at Port Hedland are typically rounded to the nearest five minutes.

The high time resolution of the model means that even though the number of variables is small, the domain size of each variable is large, which results in a longer calculation time. Continuous-time formulations for this problem have not yet been investigated, but they are a possible avenue for future work. Modelling the time as a continuous variable is difficult, since the function is too complex to be specified by any simple equation, and would need to be adjusted for each high tide. A continuous time function is thus only possible by creating a partially-linear approximation for each high tide, based on the value of the draft at five-minute increments. Investigation of this approach is left for future work.

Sequence-Dependent Setup Times

Ports with draft restrictions usually have a channel which is used to access the berths. Port channels are often narrow and can only be transited by ships one at a time. This results in ports having safety restrictions on the minimum separation time required for ships transiting through the channel, to ensure that ships do not travel too close together, and to avoid the risk of ships colliding or not having enough room to manoeuvre. Since berths are positioned at different locations along the channel, the separation time between a pair of ships depends on the order in which ships sail. The sequence-dependent constraints do not propagate well, which makes it hard to find optimal solutions. This effect is exacerbated by the even more complex and highly sequence-dependent constraints on the availability of tugs, which will be discussed in Section 3.3.

3.2 Constraint Programming Model

3.2.1 Variables and Parameters

This section presents a Constraint Programming model for the BPCTOP.

Parameters

Let V be the set of vessels to be scheduled. Let $[1, T_{max}]$ be the range of discretised time indices. Each vessel v has an earliest departure time $E(v)$, which specifies the earliest possible time when vessel v may start sailing.

$D(v,t)$ defines the maximum allowable sailing draft for the vessel v if it starts to sail at time slot t, accounting for all safety rules at this port, including the effects of tide, waves, squat, etc. A ship may take several hours to transit through the channel, so the value of $D(v,t)$ is calculated based on the minimum under-keel clearance the ship will encounter at any point along the channel if it starts to sail at time t. The maximum draft for each ship may also be limited by the maximum capacity of the ship, which is incorporated into the value of the $D(v,t)$ parameter. At some ports, the total amount of cargo available for transport may be less than the maximum capacity of all available ships, in which case the amount of cargo would also constrain the maximum drafts; however, at Port Hedland, there is generally always more iron ore available for shipping. The model presented in this chapter does not include a constraint on maximum cargo available, however this could easily be added if required for other ports.

$ST(v_i, v_j)$ specifies the minimum separation time between the sailing times of every ordered pair of ships v_i, $v_j \in V$, as required by safety rules at the port, to prevent ships sailing so close together that they would run the risk of collision, or not have enough room to manouevre.

Let B be the set of pairs of incoming and outgoing ships sharing the same berth. For each such pair $b \in B$, let $B_i(b)$ be the incoming ship and $B_o(b)$ the outgoing ship. For every such pair of ships, $d(b)$ is the minimum delay between their sailing times, which allows enough time for the outgoing ship $B_o(b)$ to leave the berth safely prior to the arrival of the inbound ship $B_i(b)$. This model assumes that berths for each ship have already been assigned. This is a reasonable assumption, since berth assignment is considered on a time scale of days, whereas scheduling of precise ship sailing times is considered on a time scale of minutes, so these are generally modeled as two separate problems by schedulers at ports. By the time a ship's exact sailing time on the high tide is being scheduled, it already has a designated berth. However, dynamic allocation of berths could be a potential future extension to this problem.

Each ship $v \in V$ – incoming or outgoing – has an associated berth. For each pair of ships there is a minimum separation $ST(v_i, v_j)$ between their sailing slots, which depends on their associated berths. If v_i is an incoming ship whose berth is the same as that of outgoing ship v_o, in addition to their minimum separation, v_i can only be scheduled after v_o. If v_o is not included in the schedule, then nor can v_i be, since the berth that v_i is sailing to would still be in use by the previous ship, v_o. Note that the separation time is used to enforce a safe distance between successive ships in the channel, whereas the minimum delay between two ships using the same berth is used to ensure that the outgoing ship has enough time to clear the berth before the incoming ship arrives.

Decision variables

$s(v) \in [0, 1]$ is a binary variable which specifies whether the ship v is included in the schedule, since it is possible that some ships cannot be scheduled at all, if, for example, a scheduler tries to schedule too many ships on one tide.

$T(v) \in [1, T_{max}]$ defines the time slot at which vessel v is scheduled to sail.

Objective Function Parameters

Some additional parameters are used to describe complex port-specific objective functions.

Let $C(v)$ be the tonnage per centimetre of draft for the vessel v. The relationship between tonnage and draft depends on the shape of the ship; however, loaded bulk carriers generally have close to vertical sides above the waterline, so a linear function is a good approximation for the relationship between tonnage and draft for the range of drafts that would be considered for bulk carriers in a cargo throughput optimisation problem. Note that the minimum draft considered for a ship would be above the draft for an empty ship, since only a draft range that is reasonable to sail with is considered.

Let $MD(v)$ be the minimum target draft requested by the shipping agent for the vessel v – if a target draft was specified in the shipping contract, there may be a large financial penalty for sailing with less draft. Any additional cargo carried above the target draft would be a bonus, but the financial incentive to carry extra cargo may not be as high as that for reaching the target draft.

Let $R(v)$ be a dependent binary variable which is 1 iff the scheduled draft $D(v, T(v))$ for vessel v is greater than or equal to the minimum target draft $MD(v)$ requested by the shipping agent for that vessel. $R(v)$ is specified by Equation (3.1):

$$D(v, T(v)) \geq MD(v) \leftrightarrow R(v) = 1 \tag{3.1}$$

$P(v)$ represents the priority for the vessel v getting its minimum target draft $MD(v)$ in the schedule, for a port where priorities are allocated based on ship queueing rules.

Let W define the weighting in a two-objective function given to the primary objective over the secondary objective.

Note that $P(v)$ specifies the weighting between ships in the objective function, whereas W specifies the weighting between objectives. Also, $P(v)$ may change in response to changing priorities, eg. if a ship gets delayed due to an equipment breakdown and has to move to the next tide, its priority $P(v)$ may be adjusted in the calculation for the next tide's schedule.

Summary

The following table summarises the parameters and variables used in the CP model so far for easy reference.

	Parameters
V	Set of vessels.
$[1, T_{max}]$	Range of discretised time indices.
$E(v)$	Earliest departure time for vessel $v \in V$.
$D(v,t)$	Maximum allowable draft for vessel $v \in V$ starting to sail at time t.
$ST(v_i, v_j)$	Minimum separation time between vessels $v_i, v_j \in V$.
B	Set of pairs of incoming and outgoing ships using the same berth.
$B_i(b), B_o(b)$	Incoming and outgoing ships in pair $b \in B$.
$d(b)$	Minimum delay between the sailing times of vessels $B_i(b), B_o(b)$ for $b \in B$.
	Objective Function Parameters
$C(v)$	Tonnage per centimetre of draft for vessel $v \in V$.
$MD(v)$	Minimum target draft requested for vessel $v \in V$.
$R(v)$	Specifies whether vessel $v \in V$ is scheduled to sail with its target draft $MD(v)$.
$P(v)$	Priority for vessel $v \in V$ getting its target draft $MD(v)$.

W	Weighting of the primary objective in a 2-objective function.
	Variables
$s(v)$	Binary variable specifying whether vessel $v \in V$ is included in the schedule.
$T(v) \in [1, T_{max}]$	Variable specifying the scheduled sailing time for vessel $v \in V$.

3.2.2 Constraints

Earliest Departure Time Constraint

$$s(v) = 1 \Rightarrow T(v) \geq E(v), \ \forall\, v \in V \tag{3.2}$$

Specifies the earliest departure time for each vessel. Any vessel v that is included in the schedule ($s(v) = 1$) must be scheduled to sail no earlier than its earliest possible departure time $E(v)$.

Berth Availability Constraint

$$\begin{aligned} s(B_i(b)) = 1 \Rightarrow \\ s(B_o(b)) = 1 \wedge T(B_o(b)) \leq T(B_i(b)) - d(b), \\ \forall\, b \in B \end{aligned} \tag{3.3}$$

Ensures that the berths for any incoming ships are empty before the ship arrives. For every pair of incoming and outgoing ships $B_i(b)$ and $B_o(b)$ that share the same berth as their destination and origin, if the incoming ship is included in the schedule, then the outgoing ship must also be included in the schedule, and the scheduled sailing time $T(B_i(b))$ of the incoming ship must be later than the sailing time $T(B_o(b))$ of the outgoing ship by a delay of at least $d(b)$, which gives the outgoing ship enough time to clear the berth.

Separation Time Constraints

$$\begin{aligned} s(v_i) = 1 \ \wedge s(v_j) = 1 \Rightarrow \\ T(v_j) - T(v_i) \geq ST(v_i, v_j) \vee \\ T(v_i) - T(v_j) \geq ST(v_j, v_i), \\ \forall\, v_i,\ v_j \in V \end{aligned} \tag{3.4}$$

Ensures that there is sufficient separation time between successive ships to meet port safety requirements. For each pair of ships v_i, v_j either v_i sails first, followed by v_j with a minimum separation time $ST(v_i, v_j)$ between their scheduled sailing times, or v_j sails first, followed by v_i with a minimum separation time $ST(v_j, v_i)$ between them. The minimum separation time ST may be different depending on the order in which the ships sail, ie. $ST(v_i, v_j)$ may be different from $ST(v_j, v_i)$.

3.2.3 Objective Function

The objective function for the Bulk Port Cargo Throughput Optimisation Problem may vary at different ports. Some ports may have an objective function that purely optimises throughput; other ports may need to prioritise fairness to competing clients above optimising the total throughput for the port. Two examples of objective functions are shown below.

Equation (3.5) shows an objective function that maximises the total cargo throughput at the port by maximising the sum of the drafts $D(v, T(v))$ at the scheduled sailing time $T(v)$ for each vessel v, weighted by the tonnage per centimetre of draft $C(v)$ for each ship, since the amount of extra cargo allowed by an increase in draft varies depending on the size and shape of the ship. The objective function is also weighted by the binary variable $s(v)$ that specifies whether the ship was included in the schedule.

$$\text{maximise} \sum_{v \in V} s(v) \cdot C(v) \cdot D(v, T(v)) \tag{3.5}$$

Equation (3.6) shows an alternative objective function, for a port with more complex operating procedures, which allows shipping agents to request minimum target drafts for each ship. $R(v)$ is a binary variable which specifies whether the vessel v was scheduled to sail with a draft $D(v, T(v))$ that was greater than or equal to the minimum target draft $MD(v)$ requested by the shipping agent for that vessel.

$$\begin{aligned} \text{maximise} \sum_{v \in V} s(v) \cdot W \cdot P(v) \cdot R(v) + \\ \sum_{v \in V} s(v) \cdot C(v) \cdot D(v, T(v)) \end{aligned} \tag{3.6}$$

Ships are allocated sailing slots based on priority $P(v)$, which is determined by the port's ship queueing rules. An in-depth discussion of queueing rules at ports is beyond the scope of this thesis; however, these have been discussed briefly in Chapter 2, and investigated in depth by Tengku-Adnan, Sier, and Ibrahim [2009] among others.

Finally, the parameter W defines the weighting given to the primary objective of ships getting their minimum target draft $MD(v)$, compared to the secondary objective of maximising the total cargo carried, $C(v) \cdot D(v, T(v))$.

The rest of this thesis considers problems with the simple objective function defined by Equation (3.5), as this is a fairly general function that would be applicable at many ports. In practice, we've found that using the more complex multi-objective function given by Equation 3.6 actually makes it easier to find optimal solutions, as the large contribution of the requested draft objective makes it easier to narrow down the search to good solutions quickly.

3.3 Modelling Tugs

The above CP model was included in a prototype application for scheduling ship sailing times at Port Hedland, which is described in more detail in Chapter 5. One piece of user feedback from initial testing of this prototype was that the availability of tugs sometimes constrained the schedule. Tug constraints were not included in the original prototype; however, a model that did not include tug constraints could produce unrealistic schedules.

Many ports require tugs to be used to guide large cargo ships in and out of the port, to improve ship manoeuverability and therefore increase safety. Tugs are attached to outgoing ships at berth, and detach from the ship some time after the ship clears the most narrow, dangerous part of the channel. Tugs attach to incoming ships while the ship is at sea, and detach from the ship as it arrives at the berth. At some ports, tugs may also be required to push incoming ships onto the berth. The number of tugs required to assist in berthing may vary depending on the size of the ship and the location of the destination berth.

Tugs working on two successive outbound or inbound ships must travel back to berth or to sea after completing the first job. Since tugs are much smaller than cargo ships, they are not constrained by draft, and can therefore travel around other ships in the channel. Tugs working on an incoming ship followed by an outgoing ship, or vice versa, have a much shorter travel time from the end of their first job to the start of the second, as they do not need to travel back along the channel to reach their next ship.

See Port Hedland Port Authority [2013c] for a detailed description of the tug guidelines for ships currently in effect at Port Hedland. Other ports may have slightly different rules regarding tugs, or may need additional resource constraints to be added, such as constraints on the availability of ship pilots, for example. However, the general approach presented in this thesis, where tugs must be in use for the duration of the ship transit, and some length of time is required for tugs to transfer between successive ships, would apply to all other ports where tugs are used to assist in ship arrivals and departures, even if the specifics of the times required may change.

There has been some prior research on tug scheduling, such as the work of Yan et al. [2009]. However, in our problem, consideration of tug availability is only required as a constraint in the larger ship schedule optimisation problem, so assigning all individual tugs to ships may not be required.

3.3.1 Modelling Approaches

Tug availability depends on the number of tugs currently in operation at the port, the number of tugs required to bring each ship into or out of the port, and the length of time the tugs are in use for on each ship. The length of time required for a tug to become available again after assisting a ship depends on the origin and destination of the ship; the origin of the tug's next ship (since this affects the travel time of the tug from one ship to the next); whether the tug is required to push an incoming ship onto the berth; and port-specific operational rules, such as the location in the channel where tugs detach from outgoing ships and attach to incoming ships. This results in action durations in the model being highly sequence dependent.

Our first attempt at modelling the tug constraints involved assigning individual tugs to ships. However, this approach turned out to be too slow to find the optimal solution, since the already large number of interactions caused by the sequence-dependent waiting times between ships were compounded by the highly sequence-dependent waiting times between tugs becoming available for successive ships.

To address this issue, we created a modified model which only tracked the number of tugs busy at each point in time, rather than explicitly allocating tugs to ships. However, because the delay between the start of successive jobs for a tug was dependent on the sequence of jobs performed by each tug, this new model still required tracking origins and destinations for tugs, which was still too slow to find an optimal solution.

3.3.2 Successful CP Model

Our third attempt at modelling the tug constraints successfully used features of the problem to simplify the constraints, enabling realistic-sized ship schedules to be optimised within a few minutes.

There are two problem features, or simplifying assumptions, that turned out to be critical to simplifying the tug constraints.

1. The port has a single channel leading from sea to the berths, which can only be used in one direction at a time.

2. The berths are close enough together that the travel time for tugs moving from a berth to sea or vice versa can be considered independent of the berth location.

Assumption 1 holds at Port Hedland, as well as the majority of other draft-constrained ports, with some exceptions. Multiple channels would significantly increase the complexity of the tug constraints, so extending this model to ports with multiple channels would be a non-trivial task. Assumption 2 is also the case at Port Hedland, and is likely to apply at the majority of ports. However, at ports where berths are spaced far apart compared to the length of the channel, a model based on this assumption would need extension to avoid reducing schedule quality.

Assumption 1 implies that a schedule of ships can be split up into component scenarios of four possible types:

1. A sequence of outgoing ships

2. A sequence of incoming ships

3. An outgoing ship followed by an incoming ship

4. An incoming ship followed by an outgoing ship

The tug availability constraints can be considered separately for the incoming and outgoing scenarios, thus making the combined ship schedule optimisation problem much simpler. Figure 3.3 shows a schedule broken up into these four component

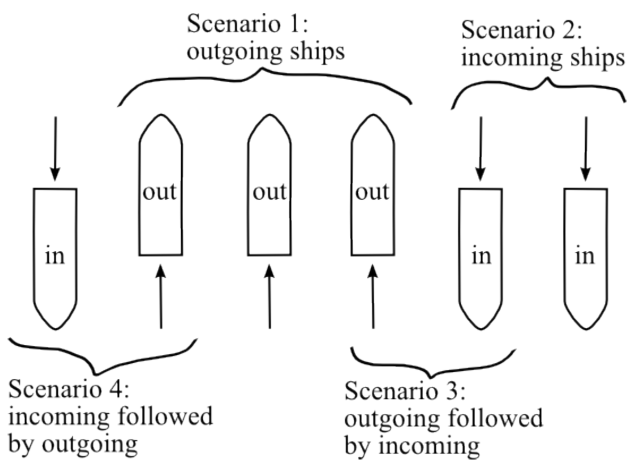


Figure 3.3: Splitting a schedule of ships into scenarios for tug counting

scenarios. In practice, at an export port, outgoing ships are generally full and incoming ships are empty, so good quality schedules will generally schedule incoming ships on the shoulders of the tide, and outgoing ships around high tide. Because of the long time delay required between ships using the channel in different directions, good quality schedules are unlikely to involve ships switching direction more than once from incoming to outgoing, and once from outgoing to incoming, as shown in Figure 3.3.

Scenarios 1 and 2: For a sequence of outgoing ships or a sequence of incoming ships, the turnaround time between successive jobs for a given tug is independent of berth location (Assumption 2), and therefore does not depend on the sequence of jobs, as long as both jobs are in the same direction.

Scenario 3: At Port Hedland, the tugs transfer from the outgoing ship to the incoming ship while the ships are in transit, so no additional delay is required. For ports where there is a delay for tug transfer from an outgoing ship to an incoming ship, Scenario 3 can be modelled similarly to Scenario 4 below.

Scenario 4: Tugs moving from an incoming ship followed by an outgoing ship require a delay between the end of the first job and the start of the second job. This can be modelled by introducing an extra variable for outgoing ships, to specify how many tugs are still busy prior to that ship's departure time due to having recently completed an incoming job. We do not need to consider tugs used for all incoming ships when calculating the number of tugs available for outgoing ships; only tugs used on any ship that arrives just before an outgoing ship leaves – one ship at most in any given schedule, due to the separation time constraints between ships. The duration of the extra delay for tugs moving from an incoming to an outgoing ship may vary based on the locations of the destination and origin berths.

Our initial approach to tug constraints had to calculate whether the number of tugs used by both incoming and outgoing ships at each point in time met the constraint on the number of tugs available, which was very complex due to the minimum time between successive ships being different depending on the previous and next ship for each tug. However, by splitting the schedule into scenarios, we can instead have two separate constraints – a constraint to ensure that the number of tugs used by incoming ships (plus tugs still busy after being used for the latest outgoing ship – always zero for Port Hedland) never exceeds the number of tugs available; and a second constraint to ensure that the number of tugs used by outgoing ships (plus tugs still busy after being used for the latest incoming ship) never exceeds the number of tugs available.

The main difference between this approach and the second attempt to model tugs is: instead of counting the number of tugs busy at each point in time for the entire sequence of ships, this approach counts the number of tugs that are unavailable for an outgoing ship due to still being in use on other outgoing ships, or on a recent incoming ship whose time overlaps with an outgoing ship, and separately counts the number of tugs that are unavailable for use on an incoming ship due to still being in use from a previous incoming ship. Tugs that were working on outgoing ships are never considered busy for incoming ships, since the channel can only be used in one direction at once, so outgoing tugs can transfer straight to the next incoming ship without having to wait. The separation time constraints already allow for all the waiting time required between an outgoing and the next incoming ship. The set of tugs busy for incoming ships and the set of tugs busy for outgoing ships can be separately checked to ensure they do not exceed the number of tugs available, since the tugs used for incoming ships cannot affect outgoing ships other than as described here, and vice versa. This separates the very complex tug constraints produced by tracking tugs on all ships at once into two much simpler, less strongly connected sets of constraints.

3.3.3 Tug Variables and Parameters

Adding tug availability constraints to the CP formulation of the ship scheduling problem with time-varying draft requires the following additional parameters and dependent variables.

Let U_{max} be the number of tugs available at the port. Let I and O be the sets of incoming and outgoing ships.

$G(v)$ is the number of tug groups required for vessel v, where a tug group is a set of tugs that spend the same length of time working on that ship. For example, an incoming ship that requires four tugs to bring it into port, and two of those tugs to push it onto the berth would have two groups containing two tugs each.

Let $H(v, g)$ be the number of tugs in group g of vessel v. G_{max} is the maximum number of groups of tugs required for any ship.

$r(v, g)$ is the "turnaround time" – the time taken for tugs in group g of ship v to become available for another ship in the same direction (incoming vs outgoing).

$X(v_i, v_j)$ is the extra time required for tugs from incoming vessel v_i to become available for outgoing vessel v_o. $X(v_i, v_j)$ is 0 for tugs moving from an outgoing ship to an incoming ship for Port Hedland, and for all the problems we consider in this paper, since tugs can transfer from an outgoing ship to an incoming ship on the move; however, this may not be the case for some ports.

$U(v, t, g)$ is a dependent variable that specifies the number of tugs busy in tug group g of vessel v at time t, assuming the next ship for these tugs is in the same direction. $x(v, t)$ defines the number of extra tugs that are busy at time t for an outgoing vessel v, due to still being in transit from the destination of an earlier incoming ship.

Finally, $L(v, t)$ specifies that the extra tug delay time for incoming vessel v overlaps with the sailing time of an outbound vessel at time t, so these tugs still need to be taken into account in the tug availability constraints for the set of ships going in the opposite direction from v.

Parameters

The following tables summarise the parameters and variables used to add tug constraints to the CP model for easy reference. The model uses the following parameters:

U_{max}	Number of tugs available.
O	Set of outgoing ships.
I	Set of incoming ships.
$G(v)$	Number of groups of tugs required for vessel $v \in V$.
$H(v, g)$	Number of tugs in group g of vessel v.
G_{max}	Maximum number of groups of tugs required for any vessel $v \in V$.
$r(v, g)$	"Turnaround time" – the time taken for tugs in group g of vessel $v \in V$ to become available for another ship in the same direction (incoming vs outgoing).
$X(v_i, v_j)$	Extra time required for tugs from incoming vessel $v_i \in I$ to become available for outgoing vessel $v_o \in O$

Variables

The variables used in the model are as follows:

$U(v, t, g)$	Number of tugs busy in tug group g of vessel $v \in V$ at time t, assuming the next ship for these tugs is in the same direction.
$x(v, t)$	Number of extra tugs busy at time t for an outgoing vessel $v \in O$, due to still being in transit from the destination of an earlier incoming ship.
$L(v, t)$	Binary variable specifying that the extra tug delay time for incoming vessel $v \in I$ overlaps with the sailing time of an outbound vessel at time t

3.3.4 Tug Constraints

Scenarios 1 and 2: One-Directional Sequence of Ships

$$\begin{aligned} s(v) = 1 \land t \geq T(v) \land t < T(v) + r(v,g) \Rightarrow \\ U(v,t,g) = H(v,g), \\ \forall\, v \in V,\ t \in [1, T_{max}],\ g \in [1, G_{max}] \end{aligned} \tag{3.7}$$

$$\begin{aligned} s(v) = 0 \lor t < T(v) \lor t \geq T(v) + r(v,g) \Rightarrow \\ U(v,t,g) = 0, \\ \forall\, v \in V,\ t \in [1, T_{max}],\ g \in [1, G_{max}] \end{aligned} \tag{3.8}$$

These constraints specify the number of tugs busy at each time t for a one-directional sequence of ships. The number of tugs busy $U(v,g,t)$ at time t in group g of vessel v is equal to the total number of tugs in that tug group, $H(v,g)$, if and only if the vessel is included in the schedule ($s(v) = 1$), and the vessel has already begun sailing at time t (the scheduled sailing time $T(v)$ is no later than t), and the turnaround time $r(v,g)$ for that tug group to become available for the next job in the same direction has not yet passed.

Scenario 4: Incoming Followed By Outgoing

$$\begin{aligned} L(v_i,t) \iff \exists\, v_o \in O\ \ s.t. \\ t = T(v_o) \land T(v_i) \leq T(v_o) \land \\ T(v_i) + \max_{g \in [1, G(v_i)]} r(v_i,g) + X(v_i,v_o) > T(v_o), \\ \forall\, v_i \in I,\ t \in [1, T_{max}] \end{aligned} \tag{3.9}$$

$$\begin{aligned} x(v_i,t) = \text{bool2int}(L(v_i,t)) \cdot \sum_{g \in [1, G(v_i)]} H(v_i,g), \\ \forall\, v_i \in I,\ t \in [1, T_{max}] \end{aligned} \tag{3.10}$$

The constraints in Equation (3.9) specify that the "overlap" flag, $L(v_i,t)$ is true iff the extra tug delay time $X(v_i,v_o)$ overlaps with the scheduled sailing time $T(v_o)$ of at least one incoming vessel, v_o.

The constraints in Equation (3.10) express the requirement that for any incoming vessel v_i, the tugs from that vessel are still considered busy for vessels travelling in the opposite direction if the "overlap" flag $L(v_i,t)$ is true. This constraint is over-conservative in the event of different groups of tugs on an incoming ship being in use for different periods of time, for example, if an incoming ship needs 4 tugs to guide it into port, then 2 tugs for an additional half-hour to push the ship onto the berth. For a port where such a situation occurs often and negatively affects schedules, these constraints can be made less conservative by making the parameters $L(v_i,t)$

and $X(v_i, v_o)$ dependent on tug group, and modifying Equations (3.9) and 3.10 to calculate the extra delay for each tug group separately.

Scenario 3: Outgoing Followed By Incoming

$$x(v_o, t) = 0, \ \ \forall\, v_o \in O, \ t \in [1, T_{max}] \tag{3.11}$$

The constraints in Equation (3.11) specify that there is no additional delay required for tugs to transfer from an outgoing ship. For ports where this is not the case, the additional delay for tugs to transfer from an outgoing ship to an incoming ship can be modelled similarly to the Scenario 4 constraints above.

Tug Availability Constraints

$$\sum_{v \in I} \sum_{g \in G(v)} U(v, t, g) \leq U_{max}, \ \ \forall\, t \in [1, T_{max}] \tag{3.12}$$

$$\sum_{v_o \in O} \sum_{g \in G(v_o)} U(v_o, t, g) + \sum_{v_i \in I} x(v_i, t) \leq U_{max}, \ \ \forall\, t \in [1, T_{max}] \tag{3.13}$$

The constraints expressed in Equation (3.12) specify that at each time t, the number of tugs busy does not exceed the total number of tugs available at the port, U_{max}. The total number of tugs in use is given by the sum of $U(v, t, g)$ over all tug groups g, for all incoming vessels $v \in I$. These constraints ensure that the resulting schedule satisfies the tug availability constraints for all sequences of incoming vessels – Scenario 2 in our set of four possible sequences of vessels described in Section 3.3.2.

The constraints expressed in Equation (3.13) represent the same requirement for outgoing vessels – Scenario 1. However, the total number of busy tugs also includes any tugs were still busy at time t due to having recently completed an incoming job and not yet having had time to transfer to the outgoing ship $(X(v_i).(t = T(v))$. Thus the constraints in Equation (3.13) ensure that the resulting schedule satisfies the tug availability constraints for both Scenarios 1 and 4.

As mentioned in Section 3.3.2, for problems considered in this thesis, Scenario 3 is ignored, since tugs moving from an outgoing ship to an incoming ship are assumed to not require any extra time to make the transfer. However, for a port where this assumption does not hold, Scenario 3 can be modelled similarly to Scenario 4.

The following chapter investigates different search strategies and improvements to this model aimed at improving scalability, as well as investigating alternative models to the CP model presented in this chapter. Chapter 5 introduces a prototype software system that uses this CP model to solve the BPCTOP for a real bulk export port, and presents an analysis of the quality of schedules produced by this model as compared to human schedulers at Port Hedland.

Chapter 4

Scalability Improvements: Search Strategies and Alternative Models

As discussed before in Chapter 2, the performance of CP solvers is highly dependent on the strategies used to choose variables and split domains in the backtracking search. This chapter therefore investigates different search strategies for solving the Bulk Port Cargo Throughput Optimisation Problem with time-varying draft restrictions.

While a good choice of search strategy was able to produce results for realistic-sized problems within a reasonable time, if this problem is to be used as a subproblem in larger routing and scheduling problems, further improvements to the calculation speed would be beneficial. This chapter therefore also investigates several improvements to the CP model to improve scalability for this problem. Finally, the CP model is compared against a MIP model and a Benders decomposition approach, which are more traditionally used to solve optimisation problems in maritime transportation.

The specific results of the search strategy and CP model comparisons presented in this chapter are quite implementation-dependent and may not apply to all CP solvers, but these results have broader implications, both in terms of investigating how much effect search strategies and modelling approaches can have on solver runtime, and in terms of identifying possible improvements to the performance of the CP solver used in this chapter for other problems.

4.1 Experimental Data

In Chapter 5, the schedules produced by the CP model are compared against real schedules created by human schedulers at Port Hedland, to investigate how optimal schedules differ in quality. However, this chapter investigates the calculation time of different approaches to solving the BPCTOP CP model, so we used generated data instead of a real data set, as this allowed us more control to systematically vary properties of the problems to see how problem characteristics affected performance. Using generated data also allowed us to test the performance of each approach with larger problem sizes than are currently found in the real world, to ensure that performance is sufficient for near-future growth in port capacity.

4.1.1 Port Characteristics

To test performance of different search strategies and modelling approaches, we generated a data set based on a fictional but realistic port, similar to the ship_scheduling data set used for the 2011 MiniZinc Challenge [University of Melbourne, 2011]. The reason for using a fictional port was again, so that the maximum problem size in our data set would not be constrained by the number of berths currently available at Port Hedland. We wanted to test the performance of our approach on larger problem sizes than the largest bulk export port in existence today to allow for future growth in port capacity.

The geography of the fictional port was generated manually by defining a number of berths at varying distances from the most constrained point of the channel. The port safety rules were based on the ones at Port Hedland, as defined by the Port User Guidelines and Procedures [Port Hedland Port Authority, 2013c]. Minimum separation time requirements for our fictional port were calculated manually between every possible ordered pair of ships sailing from different berths as follows, based on the separation times defined in the Port User Guidelines and Procedures for Port Hedland:

- *Outbound followed by outbound*: the two ships should have a separation time of 30 minutes at a key waypoint along the channel, past the locations of the berths. This ensures that the ships will have sufficient separation time between them for the entire transit, regardless of which berths they leave from.
- *Inbound followed by inbound*: inbound ships must berth facing towards the channel exit, to ensure faster departure speeds for the more tightly constrained outbound (loaded) ships. This means that inbound ships must sail to the end of the channel and turn around prior to berthing, resulting in longer separation times being required between inbound ships. Separation times between inbound ships were calculated to ensure that there would always be sufficient separation distance between ships based on the path taken to reach their berth locations.
- *Outbound followed by inbound*: the inbound vessel should enter the channel only after the outbound vessel has passed the selected channel entry point. The separation time therefore depends on the time taken for the outbound vessel to reach the inbound vessel's channel entry point, which varies with the berth the outbound vessel sails from, but does not vary with the destination berth of the incoming ship.
- *Inbound followed by outbound*: the outbound vessel should only begin sailing when it is safe to do so, ie. when the inbound vessel has passed the berth (in the case of an outbound vessel sailing from a berth closer to the channel exit)

or when the inbound vessel has arrived at its destination berth and is no longer blocking the channel, in the case of an outbound vessel sailing from further away from the channel exit. The separation times vary with the berths of both ships. The Port User Guidelines and Procedures for Port Hedland also describe a manoeuvre called a "double shuffle", where an incoming ship can enter the port prior to the departure of the outgoing vessel from its destination berth. The outgoing vessel then departs while the incoming ship is turning around. This manoeuvre is also taken into account in the safety rules defined for the data set used in this chapter.

Tug requirements for each ship are also defined in the Port User Guidelines and Procedures, specifying the number of tugs required for each section of the transit and for each ship size category. For the generated data set, we assumed that all ships are of the two largest sizes specified in the Port User Guidelines and Procedures, requiring 3-4 tugs each, since smaller ships are much less tightly constrained and can sail on the shoulders of the tide. Smaller ships also use fewer tugs, so the schedule for a set of smaller ships will be constrained by the separation time between ships, not the number of tugs available. The number of tugs available at the port was set to 12 – large enough to have three large ships underway simultaneously, but small enough to constrain the schedule, as is currently the case at Port Hedland.

4.1.2 Problem Instances

In each problem instance, ships were generated to sail to or from one berth. All outgoing ships sailed from different berths; some incoming ships sailed to a berth that was initially empty, whereas other ships sailed to a berth that was also used by an outgoing ship, thus requiring that the outgoing ship clear the berth before the incoming ship arrived.

Each ship had a corresponding earliest departure time, with at least 50% of ships in each problem able to sail from the start of the time range, but some ships needing to wait until later in the planning period, up to the peak of the high tide, to simulate a ship running late loading.

The planning horizon was 6 hours for each problem instance, with a maximum draft being defined for each ship at every 5 minutes using a sinusoidal function. In real data, the maximum draft function will also be affected by the height of the waves at different frequencies, the shape of the ship, the speed of the ship at each point along the transit, and other factors. However, while these factors are all vital to the quality of schedules calculated from a real data set, they do not have any significant effect on the calculation time required to solve the ship scheduling problem, since the scheduling problem combines all of these factors into a single maximum draft value for each ship at each five-minute time increment. Therefore, for the purposes of investigating the calculation time of different approaches in this chapter, the maximum draft for each ship was modelled as a simple sine function, with a plateau around the peak for some ships, to simulate structural limitations capping the draft of some ships. Each ship also had a minimum draft that it could sail with, to simulate the fact that some ships may already be partially loaded at the time the schedule is being calculated, as well as the fact that it could be more economical for a ship to wait until the next high tide rather than sail with a huge reduction in amount of cargo carried.

The data set used for performance comparisons contained problems of four subsets that represent different physical situations, varying in terms of how tightly constrained the problems are.

1. OneWay_Narrow: The most tightly constrained problems: all ships are sailing in the same direction (outbound), and the outbound ships have high maximum drafts, so each ship is only able to sail with its maximum draft during a narrow window around the peak of the tide. The peak of the tide is oversubscribed – not all ships will be able to sail with their maximum drafts.

2. Mixed_Narrow: Moderately constrained problems: ships are split evenly between inbound and outbound, and outbound ships have high maximum drafts with narrow peak draft windows. Inbound ships are not draft constrained and may sail at any time, subject to separation time and tug constraints.

3. OneWay_Wide: Moderately constrained problems with all outbound ships, but with lower maximum drafts, leading to wider time windows when ships can sail with their maximum draft, and a less tightly constrained schedule.

4. Mixed_Wide: The least tightly constrained problems, with ships evenly split between inbound and outbound, and with low maximum drafts and wide sailing windows for outbound ships.

Wide problems are less constrained than Narrow problems, and Mixed problems are less constrained than OneWay problems, though Mixed problems also result in more complex tug constraints coming into effect due to the interaction between inbound and outbound ships.

Each problem subset consisted of 7 problems ranging from 4 to 10 ships sailing on a single tide. Port Hedland set a record of five draft-constrained ships sailing on a single tide in 2009 [OMC International, 2009], so 4-10 ships sailing on a single tide are are realistic sized problems, and the largest problem sizes allow for some growth in port capacity.

Each problem type was considered both with and without tug constraints, to investigate how tug constraints affect performance for different approaches, since not every port will have tugs as the bottleneck in ship scheduling. This resulted in a total of $4 \cdot 7 \cdot 2 = 56$ different problem instances.

4.2 Search Strategies

The model described in Chapter 3 was formulated in the MiniZinc 1.4 modelling language, described by Nethercote et al. [2007, 2010], and solved with the finite domain

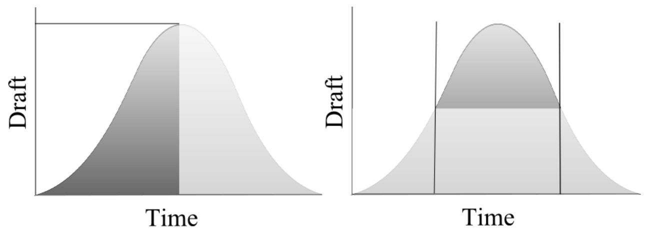


Figure 4.1: (a) Time search. (b) Draft search.

CP solver included in the G12 optimisation platform [Stuckey et al., 2005]. The G12 finite domain CP solver uses standard backtracking search, and allows a choice of variable selection and domain reduction strategies to be used for solving the problem.

There were two main reasons for the choice of MiniZinc and the G12 FD solver: firstly, the model was intended to be used in a commercial ship scheduling system, as discussed in Chapter 5, and due to budget limitations of the initial prototype project, a solver that was free for commercial use was preferred. Secondly, we wanted to model our problem in MiniZinc, since it is a generic modelling language which can be used as input to many different solvers, thus leaving the option open of switching to a different solver in future if our first choice of solver turned out to be unable to solve realistic sized problems. The G12 FD solver is usable with MiniZinc and is BSD-licensed, which made it a good first choice for the prototype.

4.2.1 Draft vs Time Search

In the CP model for the BPCTOP presented in Chapter 3, the only decision variables are the sailing times $T(v)$ for each vessel v and the boolean variables $s(v)$ which specify whether each ship v is included in the schedule. All approaches investigated in this chapter initially search on $s(v)$. The optimal schedule always includes as many ships as possible, so $s(v)$ is set to 1 initially for every ship before searching schedules where some ships are not included. Note that in this chapter, all vessels can be included for every problem.

The values of the other variables – the draft $D(v, T(v))$, the objective function component for each ship ($s(v).C(v).D(v, T(v))$ for the simple objective function that maximises total throughput), and the dependent variables used in the tug constraints – are all functions of time. However, for this problem, it may be more efficient to search on a dependent variable that directly impacts the objective function, such as draft $D(v, T(v))$, or the objective function component for each vessel, rather than searching on ship sailing time directly after setting $s(v)$ to 1.

Figure 4.1(a) shows that splitting the time domain in two might not reduce the draft domain at all. Searching on time may therefore result in searching regions of poor-quality solutions first, so searching on time is likely to be slow to converge on optimal solutions.

Figure 4.1(b), on the other hand, shows that splitting the draft domain always reduces the time domain at any port with enough tidal variation to affect vessel drafts. Searching on draft would thus allows the search to quickly cut out regions of poor quality solutions, so the search would converge on the optimal solutions much faster.

Experimental Results

The CP model presented in Chapter 3 was formulated in MiniZinc 1.4 and solved to optimality for each problem instance using the G12 finite domain CP solver. All calculations were run on a Windows 7 machine with an Intel i7-930 quad-core 2.80 GHz processor and 12.0 GB RAM, and with a 5-minute (300-second) cutoff time.

Three search strategies – searching first on time, draft, and objective function component of each ship – were compared for the data set of 56 problem instances – 7 problems of 4 different types, with and without tugs. Table 4.1 presents the number of ships for the largest problem of each type that could be solved to optimality with each search strategy within the 300-second cutoff time. (Suboptimal solutions are not presented.) The numbers in brackets indicate the CPU time in seconds for calculating the optimal solution for that problem size. The search strategy with the largest problem solved within the time limit, or the fastest CPU time in the event of a tie, is highlighted in bold for each problem type.

Each search strategy splits the domain of the corresponding variable about the mean of its lower and upper bounds. The Time search strategy selects variables with the smallest initial domain (a *first_fail* approach), whereas the Draft and ObjFun search strategies select variables with the largest initial value first to focus on the highest-quality solutions.

The first step in all search strategies, prior to searching on either the time or the draft, was to try to include all ships in the schedule by setting $s(v)$ to 1. In every example problem, all ships could be included. If some ships could not be included, the ship or ships with the smallest impact on the objective function would be left out of the schedule, and would have to be moved to the next tide. For an objective function that considers user-defined priorities for ships, such as the more complex objective function described in Chapter 3, any delayed ships could have their priorities increased by the user to ensure that the same ship wouldn't get delayed multiple times.

Alternative search strategies such as selecting the maximum value in the domain first, or choosing the variable with the smallest domain for the Draft and ObjFun search strategies, are discussed in Section 4.2.2.

Discussion

The results in Table 4.1 show that, as expected, the addition of tug constraints significantly increases the CPU time required to solve the problems, so that the maximum number of ships that can be scheduled within the cutoff time gets reduced by 1-2 ships on average with the addition of tug constraints.

Problem	ObjFun	Draft	Time
Mixed_Wide	10 (5.91)	**10 (3.07)**	4 (26.73)
OneWay_Wide	10 (18.5)	**10 (16.9)**	4 (11.47)
Mixed_Narrow	8 (160)	**8 (20.8)**	6 (195.46)
OneWay_Narrow	**8 (48.7)**	8 (74.3)	7 (63.29)
Mixed_Wide_Tugs	10 (175)	**10 (83.3)**	4 (40.85)
OneWay_Wide_Tugs	8 (39.7)	**8 (16.3)**	4 (8.67)
Mixed_Narrow_Tugs	7 (250)	**7 (24.4)**	5 (137.71)
OneWay_Narrow_Tugs	6 (23.2)	**6 (15.4)**	6 (76.82)

Table 4.1: ObjFun vs Draft vs Time search – number of ships in the largest problem solved within the cutoff time, with CPU time in seconds in brackets.

OneWay problems are more severely affected by the addition of tug constraints, probably because OneWay problems are more tightly constrained than Mixed problems, due to the incoming ships in the Mixed problems having low draft and therefore wide sailing windows. This results in tug constraints causing more disruption to OneWay problems than to Mixed problems.

Table 4.1 also shows that, as expected, the Draft and ObjFun search strategies can solve larger problems to optimality for every problem type. This approach of searching on a dependent variable that is more closely linked to the objective rather than searching on time may be useful for other problems with complex time-varying objective functions.

4.2.2 Searching on Draft

Section 4.2.1 showed that searching on draft or the objective function component of each ship allowed larger problems to be solved to optimality than searching on time directly. It is clear that the choice of search strategy can have a large impact on the time taken to prove optimality, so this section investigates search strategies in more depth, comparing the effects on CPU time of the variable selection and domain splitting methods when searching on draft or objective function component.

The first step in all search strategies, as in Section 4.2.1, tries to include all ships in the schedule by setting $s(v)$ to 1. After including all ships in the schedule, two different domain splitting methods and two different variable selection methods are compared for both Draft and ObjFun search.

The two domain splitting methods considered in this section are Max, which sets the domain of a variable to its maximum possible value, and Mean, which splits the domain of a variable about the mean of its lower and upper bounds and searches the upper half first. Both approaches backtrack and continue searching if no solution is found.

The two variable selection methods considered in this section are Largest, which chooses the variable with the largest upper bound to search on first; and FirstFail,

Problem	Largest_Mean	Largest_Max	FirstFail_Mean	FirstFail_Max
MW	10 (3.93)	10 (4.15)	10 (3.95)	**10 (3.51)**
OW	10 (1.31)	10 (1.55)	***10 (1.11)***	10 (1.22)
MN	8 (148)	7 (299)	***10 (0.45)***	10 (0.77)
ON	8 (44.1)	***8 (31.5)***	8 (70.7)	8 (57.7)
MWT	**10 (117)**	10 (184)	**10 (117)**	10 (181)
OWT	***8 (1.65)***	8 (1.76)	8 (2.42)	8 (2.53)
MNT	7 (184)	6 (60.1)	***8 (2.31)***	8 (2.86)
ONT	***6 (7.33)***	6 (10.8)	6 (17.7)	6 (18.7)

Table 4.2: ObjFun search – number of ships in the largest problem solved within the cutoff time, with CPU time in seconds in brackets.

Problem	Largest_Mean	Largest_Max	FirstFail_Mean	FirstFail_Max
MW	10 (3.17)	10 (3.07)	10 (3.38)	***10 (2.64)***
OW	**10 (0.45)**	10 (1.22)	10 (1.42)	10 (1.44)
MN	**10 (7.99)**	9 (299)	10 (141)	8 (114)
ON	8 (42.5)	8 (48.2)	8 (50.5)	**8 (35.0)**
MWT	***10 (115)***	10 (182)	***10 (115)***	10 (180)
OWT	**8 (1.76)**	**8 (1.76)**	**8 (1.76)**	**8 (1.76)**
MNT	**8 (4.06)**	8 (143)	8 (104)	8 (210)
ONT	**6 (10.3)**	6 (25.8)	6 (16.4)	6 (14.7)

Table 4.3: Draft search – number of ships in the largest problem solved within the cutoff time, with CPU time in seconds in brackets.

which searches first on the variable with the smallest domain.

Each domain splitting and variable selection method was used to solve all problem types, with both the Draft and ObjFun search strategies, resulting in eight different search strategy combinations. The results comparing these eight search strategies are presented in Tables 4.2 and 4.3.

Experimental Results

The same set of problems as in Section 4.2.1 were solved with each of the above eight search strategies. As in Section 4.2.1, all calculations were run with a 300-second cutoff time.

Tables 4.2 and 4.3 show the largest problem solved to optimality within the cutoff time for each problem type and each search strategy. CPU time is given in brackets, and bold font indicates the search strategy (Largest_Mean, Largest_Max, FirstFail_Mean or FirstFail_Max) that was able to solve the largest problem, or the one with the shortest CPU time in case of a tie. Italicised bold numbers indicate that this search strategy was fastest out of the fastest Draft and fastest ObjFun approaches.

Table 4.2 compares each domain splitting and variable selection combination for

ObjFun search, for each of the 4 problem types, with and without tugs. Table 4.3 compares CPU times for Draft search with each domain splitting and variable selection combination, using the same set of example problems.

Wide vs Narrow

For all problem types and search strategies in Tables 4.2 and 4.3, problems with Narrow windows are significantly harder to solve than problems with Wide windows. In the Narrow problem, there is not enough time for all ships to sail close to the peak of the high tide, so not all ships can get their optimal objective function values. Narrow windows thus result in a much more tightly constrained problem, whereas wide windows allow enough time for all ships to sail at or close to their peak draft so the problem is not as tightly constrained. For each of the problems and search strategies investigated, narrow windows resulted in 1-4 less ships able to be scheduled within the cutoff time due to the more constrained problem increasing the difficulty of finding the optimal schedule.

OneWay vs Mixed

Tables 4.2 and 4.3 show that while OneWay problems where all ships are outbound are usually harder to solve than Mixed problems where ships are evenly split between inbound and outbound, this is not always the case. Mixed problems are easier to solve most of the time, probably due the fact that incoming ships sail empty, and therefore have very wide windows, resulting in Mixed problems being less tightly constrained than OneWay problems. However, in some circumstances, this may be counteracted by the fact that Mixed problems have additional tug constraints and berth availability constraints coming into effect. The more complex Scenario 4 tug constraints need to be considered whenever we have a problem that includes both incoming and outgoing ships. Constraints on the berth for the incoming ship being available are also only considered when the problem includes both incoming and outgoing ships.

Search Strategies

Search strategies which searched on the objective function component of each ship (ObjFun) were faster than the search strategies that searched on Draft for all problems except Mixed_Wide problems with and without tugs – the least constrained type of problem, which was solved quite quickly by all search strategies.

For both Draft and ObjFun search, splitting the domain at the Mean produced better results than choosing the Max value in the domain for the majority of problems. Also, the Mean domain reduction strategy solved problems with the same number of ships as the best solution for all cases, whereas the Max domain reduction strategy failed to solve problems of this size for several problem types. In almost all cases, therefore Mean is the more effective domain reduction strategy for both ObjFun and Draft search.

For Draft search with Mean domain reduction strategy, the Largest variable selection strategy performed better than FirstFail on all problem types. For ObjFun search with Mean domain reduction, the Largest variable selection strategy was faster to find the optimal solution for more problems, but the FirstFail variable selection approach offered a much larger reduction in CPU time for those problems where it outperformed the Largest variable selection strategy.

Draft with Largest variable selection and Mean domain reduction was the most consistent search strategy, finding optimal solutions for all problems within 11 seconds of the best time. Four of the other search strategies – ObjFun_Largest_Mean, ObjFun_Largest_Max, Draft_Largest_Max and Draft_FirstFail_Max – fail to solve problems of the same size as the best solution for some problems. The remaining three search strategies – ObjFun_FirstFail_Mean, ObjFun_FirstFail_Max and Draft_FirstFail_Mean – all exceed the calculation time of the fastest solution by at least 30 seconds for at least one problem.

Summary

It is clear that the choice of search strategy has a huge impact on the time to prove optimality. The best choice of search strategy is likely to vary for different ports, which may have slight variations in the tug constraints and in how oversubscribed or undersubscribed the problem is. The best search strategy may also vary between different solvers, due to the interaction between solver implementation and the user-defined search strategy.

For the G12 finite domain solver and the data set considered in this chapter, which is similar to problems at Port Hedland, the Mean domain reduction strategy was faster than Max for almost all problems, and Largest variable selection was consistently faster for Draft search, whereas FirstFail performed better on average for ObjFun search. While ObjFun search was fastest to find a solution for the largest number of problems, Draft with Largest variable selection and Mean domain reduction was the most consistently fast across the full set of problems.

In the rest of this chapter, Draft search with Mean domain reduction and Largest variable selection will therefore be used unless stated otherwise, since for the G12 finite domain solver, this search strategy was consistently close to the shortest calculation time for all problem types.

4.3 Mixed Integer Programming Model

This section presents a Mixed Integer Programming (MIP) model for the Bulk Port Cargo Throughput Optimisation Problem (BPCTOP), and compares the effectiveness of the G12 OSI CBC MIP solver against the G12 finite domain CP solver for proving optimality for this problem.

4.3.1 Variables and Parameters

The MIP model uses the same parameters as those introduced for the CP model in Chapter 3; however, some of the variables in the MIP model differ from the CP model.

$s(v,t) \in [0,1]$ is a binary variable which specifies whether the ship v is scheduled to sail at time t.

$T(v) \in [1, T_{max}]$ defines the time slot at which vessel v is scheduled to sail. $T(v)$ is specified by the equation:

$$T(v) = \sum_{t \in [1, T_{max}]} s(v,t) \cdot t \;\; \forall \, v \in V \tag{4.1}$$

Summary

The following table summarises the parameters and variables used in the MIP model with no tug constraints. The parameters are identical to those used for the CP model in Chapter 3; they are only summarised here for convenience.

	Parameters
V	Set of vessels.
$[1, T_{max}]$	Range of discretised time indices.
$E(v)$	Earliest departure time for vessel $v \in V$.
$D(v,t)$	Maximum allowable draft for vessel $v \in V$ at time t.
$ST(v_i, v_j)$	Minimum separation time between vessels $v_i, v_j \in V$.
B	Set of pairs of incoming and outgoing ships using the same berth.
$B_i(b), B_o(b)$	Incoming and outgoing ships in pair $b \in B$.
$d(b)$	Minimum delay between the sailing times of vessels $B_i(b), B_o(b)$ for $b \in B$.
	Objective Function Parameters
$C(v)$	Tonnage per centimetre of draft for vessel $v \in V$.
$MD(v)$	Minimum target draft requested for vessel $v \in V$.
$R(v)$	Specifies whether vessel $v \in V$ is scheduled to sail with its target draft $MD(v)$.
$P(v)$	Priority for vessel $v \in V$ getting its target draft $MD(v)$.
W	Weighting of the primary objective in a 2-objective function.

	Variables
$s(v,t)$	Binary variable specifying whether vessel $v \in V$ sails at time t.
$T(v) \in [1, T_{max}]$	Dependent variable specifying the scheduled sailing time for vessel $v \in V$.

4.3.2 Constraints

Ship Uniqueness Constraint

$$\sum_{t \in [1, T_{max}]} s(v,t) \leq 1, \;\; \forall \, v \in V \tag{4.2}$$

Specifies that each vessel $v \in V$ sails at most once in the schedule.

Earliest Departure Time Constraint

$$T(v) \geq E(v), \;\; \forall\, v \in V \tag{4.3}$$

The Earliest Departure Time constraint is identical to the CP model in Chapter 3, but the MIP constraint does not need to consider whether the ship is included in the schedule. This constraint has no effect in the MIP model if the ship is not included in the schedule, due to the separation time and tug constraints being ignored for ships that are not in the schedule.

Berth Availability Constraints

$$T(B_o(b)) \leq T(B_i(b)) - d(b), \qquad \forall\, b \in B \tag{4.4}$$

$$\sum_{t\in[1,T_{max}]} s(B_o(b),t) \geq \sum_{t\in[1,T_{max}]} s(B_i(b),t), \qquad \forall\, b \in B \tag{4.5}$$

These constraints ensure that the berths for any incoming ships are empty before the ship arrives. Equation (4.4) specifies that for every pair of incoming and outgoing ships $B_i(b)$ and $B_o(b)$ that share the same berth as their destination and origin, the scheduled sailing time $T(B_i(b))$ of the incoming ship must be later than the sailing time $T(B_o(b))$ of the outgoing ship by a delay of at least $d(b)$. Equation (4.5) specifies that if the incoming ship is included in the schedule – $\sum_{t\in[1,T_{max}]} s(B_i(b)) > 0$ – then the outgoing ship must also be included in the schedule.

Separation Time Constraints

$$\begin{aligned} s(v_i,t) + \sum_{t'\in[t,\min(T_{max},t+ST(v_i,v_j)-1)]} s(v_j,t') &\leq 1 \\ s(v_j,t) + \sum_{t'\in[t,\min(T_{max},t+ST(v_j,v_i)-1)]} s(v_i,t') &\leq 1 \\ \forall\, v_i,\; v_j \in V,\; t \in [1,T_{max}]& \end{aligned} \tag{4.6}$$

These constraints are the CP Separation Time Constraints converted to linear form. These constraints represent the requirement that for each pair of ships v_i, v_j either v_i sails first, followed by v_j with a minimum separation time $ST(v_i,v_j)$ between them, or v_j sails first, followed by v_i with a minimum separation time $ST(v_j,v_i)$ between them.

4.3.3 Tug Variables and Parameters

Converting the tug constraints into Mixed Integer Programming form uses all the same variables and parameters as the CP model.

Parameters

The following tables summarise the parameters and variables used to represent tug constraints in the MIP model. These are identical to the CP model and are only reproduced here for convenience. The model uses the following parameters:

U_{max}	Number of tugs available.
O	Set of outgoing ships.
I	Set of incoming ships.
$G(v)$	Number of groups of tugs required for vessel $v \in V$.
$H(v,g)$	Number of tugs in group g of vessel v.
G_{max}	Maximum number of groups of tugs required for any vessel $v \in V$.
$r(v,g)$	"Turnaround time" – the time taken for tugs in group g of vessel $v \in V$ to become available for another ship in the same direction (incoming vs outgoing).
$X(v_i,v_j)$	Extra time required for tugs from incoming vessel $v_i \in I$ to become available for outgoing vessel $v_o \in O$.

Variables

The variables used in the model are as follows:

$U(v,t,g)$	Number of tugs busy in tug group g of vessel $v \in V$ at time t, assuming the next ship for these tugs is in the same direction.
$x(v,t)$	Number of extra tugs busy at time t for an outgoing vessel $v \in O$, due to still being in transit from the destination of an earlier incoming ship.
$L(v,t)$	Binary variable specifying that the extra tug delay time for incoming vessel $v \in I$ overlaps with the sailing time of an outbound vessel at time t.

4.3.4 Tug Constraints

Scenarios 1 and 2: One-Directional Sequence of Ships

Converting the constraints for a one-directional sequence of ships to linear form for the MIP model results in the following constraints.

$$U(v,t,g) = H(v,g) \cdot \sum_{t' \in [\min(1, t-r(v,g)+1), t]} s(v,t') \tag{4.7}$$
$$\forall\, v \in V,\ t \in [1, T_{max}],\ g \in [1, G_{max}]$$

These constraints express that for a one-directional sequence of ships, at each time t, the number of tugs busy $U(v,t,g)$ in group g of vessel v is equal to the total number of tugs in that tug group, $H(v,g)$, if and only if the vessel has already sailed, but the turnaround time $r(v,g)$ for that tug group to become available for the next job in the same direction has not yet passed.

Scenario 4: Incoming Followed By Outgoing

Converting the Scenario 4 constraints to linear form for the MIP model results in the following constraints:

$$x(v_i, t) = L(v_i, t) \cdot \sum_{g \in [1, G(v)]} H(v_i, g) \tag{4.8}$$
$$\forall\, v_i \in I,\ t \in [1, T_{max}]$$

$$L(v_i, t) \geq s(v_o, t) + \sum_{t' \in [\max(1, t - t_{range}), t]]} s(v_i, t') - 1 \tag{4.9}$$
$$\forall\, v_i \in I,\ v_o \in O,\ t \in [1, T_{max}]$$

where

$$t_{range} = t - X(v_i, v_o) - \max_{g \in [1, G(v_i)]} (r(v_i, g)) + 1$$

The constraints in Equation (4.9) specify that the "overlap" flag, $L(v_i, t)$ is true if the sailing time t for an outgoing ship v_o overlaps with the extra tug delay time $X(v_i, v_o)$ for this incoming vessel v_i.

The constraints in Equation (4.8) express the requirement that for any incoming vessel v_i, the tugs from that vessel are still considered busy for vessels sailing in the opposite direction at time t if the overlap flag $L(v_i, t)$ is true.

Scenario 3: Outgoing Followed By Incoming

These constraints are identical to the CP model.

Tug Availability Constraints

These constraints are identical to the CP model.

4.3.5 MIP vs CP Comparison

The MIP model was implemented in MiniZinc 1.4 and solved for all the same example problems as those described in Section 4.2, using the MIP OSI CBC solver [Lougee-Heimer, 2003] included in the G12 optimisation system described by Stuckey et al. [2005]. Table 4.8 presents the number of ships for the largest problem solved for each problem type, with the CPU time in brackets, for the MIP model compared against the fastest CP search strategy – OBJFUN_FIRSTFAIL_MAX. For MIP, the search strategy is set by the solver; however, all CP search strategies that search first on draft or objective function component outperform the MIP solver, with the exception of the MIXED_NARROW problems without tug constraints, where the MIP solver outperforms each of the OBJFUN_LARGEST_MEAN, OBJFUN_LARGEST_MAX and DRAFT_FIRSTFAIL_MAX search strategies, but is still outperformed by all the other CP approaches.

Problem	CP	MIP
Mixed_Wide	**10 (3.17)**	8 (39.1)
OneWay_Wide	**10 (0.45)**	9 (41.8)
Mixed_Narrow	**10 (7.99)**	8 (11.4)
OneWay_Narrow	**8 (42.5)**	7 (11.4)
Mixed_Wide_Tugs	**10 (115)**	6 (180)
OneWay_Wide_Tugs	**8 (1.76)**	8 (273)
Mixed_Narrow_Tugs	**8 (4.06)**	6 (126)
OneWay_Narrow_Tugs	**6 (10.3)**	5 (7.66)

Table 4.8: Largest problem solved, with solution time in seconds for MIP vs CP.

Table 4.8 shows that CP with a good choice of search strategy was able to solve larger problem sizes to optimality within the cutoff time for almost all problem types, and was the fastest to solve all problems. The MIP solver was particularly slow for Mixed problems with tugs, possibly indicating that the tug constraints for Scenario 4 – an incoming ship followed by an outgoing ship, which occurs only in Mixed problem types – were particularly inefficient in the MIP implementation.

One possible reason for the CP solver being faster than the MIP solver for this set of problems is that the Draft and ObjFun search strategies allow large areas of the search space to be eliminated quickly by the finite domain solver. The search strategy chosen by the MIP solver may be less efficient.

While the specific MIP solver considered in this chapter resulted in much longer CPU times than the CP solver, the use of MIP for solving this scheduling problem may be worth investigating further. In particular, there may be ways to improve the constraints used in the MIP model to make them more efficient. Other solvers (Gecode, CPLEX and Gurobi) were also explored briefly, and though calculation times varied, the overall picture was similar, namely that the fastest CP solver (Gecode) with a good choice of search strategy was still faster to solve most problems than the fastest MIP solver (CPLEX).

4.4 Improvements to the Constraint Programming Model

Earlier sections showed that user-defined search strategies had a significant impact on the time taken for a solver to prove optimality for this problem, and that the choice of model and solver also significantly affected the calculation time. This section investigates the impact of CP modelling approaches on solver calculation time by comparing several modifications the CP model that may make it faster to solve.

The calculation times presented in this chapter are quite implementation-specific for the G12 finite domain CP solver; however, the more general issue is that solver implementation, user-defined search strategies, as well as the CP modelling approach all affect calculation time. The interaction between these factors needs to be taken into account when designing CP models, choosing solvers, and selecting search strategies. Also, G12 as a product can benefit from the experience gained as a result of this investigation into its effectiveness for this problem, and several improvements to G12 are suggested at the end of this section.

4.4.1 Sequence Variables and Time Search

The results in Section 4.2 showed that searching on draft or on the objective function component of each ship finds optimal solutions faster than searching directly on time. However, since the draft and objective function component can have the same value for multiple time slots, the exact time for each ship still needs to be determined after the draft or objective function component have been decided. As a result, it is worth investigating several search strategies on time in more detail.

This subsection investigates the effects of the variable selection and domain reduction methods on calculation time when searching on time as a second search step, after first including all ships in the schedule, and searching on draft or objective function component.

Two variable selection methods are compared with a search strategy that searches directly on the time variables for each ship – Smallest, which chooses the variable with the smallest lower bound, ie. the ship which has the earliest allowable sailing time; and FirstFail, which chooses the variable with the smallest time domain.

An alternative method of searching on time – SeqVar – is also investigated. This approach extends the CP model for the BPCTOP with redundant sequence variables specifying ordering between every pair of vessels in the schedule. The SeqVar search strategy searches on these variables first, prior to setting the exact sailing time for each ship. This modification investigates whether setting the order in which ships sail prior to choosing the exact sailing times would reduce the search, and thus speed up the time required to find an optimal solution.

New Variables

The following new variables and constraints specify the modified CP model for the BPCTOP with redundant sequence variables specifying the ordering between every pair of vessels in the schedule. This modification required the addition of a new set of dependent variables to the original CP model presented in Chapter 3, as well as the modification of several constraints.

$sb(v_i, v_j) \in [0,1]$ – SailsBefore(v_i, v_j) – is a binary variable which is set to 1 if the vessel v_i sails earlier than the vessel v_j, ie. if $T(v_i) < T(v_j)$, and 0 otherwise.

Sequence Variable Constraints

$$sb(v_i, v_j) = 1 \leftrightarrow T(v_i) < T(v_j), \qquad \forall\, v_i,\ v_j \in V; v_i \neq v_j \tag{4.10}$$

$$sb(v_i, v_j) = 1 \leftrightarrow sb(v_j, v_i) = 0, \qquad \forall\, v_i,\ v_j \in V; v_i \neq v_j \tag{4.11}$$

$$sb(v_i, v_i) = 0, \qquad \forall\, v_i \in V \tag{4.12}$$

The constraints in Equations (6.16), (4.11) and (6.19) define the values of the sequence variables $sb(v_i, v_j)$ introduced above.

Problem	FirstFail_Min	Smallest_Min	SeqVars
Mixed_Wide	10 (5.48)	**10 (1.53)**	10 (2.09)
OneWay_Wide	**10 (1.22)**	10 (108)	10 (30.7)
Mixed_Narrow	**9 (292)**	8 (80.5)	9 (298)
OneWay_Narrow	8 (75.5)	8 (203)	**8 (57.9)**
Mixed_Wide_Tugs	10 (219)	**10 (9.83)**	10 (14.1)
OneWay_Wide_Tugs	**8 (1.76)**	**8 (1.76)**	8 (14.4)
Mixed_Narrow_Tugs	**8 (138)**	8 (139)	**8 (137)**
OneWay_Narrow_Tugs	6 (20.8)	6 (21.2)	**6(13.0)**

Table 4.9: Search strategies on the time dimension - largest problem solved, with CPU times in seconds.

Equation (6.16) specifies that the vessel v_i "sails before" v_j if the scheduled sailing time $T(v_i)$ for v_i is earlier than the scheduled sailing time $T(v_j)$ for v_j.

Equation (4.11) requires that if vessel v_i "sails before" v_j, then v_j cannot sail before v_i, and Equation (6.19) specifies that no vessel can sail before itself.

Separation Time Constraints

$$\begin{aligned} s(v_i) = 1 \ \wedge s(v_j) = 1 \Rightarrow \\ (sb(v_i, v_j) = 1 \Rightarrow T(v_j) - T(v_i) \geq ST(v_i, v_j)), \\ \forall \ v_i, \ v_j \in V \end{aligned} \tag{4.13}$$

The separation time constraints originally introduced in Chapter 3 are modified to depend on the "sails before" sequence variables $sb(v_i, v_j)$. The modified constraints above represent the requirement that for each pair of ships v_i, v_j, either v_i sails first ($sb(v_i, v_j) = 1$), followed by v_j with a minimum separation time $ST(v_i, v_j)$ between their scheduled sailing times, or v_j sails first ($sb(v_j, v_i) = 1$), followed by v_i with a minimum separation time $ST(v_j, v_i)$ between them.

Experimental Results

The same set of problems as in section 4.2 were solved with each of the FirstFail, Smallest and SeqVars variable selection strategies. These time search strategies were all applied after the Draft search strategy with Largest variable selection and Max domain reduction. As in earlier sections, all calculations were run with a 300-second cutoff time.

Table 4.9 presents the largest problem that could be solved to optimality within the 300-second cutoff time by each variable selection strategy, for each problem type with and without tugs, with the corresponding CPU time in seconds. Table 4.9 shows that the Smallest_Min search strategy was fastest for the least constrained problems – Mixed_Wide with and without tugs. This is not surprising, since selecting the earliest allowable sailing time will find a solution quickly when the problem is underconstrained.

However, for the more tightly constrained problem sets – OneWay_Wide and Mixed_Narrow with and without tugs – FirstFail_Min was the fastest search strategy. For the most tightly constrained problems – OneWay_Narrow with and without tugs – the sequence variable model and SeqVars search strategy resulted in the fastest solution times.

One possible reason for this behaviour is that the added complexity of the additional variables used in the sequence variable model slows down calculation times for the underconstrained problems; however, for over-constrained problems, larger areas of the problem domain can be cut out by searching on sequence variables before searching on time. This speedup in search is only an effective tradeoff for the most tightly constrained OneWay_Narrow problem type.

4.4.2 Other Improvements

Other improvements to the CP model were also investigated. One variation to the CP model which was effective in speeding up the calculation time, particularly for the model with sequence variables, was to convert multi-dimensional array lookups with a variable index to use one-dimensional arrays instead. The objective function uses the term $D(v, T(v))$, where v is a constant and $T(v)$ is a variable. Constraints on variables which index into multi-dimensional arrays are inefficiently handled in MiniZinc, since this type of constraint is implemented using an element constraint with a flattened array as well as a second constraint which models the relationship between the indices of the multi-dimensional array and the index of the flattened array. Pruning on the value of the array lookup gives little information about the values of the original index variables. A better model is achieved by replacing $D(v, T(v))$ with $D'_v(T(v))$ where D'_v is the projection of D on v, since a one-dimensional array lookup is implemented with only an element constraint, so pruning the value of the array lookup gives maximal information about the index variables[1].

Another modification that improved calculation time was sorting ships into ascending order of maximum objective function component $\max_{t \in [1..T_{max}]} D(v,t) \cdot C(v)$, where $D(v,t)$ is the maximum allowable draft for vessel v at time t, and $C(v)$ is the tonnage per centimetre of draft for vessel v. This improved the efficiency of the search slightly, as it allowed sailing times to be searched in order of ship size. Sorting the array of ships is required as a workaround, since MiniZinc does not implement a user-defined search strategy which sorts one variable by another variable with the same index. Sorting is thus a proxy for a more complex user-defined variable ordering heuristic.

Experimental Results

The modified CP models were compared against the original model introduced in Chapter 3, with the same set of example problems as used for earlier experiments.

[1]Note: the implementation of multi-dimensional array lookups with variable indices has been changed in the latest version of MiniZinc.

Problem	OLD	ONEDIM	SORT	ONEDIMSORT
MIXED_WIDE	10 (3.17)	**10 (0.22)**	10 (1.00)	10 (0.56)
ONEWAY_WIDE	10 (0.45)	10 (2.62)	10 (0.56)	**10 (0.34)**
MIXED_NARROW	10 (7.99)	**10 (0.22)**	10 (8.30)	10 (0.56)
ONEWAY_NARROW	8 (42.5)	10 (190)	8 (37.1)	**10 (157)**
MIXED_WIDE_TUGS	10 (115)	**10 (14.3)**	10 (124)	10 (19.4)
ONEWAY_WIDE_TUGS	8 (1.76)	9 (124)	8 (2.09)	**9 (100)**
MIXED_NARROW_TUGS	8 (4.06)	10 (166)	8 (4.16)	**10 (121)**
ONEWAY_NARROW_TUGS	6 (10.3)	9 (228)	6 (7.75)	**9 (211)**

Table 4.10: Comparison of original vs modified CP models - largest problem solved, with CPU time in seconds.

The original CP model was tested with the fastest search strategy as found in Section 4.2 – DRAFT_LARGEST_MEAN. The improved CP models were tested with the search strategies that were fastest for each model, as investigated further in this section.

The results of these comparisons are shown in Table 4.10. The CP model with one-dimensional arrays performed significantly better than the fastest original CP model for all problem types, successfully solving problems with an average of 2 more ships than the original CP model. The CP model with sorted ships only resulted in a very small improvement on the calculation times of the original CP model, and in some cases resulted in slower calculation time. However, when combined with one-dimensional arrays, sorting the ships resulted in faster calculation times for the most difficult problem types – ONEWAY_NARROW with and without tugs, and MIXED_NARROW and ONEWAY_WIDE with tugs.

4.4.3 Search Strategies for the Improved CP Model

A number of search strategies were also compared for the modified CP models, to test whether the modifications affected which search strategies were able to find optimal solutions fastest. The search strategies compared in this section include:

1. DRAFTFF – Searching on draft first, with the FIRSTFAIL variable selection strategy and MEAN domain reduction.
2. DRAFTLARGEST – Searching on draft first, with the LARGEST variable selection strategy and MEAN domain reduction.
3. OBJFF – Searching on objective function first, with the FIRSTFAIL variable selection strategy and MEAN domain reduction.
4. OBJLARGEST – Searching on objective function first, with the LARGEST variable selection strategy and MEAN domain reduction.
5. SEQVARSFIRST – Searching on sequence variables first, followed by DRAFT with the LARGEST variable selection strategy and MEAN domain reduction.

Problem	DraftFF	DraftLargest	ObjFF	ObjLargest	SeqVarsFirst
MW	10 (0.55)	**10 (0.45)**	**10 (0.44)**	**10 (0.44)**	10 (0.56)
OW	**10 (0.34)**	**10 (0.33)**	**10 (0.33)**	**10 (0.33)**	**10 (0.34)**
MN	**10 (0.34)**	**10 (0.33)**	**10 (0.33)**	**10 (0.33)**	10 (0.56)
ON	8 (23.0)	8 (22.8)	8 (29.4)	8 (17.9)	**10 (157)**
MWT	10 (50.7)	10 (50.7)	10 (51.5)	10 (50.9)	**10 (19.4)**
OWT	8 (0.89)	8 (0.89)	8 (0.89)	9 (262)	**9 (100)**
MNT	9 (277)	9 (40.1)	9 (84.0)	8 (1.33)	**10 (121)**
ONT	7 (138)	7 (119)	7 (171)	7 (132)	**9 (211)**

Table 4.11: Comparison of search strategies for the CP model with all three improvements.

Problem	DraftLMean	SeqVarsFirst
MW	**10 (0.22)**	**10 (0.22)**
OW	**10 (2.40)**	10 (2.62)
MN	**10 (0.22)**	**10 (0.22)**
ON	8 (18.4)	**10 (190)**
MWT	10 (48.8)	**10 (14.3)**
OWT	8 (1.43)	**9 (124)**
MNT	9 (27.8)	**10 (166)**
ONT	7 (98.3)	**9 (228)**

Table 4.12: Searching by draft vs sequence variables for improved model with one-dimensional arrays only.

All of the DraftFF, DraftLargest, ObjFF and ObjLargest search strategies also searched on sequence variables as a second search step.

Improved Model with One-Dimensional Arrays and Sorted Inputs

Table 4.11 shows the largest problem solved and corresponding CPU time in seconds for each of these five search strategies using the CP model with all three improvements: sequence variables, one-dimensional arrays and sorted inputs. The SeqVarsFirst search strategy was significantly faster than all the other search strategies on all the most difficult problems, successfully solving larger problem sizes for all problems with tugs, as well as for the most difficult OneWay_Narrow problem without tugs.

When the problem is expressed with one-dimensional arrays and sorted inputs, searching on sequence variables first is faster than searching on draft first. Tables 4.12 and 4.13 show that this effect is repeated for one-dimensional arrays only, but not for sorted inputs only. This implies that sequence variable constraints in particular propagate better when expressed with one-dimensional arrays, rather than as multi-dimensional array lookups with a variable index.

Problem	DraftLMean	SeqVarsFirst
MW	**10 (1.00)**	8 (179)
OW	**10 (0.56)**	8 (136)
MN	**10 (8.30)**	8 (54.5)
ON	8 (37.1)	**9 (204)**
MWT	**10 (124)**	7 (9.83)
OWT	**8 (2.09)**	8 (257)
MNT	**8 (4.16)**	7 (21.9)
ONT	6 (7.75)	**8 (282)**

Table 4.13: Searching by draft vs sequence variables for improved model with sorted ships only.

Improved Model with One-Dimensional Arrays Only
Table 4.12 shows the largest problem solved and corresponding CPU time for the two search strategies that were tested with one-dimensional arrays only. As for the improved model with both one-dimensional arrays and sorted inputs, searching on sequence variables first allows larger problem sizes to be solved for the most complex problems.

Improved Model with Sorted Inputs Only
Table 4.13 shows the largest problem solved and corresponding CPU time for the two search strategies that were tested with sorted inputs only. For sorted inputs only, searching on sequence variables first results in slower calculation times for most problems. Sequence variable search is also only able to solve problems that are 1-2 ships smaller than the largest problems solved by draft search for most problem types, with the exception of OneWay_Narrow problems, where sequence variable search is able to find solutions for larger problem sizes.

For all search strategies, the improved model with sorted inputs only always performs worse than the improved model with one-dimensional arrays, or the improved model with both one-dimensional arrays and sorted inputs.

Discussion Improvements to the CP model were found to improve calculation speed significantly, and were much more effective than any of the small improvements produced by the draft and objective function search strategies investigated in Section 4.2.2. In particular, expressing multi-dimensional array lookups with a variable index as one-dimensional arrays provided the most significant improvements, particularly when combined with sequence variable search. These improvements both arose from the implementation details and limitations of the G12 finite domain solver, showing that solver implementation has a significant impact on what modelling approaches and search strategies are most effective for a particular problem.

Chapter 3 discussed aspects of the CP model that led to improved calculation time, such as the approach of splitting the tug constraints into four scenarios. These improvements to the model were highly problem-dependent, and could only be identified by a human; however, converting multi-dimensional arrays to one-dimensional arrays is a more mechanical feature of a CP model, which can be built into a CP solver or modelling language. Automatically translating multi-dimensional array lookups to one-dimensional arrays may therefore be worth investigating as a potential improvement to the MiniZinc modelling language and the G12 system.

Sorting the inputs prior to solving the model also improved calculation time. This has more general implications for other CP problems – any precomputation of inputs, such as sorting inputs for easier search, may result in improved calculation speed.

Precomputation of inputs was also used in the basic ship scheduling constraints introduced in Chapter 3. The asymmetric separation times between ships originate from symmetric separation times between ships at waypoints (usually narrow or shallow points) along the channel. An early version of the CP model presented in Chapter 3 defined several waypoints with a fixed separation time at each waypoint, and distances between berths and waypoints along the channel.

However, solving this subproblem every time the separation time constraints needed to be evaluated significantly increased the time required to find optimal schedules. Precomputing separation times between every pair of ships could be done easily by an external script, and resulted in significantly faster runtimes for the CP solver than modelling separation times separately at a series of waypoints along a channel.

This suggests a possible improvement to the G12 optimisation system or other CP solvers: automatically identifying easily solvable subproblems which could be precomputed for every possible combination of inputs rather than being solved repeatedly as part of the optimisation process. If the process of finding subproblems for precomputation can be automated, many other problems are likely to benefit from the increase in calculation speed, and this will significantly reduce the amount of work required for human experts in identifying such subproblems manually.

4.5 Benders Decomposition

The results presented in Sections 4.2 to 4.4 show that the addition of tug constraints significantly increases the difficulty of this ship scheduling problem. For all problems and search strategies investigated in this chapter, the addition of tug constraints reduced the largest problem size that could be solved within the cutoff time by around 2 ships on average.

The tug constraints are clearly the hardest part of the problem; however, the scheduling problem can be solved with or without tug constraints. This leads to the possibility of speeding up the solution time of large scheduling problems by decomposing the problem into scheduling and tug availability subproblems.

Benders decomposition, first introduced by Benders [1962], as discussed in Chapter 2, is a commonly used method for decomposing hard linear programming problems into a master and subproblem which are smaller and more tractable than the combined problem. The problems are then solved iteratively, with the master problem providing a bound on optimal solution quality, and the subproblem providing cuts that are used to remove infeasible areas of the master problem domain from the search. Logic-based Benders decomposition is an extension of the original Benders decomposition method to CP problems [Hooker and Ottosson, 2003]. This seems a suitable way to apply Benders decomposition to the BPCTOP, since CP was found to be more effective than MIP for the BPCTOP in Section 4.3.

In logic-based Benders decomposition, the master problem generates a candidate solution which is used as a bound on the optimal solution quality. The subproblem is then used to generate *cuts* – new constraints – which are added to the master problem for future iterations.

Efficient logic-based Benders cuts for scheduling problems have been investigated for a number of objective functions, such as minimum makespan and minimum total tardiness. However, the time-varying value objective function of the BPCTOP does not fit into any of these categories, so new Benders cuts need to be identified for this model.

This section presents an initial approach to a logic-based Benders decomposition model of the Bulk Port Cargo Throughput Optimisation Problem, with the ship scheduling problem as the master problem and constraints on tugs (Equations (3.7) to (3.13)) as the subproblem. For any time slots where the tug constraints are breached in the subproblem, the tug constraints are then added back into the master problem in the next iteration. We call this the *tug cuts* approach, to differentiate it from the alternative approach with a different set of Benders cuts (*separation time cuts*) presented later in Section 4.5.3. The performance of both CP and MIP models for the master and subproblem are compared using the same problem set as earlier in this chapter.

4.5.1 Benders CP and MIP Models

To implement the *tug cuts* Benders decomposition approach, the CP model for the BPCTOP had to be split into two halves – a ship scheduling master problem and tug constraints subproblem.

Master Problem

The master problem contained all the constraints and variables introduced in Chapter 3, without tugs. After the first iteration, tug constraints get added back in, but only for time points where these constraints were breached in the subproblem. This requires the addition of two new parameters in the tug constraints – T_{start} and T_{end} – which specify the range of time points where the tug constraints need to be applied.

All the tug constraints that apply for all times $t \in [1, T_{max}]$ are modified in the Benders master problem to apply for $t \in [T_{start}, T_{end}]$ instead. These include the constraints presented in equations (3.7) to (3.13) in Chapter 3.

Subproblem

Output of Master Problem – The output of the master problem is a schedule which specifies whether each vessel v_i is included in the schedule – $s(v_i)$ – as well as scheduled sailing time $T(v_i)$ for each ship. These outputs are then used to set the values of the corresponding variables in the subproblem, for validation of the constraints on tugs.

Variables – The subproblem has a new boolean decision variable $a(t)$ which is true if the tug availability constraints can apply for time $t \in [1, T_{max}]$, and false if the tug availability constraints were breached for the given schedule at time t.

Constraints – The subproblem only contains the constraints on tugs; all other constraints are left out of the subproblem, as the solution to the master problem will already satisfy all those constraints.

The tug availability constraints introduced in Equations (3.12) and (3.13) in Chapter 3 are modified in the subproblem to only hold for time points when tug constraints apply, ie. $a(t)$ is true. The modified tug availability constraints are shown below.

$$\begin{aligned} \sum_{v \in I} \sum_{g \in G(v)} U(v,t,g) \leq U_{max} \wedge \\ \sum_{v_o \in O} \sum_{g \in G(v_o)} U(v_o,t,g) + \sum_{v_i \in I} X(v_i,t) \leq U_{max} \\ \rightarrow a(t) = 1, \;\; \forall\, t \in [1, T_{max}] \end{aligned} \tag{4.14}$$

$$\sum_{v \in I} \sum_{g \in G(v)} U(v,t,g) > U_{max} \rightarrow a(t) = 0, \;\; \forall\, t \in [1, T_{max}] \tag{4.15}$$

$$\sum_{v_o \in O} \sum_{g \in G(v_o)} U(v_o,t,g) + \sum_{v_i \in I} X(v_i,t) > U_{max} \rightarrow a(t) = 0, \;\; \forall\, t \in [1, T_{max}] \tag{4.16}$$

Equation (4.14) specifies that if the total number of tugs used for both incoming and outgoing ships at time t is less than or equal to the total number of tugs U_{max} available at the port, then tug constraints apply and $a(t)$ must be true (1).

Equations (4.15) and (4.16) specify that if the number of tugs used for either incoming or outgoing ships at time t is greater than the total number of tugs U_{max} available at the port, then the tug constraints are breached at time t, and $a(t)$ is false (0).

Objective Function – The objective for the subproblem, given by Equation (4.17), maximises the total number of time slots when the tug constraints apply.

$$\max \sum_{t \in [1, T_{max}]} a(t) \tag{4.17}$$

Output – The result of the subproblem is the list of times when tug constraints

Problem	BendersCP	BendersMIP	Comb.
MWT	9 (15.9)	7 (16.6)	**10 (115)**
OWT	8 (16.3)	7 (22.3)	**8 (1.75)**
MNT	**8 (2.98)**	7 (200)	8 (4.06)
ONT	**6 (6.58)**	5 (5.15)	6 (10.3)

Table 4.14: Benders CP and MIP models vs. combined CP model – largest problem solved for each type, with corresponding CPU time in seconds.

Problem	MP_CP	MP_MIP	SP_CP	SP_MIP
MWT_7	**0.67**	16.0	0.89	**0.67**
OWT_7	**1.42**	1.64	0.67	**0.45**
MNT_7	**0.45**	194	0.87	**0.67**
ONT_5	**0.55**	1.97	0.55	**0.44**

Table 4.15: MIP vs CP models for the Benders Master and Subproblems – CPU times for the final iteration in seconds.

were breached, which is then passed into the master problem to add back the tug constraints for the range of times where tug constraints were breached.

Benders MIP Model The MIP model introduced in Section 4.3 also required modification to be converted into a master and subproblem for Benders decomposition. These modifications are all identical to those required for the CP model above.

4.5.2 Experimental Results

MiniZinc and G12 do not yet support Benders decomposition, so the iteration between master and subproblem was implemented externally via a Python script. The Benders CP and MIP implementations were tested with the same set of example problems as used for tests in earlier sections of this chapter; however, only the problems with tug constraints were used, as the subproblem would not exist for any problem without tugs.

Table 4.14 shows the largest problem solved for each problem type using the Benders CP and MIP implementations, as well as a combined CP model, with the corresponding CPU time. Table 4.15 shows the calculation time for MIP and CP for the Benders master or subproblem in the final iteration of the largest problem to be solved by both MIP and CP. This allows the calculation speed to be compared for the master and subproblem individually. The search strategy used for the Benders Master problem was the search strategy found to be the fastest for the combined problem – Draft_Largest_Mean.

Table 4.14 shows that, as with the combined problem, CP outperforms MIP for every problem type. However, looking at the master and subproblem individually, MIP was always faster for the subproblem, even though CP was always faster for the master problem. Even though the subproblem CPU times are small compared to the master problem, a combined approach using CP for the master problem and MIP for the subproblem would result in slightly faster calculation times than either a pure MIP or pure CP approach.

The Benders *tug cuts* approach provides a simple way to validate that Benders decomposition can be used to decompose the scheduling and tug availability problems. However, the results in Table 4.14 show that this approach is quite inefficient – the Benders approach did not solve any larger problems than the combined model. The Benders model provided a small speed improvement for two problem types, but was slower than the combined model for the other two.

The likely reason for this is that the *tug cuts* approach is quite inefficient, since the tug constraints get added back to the master problem at exactly the most tightly constrained times in the schedule. This means that the later iterations of the master problem are almost as slow as solving the combined problem.

However, this was only the first, simplest attempt at implementing Benders decomposition for this problem. Simpler, more efficient Benders cuts may result in much faster calculation times, possibly improving on the combined problem results.

4.5.3 Alternative Benders Cuts

An alternative set of logic-based Benders cuts was implemented, to investigate the performance of simpler cuts which have shorter calculation time, but which may require more iteration.

Similarly to the previous simple Benders approach, ships are scheduled in the master problem and constraints on tugs are checked in the subproblem. However, if the tug constraints are breached in the subproblem, a different set of cuts – new constraints – is added to the master problem. These new cuts specify that for ships involved in breaching the tug constraints, either their sailing order must change, or the sum of the separation times between ships must increase. We refer to this as the *separation time cuts* approach.

The separation time cuts approach never cuts out any valid solutions, since if the tug constraints are breached for a given set of ships with a given set of scheduled sailing times, then the tug constraints will still be breached in any schedule with an equal or smaller sum of separation times between those ships, and with the same sailing order.

The separation time cuts also ensure that the successive iterations of the Benders master problem will eventually find a solution that satisfies the tug constraints if one exists, since the increase in separation times will eventually push any set of ships that breach the tug constraints further apart, until enough tugs are available at each point in time.

4.5.4 New Benders CP Model

The subproblem uses the same CP model as presented in Section 4.5.1.

The CP model for the master problem no longer includes any constraints on tugs at all. The master problem model contains all constraints from the original CP model presented in Chapter 3 without tugs, with the following additional parameters and constraints.

Parameters

N_c defines the number of separation time cuts added to the original master problem.

$N_v(c)$ is the number of ships that contributed to the breach of tug constraints, and therefore that are involved in the separation time cut c.

$O(c, v)$ defines the ordering of vessels $v \in [1, N_v(c)]$ that led to a breach of tug constraints, and that therefore must be avoided in separation time cut c.

ST_{min} is a lower bound on the sum of the separation times between the ships listed in $O(c, v)$.

The values of the above parameters are set from the results of each iteration of the subproblem. N_c increases by 1 after each iteration, since all previous cuts still need to be applied. The results of the subproblem allow the ships that contributed to the tug constraint breach to be identified, as all ships which sailed during or before the tug constraint breach – T_{start} to T_{end} introduced in Section 4.5.1 – and for which at least some tugs had not yet become available for their next job at the start of the tug constraint breach, ie. the tug turnaround time $r(v, g)$ had not yet passed. Ships contributing to the tug constraint breach are those vessels v that satisfy Equation (4.18), below.

$$T(v) \leq T_{end} \wedge T(v) + \max_{g \in G(v)} r(v, g) > T_{start} \tag{4.18}$$

Constraints

The master problem with separation time cuts requires the addition of the following new constraints.

$$\begin{aligned} \sum_{v \in [1, N_v(c)]} (T(O(c, v)) - T_{earliest}) > ST_{min} \vee \\ \exists\ v_1, v_2 \in [1, N_v(c)]\ \ s.t. \\ v_1 < v_2 \wedge T(O(c, v_2)) < T(O(c, v_1)) \\ \forall\ c \in [1, N_c] \end{aligned} \tag{4.19}$$

where

$$T_{earliest} = \min_{v \in [1, N_v(c)]} T(O(c, v)) \tag{4.20}$$

The constraints in Equations (4.19) and (4.20) specify that for every separation time cut c added to the master problem, the vessels $v \in [1, N_v(c)]$ that contributed

Problem	TugsCP	SepTime	Combined
MWT	9 (15.9)	8 (10.6)	**10 (115)**
OWT	8 (16.3)	7 (54.8)	**8 (1.75)**
MNT	**8 (2.98)**	8 (10.6)	8 (4.06)
ONT	**6 (6.58)**	5 (16.9)	6 (10.3)

Table 4.16: Benders approach with separation time cuts vs. tug cuts and the combined CP problem – CPU time in seconds.

	No. Iterations		Calc Time	
Problem	Tugs	SepTime	Tugs	SepTime
MWT_8	2	10	1.11	1.22
OWT_7	3	43	1.44	0.76
MNT_8	2	10	0.56	0.55
ONT_5	3	23	0.44	0.33

Table 4.17: Number of iterations and CPU times in seconds for the final iteration of the master problem for the largest problem solved by Benders with tug cuts vs. separation time cuts.

to the breach of the tug constraints in the previous iteration of the subproblem must either have a larger sum of separation times than the earlier bound ST_{min}, or must sail in a different order from that defined by the ordering parameters $O(c, v)$ for this cut.

4.5.5 Experimental Results for New Approach

Table 4.16 shows that the separation time Benders cuts are much slower than the tug cuts for every problem type, and for most problems the separation time cuts approach can only solve problems that are one ship smaller than the tug cuts approach. The much longer CPU times are likely due to the number of iterations being much higher for separation time cuts, as shown by Table 4.17, while the calculation time for each iteration is not reduced significantly for those problems that can be solved by Benders decomposition with both types of cuts.

Investigating the specific cuts that get added in each iteration for the separation time cuts approach shows that the ships move apart very slowly, with each iteration changing by only one time step, and with frequent iteration between different orderings that do not resolve any breaches in tug constraints.

The number of iterations taken to find an optimal solution with the separation time cuts approach can perhaps be reduced significantly by using smarter cuts that cut out more of the problem domain in each iteration, but without becoming as complex as adding the tug constraints back entirely. For example, the separation time cuts defined by Equations (4.19) and (4.20) only require that the sum of the separation times between ships be less than that for a schedule that breached the tug constraints. However, we may be able to use a stronger bound by requiring that the sum of the separation times be reduced by at least the number of time steps at which the tug constraints were breached. If the tug constraints were breached at 10 time steps, it is clear that moving one ship by one time step will not resolve the breach.

There may be a middle ground between complex cuts which are slow for each iteration with larger problems, and overly simple cuts which are fast for each iteration even with large problems, but which require too many iterations to find a solution. However, investigation of further logic-based Benders cuts for this problem is left for future work.

4.6 Summary

This chapter investigated a number of approaches to solving the Bulk Port Cargo Throughput Optimisation Problem (BPCTOP) introduced in Chapter 3, including several user-defined search strategies for the CP solver, a comparison against a new MIP model and MIP solver, a number of improvements to the CP model that took into account CP solver implementation details, and finally a Benders decomposition approach. While not all approaches were effective at solving the problem quickly, some approaches were found that significantly improved calculation time, and allowed problems with 9 ships sailing on one high tide to be solved within 5 minutes – this is 50% more than the current record of 6 ships sailing on one high tide at Port Hedland, the world's largest bulk export port [Port Hedland Port Authority, 2013d].

The results in this chapter showed that the choice of solver, model and search strategy all significantly influenced the solution time for this problem, and more importantly, that the interaction between these three factors also had a significant effect. Some approaches to modelling the problem resulted in problem instances being solved faster due to aspects of the model being implemented more efficiently in the solver, and some variations to the model also changed which search strategy was more effective. Similar results are observed for another CP model for a different maritime transportation problem discussed in Chapter 7, so the more general result of this chapter is that the choice of model, solver, and search strategy needs to be considered as an interacting system, rather than in isolation.

This chapter also presents some recommendations for improvement to the G12 finite domain CP solver that arose from the efficiency investigations for the BPCTOP. While these findings are specific to the G12 finite domain solver, other solvers may also have similar implementation issues and may benefit from these ideas. Specifically, this chapter found that multidimensional arrays were handled inefficiently compared to one-dimensional arrays, so automatic conversion of multidimensional arrays to one-dimensional arrays as a preprocessing step may improve performance for some problems. Also, preprocessing of inputs, such as sorting variables or precomputing all possible values for a small subproblem rather than solving it repeatedly, was found to improve calculation speed. This may also be worth investigating as a possible target for automation, since opportunities for preprocessing are hard to identify manually, and require a good understanding of both the problem, and solver limitations.

The next chapter discusses the prototype software for scheduling ship sailing times at a port that was implemented based on the work presented in this chapter, and compares the quality of the solutions produced by the CP model against real schedules created by human schedulers at Port Hedland.

Chapter 5

Real-World Impact

The CP model for the Bulk Port Cargo Throughput Optimisation Problem (BPCTOP) presented in Chapter 3 was incorporated into a commercial ship scheduling system, DUKC® Optimiser, developed by OMC International. The system uses the Dynamic Under-Keel Clearance (DUKC®) software developed by OMC International to calculate draft restrictions. These restrictions are then used as input to the CP model, and solved using the G12 finite domain solver [Stuckey et al., 2005].

This chapter presents the prototype port optimisation system, including user interface, system architecture and a discussion of improvements made in response to user feedback. This chapter also compares the schedules produced by the system against approaches typically used by human schedulers, as well as against a set of real schedules created by human schedulers at Port Hedland.

5.1 The DUKC Optimiser System

5.1.1 Initial Prototype

A command-line prototype of DUKC® Optimiser was developed and tested by port schedulers in late 2010, and demonstrated at the International Conference on Automated Planning and Scheduling (ICAPS) 2011 [Kelareva, 2011]. An updated model containing improvements based on user feedback was incorporated in a commercial system in 2012. This system has now been commissioned and is in use at Port Hedland. DUKC® Optimiser won the NASSCOM Innovation Student Award for IT-Enabled Business Innovation in 2013 [Consensus Group, 2013].

The initial prototype was command-line based, which was sufficient to gather initial user feedback on schedule quality, but would have been inconvenient for operational use. The scheduling system was therefore incorporated into a commercial web-based dynamic under-keel clearance management system - DUKC® Series 5, developed by OMC International.

The initial prototype also did not include tug constraints; however, testing of the prototype on real ship scheduling problems showed that the availability of tugs sometimes constrained the schedule, so schedules produced by the model without tug constraints could be infeasible in practice. Tug constraints therefore needed to be incorporated before the system could be used in practice, which led to the development of the tug constraint model discussed in Chapter 3. The updated system described in the rest of this chapter, which is now in use at Port Hedland, does include tug constraints.

5.1.2 User Interface

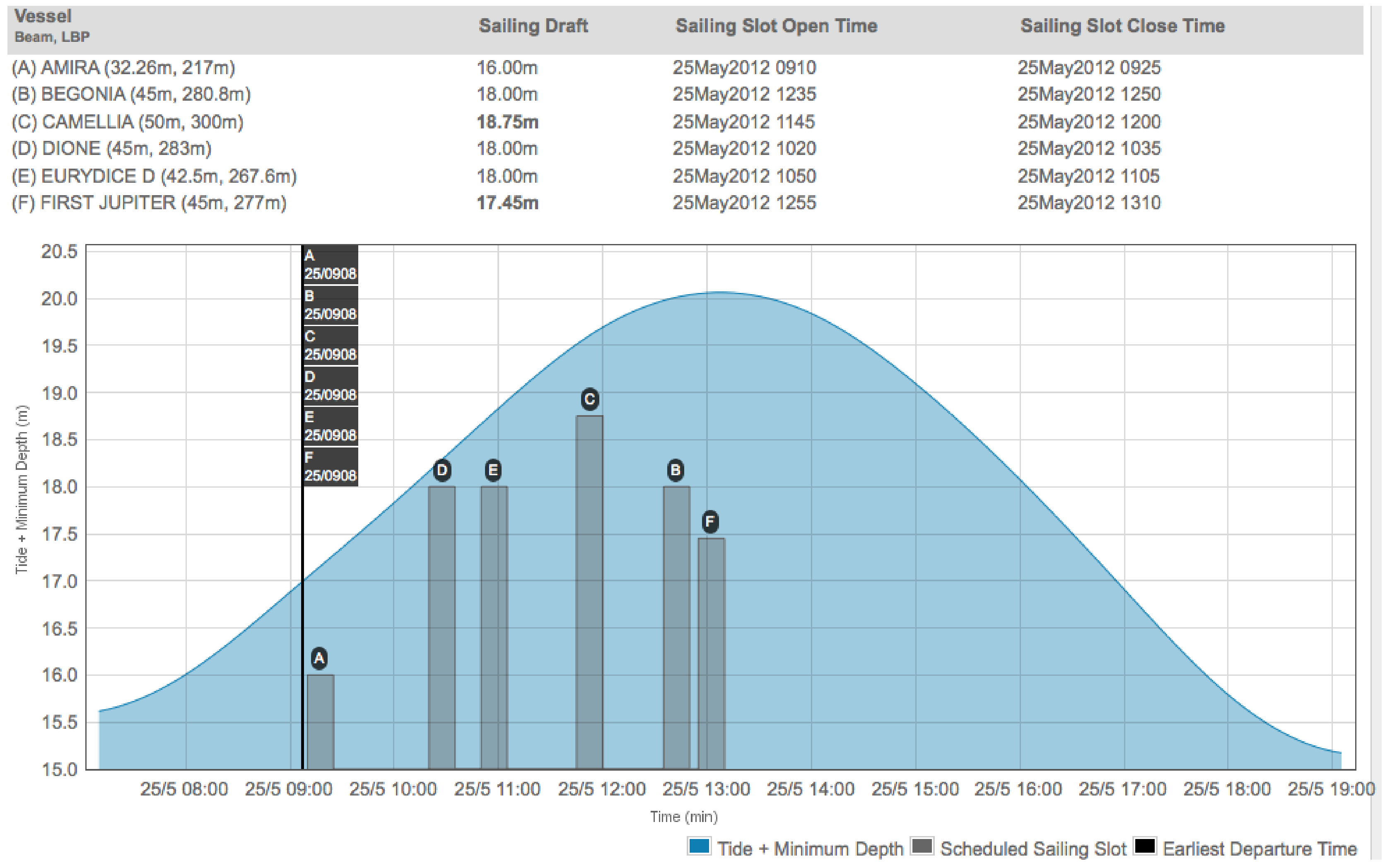
To run a DUKC® Optimiser calculation, the scheduler must enter parameters such as length and beam for each ship that are used to calculate under-keel clearance. Other inputs required to calculate a schedule include the earliest sailing time for each ship, the number of tugs required for each ship, the range of drafts to calculate, and a priority number that is used to ensure fairness to different companies using the port.

Figure 5.1 shows an output schedule for a set of six ships sailing on one tide. In the graph on the bottom half of the screen, each bar represents a ship, with the height of the bar corresponding to the draft that the ship is scheduled to sail with, and the location of the bar along the x-axis indicating the time at which the ship is scheduled to sail. Each ship is scheduled with a time slot of 15 minutes, rather than a fixed time point, as it is impractical to expect a large bulk carrier to sail precisely at a given minute.

The blue curve indicates the minimum water depth along the channel plus the height of the astronomical tide prediction at that point in time. *Astronomical tide* is the prediction of the tide height based on gravitational forces from the moon and sun; this prediction does not allow for short-term variation due to weather conditions, which can be up to 1 metre. The blue line on the graph also does not take into account the ship's wave response, squat and heel, as well as a conservative safety allowance, all of which can decrease the amount of water available to the ship. This results in the height of the blue curve being significantly above the height of the bars indicating the scheduled draft for each ship, since the allowable sailing draft calculation does take all of these factors into account.

The decision to show the height of the tide instead of the allowable draft was made because each ship sails from a different berth, and responds differently to waves and other environmental conditions, so the draft function is slightly different for each ship. If the draft were shown instead, this would result in a more cluttered graph with 6 similar overlapping lines, so the simpler approach of displaying the height of the tide was chosen instead.

The shallowest point along the channel is some way out from the berths at most ports, so the maximum draft that can be obtained for any ship occurs when a ship starts sailing some time before the high tide. In this example, ship C is scheduled with the highest draft, and sails about an hour before the peak of the high tide. Ships that start sailing at high tide will reach the shallowest point of the channel when the tide is already falling, and thus ships B and F must sail with a lower draft.



Vessel Beam, LBP	Sailing Draft	Sailing Slot Open Time	Sailing Slot Close Time
(A) AMIRA (32.26m, 217m)	16.00m	25May2012 0910	25May2012 0925
(B) BEGONIA (45m, 280.8m)	18.00m	25May2012 1235	25May2012 1250
(C) CAMELLIA (50m, 300m)	**18.75m**	25May2012 1145	25May2012 1200
(D) DIONE (45m, 283m)	18.00m	25May2012 1020	25May2012 1035
(E) EURYDICE D (42.5m, 267.6m)	18.00m	25May2012 1050	25May2012 1105
(F) FIRST JUPITER (45m, 277m)	**17.45m**	25May2012 1255	25May2012 1310

Figure 5.1: DUKC® Optimiser Output

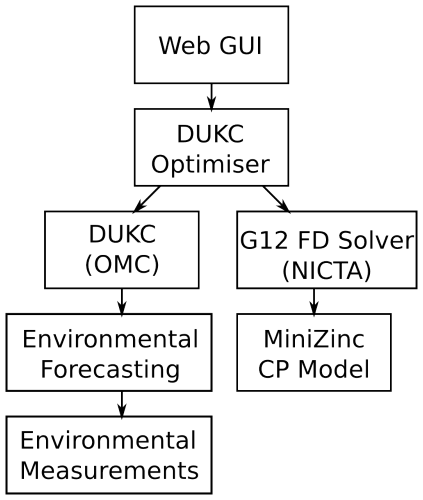


Figure 5.2: DUKC® Optimiser System Architecture

5.1.3 Other Features

The web-based DUKC® Series 5 system includes a number of additional features to assist port schedulers, pilots and harbourmasters in making ship scheduling and sailing decisions. Port administrators may limit user permissions to only access schedules for ships belonging to their organisation, or to only access features that are required for their job. The system can also display live environmental data, such as measurements from wave buoys and tide gauges. The system also includes the DUKC® calculations used to provide under-keel clearance advice to pilots and ship captains immediately prior to sailing, and an in-transit monitoring tool that tracks the locations of ships at the port.

5.1.4 System Architecture

Figure 5.2 summarises the high-level system architecture. When the scheduler selects a set of ships to be scheduled in the web-based GUI, a query is sent to the DUKC® Optimiser server. Upon receiving a schedule query, DUKC® Optimiser converts it into a set of queries to OMC's DUKC® software. The DUKC® software uses real-time environmental forecasts and measurements to analyse each ship's motion, and thus to calculate the ship's under-keel clearance – the amount of water under the ship at each point in the transit. This produces sailing windows for a range of drafts for each ship.

DUKC® Optimiser then converts the user inputs and the results of the DUKC® calculations into the input data required for the Constraint Programming (CP) model introduced in Chapter 3. The model is specified in the MiniZinc optimisation language [Nethercote et al., 2007] and solved using the G12 finite domain solver [Stuckey et al., 2005]. The DUKC® Optimiser GUI then displays the resulting schedule.

5.2 Schedule Quality Analysis

This section presents comparisons of schedules produced by our model for the BPC-TOP against fixed-draft approaches typically used in academia, as well as some typical heuristics used by human schedulers at ports. This section compares these approaches on some simple examples, to illustrate why common heuristics may fail to find optimal solutions even for small problem sizes with only 3 ships. The next section presents a more extensive comparison of our approach against a data set of 2 months' worth of real schedules created by human schedulers at Port Hedland.

5.2.1 Comparison Against Fixed Draft

In existing ship scheduling and routing problems, draft restrictions are generally considered as fixed constant maximum drafts, for example Christiansen et al. [2011]; Korsvik, Fagerholt, and Laporte [2010]; Bausch, Brown, and Ronen [1998]. Fixed-time draft restrictions produce good schedules for problems with small (ie non-draft-restricted) ships, or for non-tidal ports. However, the majority of the world's sea ports are affected by tides, which will cause the draft restrictions at the port to vary with time. For ports that use time-varying draft restrictions, scheduling draft-limited ships using fixed-time draft restrictions can result in sub-optimal schedules.

Figure 5.3 shows one example of a schedule with three ships sailing on a tide, with time-varying or fixed-time draft restrictions. The actual, time-varying, draft restrictions used at the port will allow ship A to sail with 18.1m draft, as shown in Figure 5.3(a). A scheduling algorithm with fixed constant draft restrictions, on the other hand, has to set the draft restriction low enough to ensure that all ships will be able to sail with this draft. This results in all three ships A, B and C sailing with 18.0m draft, as shown in Figure 5.3(b).

Assuming that all ships can carry 130 tonnes of cargo per centimetre of additional draft, as is the average for iron ore bulk carriers at Port Hedland [Port Hedland Port Authority, 2011b], the sub-optimal fixed-draft schedule shown in Figure 5.3(b) will result in 1300 tonnes less iron ore being carried on this set of ships. This is around US$175,500 less iron ore, at the January – July 2013 average iron ore price of around US$135/tonne [Index Mundi, 2013]. Note that all cost and revenue figures in this thesis are given in US dollars unless stated otherwise, which was close to parity with the Australian dollar at the time of writing.

Around 1300 ships sailed from Port Hedland in the 2009-10 financial year [Port Hedland Port Authority, 2011a]. A 10cm reduction in draft of every 3rd ship will result in 563Kt less iron ore being shipped on the same set of ships over the course of a year, or around US$76 million less iron ore per year. Given that a typical iron ore bulk carrier can transport 170,000 tonnes of iron ore [Port Hedland Port Authority, 2011a], this results in 3.3 extra voyages needing to be made per year.

In reality, using fixed constant draft restrictions may have an even larger reduction in draft for some ships, since the height of each tide is not constant throughout the year, but varies with the spring-neap tidal cycle by as much as several metres. Any

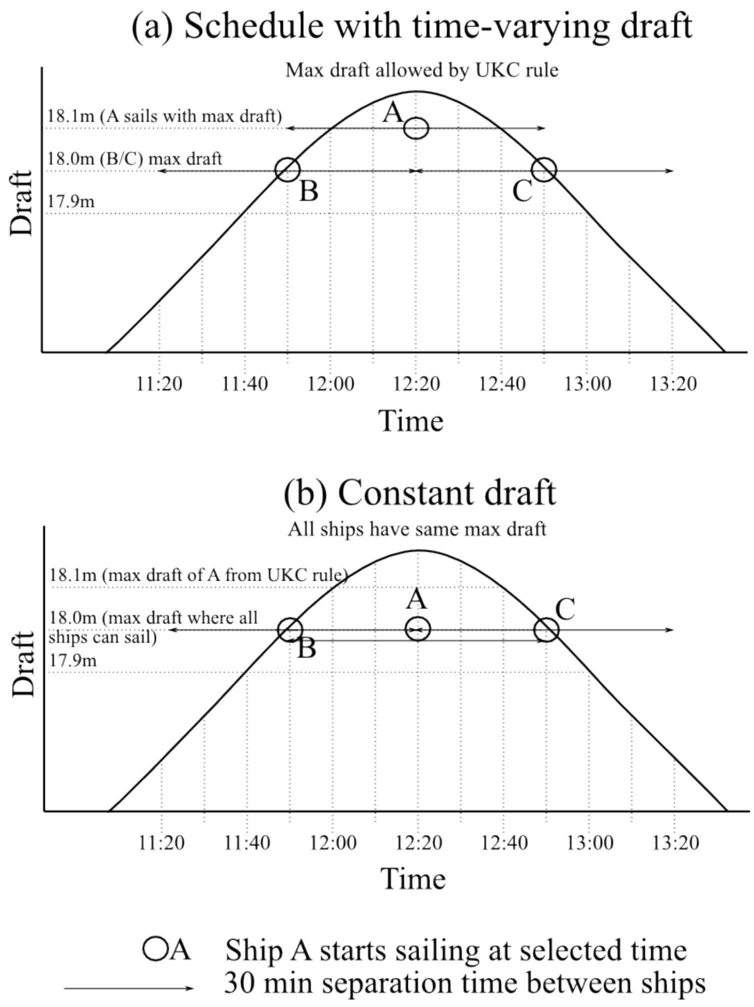


Figure 5.3: Schedule with time-varying draft vs. constant draft.

scheduling approach which uses fixed-time draft restrictions would have to set the draft restriction to a value that is low enough to allow ships to sail even at neap tides, thus reducing the draft of ships sailing at higher tides even further.

This shows that even though constant draft restrictions are good enough for scheduling small non-draft-restricted ships, or for scheduling at non-tidal ports, problems involving large draft-restricted ships sailing through tidally-affected ports have a high potential for improvement in schedule quality by incorporating time-varying draft restrictions.

As seen in Chapter 4, incorporating time-varying draft restrictions significantly increases the complexity of a ship scheduling problem; so much so that finding optimal schedules is a non-trivial problem even for ships sailing from a single port. Larger problems may be solvable efficiently by using a decomposition approach, for example, first assigning ships to tides with a fixed constant draft for each tide, then optimising the schedule for each tide individually to maximise drafts more accurately for that set of ships. Other non-optimal heuristic approaches may also be used, as in Chapter 6, which investigates the much larger ship routing and scheduling problem with time-varying draft restrictions and speed optimisation.

5.2.2 Comparison Against Manual Scheduling

The schedules produced by solving the Bulk Port Cargo Throughput Optimisation Problem can be compared against schedules produced by simple manual optimisation algorithms similar to those typically used by human schedulers. One algorithm that can be used for manual scheduling is to schedule the ship that has the highest weighting in the objective function first (ie. the ship with the highest tonnage per centimetre of draft, or the highest priority for more complex priority queue-based objective functions), and to schedule each ship at the earliest time it can sail with its highest possible draft. This manual scheduling algorithm will be referred to as ObjFunFirst.

Note: each ship to be scheduled has a maximum draft it can be loaded to, which may depend on the structural engineering of the ship, and at some ports may be constrained by the amount of cargo available for transport. The maximum draft for each ship may be below the maximum draft that would be allowed by the under-keel clearance rule at the peak of the tide, in which case the draft equation $D(v,t)$ for the ship will have a plateau at the peak of the tide.

5.2.2.1 ObjFunFirst vs Optimal Scheduling

In some situations, the ObjFunFirst algorithm may produce suboptimal schedules, where ships carry less cargo in total. Figure 5.4 shows one example problem where ObjFunFirst scheduling results in less cargo being carried by ship D.

Figure 5.4 shows a simple example schedule with four outgoing ships A to D, having maximum drafts of 18.1, 18.0, 17.9. and 17.8 metres respectively. The tonnage

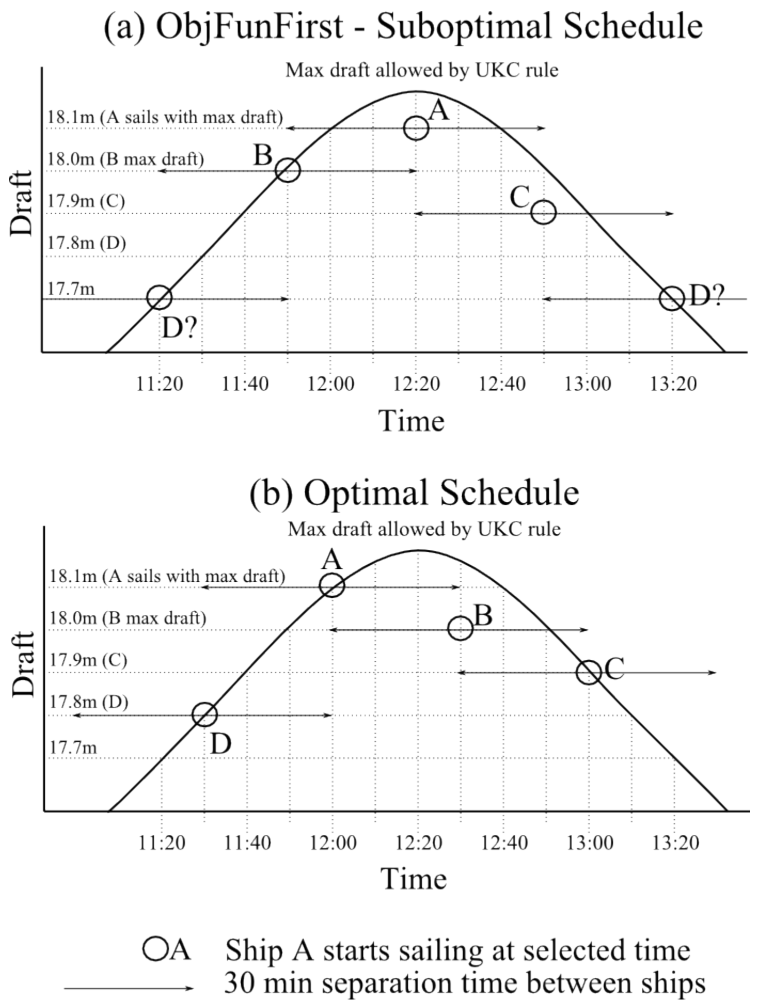


Figure 5.4: Manual schedule using ObjFunFirst vs optimal schedule.

per centimetre of draft is 130, 140, 125 and 120 respectively. (The average for iron ore bulk carriers at Port Hedland is 130 tonnes of cargo per centimetre of draft [Port Hedland Port Authority, 2011b]). The sailing windows at their respective maximum drafts are 12:00–12:40 for ship A, 11:50–12:50 for ship B, 11:40–13:00 for ship C, and 11:30–13:10 for ship D. For simplicity, we assume that the separation time between each pair of ships is 30 minutes, regardless of order.

The ObjFunFirst algorithm will schedule ship B first, as it has the highest tonnage per centimetre of draft. Ship B will be scheduled to sail at its earliest possible time when it can get its maximum draft, that is, 11:50. Ship A will be scheduled next, sailing at 12:20. Ship C will be scheduled next, sailing at 12:50. The 30-minute separation requirement now does not allow ship D to sail with its maximum draft, either at the start or end of the schedule. Ship D will have to sail 10 minutes before or 10 minutes after it can get its maximum draft. This sub-optimal schedule is shown in Figure 5.4(a).

Optimal scheduling, on the other hand, finds a solution that will allow all ships to sail with their maximum drafts, scheduling ship A to sail at 12:00, B at 12:30, C at 13:00, and D at 11:30, as shown in Figure 5.4(b).

5.2.2.2 MaxDraftFirst vs Optimal Scheduling

Another algorithm that could be used for manual scheduling is to schedule the ship with the highest allowable draft first (referred to in this chapter as MaxDraftFirst. In the example shown in Figure 5.4, MaxDraftFirst would have found the optimal schedule. However, there are other example problems where MaxDraftFirst scheduling would result in suboptimal schedules. One such example is shown in Figure 5.5.

Figure 5.5 shows an example schedule with three outgoing ships A to C, having maximum drafts of 18.1m for ship A, and 18.0m for ships B and C. The sailing windows at their respective maximum drafts are 12:00 - 12:40 for ship A, and 11:50 - 12:50 for ships B and C. As for the earlier example shown in Figure 5.4, we assume that the separation time between each pair of ships is 30 minutes, regardless of order.

The MaxDraftFirst algorithm will schedule ship A first, as it has the highest maximum draft. Ship A will be scheduled to sail at the earliest possible time when it can get its maximum draft, that is, 12:00. Ship B will then be scheduled to sail at 12:30, resulting in Ship C having to sail at 13:00, 10 minutes outside the time range when it can get its maximum draft. This sub-optimal schedule is shown in Figure 5.5(b).

Optimal scheduling, on the other hand, finds a solution that has Ship A sailing in the middle of its maximum draft time range, thus allowing both ships B and C to sail with their maximum drafts, as shown in Figure 5.5(b).

These examples are obviously much simpler than a real-world scenario, which would include slightly different draft functions for each ship, tug availability constraints, and asymmetric separation times between ships. However, the drafts, amounts

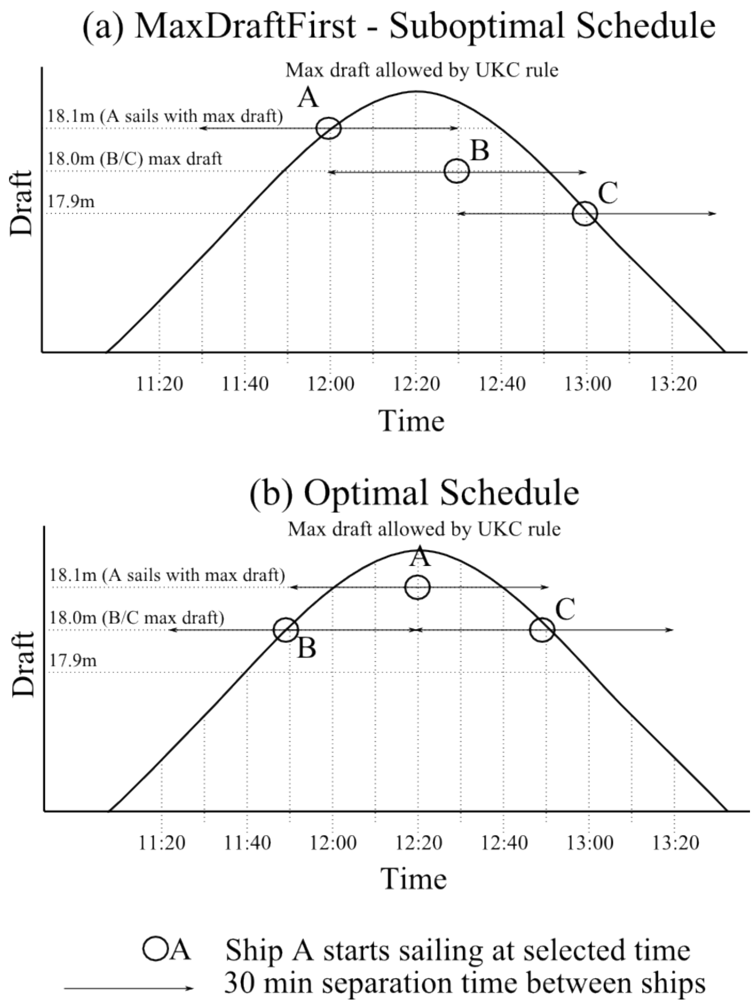


Figure 5.5: Manual schedule using MaxDraftFirst vs optimal schedule.

of cargo, and sailing windows for the maximum drafts used in these examples are realistic, though simplified. The more complex constraints found in practice at most ports will make the scheduling problem more complex, and less likely to be solved optimally by simple rules such as MaxDraftFirst or ObjFunFirst which are typically used in manual scheduling. For the larger, more realistic example problems used for experimental results in Chapter 4, such as OneWay_Narrow problems without tugs, the optimal schedules allowed an average of 120cm more draft for the set of ships compared to scheduling with constant draft constraints, and 15.8cm more draft per set of ships compared with manually scheduling the biggest ships first.

As shown by the examples in this section, even a small difference from the optimal schedule has a high cost, which makes this problem worth solving optimally even for a single port.

5.3 Comparison with Real Schedules

This section compares the schedules produced by solving the models introduced in Chapters 3 and 4 against schedules produced by human schedulers at a real port. Actual ship sailing schedules from Port Hedland were compared against schedules produced by solving the CP model to optimality for the same set of ships. Schedule comparisons were done for all high tides that had three or more ships sailing on a single tide over February and March 2012 – a set of 32 schedules in total. Schedules with less than three ships sailing on a tide were ignored, assuming that a human scheduler would have been able to find an optimal schedule in these cases, thus there would have been no room for improvement.

All ships have a *design draft* which limits the maximum amount of cargo that can be loaded onto a ship based on its structural specifications. Most ships to be scheduled were large Capesize class vessels, and some were given drafts of up to 18.42m by human schedulers. To avoid scheduling ships with unrealistically high drafts in our schedules, ship drafts were capped at 18.5m; however, the design draft for every individual vessel was not investigated in depth, so some ships may have been limited in practice by lower design drafts.

We also assumed that any ships that were scheduled by human schedulers with drafts at least 1m lower than the largest ship sailing on the same tide were smaller ships that could not load more cargo, (ie. that human schedulers had a good reason for scheduling ships to sail with significantly less cargo) so we did not allow any draft increase for these ships in our model. This included all Panamax vessels seen in the set of schedules. For large ships, we allowed the draft to increase if there was enough water available at the ship's scheduled sailing time, up to the 18.5m maximum.

We did not allow any more ships to be scheduled than were included in the original schedules, since we had no way to tell if the assignment of ships to tides was done because ships sailing on future tides were not ready to sail earlier, or because an overly crowded tide resulted in the human scheduler deciding to move a ship to the following tide. This may have resulted in our schedules missing some opportunities to allow ships to sail on an earlier tide. However, our model allowed the order as well as the scheduled sailing time of ships to vary, which allowed many opportunities for improving the schedule.

We had no way of knowing the time at which the ship finished loading, so there may have been some instances when a human scheduler scheduled a given ship to sail at the end of the tide in a suboptimal time slot due to the ship running late loading. In such situations, the improvement found by our approach would not have been feasible in practice; however, in most cases, ships aim to be ready to sail some time before the high tide, to ensure that minor loading delays do not disrupt the schedule, so the assumption that ships could have been scheduled to sail slightly earlier on the same tide is generally valid.

5.3.1 Experimental Results

For the 2-month period of February and March 2012, optimal schedules found by solving the CP model would have allowed ships to be scheduled with a total of 26.20m more draft. This is a modest relative improvement of 1.36%, however with the huge volumes of cargo transported, even a modest relative improvement translates to significant absolute savings.

Each 1cm of extra draft translates to around 130 tonnes of extra iron ore on a typical Capesize bulk carrier [Port Hedland Port Authority, 2011b]. Therefore the extra draft allowed by our schedules translates to 2.04Mt extra iron ore shipped per year. Given that a typical iron ore bulk carrier can transport 170,000 tonnes of iron ore [Port Hedland Port Authority, 2011a], this saves approximately 11 voyages per year.

Translating the number of voyages to an estimate of cost savings or profit increase for the shipper is complex, as it depends on a number of variables including:

- Freight rates, which are highly volatile.
- Whether the ships are on a time or voyage charter.
- Whether the shipping contract specifies that the importer or the supplier pays for the cost of shipping.

However, it is possible to make some estimates of cost savings or profit increases for a number of scenarios, using current freight rates and iron ore prices as a guide.

First, consider a cost per tonne perspective, assuming that the same amount of iron ore is shipped over a year, but that fewer ships are needed. The shipping cost of iron ore from Western Australia to China was approximately \$10 per tonne for Capesize vessels as of July 2012 [Curtis, 2012]. Therefore, saving the shipping costs on 2.04Mt of iron ore translates to \$20.4 million in cost savings per year. These cost savings occur even if there are bottlenecks in production which constrain the amount of iron ore available for shipping.

On the other hand, if the number of ships stays the same, but the amount of cargo transported on each ship increases (eg. if the port is the bottleneck and there are no constraints in how much iron ore can be produced) using optimal schedules allows 2.04Mt more iron ore to be transported on the same set of ships. The average price of iron ore for the first six months of 2013 was $135 per tonne [Index Mundi, 2013], and the profit for the exporter of shipping iron ore from Western Australia to China was estimated to be about $75/tonne in 2012 [Curtis, 2012]. The extra 2.04Mt of iron ore thus translates to a $275 million annual increase in revenue or a $153 million annual increase in profit.

In practice, these estimates may vary significantly based on freight rates, the price of iron ore, and other factors such as the classes of ships chartered, and the details of the shipping contracts. However, these figures provide an order-of-magnitude estimate of the benefit that could be obtained by using optimal ship schedules with time-varying drafts. Even allowing for large fluctuations in the ship charter rate, the price of iron ore and other factors, these estimates clearly show that using DUKC® Optimiser to produce optimal schedules for ship sailing times at draft-restricted bulk export ports has an enormous potential benefit to industry.

5.4 Limitations and Future Work

One possible limitation of the approaches presented in this chapter is the robustness of the resulting schedules, as robustness is not considered by any the scheduling algorithms presented in this thesis. The allowable sailing draft for each vessel at each time is calculated using conservative predictions of all factors that affect safe drafts, and the separation times between ships are also conservative enough to allow for some inaccuracy in ship sailing times. The resulting schedules are robust enough to be used in practice at Australia's largest export port. However, an in-depth investigation of the tradeoff between robustness and cost optimisation has not been conducted and would be a worthwhile avenue of future research.

Note that the risk of using any of the algorithms presented in this thesis for scheduling ship sailing times is not grounding but delay. Ports using these approaches still need to do final safety checks before sailing, and if weather conditions have changed beyond the conservative forecasts used to calculate draft restrictions during planning, ships may need to be delayed to the following high tide rather than sailing on schedule.

5.5 Summary

This chapter presented the DUKC® Optimiser commercial system for optimising ship schedules at a bulk export port that was implemented based on the research presented in Chapters 3 and 4 of this thesis, and compared the optimal schedules produced by solving the CP model from Chapter 3 against real schedules created by human schedulers at Port Hedland. DUKC® Optimiser was estimated to have the potential to increase annual revenue at the port by $275 million, or to reduce costs by $20.4 million in the event of bottlenecks in the supply chain constraining iron ore production.

A comparison of optimal schedules against simple heuristics that may be easily applied by human schedulers was also presented, to demonstrate how such heuristics may fail to find the optimal schedule even for a simple problem with three ships. Optimal schedules were also compared against results that would be produced by a system that used a fixed draft limit for the port.

The next chapter investigates the impact of considering time-varying draft restrictions in a larger maritime transportation problem – the cargo routing problem with time-varying ship speeds. The problem takes into account profits resulting from delivery of cargoes, routing costs, as well as fuel usage as a function of ship speed, and the impact of time-varying draft restrictions on available ship arrival times at ports.

Chapter 6

Ship Routing and Scheduling with Variable Speeds and Drafts

6.1 Introduction

This chapter extends time-varying draft restrictions to a larger problem in maritime transportation – the cargo routing problem which involves finding a minimum cost solution for routing a set of cargoes between origin and destination ports using a fleet of ships. Another important consideration in cargo routing in recent years has been optimisation of ship speeds to reduce fuel usage. The approaches investigated in this chapter build on the recent work of Fagerholt, Laporte, and Norstad [2010] and Norstad, Fagerholt, and Laporte [2011] on ship speed optimisation on a fixed route and in a cargo routing problem respectively. These approaches are extended to also consider time-varying restrictions on ship draft at each port, including the impact this has on the allowable ship sailing times and therefore on optimal speeds.

6.1.1 Calculating Shipping Costs

The costs that need to be considered for shipping some amount of cargo on a fixed shipping route include the fuel costs, which increase with faster ship speed, and other costs such as ship charter rates which are linear over time and thus increase with slower ship speeds and longer travel times.

Fuel consumption (and therefore fuel cost) in a given period of time is proportional to a cubic function of the ship speed above a certain speed, and constant below this speed [Ronen, 1982; Ryder and Chappell, 1980]. This relationship is confirmed by empirical observations [Vernimmen, Dullaert, and Engelen, 2007]. The fuel cost for travelling a given distance is therefore proportional to the square of the ship speed. Other costs that are constant over time are inversely proportional to the ship's speed for a given distance.

Psaraftis and Kontovas [2009] modelled the fuel consumption in tonnes per day for a fully loaded Panamax bulk carrier as $F = k_1 \cdot V^3$ where $k_1 = 1.4946 \cdot 10^{-7}$ and V is in kilometres per day. This is supported by empirical data [Clarkson Research, 2012].

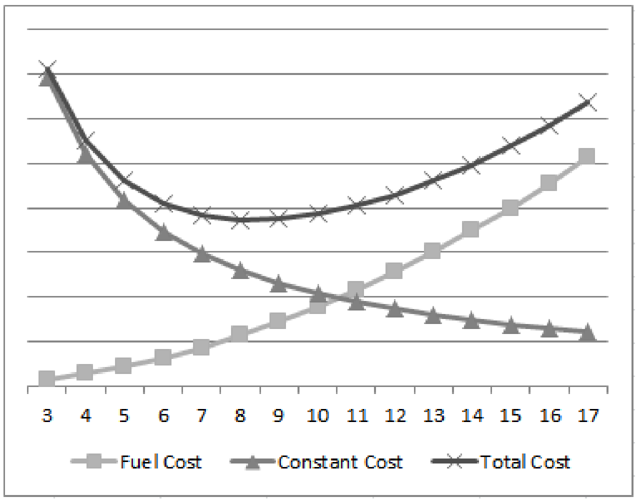


Figure 6.1: Fuel cost, constant cost, and total cost for travelling a fixed distance with a range of ship speeds (in knots).

Converting V to knots (nautical miles per hour), we find that the fuel cost for a Panamax bulk carrier travelling distance D (in nautical miles) with velocity V (in knots) can be modelled by Equation (6.1), where $k_2 = 5.4671 \cdot 10^{-4}$, and p is the price of bunker fuel, around US$650 per tonne as of 2012 [Bunkerworld, 2012].

$$\textsc{fuel\_cost}(V, D) = p \cdot k_2 \cdot D \cdot V^2 \tag{6.1}$$

A ship scheduled to travel below its minimum speed V_{min} will actually travel at V_{min} and wait when it arrives at its destination. The $\textsc{fuel\_cost}(V, D)$ is replaced in Equation (6.2) by $\textsc{fuel\_cost}(V_{min}, D)$, whereas the constant cost is calculated based on $V < V_{min}$.

Costs that are constant over time also need to be taken into account. In this chapter, constant costs are approximated by the average time charter rate for a Panamax bulk carrier over the financial year 2011-12 – around US $10,000 per day [DryShips Inc., 2012; CoalSpot.com, 2012]. The total shipping cost, given by Equation (6.2), is the sum of FUEL_COST and the $10,000 per day = $416.67 hourly constant cost h. We do not consider inventory costs in this chapter, as these are generally much lower for bulk cargo than for container cargo, however, these could be incorporated similarly.

$$\begin{aligned} \textsc{total\_cost}(V, V_{min}, D) &= \textsc{fuel\_cost}(V, D) + \textsc{hourly\_cost}(V, D) \\ &= p \cdot k_2 \cdot D \cdot \max(V, V_{min})^2 + h \cdot \frac{D}{V} \end{aligned} \tag{6.2}$$

Figure 6.1 shows the sum of FUEL_COST and the constant cost for travelling a fixed distance with a range of ship speeds. This demonstrates that for each ship, there is an optimal speed below which the costs increase due to a longer voyage time, and above which the costs increase due to higher fuel consumption.

6.1.2 Effect of Draft on Shipping Costs

The models presented in this chapter consider time-varying draft constraints at waypoints, and the amount of cargo carried on each ship is a decision variable. The models therefore also need to consider the effect that sailing with increased draft has on shipping costs.

Sailing with an increased draft reduces annual shipping costs for a route by allowing more cargo to be carried on each ship, thus allowing fewer ships to be chartered over the long term. Increased tonnage transported on a ship can be translated to an estimate of costs saved by calculating the average cost of shipping one tonne of cargo over the given route, as was done in Chapter 5. However, increasing ship loading also increases fuel usage, and the increased fuel costs need to be subtracted from the cost reduction of carrying more cargo on the same ship. Empirical results show that ballast (unloaded) voyages use approximately 20% less fuel at the same speed compared to fully loaded voyages [Endresen et al., 2004; Wijnolst and Wergeland, 1997].

For a ship that has a 10m difference in draft between fully loaded and unloaded states, sailing with 10cm deeper draft allows the ship to carry 1% extra cargo, thus resulting in a cost savings of 1% of the normal cost of a voyage. However, assuming that the 10m difference in draft between the fully loaded and unloaded states translates to a 20% difference in fuel usage, and that fuel usage increases linearly with the draft increase, the 1% increase in draft also results in a 0.2% increase in fuel consumption for the voyage, thus increasing the fuel cost by 0.2%.

The fuel cost can comprise up to 75% of the total shipping cost for a voyage [Ronen, 2011], thus the increase in fuel usage caused by sailing with a higher draft may reduce the cost savings of carrying more cargo by as much as 15%. Therefore the increased fuel usage caused by a higher ship loading, while it is much lower than the cost savings produced by sailing with more cargo, is still significant and needs to be taken into account in the cost optimisation model with variable speed and loading presented in this chapter.

An example with realistic figures is: the cost of shipping iron ore from Australia to China is approximately around US$10 per tonne as of July 2012, or $1.5 million for a Capesize bulk carrier carrying 150,000 tonnes of ore [Curtis, 2012]. If 10% more cargo (15,000 tonnes) could be loaded on each ship, this would result in savings of 10% of a voyage per ship ($150,000), but increase the fuel cost by 2% for each ship. If we assume that the fuel cost comprises 75% of the total cost of the voyage, the increased fuel usage increases the cost of this voyage by $30,000, resulting in a total savings of $120,000.

The models presented in this chapter do not consider port charges and canal fees since, as observed by Ronen [2011], these are constant for a container fleet operating on a weekly schedule. These will also be constant for a ship travelling on a fixed route. For a ship routing and scheduling problem, the port charges may vary depending on which ship in the fleet is used to transport each cargo; however, we assume that the fleet is composed of ships of a similar size (eg. all Panamax bulk carriers) so port charges would be very similar for all ships. For a heterogeneous fleet composed of many different vessel sizes, port charges may vary depending on the ship used to transport each cargo, and may need to be considered in more detail.

In this chapter, the cost savings for an increased draft is approximated by Equation (6.3), where c is the cost of shipping one tonne of cargo for this voyage, t is the increased tonnage per centimetre of draft, d_i is the draft for which the cost savings is being calculated, d_0 is the minimum draft being compared against, D_j is the distance for leg j of the voyage, and v_j is the speed for leg j, V_j is the speed for leg j, V_{min} is the minimum speed at which the ship can actually travel, d_{max} is the ship's maximum draft, and d_{ul} is the draft of an unloaded ship. The total cost savings, described by Equation (6.3), is the cost of shipping one tonne of cargo c multiplied by the number of tonnes of extra cargo shipped by increasing draft – $t \cdot (d_i - d_0)$ – minus the cost of the extra fuel used due to sailing with more cargo.

$$\begin{aligned} \textsc{cost\_savings}(d_i) = c \cdot t \cdot (d_i - d_0) - \\ 0.2 \cdot \frac{d_i - d_0}{d_{max} - d_{ul}} \cdot \sum_{j \in [0, \text{nLegs}-1]} \textsc{fuel\_cost}(\max(V_j, V_{min}), D_j) \end{aligned} \tag{6.3}$$

Finally, the objective is to minimise total cost = fuel cost + constant cost – cost savings, where fuel cost + constant cost is given by Equation (6.2) and cost savings is given by Equation (6.3).

6.1.3 Existing Approaches to Cargo Routing with Variable Ship Speed

The two most closely related papers to the problems investigated in this chapter are Fagerholt, Laporte, and Norstad [2010] and Norstad, Fagerholt, and Laporte [2011]. Fagerholt, Laporte, and Norstad [2010] optimised the ship speed on each leg of a fixed liner shipping route, with time windows for arrival at each waypoint. The problem assumes that there is no waiting time at waypoints. Three approaches were compared:

1. Non-linear programming model with speed as the primary decision variable.
2. Non-linear programming model with sailing time as the primary decision variable.
3. Discretising the arrival times within the time window at each waypoint and solving the speed optimisation problem as a shortest path problem on a directed acyclic graph.

The third approach of discretising the arrival times and solving using a shortest path algorithm was found to be the most efficient. The arrival time was chosen as the variable to discretise because this results in a graph where the number of nodes is linear with the number of discretisations and the number of waypoints in the route. Discretising speeds or sailing times between each pair of waypoints results in a graph that grows exponentially with the number of waypoints or the number of discretisations.

This problem of speed optimisation of a fixed route with time windows is very similar to the problem presented in this chapter, but this research also considers time-varying draft constraints which result in variable time windows for different ship drafts, as well as resource conflicts caused by ships arriving at waypoints at the same time.

Norstad, Fagerholt, and Laporte [2011] investigated a ship routing and scheduling problem with speed optimisation on each leg of each route. The approach presented by Fagerholt, Laporte, and Norstad [2010] was used to determine the profit contribution of each candidate route by optimising speeds for a single ship on each candidate route. The routing and scheduling problem was solved by generating a number of initial solutions and using a multi-start local search heuristic to improve the best solutions. This method is discussed in more detail in Section 6.5, which builds on the work of Norstad, Fagerholt, and Laporte [2011] to incorporate time-varying draft restrictions into a ship routing and scheduling problem with variable ship speeds.

6.2 Speed Optimisation for a Fixed Route

6.2.1 Problem Description

As a first step towards solving the cargo routing problem with ship speed optimisation and time-varying draft, this section extends the work of Fagerholt, Laporte, and Norstad [2010] on the speed optimisation problem for a single ship on a fixed route to include time-varying draft constraints at all waypoints along the route. This problem is also extended to be able to consider multiple ships travelling along the same route, with resource conflicts due to several ships needing to travel through a draft-constrained waypoints on a single high tide.

This problem corresponds to real-world situations such as ships travelling through the draft-constrained Torres Strait, north of Australia. One of the problem instances considered in this chapter is similar to the Rio Tinto shipping route transporting bauxite from the mining port of Weipa, Australia, through the Torres Strait, to the aluminium refinery at the port of Gladstone (see map in Figure 6.2). However, other shipping routes with draft-constrained ports and waterways may also benefit from speed optimisation with time-varying draft constraints, eg. shipping routes through the Malacca Strait, or shipping routes through a series of draft-constrained ports such as those considered by Rakke et al. [2012].

For the fixed-route problem, we assume that the draft remains constant for the entire route, ie. that the ship does not load or unload cargo at intermediate waypoints other than the start and end of the route. This models a route between two ports, with draft-restricted waterways such as the Torres Strait or Panama Canal as intermediate waypoints. This approach can be extended to optimise speeds and drafts for a route with multiple ports by splitting it into subroutes between each pair of ports, and

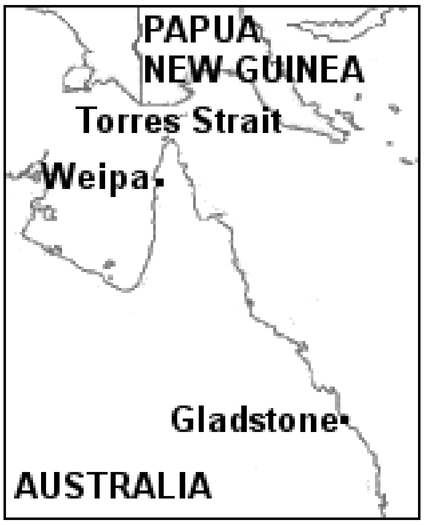


Figure 6.2: Map of Weipa – Torres – Gladstone shipping route.

optimising the speeds and drafts for each subroute independently.

The key features of this problem are:

1. Let S be a set of ships, each with a range of possible drafts and speeds, and a cargo capacity per centimetre of draft within the allowable range.
2. All ships sail along a fixed route between two ports with several waypoints.
3. Each port and waypoint has time-dependent draft constraints that vary with tide, waves, current, and other environmental conditions.
4. Draft constraints can be converted to multiple time windows for each possible ship draft. These time windows may vary with ship draft (if they are entirely dependent on draft constraints) or may be static, for example if the earliest arrival time at the destination is determined by needing to have enough storage space to unload the ship's cargo.
5. Loading and unloading time at ports are ignored, similarly to Fagerholt, Laporte, and Norstad [2010]. However, ships may wait at waypoints if required, to ensure that minimum speeds do not make time windows infeasible.
6. At each port and waypoint, there may be a minimum separation time between ships, to avoid ships sailing too close together in a narrow draft-restricted waterway.
7. The objective is to minimise the sum of the total costs and cost savings for the set of ships, including fuel costs, constant-time costs, cost savings for sailing with increased draft, and increased fuel costs caused by sailing with a higher loading.

This section and Section 6.3 investigate the Speed Optimisation Problem with Time-Varying Draft (SOPTVD) for a single ship, to find the optimal speeds and optimal draft for a fixed route with time-varying draft constraints at waypoints. Section 6.4 then extends this approach to solve the Multi-Ship Speed Optimisation Problem with Time-Varying Draft (MS-SOPTVD) which includes resource conflicts between ships scheduled close together at draft-restricted waypoints, and Section 6.5 investigates the cargo routing and scheduling problem with speed optimisation and time-varying draft restrictions with the single-ship problem presented in this section used as a subproblem to solve the larger routing problem.

6.2.2 Formal Problem Definition

We initially solve the Speed Optimisation Problem with Time-Varying Draft for a ship with a fixed draft, but with time-varying draft restrictions at each waypoint. Section 6.3 extends this problem to also find the optimal draft for the ship from a range of drafts.

The following problem definition for the Speed Optimisation Problem with Time-Varying Draft (SOPTVD) for a ship with a fixed draft is based on the problem definition for the Speed Optimisation Problem presented by Fagerholt, Laporte, and Norstad [2010] and Norstad, Fagerholt, and Laporte [2011].

Given a route, i.e. a sequence of waypoints that a given ship must sail past, with corresponding allowable time windows for the ship's fixed draft, the objective of the SOPTVD is to determine the speed for each leg of the route so that the total cost for the route is minimised, where the cost is specified by Equation (6.2) as the sum of the fuel cost and constant cost per time.

Let the waypoints along the route be indexed by $i = 0, ..., N$, with $V_{i,i+1}$ and $D_{i,i+1}$ being the speed and distance respectively from waypoint i to $i+1$. Let T_i be the arrival time at waypoint i. Let V_{max} be the maximum possible speed for the ship, and V_{min} be the minimum speed the ship can travel at in practice, used to calculate the minimum fuel cost for the voyage if the ship travels below this speed and waits.

Let W_i be the set of draft-restricted time windows w at waypoint i for the given ship with the given draft, with each time window specified by the time interval $\underline{w}, \overline{w}$. Each waypoint also has earliest and latest arrival times E_i and L_i respectively to represent any restrictions on cargo pickup and delivery times. For a waypoint where there is no limit on arrival times except for the draft restriction, E_i and L_i can be set to the start or end of the time range.

Let $\textsc{total\_cost}(V_{i,i+1}, V_{min}, D_{i,i+1})$ specify the total cost for travelling the leg from i to $i+1$ at speed $V_{i,i+1}$, including both fuel and other costs, as given by Equation (6.2) in Section 6.1.1. V_{min} is used in this equation to place a lower bound on the fuel cost for each leg as the fuel used by the ship travelling at minimum speed.

The SOPTVD can now be defined as follows:

$$\text{minimise} \sum_{i=0}^{N-1} \textsc{total\_cost}(V_{i,i+1}, V_{min}, D_i) \tag{6.4}$$

subject to

$$V_{i,i+1} = D_{i,i+1}/(T_{i+1} - T_i), \forall i \in [0, N-1] \tag{6.5}$$
$$0 < V_{i,i+1} < V_{max}, \forall i \in [0, N-1] \tag{6.6}$$
$$E_i \leq T_i \leq L_i, \forall i \in [0, N] \tag{6.7}$$
$$\exists w \in W_i \text{ s.t. } \underline{w} \leq T_i \leq \overline{w}, \forall i \in [0, N] \tag{6.8}$$

The objective function (6.4) minimises the sum of the fuel cost and waiting cost for the route. Equation (6.5) specifies the speed as a function of time and distance for each leg. Equation (6.6) ensures that no ship travels faster than its maximum speed, and that each ship keeps moving at a speed above zero. Note that Norstad, Fagerholt, and Laporte [2011] also specified a minimum speed for each ship, and allowed waiting by extending the time between waypoints. On the other hand, we incorporate waiting by adjusting the speed for each leg, and using a minimum speed to place a lower bound on the fuel cost. Finally, Equation (6.7) specifies the earliest and latest arrival times, and equation (6.8) specifies that the arrival time at each waypoint must lie within one of the draft-restricted time windows for that waypoint, i.e. that the tide must be high enough for the ship to sail.

The final draft-restricted time window constraint and the travel time being taken into account in the cost function are the only substantial differences from the Speed Optimisation Problem defined by Norstad, Fagerholt, and Laporte [2011]. Other changes are cosmetic only.

6.2.3 Shortest Path Approach

The Speed Optimisation Problem (SOP) was solved by Fagerholt, Laporte, and Norstad [2010] for a single ship with a fixed route, no draft constraints and with fixed windows for delivery times at waypoints. Their approach worked by discretising arrival times and converting the SOP into a shortest-path problem where nodes are the discretised arrival times at each waypoint, and the costs of each arc corresponds to the cost of travelling between those waypoints at those times.

This approach was able to find close-to-optimal solutions quickly, and the error could be made arbitrarily small by choosing a fine enough discretisation. Arrival time discretisation was used instead of speed discretisation since the latter led to the search tree growing exponentially with the number of waypoints, whereas arrival time discretisation resulted in the search tree growing linearly with the number of waypoints.

To extend the shortest path approach to multiple time windows, discretisation points were selected to be evenly spaced inside the entire time range for a given waypoint, from the first allowable sailing time to the last allowable sailing time, including infeasible regions between time windows. Then any discretisation points that were initially outside a time window were adjusted to the nearest time window, and the points inside each time window were evenly spaced. This results in the start and end of each interval always being included in the discretised values as long

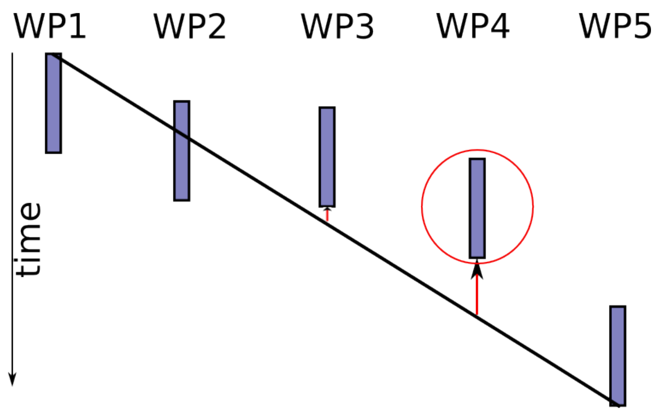


Figure 6.3: RSA Step 1: Calculate optimal speed for route, ignoring time windows except at first and last waypoints. Then find waypoint with largest time window violation and adjust its arrival time to the nearest feasible time.

as the number of discretisations is high enough. This discretisation approach was chosen because the optimal path is more likely to go through the start or end of a time window than a random point in the middle.

6.2.4 Recursive Smoothing Algorithm

The Speed Optimisation Problem for a single ship with a fixed route and no draft constraints was also solved using a recursive smoothing algorithm (RSA), which was able to find optimal solutions, and was also found to be faster than the shortest path approach [Norstad, Fagerholt, and Laporte, 2011].

In this chapter, we implement both the shortest path method and the recursive smoothing algorithm for solving the single ship fixed-route problem, to investigate which approach works best when the single ship solutions are used as a subproblem of the Multi-Ship Speed Optimisation Problem with Time-Varying Draft (MS-SOPTVD).

The recursive smoothing algorithm presented by Norstad, Fagerholt, and Laporte [2011] is based on the idea that, if there were no time windows constraining the sailing times for each waypoint, the optimal speed for each leg would be constant. For a cost function that is convex and non-decreasing over the allowable range of speeds, the optimal speed is also the minimum speed. However, this optimal speed may result in an arrival time at one of the waypoints falling outside the time windows at that waypoint. Both Fagerholt, Laporte, and Norstad [2010] and Norstad, Fagerholt, and Laporte [2011] only consider problems with a single time window at each waypoint.

The algorithm starts by calculating an optimal speed in the absence of time window constraints, except at the first and last waypoints. Because the cost function is monotonic, the optimal speed is always the slowest possible speed, departing from the first waypoint at the earliest possible sailing time, and arriving at the final waypoint at the latest possible arrival time, as shown in Figure 6.3. If no time windows

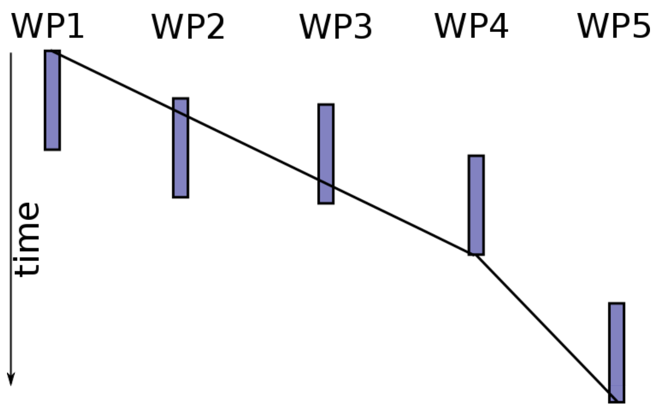


Figure 6.4: RSA Step 2: Split route at waypoint with largest time window violation and recalculate subroutes recursively.

are violated, this is the optimal solution. If any time window is violated, as is the case for Waypoints 3 and 4 in Figure 6.3, the time for that waypoint is adjusted to the start or end of the time window, whichever is nearest, as shown in Figure 6.3. The route is then split into two subroutes at that waypoint, as shown in Figure 6.4, and the procedure repeated recursively until all time window constraints are met. In this example, after Figure 6.4 there are no more time window violations, so no further route splitting is required.

Algorithm 1 describes the recursive smoothing algorithm introduced by Norstad, Fagerholt, and Laporte [2011]. Let t_i be the arrival time at node i and $v_{i,i+1}$, $d_{i,i+1}$ the speed and distance respectively from i to $i+1$. Let s and e be the start and end nodes for the partial route, respectively. The time window for node i is $[\underline{t}_i, \overline{t}_i]$. The algorithm checks each waypoint and splits the route on node p with the largest time window violation δ, regardless of whether the arrival time is too early or too late.

Before the algorithm is called, the start and end time of the route must be set to $t_s = t_0$, which is the starting position for the ship, and $t_e = \overline{t}_e$ respectively, since the original recursive smoothing algorithm presented by Norstad only considers monotonic cost functions where the lowest speed is the optimal speed, so the optimal route must start at the earliest possible departure time from the first waypoint, and end at the latest possible arrival time at the last waypoint.

6.2.5 Extending Norstad's RSA for Multiple Time Windows

The introduction of time-varying draft constraints at waypoints creates multiple time windows at each waypoint around each high tide, not only a single time window defined by an earliest and latest arrival time. The cost function for our problem is also modelled as non-monotonic, since ships may wait for the start of a time window, thus removing any constraint on minimum speed. We therefore need to extend the recursive smoothing algorithm presented by Norstad, Fagerholt, and Laporte [2011] to be able to deal with multiple time windows at each waypoint and a non-monotonic cost function.

Algorithm 1 RECURSIVE_SMOOTHING_ALGORITHM(s, e), from Norstad, Fagerholt, and Laporte [2011]

```
$\delta \leftarrow 0$
$p \leftarrow 0$
$v^* \leftarrow \sum_{i=s}^{e-1} d_{i,i+1}/(t_e - t_s)$
**for** $i \leftarrow s$ to $e$ **do**
    $i \leftarrow i + 1$
    $v_{i-1,i} \leftarrow v^*$
    $t_i \leftarrow t_{i-1} + d_{i-1,i}/v_{i-1,i}$
    **if** $t_i - \bar{t}_i > |\delta|$ **then**
        $\delta \leftarrow t_i - \bar{t}_i$
        $p \leftarrow i$
    **end if**
    **if** $\underline{t}_i - t_i > |\delta|$ **then**
        $\delta \leftarrow t_i - \underline{t}_i$
        $p \leftarrow i$
    **end if**
**end for**
**if** $\delta > 0$ **then**
    $t_p \leftarrow \bar{t}_p$
    RECURSIVE_SMOOTHING_ALGORITHM$(s, p)$
    RECURSIVE_SMOOTHING_ALGORITHM$(p, e)$
**end if**
**if** $\delta < 0$ **then**
    $t_p \leftarrow \underline{t}_p$
    RECURSIVE_SMOOTHING_ALGORITHM$(s, p)$
    RECURSIVE_SMOOTHING_ALGORITHM$(p, e)$
**end if**
```

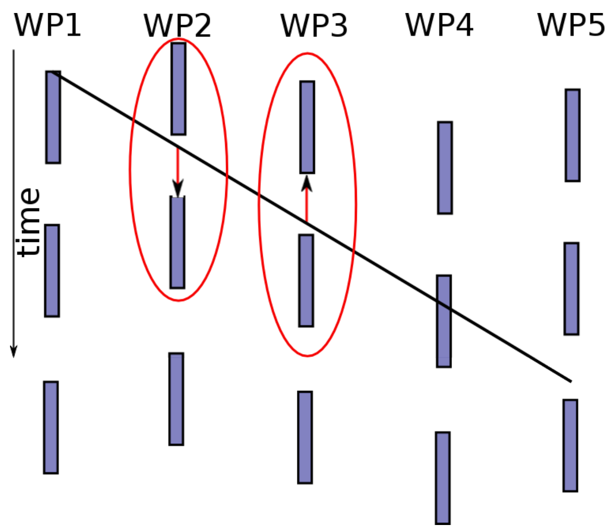


Figure 6.5: MW_RSA Step 1: Try splitting the route on both the waypoint with the largest distance to the previous time window, and the waypoint with the largest distance to the next time window. Try adjusting the time at the selected waypoint to both the nearest feasible times before and after the original optimal time.

We call our extension to Norstad's Recursive Smoothing Algorithm the Multi-Window Recursive Smoothing Algorithm (MW_RSA).

The cost function for the Speed Optimisation Problem for a ship travelling along a fixed route with a fixed draft depends only on speed. However, since there is no restriction on minimum sailing speed for each ship, and the problem involves both fuel costs and costs that are constant over time, the total cost function shown in Figure 6.1 is not monotonic.

The total cost function for travelling a given distance is a quadratic function of speed, given by Equation (6.2). The optimal speed is given by differentiating the cost function and solving for the optimal speed V_{opt}, as shown in Equation (6.9). However, if this optimal speed results in an arrival time at a waypoint that is outside the time windows, the optimal arrival time may be either at the end of the previous time window, or the start of the next time window. In Equation (6.9), V is the speed of the ship, D is the distance travelled, p is the price of one tonne of fuel, k_2 is a constant used to calculate fuel usage for the ship, and h is the hourly cost of operating the ship.

$$\begin{aligned}
& \textsc{total\_cost}(V,D) = p \cdot k_2 \cdot D \cdot V^2 + h \cdot \tfrac{D}{V} \qquad (6.9)\\
\Longrightarrow \quad & \tfrac{d}{dV}(p \cdot k_2 \cdot D \cdot V_{opt}^2 + h \cdot \tfrac{D}{V_{opt}}) = 0\\
\Longrightarrow \quad & 2p \cdot k_2 \cdot D \cdot V_{opt} - h \cdot (\tfrac{D}{V_{opt}^2}) = 0\\
\Longrightarrow \quad & V_{opt} = \sqrt[3]{(\tfrac{h}{2p \cdot k_2})}
\end{aligned}$$

MW_RSA first calculates optimal speeds for the route in the absence of time

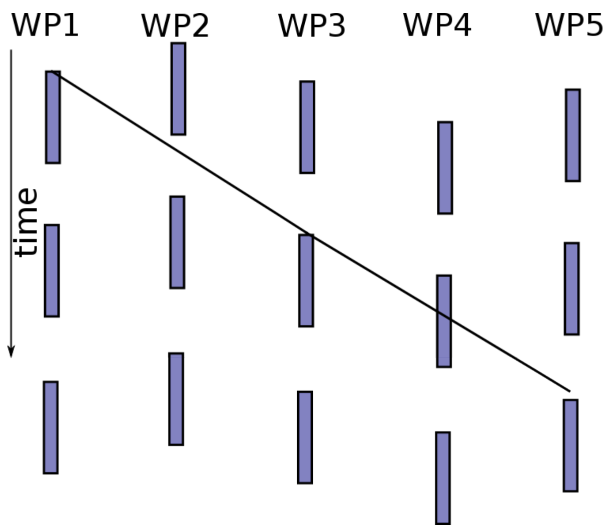


Figure 6.6: MW_RSA Step 2: After adjusting arrival time at selected waypoint, split the route at that waypoint, and recursively optimise both subroutes until all arrival times at waypoints are within valid time windows. In this case, both subroutes WP1 – WP3 and WP3 – WP5 will need to be adjusted further.

windows, based on Equation (6.9). If there are no time window violations, this solution is optimal.

If time window constraints are breached at any waypoint, we need to choose a waypoint at which to split the route. For the single-window RSA, we could always pick the waypoint with the largest time window violation, regardless if the original optimal time was before or after the available sailing window. For the multi-window case, we need to consider splitting the route at two possible waypoints: the waypoint with the largest distance to the end of the previous time window, and the waypoint with the largest distance to the start of the next time window, as shown in Figure 6.5. If we only consider splitting the route at one waypoint, we risk cutting off valid time windows for adjacent waypoints, and missing the optimal solution, as shown in Figures 6.7 and 6.8.

After selecting a waypoint to split the route at, we need to consider adjusting the arrival time at that waypoint to both the nearest valid sailing times before and after, since again we do not know whether sailing slightly faster or slightly slower will produce the better solution. After adjusting the arrival time at the selected waypoint, we split the route into two subroutes as shown in Figure 6.6, and recursively calculate the optimal speeds for the two subroutes.

Once subroute costs are calculated, we pick the waypoint and arrival time with the lowest total cost for the two subroutes. If the optimal speed for any subroute is above the ship's maximum speed, this branch of the problem is infeasible, and the algorithm returns an infinite cost for that subroute. Once all feasible branches have been explored, we select the combined route with the lowest total cost.

Note that for the single-window RSA presented by Norstad, Fagerholt, and La-

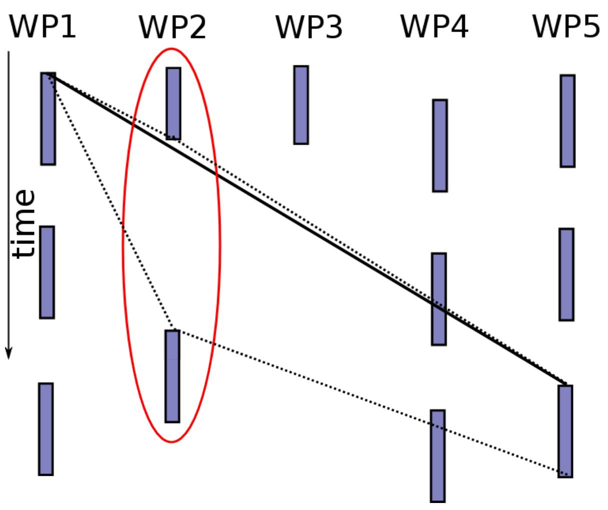


Figure 6.7: Example showing why both the waypoints with the largest time window violations before and after need to be considered for splitting the route. WP2 has the largest time window violation overall; however, if only WP2 is considered for splitting, the valid window at WP3 will not be found.

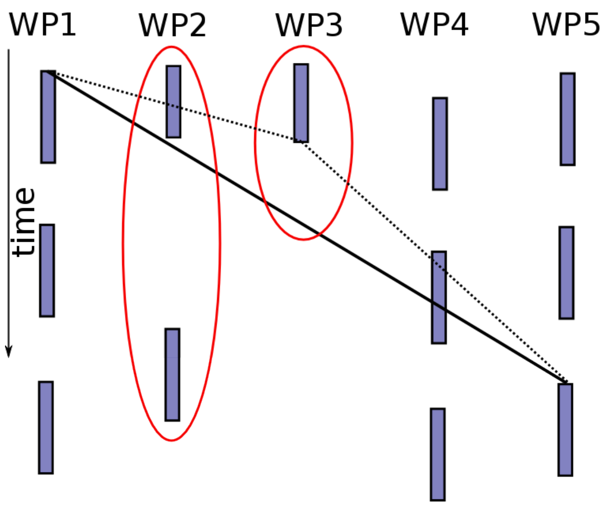


Figure 6.8: WP3 has the largest distance to the previous time window, and considering both WP2 and WP3 finds the optimal route.

porte [2011], the cost function is monotonic, so it is always possible to fix the arrival time at the final waypoint to the latest feasible arrival time (the end of the time window). For MW_RSA, the cost function can be quadratic, so the optimal arrival time may fall between two waypoints. This means that when choosing the waypoints with the largest time window violations to split the route on, the final waypoint must also be considered for splitting. After each split, the two subroutes to be calculated may be slightly different problems – the subroute on the left has a fixed arrival time its final waypoint, whereas the subroute on the right may still have a flexible arrival time at the final waypoint if the end of the subroute is the final waypoint and its arrival time has not yet been set.

6.2.6 Formal Algorithm Definition

Let the waypoints along the route be indexed by $i = 0, ..., N$, with $V_{i,i+1}$ and $D_{i,i+1}$ being the speed and distance respectively from waypoint i to $i + 1$. Let s and e be the start and end nodes for the partial route, respectively. Let T_i be the arrival time at waypoint i. Let V_{max} be the maximum possible speed for the ship, and V_{min} be the minimum speed the ship can travel at in practice, used to calculate the minimum fuel cost for the voyage if the ship travels below this speed and waits. Let V_{opt} be the optimal speed in the absence of time window constraints, given by Equation (6.9).

Let W_i be the set of draft-restricted time windows w at waypoint i for the given ship with the given draft, sorted in increasing order of time, with each time window specified by the time interval $\underline{w}, \overline{w}$. Each waypoint also has earliest and latest arrival times E_i and L_i respectively for the given ship, which can be the start or end of the time range for a waypoint where there is no limit on arrival times except for the draft restriction. Let $\textsc{total\_cost}(V_{i,i+1}, V_{min}, D_{i,i+1})$ specify the cost for travelling the leg from i to $i + i$ at speed $V_{i,i+1}$, including both fuel and other costs, as given by Equation (6.2) in Section 6.1.1. V_{min} is used in this equation to place a lower bound on the fuel cost for each leg as the fuel used by the ship travelling at minimum speed.

Let $\textsc{get\_valid\_times}(t, W_i)$ be a function that returns $(\underline{t}, \overline{t})$ – the nearest times before and after t that are within the time windows W_i for waypoint i. If t is already within a window, both $\underline{t}$ and $\overline{t}$ will be equal to t. If t is before the first window or after the last window, then an $INFEASIBLE$ flag will be returned for $\underline{t}$ or $\overline{t}$ respectively. Function $\textsc{get\_valid\_times}(t, W_i)$ is defined by Algorithm 2.

Let $\textsc{get\_TW\_violations}(T_s, ...T_e, W_s, ...W_e)$ be a function that returns $(i_{late}, i_{early}, \underline{t}_{late}, \overline{t}_{late}, \underline{t}_{early}, \overline{t}_{early})$, where i_{late} and i_{early} are the indices of the waypoints with the largest time window violations δ_{late} and δ_{early} for a ship arriving late or early respectively; $\underline{t}_{late}$ and $\overline{t}_{late}$ are the nearest valid times for the time windows before and after the optimal arrival time for waypoint i_{late}; and $\underline{t}_{early}$ and $\overline{t}_{early}$ are defined similarly for waypoint i_{early}.

If all $(T_s, ...T_e)$ are within the time windows, the flag OK is returned for $(i_{late}$ and i_{early}. If all time window violations were before or after the first or last window, then the flag $INFEASIBLE$ is returned for $\underline{t}_{late}$ and $\underline{t}_{early}$ or $\overline{t}_{late}$ and $\overline{t}_{early})$ respectively. If there is no valid time at all for some waypoint, the flag $INFEASIBLE$ is returned

Algorithm 2 GET_VALID_TIMES(t, W_i)

```
$\underline{t} = \overline{t} = INFEASIBLE$
**for** $w \in W_i$ from earliest to latest **do**
    **if** $\overline{w} \geq E_i$ and $\underline{w} \leq L_i$ **then** (Only include windows that are within $[E_i, L_i]$)
        **if** $t < \underline{w}$ or $t < E_i$ **then** ($t$ is before this window, or before the earliest time.)
            $\overline{t} = \max(E_i, \underline{w})$
            return $(\underline{t}, \overline{t})$
        **end if**
        **if** $t > L_i$ **then** ($t$ is after the latest arrival time.)
            $\underline{t} = \min(L_i, \overline{w})$
            return $(\underline{t}, \overline{t})$
        **end if**
        **if** $t > \overline{w}$ **then** ($t$ is after this window.)
            $\underline{t} = \overline{w}$
        **else** ($t$ is within this window and the allowable time range.)
            return $(t, t)$
        **end if**
    **end if**
**end for**
return $(\underline{t}, \overline{t})$
```

Algorithm 3 GET_TW_VIOLATIONS($T_s, ...T_e, W_s, ...W_e$)

```
$\delta_{late} = \delta_{early} = 0$ (Biggest time differences found so far.)
$i_{late} = i_{early} = OK$
$\underline{t}_{late} = \overline{t}_{late} = \underline{t}_{early} = \overline{t}_{early} = INFEASIBLE$
**for** $i \in [s, ..., e]$ **do**
    $(\underline{t}, \overline{t})$ = GET_VALID_TIMES($T_i, W_i$)
    **if** $\underline{t} \neq T_i$ **then** ($T_i$ is not within a window.)
        **if** $\underline{t} = \overline{t} = INFEASIBLE$ **then** (No valid times found for this waypoint.)
            $i_{late} = i_{early} = \underline{t}_{late} = \overline{t}_{late} = \underline{t}_{early} = \overline{t}_{early} = INFEASIBLE$
            return $(i_{late}, i_{early}, \underline{t}_{late}, \overline{t}_{late}, \underline{t}_{early}, \overline{t}_{early})$
        **end if**
        **if** $\underline{t} \neq INFEASIBLE$ and $T_i - \underline{t} > \delta_{late}$ **then** (Latest arrival found so far.)
            $\delta_{late} = T_i - \underline{t}$
            $i_{late} = i$, $\underline{t}_{late} = \underline{t}$, $\overline{t}_{late} = \overline{t}$
        **end if**
        **if** $\overline{t} \neq INFEASIBLE$ and $\overline{t} - T_i > \delta_{early}$ **then** (Earliest arrival found so far.)
            $\delta_{early} = \overline{t} - T_i$
            $i_{early} = i$, $\underline{t}_{early} = \underline{t}$, $\overline{t}_{early} = \overline{t}$
        **end if**
    **end if**
**end for**
return $(i_{late}, i_{early}, \underline{t}_{late}, \overline{t}_{late}, \underline{t}_{early}, \overline{t}_{early})$
```

for all of $(i_{late}, i_{early}, \underline{t}_{late}, \overline{t}_{late}, \underline{t}_{early}, \overline{t}_{early})$. GET_TW_VIOLATIONS$(T_s, ...T_e, W_s, ...W_e)$ is defined by Algorithm 3.

Algorithm 5 describes the MW_RSA algorithm for a ship with a fixed draft and fixed route. The algorithm returns the minimal cost of the route C, or the $INFEASIBLE$ flag if no route was found, eg. if one waypoint had no time windows. This algorithm uses function MIN_FEASIBLE_COST defined by Algorithm 4 to choose the minimum feasible cost for two routes, passing through the earlier or later time a given waypoint.

Algorithm 4 MIN_FEASIBLE_COST$(\underline{C}_{left}, \underline{C}_{right}, \overline{C}_{left}, \overline{C}_{right})$

```
$C = \underline{C} = \overline{C} = INFEASIBLE$
if $\underline{C}_{left} \neq INFEASIBLE$ and $\underline{C}_{right} \neq INFEASIBLE$ then
    $\underline{C} = \underline{C}_{left} + \underline{C}_{right}$
end if
if $\overline{C}_{left} \neq INFEASIBLE$ and $\overline{C}_{right} \neq INFEASIBLE$ then
    $\overline{C} = \overline{C}_{left} + \overline{C}_{right}$
end if
if $\underline{C} = INFEASIBLE$ then
    $C = \overline{C}$
end if
if $\overline{C} = INFEASIBLE$ then
    $C = \underline{C}$
end if
if $\underline{C} \neq INFEASIBLE$ and $\overline{C} \neq INFEASIBLE$ then
    $C = \min(\underline{C}, \overline{C})$
end if
return $C$
```

When MW_RSA is initially called for the complete route, the input T_s is set to the earliest allowable sailing time E_0 or the start of the next time window, since the ship can wait at any point along the route, so there is never any advantage to waiting at the first waypoint and we can assume without loss of generality that the ship always starts sailing from the first waypoint at the earliest feasible sailing time. The input T_e is the arrival time at the last waypoint for the subroute; it is set to $UNDEFINED$ when the arrival time of the final waypoint of the subroute has not yet been set, including when MW_RSA is initially called.

6.2.7 Smoothing Step

The MW_RSA algorithm described above finds the optimal time window for each waypoint, but it may make too many speed changes. An additional smoothing step is therefore needed to find the optimal sailing time within each waypoint.

Figure 6.9 shows an example of a situation where the basic MW_RSA algorithm will result in a suboptimal path. The algorithm finds the best time window for each waypoint, but because of the order in which waypoints are selected for splitting, and

Algorithm 5 MW_RSA(s, e, T_s, T_e)

```
Step 1: Calculate initial constant speed, ignoring time windows at waypoints.
if $T_e = UNDEFINED$ then
    $V_{const} = V_{opt}$
else
    $V_{const} = \sum_{i=s}^{e-1} D_{i,i+1} / (T_e - T_s)$
end if
for $i \in [s, ..., e-1]$ do (Calculate times at waypoints.)
    $T_{i+1} = T_i + D_{i,i+1} / V_{const}$
end for
Step 2: Check for time window violations
$(i_{late}, i_{early}, \underline{t}_{late}, \overline{t}_{late}, \underline{t}_{early}, \overline{t}_{early}) = \text{GET\_TW\_VIOLATIONS}(T_s, ...T_e, W_s, ...W_e)$
if $i_{late} = i_{early} = INFEASIBLE$ then (No solution possible for this route.)
    return $INFEASIBLE$
end if
if $i_{late} = i_{early} = OK$ then (No time window violations found. Calculate cost.)
    for $i \in [s, ..., e-1]$ do (Calculate cost for each leg.)
        $C_{i,i+1} = \text{TOTAL\_COST}(V_{const}, V_{min}, D_{i,i+1})$
    end for
    $C = \sum_{i=s}^{e-1} C_{i,i+1}$
    return $C$
else (Time window violations found.)
    Step 3a: Try splitting route at waypoint with latest arrival $i_{late}$.
    $\underline{C}_{left} = \underline{C}_{right} = \overline{C}_{left} = \overline{C}_{right} = INFEASIBLE$
    if $\underline{t}_{late} \neq INFEASIBLE$ then (First window feasible. Calculate costs.)
        $\underline{C}_{left}$ = MW_RSA$(s, i_{late}, T_s, \underline{t}_{late})$
        $\underline{C}_{right}$ = MW_RSA$(i_{late}, e, \underline{t}_{late}, T_e)$
    end if
    if $\overline{t}_{late} \neq INFEASIBLE$ then (Second window feasible. Calculate costs.)
        $\overline{C}_{left}$ = MW_RSA$(s, i_{late}, T_s, \overline{t}_{late})$
        $\overline{C}_{right}$ = MW_RSA$(i_{late}, e, \overline{t}_{late}, T_e)$
    end if
    $C_{late} = \text{MIN\_FEASIBLE\_COST}(\underline{C}_{left}, \underline{C}_{right}, \overline{C}_{left}, \overline{C}_{right})$
end if
if $i_{early} = i_{late}$ then (Only one waypoint to calculate.)
    return $C_{late}$
else (Two possible waypoints. Calculate cost for 2nd waypoint, return min cost.)
    Step 3b: Try splitting route at waypoint with earliest arrival, $i_{early}$.
    Calculate $C_{early}$ as Step 3a:, with $i_{early}, \underline{t}_{early}, \overline{t}_{early}$ replacing $i_{late}, \underline{t}_{late}, \overline{t}_{late}$.
    if $C_{late} \neq INFEASIBLE$ and $C_{early} \neq INFEASIBLE$ then
        return min$(C_{late}, C_{early})$
    else
        if $C_{late} = INFEASIBLE$ then
            return $C_{early}$
        else
            return $C_{late}$
        end if
    end if
end if
```

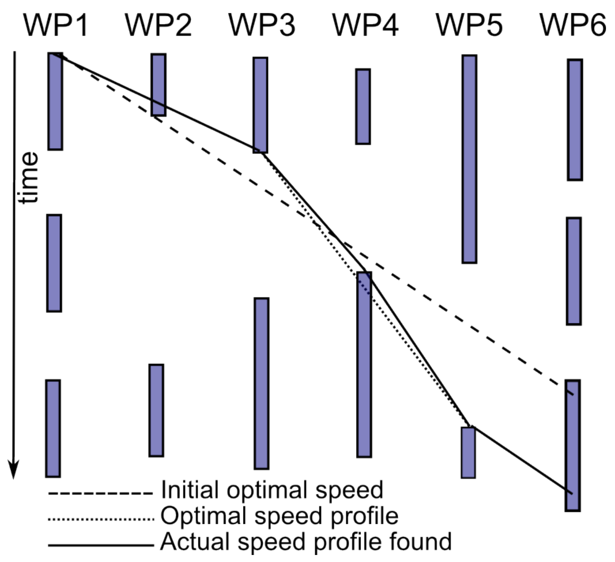


Figure 6.9: Example showing why a final smoothing step is required. WP2 or WP4 would be chosen first for splitting; however both result in a suboptimal route, as the optimal path passes through the middle of time windows at both WP2 and WP4. A smoothing step is required to remove unnecessary speed changes.

the adjustment of waypoint arrival times to the nearest feasible time, this results in the ship arriving at the edge of the window at WP4, even though a constant speed between the WP3 and WP5 arrival times would pass within the same time window at WP4, and would reduce the travel cost. WP2 and WP4 are the waypoints with the largest time window violations, and therefore the waypoints that would be selected first for splitting, and for adjusting the times to the edges of adjacent time windows. However, the optimal path passes through the middle of the time windows at both of these waypoints.

To deal with situations like this, where the algorithm finds the optimal time windows at waypoints but results in a path with too many speed changes, we need to add an additional step of smoothing the speeds between successive waypoints. We modify the basic MW_RSA algorithm presented in Section 6.2.6 to calculate each path and perform an additional smoothing step to eliminate any unnecessary changes in speed.

Algorithm 6 describes the final smoothing step of the Multi-Window RSA algorithm. Let t_i be the arrival time at waypoint i, and t'_i be the time being considered for smoothing at waypoint i. Let $v_{i,i+1}$, $d_{i,i+1}$ and $c_{i,i+1}$ be the speed, distance and cost respectively from i to $i+1$, and v_{const} be the constant speed being considered for smoothing. Let v_{min} be the slowest speed at which the ship can actually travel, used to calculate minimum fuel cost, since if a ship is scheduled to travel below this speed, it actually travels at v_{min} and waits. Let s and e respectively be the start and end waypoints for the route segment being considered for smoothing, and let v_s and v_e be the speeds immediately before and after waypoints s and e respectively. Let W_i be the set of allowable sailing windows for waypoint i. Let OUTSIDE_WINDOWS(t, W_i)

Algorithm 6 MW-RSA SMOOTHING

```
$s = 0$
$e = 2$
while $s < N - 1$ and $e < N$ do
    if $v_s \neq v_e$ then
        Step 1: Check if travelling at constant speed for subroute $s$ to $e$ is feasible (i.e. times are within windows).
        SMOOTH_SPEEDS = True
        $t'_s = t_s$
        $v_{const} = \frac{\sum_{i \in [e,s-1]} d_{i,i+1}}{t_e - t_s}$
        for $i = s + 1$ to $e$ do
            $t'_i = t'_{i-1} + \frac{d_{i-1,i}}{v_{const}}$
            if OUTSIDE_WINDOWS$(t'_i, W_i)$ then
                SMOOTH_SPEEDS = False
            end if
        end for
        if SMOOTH_SPEEDS then
            Step 2a: Times are feasible. Set constant speed for subroute $s$ to $e$.
            for $i = s + 1$ to $e$ do
                $t_i = t'_i$
                $v_{i-i,i} = v_{const}$
                $c_{i-1,i} = $ TOTAL_COST$(v_{const}, v_{min}, d_{i-1,i})$
            end for
            $e = e + 1$
        else
            Step 2b: Times are infeasible. Move on to next section of the route.
            $s = s + 1$
            if $e - s < 2$ then
                $e = e + 1$
            end if
        end if
    else
        Step 3: Speeds between waypoints $s$ and $e$ are already constant.
        $e = e + 1$
    end if
end while
```

be a function that returns true iff t is outside the time windows in the set W_i.

The algorithm iteratively considers a subsection of the complete route with at least 3 waypoints, where the speed in the first segment differs from the last. If travelling at constant speed for the subsection being considered results in all times being within windows, then speeds can be smoothed (adjusted to be constant). Otherwise, the algorithm moves on to the next subsection along the route. The algorithm returns the adjusted arrival times t_i at each waypoint i, with corresponding updated speeds $v_{i,i+1}$ and costs $c_{i,i+1}$ for each section of the route.

6.2.8 Optimality of RSA

Norstad, Fagerholt, and Laporte [2011] did not provide a proof of optimality for their Recursive Smoothing Algorithm, though they did prove that for a monotonic cost function (eg. one that only considers fuel usage), a constant speed between two points was always lower cost than travelling parts of the route at different speeds. We present here a proof of optimality of the single-window RSA; a discussion of the optimality of the Multi-Window RSA is presented in the next section.

The fuel cost for travelling a distance d at speed v is defined by Equation (6.1), $\textsc{fuel\_cost}(v, d) = k \cdot d \cdot v^2$, where k is a constant. Since $v = d/t$, the fuel cost for travelling distance d_1 in time $t_1 - x$ and distance d_2 in time $t_2 + x$ is given by Equation (6.10):

$$f(x) = k \cdot \frac{d_1^3}{(t_1 - x)^2} + k \cdot \frac{d_2^3}{(t_2 + x)^2} \tag{6.10}$$

$f(x)$ is convex within the interval $x \in (-t_2, t_1)$, with the minimum point occurring when $d_1/(t_1 - x) = d_2/(t_2 + x)$, i.e. when the speed is constant.

Therefore, any optimal path between two points with fixed times only allows the speed to change at waypoints, since for any path with a speed change between two adjacent waypoints, a lower-cost path can be found by travelling at constant speed for that leg. Likewise, any optimal path between two points with fixed times can only allow the speed to change at waypoints which the ship passes at the edge of the waypoint's time window, since if the ship passes the waypoint in the middle of a time window, the speed to and from adjacent waypoints can be adjusted to be closer to constant, thus reducing the cost. Therefore any optimal path between two points with fixed times consists of constant-speed segments between the edges of time windows at waypoints.

Consider the RSA algorithm for a subroute between waypoints s and e, with optimal times at s and e already known. Let p be the waypoint in this subroute with the largest time window violation when travelling at constant speed v between the fixed times at s and e. Let the nearest endpoint of the time window at p be at time t. Let L be the line passing between points s and e at constant speed v, and let L' be the line passing through point p at time t with constant speed v, as shown in Figure 6.10.

Suppose an optimal path P exists which passes through point p at some other point t' within its time window. Since P is an optimal path, it can only change direction at waypoints where it passes through the edge of a window. P is therefore

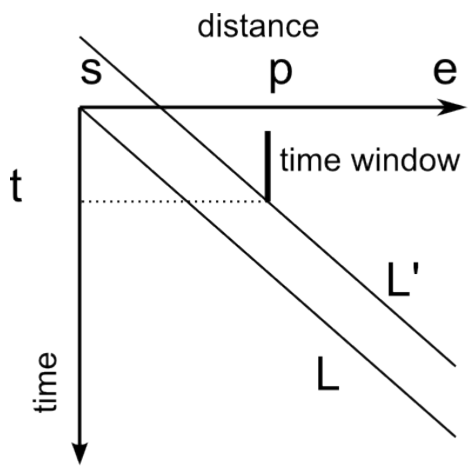


Figure 6.10: Illustration for RSA proof of optimality.

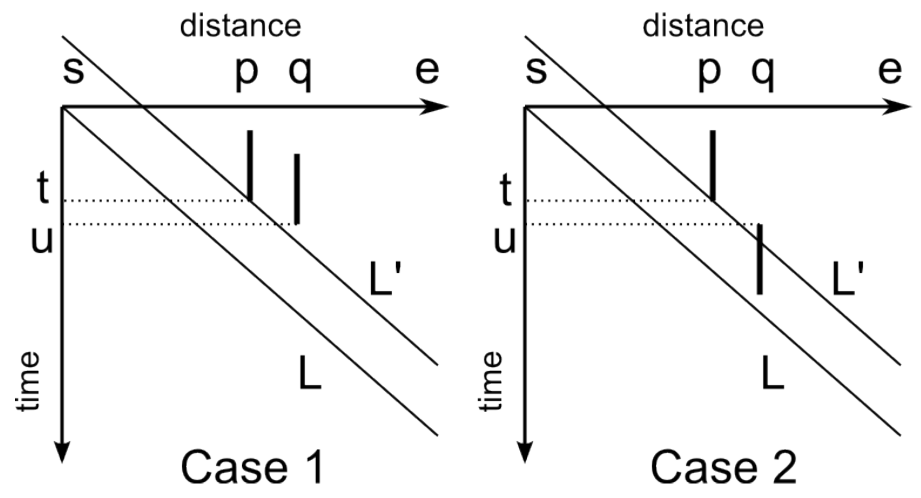


Figure 6.11: Illustration for RSA proof of optimality – 2 cases.

a piecewise-linear continuous path from (s, t_s) to (e, t_e). Since P starts and finishes on the same side of L', and passes through t' which is on the other side of L', it must change direction on the side of L' that contains t'. Since P can only change direction at waypoints when it passes through the edge of the time window, there must exist a waypoint q with time-window edge point u such that P passes through u and u is on the same side of L as t'.

There are now two cases, as illustrated in Figure 6.11:

1. The time window for q lies on the same side of L' as the time window for p. p is the point with the largest time window violation, therefore both t' and u must lie on L', so $t' = t$ and P passes through the nearest edge of the time window of point p.

2. The time window for q lies on the opposite side of L'. Therefore the cost for P can be decreased by having the path P pass through q closer to L', so P is not

the optimal path.

Therefore, if p is the point with the largest time window violation when travelling at constant speed between two waypoints with fixed start and end times, then any optimal path P must pass through p at the nearest feasible time t to the constant-speed path. Therefore, since the optimal time at p is now fixed, the route s to e can be split into two subroutes s to p and p to e, both of which have fixed optimal times at the start and end. The optimal arrival times for intermediate waypoints along both subroutes can therefore be found recursively. Therefore the Recursive Smoothing Algorithm presented by Norstad, Fagerholt, and Laporte [2011] finds the optimal solution.

6.2.9 Optimality of MW_RSA

Proving optimality for MW_RSA would be significantly more difficult than for the single-window RSA algorithm; however, we discuss differences in the two problems and how these affect optimality.

The Recursive Smoothing Algorithm presented by Norstad, Fagerholt, and Laporte [2011] is optimal due to the monotonic cost function and the fact that there is only one window for each waypoint. Neither of these is the case for Multi-Window RSA; however, for any subroute where the arrival time at the start and end is fixed, the cost function reduces to the fuel usage, since travel time is constant, so the cost function becomes monotonic.

Therefore several key results from the RSA proof of optimality still hold for MW_RSA:

1. For any path P between two waypoints with fixed times, the optimal path in the absence of windows has constant speed.

2. Therefore, any optimal path P only changes speed at waypoints.

3. Therefore, any optimal path P between two points with fixed times only changes speed at waypoints which the ship passes at the edge of one of the time windows.

4. Therefore, any optimal path P between two points with fixed times consists of constant-speed segments between the edges of time windows at waypoints.

5. Therefore, any optimal path P between two points s and e with a fixed time at s but not e consists of constant-speed segments between the edges of time windows at waypoints, except for the final waypoint, e, where the arrival time can be in the middle of the waypoint if and only if the ship travels at its minimum-cost speed V_{opt} for the final leg of the route.

6. Given a subroute between two points s and e with fixed times, let v be the constant speed between those points. Let p be the point with the largest time window violation on one side of the line L passing through s and e at constant

speed v, with t being the edge of the nearest time window on that side of L at point p. Then any optimal path P between points s and e never crosses the line L' which passes through p at time t with constant speed v. (P can touch the line L', but not cross it.

7. The above result also holds for a subroute between points s and e where the time at e is not fixed, but lines L and L' have the constant speed V_{opt}, which is the minimal-cost speed for the ship.

For the single-window RSA, the last three results above led to the conclusion that for a point p which has the largest time window violation on one side of L, any optimal path P will pass through p at the nearest feasible time t at the edge of its window. However, for MW_RSA, there is another option – an optimal path P can also pass through point p during a time window on the opposite side of L. This is why MW_RSA needs to check both the time window before and after the optimal time for the waypoints with the largest time window violation on either side of L, instead of only one time point as with the single-window RSA.

Also, if the optimal path P passes through a waypoint p on the opposite side from the window that is furthest away from L, then there is no guarantee that P passes through the edge of the time window. One of the paths P' explored by MW_RSA will pass through the same time window edges as some optimal path P, but P' may also make some unnecessary speed changes to pass through the edge points of some time windows where a lower-cost path would pass through the middle. This is why the MW_RSA algorithm described in Section 6.2.6 needs the additional smoothing step at the end to remove any unnecessary changes in speed, as described in Section 6.2.7.

6.2.10 Scalability of Multi-Window RSA

The MW_RSA algorithm has two branch points: the choice of waypoint to split the route on, and the time window chosen at that waypoint. This clearly makes the algorithm exponential in the number of waypoints; however, many branches can be pruned during the search, since some subroutes will turn out to be infeasible. If a subroute requires the ship to sail faster than its maximum speed, then that combination of waypoint and time window does not need to be explored further.

We consider the worst-case number of subroutes explored for any given number of waypoints, and compare this worst-case against the MW_RSA average performance on a set of problems.

Let $f(n)$ be the worst-case number of subroutes explored by MW_RSA for a route with n waypoints and with a fixed time at the final waypoint, and let $g(n)$ similarly specify the worst-case number of subroutes for n waypoints with the final time not fixed. We can immediately observe that $f(2) = 1$, since for a route with 2 waypoints, if the arrival time at the final waypoint is fixed, there is only one route to explore. Likewise, $g(2) = 2$, since if the time at the final waypoint is not fixed, then the optimal arrival time at the final waypoint may fall between two windows, requiring

2 time windows to be considered, but there are no intermediate waypoints that we may need to split the route on.

For a closed-ended route with 3 waypoints, $f(3) = 2 \cdot [f(2) + f(2)] = 4$, since we have only one waypoint that the route may be split on, but we need to consider adjusting the time at that waypoint to both the previous and next window. After splitting the route, we will also need to run MW_RSA again for both of the resulting subroutes with 2 waypoints.

For an open-ended route, $g(3) = 2 \cdot [f(2) + g(2)] + 2 \cdot f(3) = 14$, since there are 2 possible waypoints to consider for splitting, and the waypoint with the largest time window violation before the optimal arrival time and the largest time window violation after the optimal time may differ. For both ways of splitting the route, we need to consider adjusting the time to both the previous and next window at that waypoint. Splitting on the intermediate waypoint results in a closed-ended route with 2 waypoints and an open-ended route with 2 waypoints. Splitting on the final waypoint results in one closed-ended route with 3 waypoints. Finally, after splitting the route, we need to run RSA again for all new subroutes.

For a closed-ended route with 4 waypoints, $f(4) = 2 \cdot 2 \cdot [f(2) + f(3)] = 20$, since we have 2 possible waypoints to split on, each with 2 possible windows, and both possible splits result in one closed-ended 2-waypoint route and one closed-ended 3-waypoint route.

For closed-ended routes with more than 4 waypoints, we can see that $f(n) = 2 \cdot 2 \cdot [f(2) + f(n-1)] = 4 + 4f(n-1)$, since $f(n)$ increases exponentially, so the worst-case split will always be the one that produces one 2-waypoint route, and one route with $n - 1$ waypoints.

For an open-ended route with 4 waypoints, $g(4) = 2 \cdot [f(2) + g(3) + f(4)] = 70$, the two worst-case waypoints for splitting are the first and last waypoints that haven't already had their times set. These two waypoints continue to be worst-case for splitting for open-ended routes with more than 4 waypoints, which can be proved by induction, so for an open-ended route with more than 4 waypoints, we can see that $g(n) = 2 \cdot [f(2) + g(n-1) + f(n)] = 2 + 2 \cdot g(n-1) + 2 \cdot f(n)$.

The recurrence relation for $f(n)$ can be solved to find that the worst-case number of subroutes explored by MW_RSA for a closed-ended route with n waypoints is defined by:

$$f(n) = \frac{4^{n-1} - 1}{3} - 1 \tag{6.11}$$

Similarly, the recurrence relation for $g(n)$ can be solved to find that the worst-case number of subroutes explored by MW_RSA for an open-ended route with n waypoints is defined by:

$$g(n) = \frac{4^{n} - 1}{3} - 2^n + 1 \tag{6.12}$$

This worst-case behaviour occurs if the waypoints with the largest time window violations are always different and always at the edges of the route; the optimal time for each node is outside the time windows at each waypoint; and both the previous and next time windows result in feasible routes at each waypoint for each subroute.

N	Worst case	Average	Worst case over 8 ships
4	70	8.05	51
8	21,590	206	1,349
12	5.59 x 10^6	1,114	7,821
16	$1.43 \cdot 10^9$	5,553	29,213

Table 6.1: RSA: worst-case number of speed optimisation calculations vs. average over 8 ships x 20 drafts vs. worst-case seen over 8 ships x 20 drafts.

	Shortest Path Time Discretisations					MW_RSA
N	20	40	60	80	100	–
4	0.64	2.41	5.57	9.97	16.2	0.02
8	1.06	4.02	8.38	15.0	24.8	0.62
12	1.29	4.72	10.3	18.8	30.8	4.96
16	1.67	6.11	13.1	24.0	40.0	29.1

Table 6.2: Calculation time in seconds for optimising speeds for 8 ships x 20 drafts, for MW_RSA vs Shortest Path approach.

For the largest problem instance with 16 waypoints, the worst case number of speed optimisations is: $\frac{4^{16}-1}{3} - 2^{16} + 1 = 1.43 \cdot 10^9$. However, in practice the average number of speed optimisations calculated is much smaller, since some nodes will have the optimal time fall within a time window, and some time windows will result in an infeasible problem for one of the subroutes, so no further calculations will be done for that time window.

Table 6.1 shows the worst-case number of speed optimisations calculated for problem instances with 4, 8, 12 and 16 waypoints, as well as the actual number of speed optimisations averaged over 8 ships with different maximum speeds, and with 20 drafts each. This shows that, for realistic sized problems, the average-case number of speed optimisations calculated is small enough that the problem is tractable. In fact, for problems of 4 – 12 waypoints, the Recursive Smoothing Algorithm finds optimal speeds faster than the reasonably accurate shortest path approaches (60, 80 and 100 time discretisations) even with the exponential growth in calculation time with problem size. Table 6.2 compares the times taken to optimise speeds using the shortest path approach vs. MW_RSA for 8 ships with 20 drafts each, with 4, 8, 12 and 16 waypoints.

Note that, while the shortest path approach is faster to calculate for the largest problems, it suffers from producing lower quality solutions due to discretisation error. Solution quality comparisons between MW_RSA and the shortest path approach are presented in Table 6.4 of Section 6.3.

The final smoothing step advances either the start or end of the smoothing window at each step, and considers each waypoint within the window once at each step. The smoothing step is thus $O(n^2)$ worst-case complexity where n is the number of waypoints, which is insignificant compared to the worst-case complexity of the rest of the algorithm.

Other algorithms and speedup approaches may also be worth investigating for solving the Speed Optimisation Problem with multiple time windows, for example dynamic programming could be used to avoid re-calculating results for the same sub-routes in MW_RSA, which may provide a way to convert MW_RSA to an algorithm that is both polynomial and optimal. Further investigation of approaches for solving the single-ship fixed-route Speed Optimisation Problem is left for future work.

6.3 One Ship with Variable Draft

This section extends the Speed Optimisation Problem with Time-Varying Draft (SOPTVD) for a single ship travelling through several draft-constrained waypoints to also choose an optimal draft for the ship. To find an optimal draft and speed for a single ship, the draft range for the ship is discretised, and either the shortest path method or Multi-Window Recursive Smoothing Algorithm (MW_RSA)is used to optimise speeds between waypoints for each discrete draft. The time windows at each waypoint are wider for lower drafts, which allows a greater range of speeds to be used for travelling between waypoints, thus allowing speeds with a lower total fuel cost to be used.

The total cost for each discretised draft is then calculated by subtracting a cost savings factor for carrying more cargo than the minimum draft from the optimal fuel cost for the speed-optimised route for that draft. Finally, the draft with the lowest combined cost for each ship is selected. This algorithm for solving the Speed and Draft Optimisation Problem with Time-Varying Draft (SDOPTVD) for a single ship is described by Algorithm 7.

Let d_0 to d_N be the discretised draft values, and let M be the number of legs of the voyage with distances D_j. Let W_i be the set of time windows at each waypoint for draft d_i. Let SPEED_OPT be an algorithm for finding optimal speeds $\{V_0, ..., V_{M-1}\}$ between waypoints given the set of time windows W_i at each waypoint. Note that the optimal speeds will also depend on the speed range and cost function for the ship; however, these do not vary with draft. Let V_{min} be the minimum speed for the ship, used to calculate the minimum fuel cost for each leg.

TOTAL_COST and COST_SAVINGS are specified by Equations (6.2) and (6.3) in Sections 6.1.1 and 6.1.2 respectively. Algorithm 7 returns the minimum cost C_{min} and the optimal draft d_{opt} for this ship, as well as the optimal speeds $\{V_0, ..., V_{M-1}\}$ for that draft.

The sum of the minimal costs for a set of ships with variable drafts but with no interaction between ships at waypoints can be used as a lower bound on the cost of the Multi-Ship Speed Optimisation Problem with Time-Varying Draft (MS-SOPTVD) with resource conflicts between ships at waypoints, presented in the next section.

Algorithm 7 SOLVE_SDOPTVD: Solving the Speed and Draft Optimisation Problem with Time-Varying Draft (SDOPTVD) for one ship

```
$C_{min}$ = BIGNUM
$d_{opt} = 0$
for $i = 0$ to $N - 1$ do
    $\{V_0, ..., V_{M-1}\}$ = SPEED_OPT($W_i$)
    $C = \sum_{j \in [0, M-1]}$ TOTAL_COST($V_j, V_{min}, D_j$)
    $S$ = COST_SAVINGS($d_i$)
    if $C - S < C_{min}$ then
        $C_{min} = C - S$
        $d_{opt} = d_i$
    end if
end for
```

6.3.1 Problem Instances

Algorithm 7 was used to calculate optimal drafts and speeds for 8 ships, with constructed voyages containing 4, 8, 12 and 16 waypoints. The distances between waypoints for the 4-waypoint voyage are similar to the real-world scenario of transporting bauxite ore from the mines at Weipa through the Torres Strait to the refinery at Gladstone, shown in Figure 6.2. Each longer voyage adds 4 more waypoints to this scenario. The distances between waypoints for the 16-waypoint voyage are shown in Table 6.3.

The 8 ships and 4 routes makes a combined total of 32 problem instances. The ships had maximum drafts of 12.0 to 12.2 metres, and varying minimum and maximum speeds ranging from 5 to 7 knots for minimum speeds, and 14 to 17 knots for maximum speeds. The minimum draft was 10.5 metres for each ship, with 20 draft discretisations calculated. Ships were allowed to wait at waypoints, so in practice there was no minimum duration for travel between waypoints, even though each ship had a minimum speed below which the fuel usage function would not continue to decrease.

The maximum allowable draft at waypoints was modelled as a sine function, with the difference between high tide and low tide varying from 2.5m (spring tide) to 1.1m (neap tide). Each waypoint used the same maximum draft function, but offset in time and tide height by several hours and up to 0.5m.

Each ship also had different earliest and latest arrival and departure windows at the origin and destination of the voyage, in addition to the windows at waypoints caused by the height of the tide.

The shipping cost per tonne for the voyage, used to calculate cost savings for draft above the minimum, was estimated as US$2.5 for the shortest voyage with 4 waypoints, and scaled linearly by distance for the longer voyages. The $2.5 figure was the approximate cost per tonne of the average of the costs for a speed-optimised round-trip 4-waypoint voyage for a Panamax bulk carrier transporting 70,000 tonnes of cargo for the first half of the voyage, and sailing empty for the second half.

Waypoint	Distance to Next
1	170
2	30
3	1080
4	50
5	30
6	140
7	950
8	220
9	40
10	50
11	30
12	300
13	20
14	20
15	500

Table 6.3: Distances between waypoints for 16-waypoint route, in nautical miles.

6.3.2 Experimental Results

Algorithms for solving the shortest path problem with discretised arrival times, as well as the recursive smoothing algorithm were implemented in Python. The shortest path approach was run with 20, 40, 60, 80 and 100 arrival time discretisations. All calculations were run on a Windows 7 machine with an Intel i7-930 quad-core 2.80 GHz processor and 12.0 GB RAM.

MW_RSA vs Shortest Path

Table 6.4 compares the costs of the optimal solutions found by the recursive smoothing algorithm against solutions found by the shortest path approach with 20, 40, 60, 80 and 100 arrival time discretisations. This shows that, while the shortest path approach can find close to optimal solutions, particularly with a large number of time discretisations, the recursive smoothing algorithm still produces some improvement in results for longer routes.

20 time discretisations is clearly insufficient for long routes of 12-16 waypoints – the shortest path approach with 20 time discretisations results in costs that are around 25% higher than MW_RSA for these routes. For 60 or more time discretisations, on the other hand, the results produced by MW_RSA are only slightly better than the shortest path results.

Shipping Costs vs Opportunity Cost

Figure 6.12 shows the minimum sailing cost and under-loading opportunity costs (cost of carrying less cargo) for a range of discretised drafts for an example vessel. This shows that sailing with the maximum possible draft increases the combined cost,

	Shortest Path Time Discretisations				
	20	40	60	80	100
N	Cost Increase over MW_RSA (%)				
4	0.54	0.54	0.54	0.53	0.53
8	5.94	2.55	0.94	0.18	0.25
12	26.66	6.79	2.72	1.44	1.04
16	25.29	5.86	2.91	2.00	1.31

Table 6.4: Percentage by which Shortest Path cost with 20, 40, 60, 80 and 100 arrival time discretisations exceeds RSA cost, averaged over 8 ships.

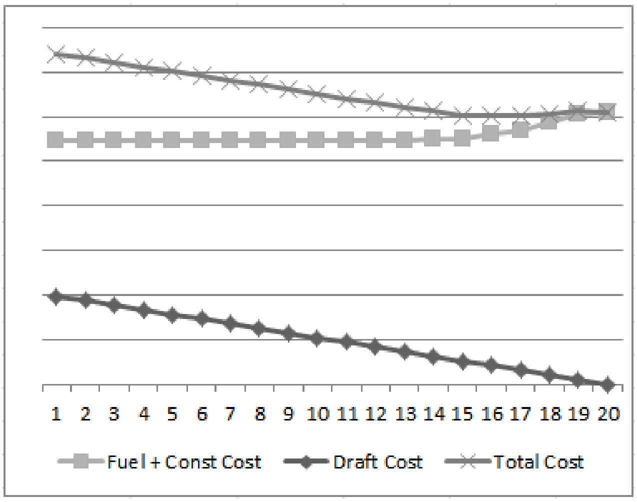


Figure 6.12: Minimum voyage cost and opportunity cost for 20 discretised draft values.

	Time Windows for Unconstrained Draft				
	4 hrs	6 hrs	8 hrs	10 hrs	12 hrs
N	Cost Improvement (%)				
4	1.53	4.55	12.29	26.18	32.10
8	2.31	4.64	7.02	21.49	28.17
12	2.49	4.77	7.13	21.74	28.47
16	2.74	4.74	6.79	19.96	26.14

Table 6.5: Percentage cost improvement of optimal draft vs. unconstrained draft, using RSA for speed optimisation, averaged over 8 ships.

since sailing with the maximum draft results in very narrow time windows at each waypoint, forcing the ship to sail with a higher speed to reach the draft-constrained waypoints within the allowable time windows, thus resulting in higher fuel costs.

On the other hand, sailing with a very low draft does not reduce fuel costs significantly beyond a certain point, as the time windows at waypoints become wide enough that further reductions in draft do not improve fuel consumption enough to offset the opportunity cost lost by sailing with an under-loaded ship. From the cost example in Figure 6.12, it is clear that there is a financial benefit to considering both draft and speed in the cost optimisation of cargo ship routes, rather than optimising speeds and draft constraints separately.

Optimal Draft vs Unconstrained Draft

Table 6.5 presents the improvement in cost resulting from using the optimal draft for each ship for the 4, 8, 12 and 16-waypoint routes, by comparing against the optimal cost for "unconstrained" ship drafts – drafts that are low enough to get windows of at least 4, 6, 8, 10 and 12 hours at each waypoint. The tides in this simulation are diurnal, with a 24-hour time difference between successive high tides, so sailing windows of at least 6 hours results in a ship being able to sail at least 25% of the time.

The cost improvement of using the optimal draft is small compared to the optimal speed for a fixed draft which gets 4-hour windows; however, the cost difference increases significantly for less tightly constrained drafts. A ship with an "unconstrained" draft that has minimum 12-hour windows at each waypoint (ie. a ship that can sail without waiting at least 50% of the time) has 30% higher shipping costs (mainly due to under-loading) compared to a ship sailing with optimal draft.

Tables 6.5 compares the costs of the optimal speeds for the optimal draft against a less constrained draft; however, the optimal speeds for both drafts are calculated with the allowable sailing windows at each waypoint taken into account.

In Table 6.6, on the other hand, we compare the optimal costs found by considering time-varying draft constraints in speed optimisation against a speed optimisation model that ignores draft constraints, leading to ships sometimes being scheduled to sail through a draft-constrained waypoint at low tide and having to wait for the tide to rise. This situation may occur if a speed optimisation model that does not include

N	Cost Improvement (%)
4	16.98
8	21.12
12	25.24
16	25.90

Table 6.6: Percentage reduction in cost of considering draft constraints in speed optimisation vs. ignoring draft windows in speed optimisation and waiting when ship arrives outside the windows, for same (optimal) draft. Averaged over 8 ships.

N	Cost Improvement (%)
4	15.64
8	10.65
12	9.84
16	11.47

Table 6.7: Percentage reduction in cost of considering draft constraints in speed optimisation vs. ignoring draft windows in speed optimisation and waiting when ship arrives outside the windows, for same (optimal) draft, when ships do not sail faster to catch up. Averaged over 8 ships.

draft constraints is used to optimise sailing speeds for a ship over a route where some waypoints are draft-constrained. The costs in Table 6.6 are calculated for the same draft (the optimal draft found by our approach), with and without considering draft constraints.

Table 6.6 shows that for the optimal draft, the costs of ignoring draft constraints in speed optimisation and waiting for the high tide range from 17% to 26% for different routes.

When draft constraints are not considered in speed optimisation, ships wait upon arrival at the waypoint if the tide is not high enough, and then sail faster afterwards to catch up. We also investigated an alternative method of dealing with waiting for the high tide, where ships continued sailing at the optimal, pre-calculated speed, even though they were running late. The cost improvement of considering draft against the waiting cost calculated using this method ranged from 9% to 16% for the different routes, and are shown in Table 6.7.

The highest cost increases for waiting using this method were incurred by the most tightly constrained waypoints, since a ship is more likely to arrive outside the windows for a tightly-constrained waypoint, and is thus going to have longer waiting times on average at more tightly-constrained waypoints. This explains why the 4-waypoint route has the highest cost increase for waiting when no catchups were permitted – one of the most tightly constrained waypoints is on the 4-waypoint route, whereas the 8- and 12-waypoint routes in particular have many waypoints that are less tightly constrained.

Note that the cost increase here is only due to the increased duration that the ship is in use. We do not consider any cost penalties for arriving late, whereas in

	Time Windows for Draft					
	2 hrs	4 hrs	6 hrs	8 hrs	10 hrs	12 hrs
N	Cost Improvement (%)					
4	18.75	9.06	7.66	5.47	0.00	0.00
8	33.26	11.03	10.31	9.37	1.38	0.00
12	34.65	14.45	12.59	11.73	3.76	1.87
16	33.79	13.93	12.27	11.54	4.27	2.33

Table 6.8: Cost improvement of considering draft restrictions in speed optimisation vs. ignoring draft restrictions and waiting, for a ship with fixed drafts with different minimum time windows at waypoints. Averaged over 8 ships.

practice, there may be financial penalties for late delivery when ships have to wait at waypoints for the tide to rise without sailing faster afterwards to catch up. The more costly method of ships sailing faster to make up for having to wait for a high tide is thus more realistic, as this method avoids late arrival penalties.

The differences in cost between the optimal speeds with and without considering draft constraints also vary for different drafts. Table 6.8 shows the cost improvements for using a speed optimisation approach with time-varying draft restrictions vs. a speed optimisation approach that ignores draft windows for ships with a range of fixed drafts, and allows ships to sail faster to catch up if they miss the high tide. The results of both methods are compared for ships fixed drafts that have windows of 2 to 12 hours around the high tide. The benefit of considering draft restrictions in speed optimisation decreases for less constrained drafts, until there is very little benefit in considering draft windows for a draft that gets minimum 12-hour windows at each waypoint.

A draft that gets minimum 12-hour windows has at least 50% chance of not having to wait at all when arriving at any given waypoint (assuming all waypoints have diurnal tides, ie. approximately one high tide every 24 hours). Some waypoints may have even wider windows, as the 12-hour minimum windows occur at even the most tightly constrained waypoints. In the 4- and 8-ship routes, a ship with a draft that gets 12-hour windows at all waypoints does not have to wait at all when sailing with optimal speeds calculated without draft restrictions. However, for all other routes and all higher drafts, ships have to wait at some of the waypoints, resulting in higher costs when draft restrictions were not considered in the speed optimisation.

Note that the 20 draft discretisations used for solving these problems introduce a discretisation error, since the actual optimal draft may lie between two discretised values. The next section shows that this discretisation error can increase shipping cost by up to 2% for some problems. Since increasing the number of draft discretisations would only result in a linear increase in calculation time, we recommend that discretising the draft at every centimetre be used in future, rather than discretising drafts at a more coarse resolution. (A finer draft resolution than 1cm is not worth calculating, since it is impractical to load a ship with such precision.)

6.4 Multiple Ships with Variable Draft

To optimise speeds for multiple ships sailing through draft-constrained waypoints, we also need to consider resource conflicts between ships at waypoints due to multiple ships trying to sail through a narrow waterway at the same time. This problem is very similar to the CP model for the Bulk Port Cargo Throughput Optimisation Problem introduced in Chapter 3 to minimise costs while meeting draft restrictions and resource constraints at each waypoint. This section combines the BPCTOP CP model with the single-ship Speed Optimisation Problem to create a new Multi-Ship Speed Optimisation Problem with Time-Varying Draft (MS-SOPTVD).

6.4.1 Waypoint Cost Optimisation CP Model

In this chapter, we use a simplified version of the BPCTOP Constraint Programming model from Chapter 3. This simplified model has no constraints on the availability of tugs, and it does not consider resource conflicts with (empty) inbound ships at the origin port. The objective function is also replaced by a function that minimises the total cost, including speed-dependent fuel costs for travelling to and from adjacent waypoints, instead of maximising the cargo carried on the set of ships as was done in the original model.

Let U be the set of all vessels to be scheduled, indexed by v. Let $[1, T_{max}]$ be the range of allowable time indices for this waypoint. Each ship has an associated earliest departure time $E(v)$, and a maximum allowable sailing draft $D(v, t)$ at each time t. Each ship also has an associated cost at each time $C(v, t)$, which combines the travel cost to and from adjacent waypoints with the cost savings due to the allowable draft at this time being above the minimum.

Each ordered pair of ships v_i, $v_j \in U$ also has a minimum separation time $ST(v_i, v_j)$ specifying the minimum delay required between their sailing slots. Let $s(v)$ be a binary decision variable specifying whether each ship v is included in the schedule (since some ships may not be able to be scheduled), and let Q be a large cost penalty for a ship not being included in the schedule. Let $T(v) \in [1, T_{max}]$ be the decision variable specifying the scheduled sailing time for each vessel v.

Note that while ships are not allowed to overtake each other at waypoints, they may pass each other between waypoints, so the order of ship sailing times may change from waypoint to waypoint. Also, in this problem ships are all travelling in the same direction; however, waypoint resource conflicts with ships travelling in the opposite direction could be implemented by incorporating the different delays into the sequence-dependent separation times, similarly to the way incoming and outgoing ships are handled in the BPCTOP in Chapter 3.

The cost optimisation problem for a set of ships at a waypoint with time-varying draft constraints can now be formulated as follows:

$$\text{minimise} \sum_{v \in U} s(v) \cdot C(v, t) + (1 - s(v)) \cdot Q \tag{6.13}$$

subject to

$$s(v) = 1 \Rightarrow T(v) \geq E(v), \;\; \forall \, v \in U \tag{6.14}$$

$$\begin{aligned} s(v_i) = 1 \; \wedge s(v_j) = 1 \Rightarrow \\ T(v_j) - T(v_i) \geq ST(v_i, v_j) \vee \\ T(v_i) - T(v_j) \geq ST(v_j, v_i), \\ \forall \, v_i, \; v_j \in U, \; v_i \neq v_j \end{aligned} \tag{6.15}$$

The objective function (6.13) minimises the cost for this set of ships at this waypoint. Equation (6.14) specifies the earliest departure time for each vessel. Equation (6.15) ensures that there is sufficient separation time between successive ships to meet safety requirements.

6.4.2 Solving the MS-SOPTVD

The above formulation for the waypoint optimisation problem with time-varying draft constraints can be combined with the mathematical formulation for the Speed Optimisation Problem presented by Fagerholt, Laporte, and Norstad [2010] to create a mathematical model for the combined Multi-Ship Speed Optimisation Problem with Time-Varying Draft. However, the methods for solving the subproblem investigated in Chapter 4 become infeasible for problem sizes larger than around 16 ships, and would therefore be unable to solve realistic-sized instances of the MS-SOPTVD with resource conflicts at all waypoints considered at once.

Since the draft varies rapidly with time, the cost function in the waypoint optimisation subproblem needs to be modelled with a 5-minute time discretisation to ensure accurate costs for different drafts around the high tide. The speed optimisation part of the MS-SOPTVD, on the other hand, can result in long time horizons as the time taken to travel between waypoints can vary by several days for slow vs fast ship speeds. The 16-waypoint example problem has a maximum 30 day duration for the total voyage, which would result in very large domains for each ship's arrival time at each waypoint.

Also, the cost optimisation at each waypoint affects the range of valid times for each ship at adjacent waypoints, as well as the range of valid times for other ships at the same waypoint. The number of choice points in the resulting combined MS-SOPTVD model is thus around nWaypoints times the number of choice points for the single-port optimisation subproblem with the same number of ships.

All of these factors would make a complete CP model for the MS-SOPTVD infeasible. We therefore have to find alternative ways of solving the MS-SOPTVD.

The optimal solution to the single-ship Speed Optimisation Problem with fixed draft provides a lower bound on the cost for a set of ships with the same draft. Note that this approach to solving the Speed Optimisation Problem with Time-Varying Draft is not optimal, since the drafts are discretised, which results in an error that depends on the number of discretisations.

We solve the MS-SOPTVD by first solving the SOPTVD with discretised drafts for individual ships, ignoring resource constraints at waypoints. Our algorithm then calculates the travel costs and cost reduction for draft for each ship for 5-minute increments around the optimal arrival time at each waypoint, based on travel speeds to and from adjacent waypoints as well as the maximum allowable sailing draft for that ship at that time. This results in a cost function for each ship to use as input to the waypoint optimisation problem. Optimal speeds are then calculated for all ships at each waypoint by solving the CP model from Chapter 3 using the optimisation approaches investigated in Chapter 4, resulting in speeds that are close to optimal, and that result in no conflicts between ships at waypoints. Multiple iterations of this approach can be used to improve the combined schedule for the set of ships.

The algorithm for solving the MS-SOPTVD is described by Algorithm 8. Let U be the set of vessels indexed by v, and let the waypoints along the route be indexed by $p = 0, ..., P$, with D_p being the distance from waypoint $p - 1$ to p. Let $d_0(v)$ to $d_N(v)$ be the discretised draft values for vessel v. Let SOLVE_SDOPTDV be Algorithm 7 used to find optimal speeds $\{V_1(v), ..., V_P(v)\}$ for a single ship v with optimal draft $d_{opt}(v)$.

Let $V_{max}(v)$ be the maximum possible speed for ship v, and $V_{min}(v)$ be the minimum speed, used to calculate minimum fuel cost. Let $W_i(v)$ be the set of time windows at each waypoint for vessel v with draft i. Let $T_{start}(v)$ be the earliest time at which ship v can start sailing from the first waypoint. Let $S(v_i, v_j, p)$ be the minimum separation time between ships v_i, v_j at waypoint p, sailing in that order. Each waypoint may also have an earliest and latest sailing time $E(v, p), L(v, p)$ specified at that waypoint for each ship.

TOTAL_COST and COST_SAVINGS are specified by Equations (6.2) and (6.3) in Sections 6.1.1 and 6.1.2 respectively. WP_OPT is the algorithm used to optimise costs for a single draft-constrained waypoint, subject to separation time constraints, using the CP model described in Section 6.4.1.

Algorithm 8 returns the optimal draft $d_{opt}(v)$ for each ship v, as well as the optimal times $T_p(v)$ for each ship v at each waypoint p, and the optimal speeds between waypoints $V_p(v)$. The algorithm also returns the total shipping cost $C_{total}(v)$ for each ship, minus cost savings for draft above the minimum.

6.4.3 Potential Algorithm Improvements

Time Blocks of Ships

There are a number of improvements that can be made to Algorithm 8. First of all, when calculating optimal speeds for individual ships to solve the single-ship Speed Optimisation Problem, the most efficient speed will be similar for similar ships. As ships sailing with a deep draft need to sail close to the high tide, the resulting schedules have ships clumped together on high tides, possibly separated by ship speed.

Ships sailing on separate high tides are far enough apart that separation time constraints at waypoints do not need to be considered. This effect of ships clumping

Algorithm 8 SOLVE_MS-SOPTDV: Solving the Multi-Ship Speed Optimisation Problem with Time-Varying Draft

```
**Step 1:** Calculate optimal speeds and arrival times at waypoints for each ship, ignoring resource conflicts at waypoints.
**for** $v \in U$ **do**
    $d_{opt}(v)$, $\{V_1(v), ..., V_P(v)\}$ = SOLVE_SOPTDV$(W_i(v), D_1..D_P, d_0(v)..d_N(v))$
    $T_0(v) = T_{start}(v)$
    **for** $p \in [0, P]$ **do**
        $T_p(v) = T_{p-1}(v) + \frac{D_p}{V_p}$
    **end for**
**end for**
**for** $p \in [0, P]$ **do**
    **Step 2:** Get time range of ship arrivals at each waypoint, and extend time range if ships are too close together.
    $T_{min}(p) = \min_{v \in U} T_p(v)$
    $T_{max}(p) = \max_{v \in U} T_p(v)$
    $S_{max}(p) = (|U| - 1) \cdot \max_{v_i, v_j \in U} S(v_i, v_j, p)$
    **if** $T_{max}(p) - T_{min}(p) < S_{max}(p)$ **then**
        $S_{diff}(p) = S_{max}(p) - (T_{max}(p) - T_{min}(p))$
        $T_{max}(p) = T_{max}(p) + S_{diff}(p)$
        $T_{min}(p) = T_{min}(p) - S_{diff}(p)$
    **end if**
**end for**
```

Continued on next page.

```
for $p \in [0, P]$ do
    for $v \in U$ do
        Step 3: Calculate cost function for each ship at each time, including speed before and after, as well as cost savings for the maximum draft able to sail at that time. For infeasible times (when the ship can't sail even with minimum draft, or times that would require speeds above the maximum between adjacent waypoints) set the cost to BIGNUM.
        for $t \in [T_{min}(p), T_{max}(p)]$ do
            Step 3a: Calculate shipping cost to and from adjacent waypoints.
            $C(v,t) = 0$
            if $p \neq 0$ then
                $V_{before} = \frac{D_p}{t - T_{p-1}(v)}$
                $C(v,t) = C(v,t) +$ TOTAL_COST$(V_{before}, V_{min}(v), D_p)$
            end if
            if $p \neq P$ then
                $V_{after} = \frac{D_{p+1}}{T_{p+1}(v) - t}$
                $C(v,t) = C(v,t) +$ TOTAL_COST$(V_{after}, V_{min}(v), D_{p+1})$
            end if
            Step 3b: Adjust shipping cost to account for cost savings due to draft.
            for $i = 0$ to $N - 1$ do
                if $t \in W_i(v)$ and ($t \ni W_{i+1}(v)$ or $i = N - 1$) then
                    $C(v,t) = C(v,t) -$ COST_SAVINGS$(d_i(v))$
                end if
            end for
            Step 3c: Set cost function to BIGNUM for infeasible times and speeds.
            if $t \notin W_0(v)$ or $t < E(v,p)$ or $t > L(v,p)$ or $V_{before} > V_{max}(v)$ or $V_{after} > V_{max}(v)$ then
                $C(v,t) =$ BIGNUM
            end if
        end for
    end for
    Step 4: Optimise costs given separation time constraints and cost function.
    $T_p(v), d_{opt}(p,v) =$ WP_OPT$(C(v \in U, t \in [T_{min}(p), T_{max}(p)]), S(v_i, v_j \in U, p))$
end for
Step 5: Calculate optimal draft and speeds between waypoints for each ship, as well as cost of resulting schedule.
for $v \in U$ do
    $d_{opt}(v) = \min_{p \in [0,P]} d_{opt}(p,v)$
    $C_{total}(v) = 0$
    for $p \in [1, P]$ do
        $V_p(v) = \frac{D_p}{T_p(v) - T_{p-1}(v)}$
        $C_{total}(v) = C_{total}(v) +$ TOTAL_COST$(V_p(v), V_{min}(v), D_p)$
    end for
    $C_{total}(v) = C_{total}(v) +$ COST_SAVINGS$(d_{opt}(p,v))$
    $d_{opt}(v) = \min_{p \in [0,P]} d_{opt}(p,v)$
end for
```

together at high tides thus allows the schedule to be broken up into non-interacting time blocks for which the waypoint optimisation problem can be solved independently, significantly speeding up the calculation time. This improvement has been implemented, whereas the other two improvements below are suggestions for future work.

Infeasible Waypoint Optimisation

The waypoint optimisation problem may be infeasible for some waypoints if the single-ship speeds used as input to the waypoint optimisation problem are very close to a ship's maximum speed. If a ship travels close to its maximum speed both to and from a waypoint, it may not be able to move its arrival time at that waypoint in either direction without exceeding its maximum speed. Multiple ships travelling close to maximum speed and arriving at the waypoint closer together than allowed by the waypoint's separation time constraint can make the waypoint optimisation problem infeasible.

This does not occur in the problem instances considered in this chapter, as sailing at maximum speed is costly, and thus rarely selected by the speed optimisation. However, it may occur more frequently for problems that have very tight deadlines, requiring ships to sail close to their maximum speed. This infeasibility can be resolved by calculating multiple sets of speeds using the Shortest Path approach, and using alternative routes with slower speeds as a fallback option if required. However, for some problems, this may not be possible if the ship speeds are too tightly constrained and only fast speeds result in arrival at the last waypoint before the deadline.

Note that ships travelling slowly do not cause a similar infeasibility, since ships may wait at waypoints, so there is no minimum speed constraining the waypoint optimisation problem.

Global Optimisation

Algorithm 8 optimises speeds for each ship independently, then resolves conflicts at waypoints. However, this may result in a suboptimal global schedule, since each waypoint optimisation problem only affects the arrival time of the ships at one waypoint, whereas a globally optimal solution may require adjusting the times at several waypoints for some ships. Global optimisation approaches for this problem is a large potential area of future research.

6.4.4 Experimental Results

Schedules were calculated for problem instances containing 4, 6 and 8 ships, with voyages containing 4, 8, 12 and 16 waypoints – a total of 12 problem instances. All ships sail from the initial waypoint on the same high tide, as ships sailing on different high tides from the initial waypoint are likely to remain on different high tides for the entire voyage, and thus not cause resource conflicts at later waypoints. Modifying any of these problem instances to consider the same number of ships sailing on the

	Number of ships		
N	4	6	8
4	-2.09	-0.71	0.19
8	-0.61	0.08	5.20
12	0.38	3.66	2.95
16	0.39	3.24	6.66

Table 6.9: Percentage increase in cost for schedule with resource constraints at waypoints vs. lower bound ignoring resource conflicts computed with RSA.

same fixed route over multiple high tides will only make the problem less tightly constrained and easier to solve.

The first waypoint is modelled as a port, with ships departing from berths at different locations, and minimum separation times between ships that depend on the order in which the ships sail. Minimum separation times at other waypoints are independent of ship ordering, and range from 30 minutes or 1 hour.

Algorithm 8 was implemented in Python and solved using both the MW_RSA and Shortest Path solutions to the single-ship Speed Optimisation Problem as input. The waypoint optimisation subproblem was implemented in MiniZinc and solved using the G12 FD solver.

Cost Increase vs Lower Bound

Table 6.9 shows the percentage cost increase of the schedule with resource conflicts at waypoints against the lower bound cost computed by optimising the cost for each ship using the Multi-Window Recursive Smoothing Algorithm, without considering resource conflicts between ships at waypoints. For small numbers of ships and for short routes, the cost increase is small.

Waypoint optimisation may even reduce costs, since the single-ship speed optimisation algorithms only considers discretised drafts, whereas the multi-ship waypoint optimisation calculates the maximum draft to the nearest centimetre every five minutes, which may result in a finer draft resolution, thus reducing the discretisation error.

This reduction in discretisation error is what causes the negative "cost increase" numbers in Table 6.9. Since increasing the number of draft discretisations only increases calculation time linearly, it may be worth increasing the number of draft discretisations from 20 to every centimetre to eliminate this source of error.

Shortest Path vs RSA

Table 6.10 presents the percentage cost increase of using 1-ship solutions calculated using the Shortest Path approach with 60 time discretisations as input to the MS-SOPTVD algorithm against 1-ship solutions calculated with MW_RSA. This shows that the Recursive Smoothing Algorithm results in lower costs for many, but not all problems.

	Number of ships		
N	4	6	8
N	Cost Increase (%) SP vs. RSA		
4	-0.17	-0.30	-0.32
8	2.35	1.64	0.39
12	3.64	0.03	1.34
16	3.60	0.88	-1.58

Table 6.10: Percentage cost increase for MS-SOPTVD with initial 1-ship solutions calculated using 60-time-discretisation Shortest Path vs. RSA.

	Number of ships		
	4	6	8
N	Cost Improvement (%)		
4	16.45	13.15	15.01
8	9.04	12.22	9.69
12	9.02	10.54	11.08
16	7.60	9.61	10.21

Table 6.11: Percentage reduction in cost caused by considering resource conflicts at waypoints, rather than optimising speeds for individual ships using RSA, and waiting at waypoints in the event of resource conflicts.

One possible reason for the Shortest Path approach sometimes finding better combined solutions is that schedules computed for different ships have more variation with the Shortest Path approach than MW_RSA, since the Shortest Path approach introduces a time discretisation error which may vary for different ships. This results in ships sailing further apart on schedules computed using the Shortest Path approach compared to MW_RSA. The Shortest Path schedules spread ships out over multiple high tides, resulting in ships being able to sail with higher drafts due to having fewer resource conflicts with other ships. This theory is supported by the fact that schedules computed using the Shortest Path approach result in ships being split into more time blocks on average than schedules computed using MW_RSA.

Cost Improvement of Considering Multiple Ships

Table 6.11 shows the percentage reduction in cost caused by considering resource conflicts between ships at waypoints, rather than optimising speeds for individual ships and having ships wait in the event of multiple ships arriving at a waypoint at once.

As expected, cost improvements are greatest for problems with more ships, as more resource conflicts arise. However, in the examples we considered, the cost improvement is also highest for the shortest route, since the most tightly constrained waypoint occurred on this route, resulting in ships having to wait for longer time periods. Longer routes had many under-constrained waypoints where ships only

	Time Windows for Draft					
	2 hrs	4 hrs	6 hrs	8 hrs	10 hrs	12 hrs
N	Cost Improvement (%)					
4	26.84	17.82	18.62	21.89	29.72	35.14
8	34.04	20.03	20.09	19.72	23.17	28.39
12	36.52	25.58	25.23	24.80	26.93	31.80
16	34.05	23.37	22.79	22.57	24.53	28.71

Table 6.12: Percentage reduction in cost for 8-ship problems caused by considering draft constraints and resource conflicts at waypoints vs. ignoring draft windows in speed optimisation for drafts with different minimum time windows at waypoints. Averaged over 8 ships.

had to wait for a very short time, resulting in lower percentage cost increases.

The results in Table 6.11 clearly show that optimising schedules for individual ships, even with consideration of time-varying draft constraints, is insufficient when there are 4 or more ships sailing along the same route at the same time, since there is a significant cost to not considering interactions between multiple ships arriving at waypoints simultaneously.

Cost Improvement vs. Ignoring Draft Constraints

Table 6.12 compares the cost of the schedules computed by Algorithm 8 with time-varying draft constraints and resource conflicts and waypoints against the cost of calculating optimal speeds without considering draft constraints, sailing with unconstrained drafts, and waiting at waypoints in the event of arrival outside the draft windows, or in the event of multiple ships arriving close together.

These results show that considering draft constraints and resource conflicts at waypoints when calculating optimal sailing speeds may reduce costs by up to 40% for 8 ships sailing along the same route. This shows that incorporating draft constraints and resource constraints at waypoints into the ship speed optimisation problem may have significant economic benefits for routes with multiple draft-constrained waypoints.

6.5 Routing and Speed Optimisation

6.5.1 Problem Description

Norstad, Fagerholt, and Laporte [2011] presented the Tramp Ship Routing and Scheduling Problem with Speed Optimisation (TSRSPSO) and used the single-ship Speed Optimisation Problem as a subproblem in the local search approach used to solve it.

This section extends the TSRSPSO with time-varying draft restrictions to form the Routing and Scheduling Problem with Speed Optimisation and Time-Varying Draft (RSPSOTVD), and uses a similar approach to Norstad, Fagerholt, and Laporte [2011] to solve this problem, with the Speed Optimisation Problem with Time-Varying Draft (SOPTVD) being used as a subproblem instead of the SOP.

The RSPSOTVD has the following features. Items 1 and 3-6 are repeated from the problem description for the SOPTVD problem description in Section 6.2.1.

1. Let S be a set of ships, each with a range of possible drafts and speeds, and a cargo capacity per centimetre of draft within the allowable range.

2. Let P be a set of ports, each of which is primarily an import or export port. Each ship is assumed to be able to dock at every port, subject to draft constraints.

3. Each port and waypoint has time-dependent draft constraints that vary with tide, waves, current, and other environmental conditions.

4. Draft constraints can be converted to multiple time windows for each possible ship draft. These time windows may vary with ship draft (if they are entirely dependent on draft constraints) or may be static, for example if the earliest arrival time at the destination is determined by needing to have enough storage space to unload the ship's cargo.

5. Loading and unloading time at ports are ignored, similarly to Fagerholt, Laporte, and Norstad [2010]. However, ships may wait at waypoints if required, to ensure that minimum speeds do not make time windows infeasible.

6. At each port and waypoint, there may be a minimum separation time between ships, to avoid ships sailing too close together in a narrow draft-restricted waterway.

7. Let C be a set of cargoes, each with a corresponding origin and destination port, as well as pickup and delivery windows, total tonnage, and revenue per tonne. Each cargo is assumed to be transportable on every ship, subject to capacity and draft constraints. Cargoes may need to be split over several ships; each ship can only carry one type of cargo at any given time.

8. The route from the origin to the destination port for each cargo may travel past a number of waypoints. Both ports and waypoints have time-varying draft constraints.

9. The objective is to maximise the total profit: the sum of the revenues for delivered cargoes, minus the sum of the total costs.

A similar method to Norstad, Fagerholt, and Laporte [2011] is used to solve the RSPSOTVD. This approach solves the routing problem using a multi-start local search heuristic, which generates a number of initial solutions, then iteratively improves them by applying several local search operators. Whenever an operator is applied, the route for any affected ship is re-evaluated by solving the corresponding Speed Optimisation Problem. The method investigated in this chapter extends the approach presented by Norstad, Fagerholt, and Laporte [2011] by considering time-varying draft restrictions in the Speed Optimisation Problem for each ship's route, and by considering resource conflicts at waypoints for ships sailing past the same waypoint within a short time window.

6.5.2 Our Approach

The Tramp Ship Routing and Scheduling Problem with Speed Optimisation (TSR-SPSO) was solved by Norstad, Fagerholt, and Laporte [2011] using a multi-start local search heuristic, which generates a number of initial solutions, then iteratively improves them by applying several local search operators. Whenever an operator is applied, the route for any affected ship is re-evaluated by solving the corresponding Speed Optimisation Problem. This section extends their approach by solving the Speed Optimisation Problem with Time-Varying Draft (SOPTVD) instead for each ship's route, and by considering resource conflicts at waypoints for ships sailing past the same waypoint within a short time window.

The SOPTVD considers time-varying restrictions on draft, but a fixed draft still needs to be chosen in order to calculate how much cargo is carried on each ship. The algorithm presented in this chapter only considers fixed drafts which give each ship 2-hour windows at each waypoint, since this was found to be close to the optimal draft in experiments with the SOPTVD and MS-SOPTVD presented earlier in this chapter. Extending the algorithm to attempt to add extra cargo to ships if there is any cargo left unscheduled is a possible avenue for future work.

Speeds only need to be calculated for ships that have been modified by the operator, and speed optimisation results are cached to avoid redundant calculations. Schedules are also checked for feasibility before performing speed optimisation, by ensuring that there is enough time for all cargoes to be picked up and delivered within their allowable windows, assuming ships sail at their maximum speed.

After a combined schedule for all ships is calculated, the algorithm identifies conflicts at waypoints by finding any ships that are scheduled to arrive at the same waypoint closer together than the minimum separation time at that waypoint. Each waypoint conflict is then resolved by solving a Waypoint Cost Optimisation Problem for ships at that waypoint that are in conflict with each other.

The algorithm ignores loading and unloading times, similarly to Fagerholt, Laporte, and Norstad [2010] and Norstad, Fagerholt, and Laporte [2011]. Including consideration of loading and unloading times for our problem would be slightly more complex than for the problems presented by these authors, since ships would participate in waypoint conflicts during both arrival and departure; however, this could be implemented similarly to the way incoming and outgoing ships are handled in the BPCTOP in Chapter 3, with each ship being involved in two different Waypoint Cost Optimisation Problems, with different sequence-dependent separation times, to take into account different separation times between incoming and outgoing ships. Berth allocation may also need to be taken into account; however, an initial simplification may be to consider all berths as identical, so there are only four different sequence-dependent separation times that need to be considered: incoming followed by incoming, outgoing followed by outgoing, incoming followed by outgoing, and vice versa.

The local search starts by generating a number of initial solutions by assigning some cargoes to random ships, then adding the remaining cargoes to the ship that can deliver them at minimum cost. After initial solutions are generated, these solutions are then improved by iteratively applying a number of local search operators:

1. One_Resequence: move one cargo assigned to ship v to the best time in the schedule of ship v.
2. Reassign: move one cargo to the best time in the schedule of the best ship.
3. Two_Interchange: swap two cargoes between ships u and v, to the best times in their schedules.
4. Two_Resequence: move two cargoes assigned to ship v to the best times in the schedule of ship v
5. Three_Interchange: Swap three cargoes between three ships to the best times in their schedules.

For more details on the local search heuristic, including discussions of ratios of random to optimal cargoes in the initial solutions, frequency of applying each local search operator, etc. see Norstad, Fagerholt, and Laporte [2011] and Brønmo et al. [2007].

6.5.3 Resolving Conflicts at Waypoints

Our approach to resolving conflicts at waypoints only considers one waypoint at a time, and only affects adjacent legs in the ship's route. This approach is suboptimal, and results could be improved by a global solution that affects a larger amount of the schedule; however, this first attempt at considering conflicts at waypoints is still preferable to ignoring waypoint conflicts entirely and having ships wait for each other when arriving at a waypoint too close together.

To resolve a waypoint conflict, a Waypoint Cost Optimisation Problem (WCOP) is created for ships participating in the conflict. This problem is a simplified version of the Bulk Port Cargo Throughput Optimisation Problem (BPCTOP) presented in Chapter 3, which maximises cargo throughput at a port, subject to time-varying draft restrictions, minimum separation time constraints, and other constraints such as constraints on tugs, which may apply at some ports. To convert this problem to a Waypoint Cost Optimisation Problem (WCOP), we maximise profit instead of cargo throughput, where profit is calculated based on the revenue per tonne of cargo, as well as the cost of travel to adjacent waypoints. The WCOP is formulated as a Constraint Programming model, presented below.

Let U be the set of vessels to be scheduled, indexed by v. Let $[1, T_{max}]$ be the range of discretised time indices. Each vessel v has an earliest departure time $E(v)$, and a profit function $P(v,t)$ which specifies the profit for vessel v at each time t, where $P(v,t)$ is 0 for any times t when the ship v cannot sail. $ST(v_i,v_j)$ specifies the minimum separation time between the sailing times of every ordered pair of ships v_i, $v_j \in U$. The binary decision variable $s(v)$ specifies whether ship v is included in the schedule. Let $T(v) \in [1, T_{max}]$ be the decision variable specifying the scheduled sailing time for vessel v. The binary variable $sb(v_i,v_j)$ – SailsBefore(v_i,v_j) – is true iff vessel v_i sails earlier than vessel v_j, defined by Eq. (6.16).

$$sb(v_i,v_j) = 1 \leftrightarrow (T(v_i) < T(v_j) \wedge sb(v_j,v_i) = 0), \quad \forall\, v_i,\ v_j \in U; v_i \neq v_j \tag{6.16}$$

The model for the Waypoint Cost Optimisation Problem is formulated as follows:

$$\text{maximise} \sum_{v \in U} s(v) \cdot P(v,T(v)) \tag{6.17}$$

$$\text{s.t.} \qquad s(v) = 1 \Rightarrow T(v) \geq E(v), \ \ \forall\, v \in U \tag{6.18}$$

$$sb(v_i,v_i) = 0, \ \ \forall\, v_i \in U \tag{6.19}$$

$$\begin{aligned} s(v_i) = 1 \ \wedge s(v_j) = 1 \Rightarrow \\ (sb(v_i,v_j) = 1 \Rightarrow T(v_j) - T(v_i) \geq ST(v_i,v_j)), \\ \forall\, v_i,\ v_j \in U \end{aligned} \tag{6.20}$$

The objective function (6.17) maximises the total profit for the set of ships. Constraint (6.18) specifies the earliest departure time for each vessel. Equation (6.19) specifies that no ship sails before itself. Equation (6.20) ensures that there is sufficient separation time between every pair of successive ships to meet safety requirements at waypoints.

We create a Waypoint Cost Optimisation Problem for each waypoint conflict, and solve it to find times for each ship that maximise the profit for travelling between adjacent waypoints. We then adjust the speeds between adjacent waypoints, and the arrival time at the affected waypoint, to create a schedule that resolves the conflict.

In a possible future extension to this algorithm, the profit function could also take into account the ability to load extra cargo onto ships by moving some ships to times where the allowable draft at the waypoint is higher. This would allow the WCOP to improve the solution to the overall schedule by loading extra cargo onto ships that have extra capacity. However, this presents some difficulties, since this would affect the available sailing windows at neighbouring waypoints, possibly making the scheduled sailing times infeasible, so for the moment we have not allowed the Waypoint Cost Optimisation Problem to load extra cargo onto ships above the draft with 2-hour windows originally scheduled. This improvement would be worth investigating in future work.

The profit function is defined by Equation (6.21),

$$\begin{aligned} P(v,t) = {} & \text{revenue}([D(v,t) - D_{empty}(v)] \cdot C(v)) \\ & -\text{fuel\_cost}(V_{prev}(v,t), V_{min}(v), d_{prev}) \\ & -\text{fuel\_cost}(V_{next}(v,t), V_{min}(v), d_{next}) \\ & \forall\, v \in V,\ t \in T \end{aligned} \tag{6.21}$$

Where $D(v,t)$ is the maximum allowable draft for vessel v at this waypoint at time t, $D_{empty}(v)$ is the draft for vessel v sailing empty, $C(v)$ is the tonnage per centimetre of draft for vessel v, $V_{prev}(v,t)$ and $V_{next}(v,t)$ are the speeds that vessel v must sail at for the previous and next legs respectively in order to arrive at this waypoint at time t, and d_{prev} and d_{next} are respectively the distances of the next and previous legs on this ship's route. fuel_cost(V, V_{min}, d) is defined by Equation (6.1) in Section 6.1.1, where $V_{min}(v)$ is the minimum speed for ship v, used to calculate minimum fuel cost.

Limitations

One limitation of this approach is that it does not yet consider variable drafts for each ship for each segment of each route. All five methods compared in this section fix the draft for each pickup-delivery voyage for each ship by initially finding the draft that gets at least 2-hour windows at every high tide for every waypoint along the route, and then increasing the draft by up to 20cm, as long as the draft still gets at least some windows at all waypoints along the route. In practice, this results in waypoints still having many windows for each ship, since 2-hour windows for every high tide would include neap tides, when the high tide is significantly lower than average.

This is a significant limitation, since this algorithm only considers one draft for each ship and each cargo, whereas it may be possible for the ship to sail with more cargo if there is sufficient tide height at waypoints, or it may be possible for ships to sail with lower speed if the draft were decreased to increase the available sailing windows. Since allowing variable draft would affect the routing problem by reducing or increasing the amount of cargo available for other ships, finding optimal drafts for each cargo is a significant challenge, and is left for future work.

6.5.4 Experimental Results

The local search heuristic was implemented in Python and solved using the RSA algorithm for the speed optimisation for individual ships. The WCOP was implemented in MiniZinc and solved using the G12 FD solver. For ships that were not in conflict, a Python script was used to choose locally optimal point of the profit function for each waypoint. The same parameter settings were used for the multi-start local search heuristic as those found by Brønmo et al. [2007] to find the best routing solutions. All experimental tests were performed on an Intel i7-930 quad-core 2.80 GHz processor and 12.0 GB RAM.

	Routing Method				
	MaxSpd	ServiceSpd	Waiting	Draft	WpOpt
NCargoes (actual)	10	9	9	9	9
Unassigned Cargo (tonnes)	7580	25660	0	0	0
Revenue ($)	2509473	2429891	2547500	2547500	2547500
Cost ($)	1092919	931496	817119	741429	739518
Profit ($)	1416554	1498394	1730380	1806070	1807981
% Profit Diff vs ServiceSpd	-5.46	0	15.48	20.53	20.66

Table 6.13: Comparing different approaches for the instance with 4 multi-ship cargoes, 4 ships and a 4-port map.

Problem instances were generated using 4 constructed maps with 4, 8, 12 and 16 ports. Ports were randomly assigned as either import or export ports, and sets of 5, 10, 15 or 20 cargoes were automatically generated, with random origins, destinations, and realistic but randomly varying tonnages and revenues. The fleet contains 4, 8, 12 or 16 ships, of realistic sizes, but with randomly generated starting locations, and with a small amount of random variation in fuel usage and maximum speed. Cargoes are generally larger than the capacity of a ship, and therefore need to be split across several ships, with possibilities for cost optimisation by splitting cargoes so as to reduce unused ship capacity. The planning horizon varies from 10 to 160 days, depending on the number of ships and cargoes. Cargo pickup and delivery windows are chosen randomly based on the size of the planning horizon and the size of the cargo. There are 64 problem instances in total.

Several different solution strategies are compared to evaluate the impact of taking time-varying draft constraints and conflicts at waypoints into account in a ship routing problem with speed optimisation. Each problem was solved with each of the following methods, with 10 random restarts of the local neighbourhood search.

1. Waiting: optimise speeds ignoring draft constraints, and have ships wait for draft windows to open and for other ships to sail upon arrival at waypoints.

2. MaxSpd: all ships sail at their maximum speed, and wait for draft windows to open upon arrival at waypoints.

3. ServiceSpd: all ships sail at their service speed (3 knots below their maximum speed), and wait for draft windows to open upon arrival at waypoints.

4. Draft: optimise speeds with draft constraints taken into account, but ignore waypoint conflicts, and have ships wait for each other at waypoints if required.

5. WpOpt: routing with speed optimisation, draft constraints and waypoint conflict resolution.

Detailed Results for One Problem

Table 6.13 presents a detailed analysis of revenue, cost and profit for one problem with a 4-port map, 4 ships, and 4 multi-ship cargoes (each cargo needs to be split over several ships). This problem was solved with all five methods.

The results show that, as expected, optimising speeds results in significantly reduced costs and higher profit compared to both the fixed-speed methods, MaxSpd and ServiceSpd. Among the methods that optimise speeds, considering draft constraints (Draft) also significantly improves results compared to optimising speeds while ignoring draft constraints and waiting for the high tide (Waiting). However, this improvement is not as high as that obtained by optimising speeds in the first place, possibly because the methods that use speed optimisation (Waiting, Draft and WpOpt) all converge on the same routing solution with the same cargoes loaded on the same ships, only the sailing speeds of the ships differ.

The fixed-speed methods not only have higher fuel costs, they also fail to load some cargoes. Note that NCargoes refers to the number of loads carried; the problem itself contains 4 cargoes which are larger than one ship load, so they each need to be split up among several ships. The MaxSpeed method finds a solution where ships carry 10 cargo loads, but a small amount of cargo is left unassigned. This may be because in order to fit the last small amount of cargo onto a ship, an earlier load of this cargo would need to be carried by the largest available ship, which also has a higher fuel usage, and is therefore more expensive at maximum speed. All other methods schedule ships to sail with a lower speed, which makes it cost-effective to have this cargo carried by the larger ship. In the case of the ServiceSpd method, one cargo gets left behind possibly because ships need to sail fast in order to deliver this cargo within the time windows, and ships sailing at service speeds are unable to fit this cargo into the schedule.

The details of what routes and cargo assignments are chosen by each schedule are clearly affected by the random variation in ship sizes and fuel usage constants in our generated data set. It would therefore be very interesting to test this algorithm on an industrial data set in future. However, the more general conclusion that can be drawn from these results is that the increased flexibility arising from using speed optimisation rather than fixed speeds results in a better choice of routes and cargo assignments, and that considering draft restrictions explicitly in speed optimisation significantly reduces the fuel usage and total cost.

The improvement obtained by considering waypoint conflicts is extremely small compared to both the improvement obtained by optimising speeds, and by considering draft constraints. There are few waypoint conflicts in this problem, so the benefit from considering waypoint conflicts is small. All routing problem instances had fewer waypoint conflicts and fewer ships participating in each conflict on average than the previous problem, since ships start out at different locations and are much less likely to arrive at any waypoint at the same time. Cargoes being randomly allocated to ports is likely to result in fewer waypoint conflicts than would occur in a real problem, since in reality, cargo origins and destinations are not evenly distributed. In Australia, for example, the top five bulk export ports – Port Hedland, Dampier, Newcastle, Hay Point and Gladstone – account for over 80% of total export tonnage, with the 40+ smaller ports contributing less than 20% [Ports Australia, 2012].

	Routing Method			
	MaxSpd	Waiting	Draft	WpOpt
NCargoes	% Profit Diff vs ServiceSpd			
5	-8.36	8.96	15.67	17.70
10	-9.91	14.98	18.61	23.72
15	-9.40	17.83	29.11	30.80
20	-10.36	19.63	26.95	27.30
All	-9.51	15.34	22.59	24.34

Table 6.14: Average percentage difference in profit for different strategies, compared against sailing at service speed, averaged over 16 problem instances for each number of cargoes.

Speed Optimisation

Table 6.14 presents the percentage increase in profit of each approach compared to sailing at service speed. The percentage increase in profit is presented for problems with 5, 10, 15 and 20 large multi-ship cargoes, averaged across 16 instances of each problem with 4, 8, 12 and 16 ships, and using the 4, 8, 12, and 16-port maps.

As found by Norstad, Fagerholt, and Laporte [2011], sailing at maximum speed (MaxSpd) reduces performance compared to sailing at service speed (ServiceSpd). All methods that optimise speeds perform significantly better on average compared to fixed-speed approaches. Even when draft constraints are ignored and ships wait for the high tide (Waiting), this still improves profit by 15.34% on average compared with sailing at the fixed service speed and waiting for high tides.

Draft Constraints

The Draft method – optimising speeds with draft constraints taken into account, but ignoring waypoint conflicts – performs significantly better than all of the ServiceSpd, MaxSpd and Waiting methods. The improvement over sailing at service speed is 22.59% on average, compared with the 15.34% improvement found by optimising speeds without considering draft. This is similar to the improvement observed for the fixed-route speed optimisation problem with time-varying draft constraints, compared to ignoring draft constraints and waiting for ships with draft windows of around 4 hours. It is clear that considering draft constraints in speed optimisation can lead to significant reduction in costs for draft-constrained ships, due to reducing the waiting time required, and reducing fuel costs due to ships not needing to sail faster to catch up after waiting for a high tide.

Waypoint Constraints

By contrast, the improvement provided by considering waypoint constraints is much smaller for the combined routing and scheduling problem than was previously observed for the fixed-route speed optimisation problem. For the WpOpt approach to the combined routing and scheduling problem, we observe only a 2% improvement on average over the Draft approach of considering draft constraints and ignoring waypoint conflicts, as compared to the approximately 11% average for the fixed-route problems investigated earlier. In a combined routing and scheduling problem, ships arriving at waypoints at the same time is much less likely than for several ships travelling along the same route starting at the same time, so there are much fewer conflicts, and their impact on the schedule is less significant. Because waypoint optimisation is also the slowest part of the calculation, a good enough tradeoff of schedule quality vs. calculation speed may be found by applying draft constraints alone, without considering waypoint conflicts.

6.6 Summary

This chapter introduced time-varying draft restrictions and resource conflicts between ships at waypoint into two new problems – the Speed Optimisation Problem (SOP) for a fixed ship travelling along a fixed route, and the Tramp Ship Routing and Scheduling Problem with Speed Optimisation (TSRSPSO) initially solved by Norstad, Fagerholt, and Laporte [2011].

For a set of ships travelling along a fixed route with draft-constrained waypoints, consideration of draft and resource constraints was found to significantly reduce shipping costs compared to ignoring draft constraints in scheduling and waiting upon arrival at waypoints if required. For the routing and scheduling problem, speed optimisation with draft constraints improved the resulting schedules by an average of about 22% compared to the traditional method of sailing at a fixed service speed. By comparison, the method presented by Norstad, Fagerholt, and Laporte [2011] of speed optimisation without draft constraints produced an improvement of 15% compared to sailing at service speed. Considering waypoint conflicts only produced a small additional improvement of about 2% on top of considering draft constraints, since there were much fewer conflicts in the routing problem compared to the problem of several ships sailing along the same route.

This chapter, along with the three preceding chapters, have clearly shown that there is a large potential benefit of considering time-varying restrictions on ship draft in maritime transportation optimisation problems which involve draft-constrained ports and waterways. However, the introduction of these time-varying draft restrictions significantly increases the complexity of the problem. Time-varying action costs are found in other problems both in the maritime industry and beyond, so the following chapter looks beyond maritime transportation problems with time-varying draft, and investigates how problems in other fields deal with time-varying action costs, and what generalisations can be drawn from these approaches.

Chapter 7

Generalisation: Scheduling with Time-Varying Action Costs

7.1 Introduction

A key feature of maritime transportation problems involving time-varying restrictions on ship draft, such as the Bulk Port Cargo Throughput Optimisation Problem (BPCTOP) investigated in Chapters 3, 4 and 5, as well as the problems with speed optimisation and time-varying draft restrictions investigated in Chapter 6, is that these problems have an objective that is a sum of time-varying costs or values for each task. This objective function increases the size and complexity of these problems, making them more challenging to solve; however, problems with similar objective functions have been investigated in related fields, so there may be scope for generalisation of approaches between these problems. Section 7.2 reviews a number of scheduling and routing problems with time-varying action costs from outside the field of maritime transportation, and discusses techniques which may be generalisable between applications.

Typical approaches to solving these types of problems include Mixed Integer Programming (MIP) and local search, or more complex solve-and-improve approaches, which find an initial solution for a simplified problem and improve it using the objective function and constraints of the full problem. However, Constraint Programming (CP) solvers have been found to be faster than MIP for some problems, including the BPCTOP as discussed in Chapter 4. CP is also a very flexible method that can be used to model a wider variety of constraints than MIP, which is limited to linear constraints. A number of recent approaches have combined CP with other techniques such as large neighbourhood search for vehicle routing [Kilby and Verden, 2011], SAT [Ohrimenko, Stuckey, and Codish, 2009] and MIP [Achterberg, 2009] in order to combine the flexibility of CP with fast algorithms for specific problems.

CP approaches may therefore be worth investigating for other problems that have traditionally been modelled with MIP. This chapter compares CP vs MIP for another problem with time-varying action costs in maritime transportation – the Liner Shipping Fleet Repositioning Problem (LSFRP). This chapter also investigates the effectiveness of Lazy Clause Generation (LCG) [Ohrimenko, Stuckey, and Codish, 2009], discussed in Chapter 2, for both the BPCTOP and the LSFRP, as LCG was found to be more effective than traditional finite domain CP solvers on a number of scheduling problems [Schutt et al., 2012; Feydy and Stuckey, 2009].

When faced with a new problem with time-dependent task costs, or when trying to add such costs to an existing problem, it is not clear which approaches will yield good results. This chapter aims to provide guidance for researchers and practitioners dealing with problems with such costs. This is done through an investigation of several key problems in the literature and the introduction of models for two maritime transportation problems. To this end, this chapter contains the following novel components: (1) a review of methods for solving problems with time-dependent task costs, (2) a CP model of the LSFRP that outperforms all previous approaches, (3) a comparison of LCG and solve-and-improve approaches on the LSFRP, (4) a solve-and-improve approach for the BPCTOP, (5) and a vehicle routing based modeling of the BPCTOP that finds high quality solutions even on large BPCTOP instances.

The work presented in this chapter is joint work with Kevin Tierney, who conducted all the experiments on the LSFRP. Parts of this research have been published in a joint conference paper [Kelareva, Tierney, and Kilby, 2013a] and a longer journal paper containing all the work in this chapter has also been submitted for publication [Kelareva, Tierney, and Kilby, 2013b].

7.2 Problems with Time-Varying Action Costs

This section presents a review of a number of scheduling and routing problems with time-dependent action costs, including net present value maximization in project scheduling [Russell, 1970] and satellite imaging scheduling [Lin et al., 2005; Wolfe and Sorensen, 2000]. This section also reviews approaches that have been used to solve these problems. Approaches used to deal with time-varying action costs in problems introduced earlier in this thesis are also summarised, including vehicle routing with soft time windows [Sexton and Choi, 1985; Figliozzi, 2012]; ship routing and scheduling with soft time windows [Fagerholt, 2001; Christiansen and Fagerholt, 2002]; and ship speed optimisation [Fagerholt, Laporte, and Norstad, 2010; Norstad, Fagerholt, and Laporte, 2011].

A variety of optimisation techniques exist for solving routing and scheduling problems, but it is often unclear which technique will offer the best results given a problem with time-dependent task costs. This section gives a detailed overview of a range of approaches that are available in the literature before several of these methods are investigated in further detail in the rest of this chapter. In doing so, this chapter aims to provide some intuition into which methods perform best on different types of time-dependent task cost problems.

Note that there are other problems besides the ones presented here where time-dependent task costs are present, such as the airline disruption problem Rosenberger, Johnson, and Nemhauser [2003]; Kohl et al. [2007], and the problem of optimisation of electricity usage whether in the home (e.g., Agnetis et al. [2013]) or in a data center (e.g., Rao et al. [2010]). We have selected several key problems where a variety of solution methods have been proposed. We use this information to try to draw conclusions on which directions seem to be the most promising, however this section is not intended to be an exhaustive review of all possible problems with time-varying costs, since a very wide range of such problems exists.

7.2.1 Satellite Imaging Scheduling

One scheduling problem with time-dependent quality functions for each action is the problem of satellite imaging scheduling, where the goal is to schedule a set of observations of an Earth observing satellite. Each observation may only be performed for a specified period of time during the satellite's orbit, and the quality of the observation drops off to zero for times before and after the peak quality window [Wolfe and Sorensen, 2000]. A satellite scheduling problem also includes resource constraints, as each satellite can only perform one observation at a time, and sequence-dependent setup times, as different observations may require different orientations of the satellite.

Satellite scheduling approaches handle the time-varying image quality objective in a number of ways. One approach used by Yao et al. [2010] and Wang et al. [2007] is to simplify the image quality function by converting it to hard time windows when a "good enough" image could be obtained. This simplifies the problem significantly, but does not account for the fact that images scheduled closer to the centre of the time window will be of higher quality.

Another approach presented by Lin et al. [2005] is to first find a feasible solution by decomposing the problem into independent subproblems using Lagrangian relaxation, and then to use heuristics to improve the solution quality by minor changes in the schedule. This approach is suboptimal, but produces better quality solutions than just using feasible time windows, with solutions being within 2% of optimality.

The quality functions in satellite imaging scheduling may be further complicated by the fact that it is possible to partially satisfy an imaging request, by taking images that cover only a portion of the requested area. The possibility of partial rewards is handled by an objective function which gives the highest weighting to complete requests, and only a low weighting to partial requests [Lemaître et al., 2002; Lin et al., 2005].

Wolfe and Sorensen [2000] compared three algorithms for satellite imaging scheduling with time-dependent task quality functions: priority dispatch, look ahead, and genetic algorithms (GA). The priority dispatch approach also used a solve-and-improve method, first building a schedule by allocating jobs in order of priority, then improving the schedule by adjusting job start and end times using a hill-climbing search to improve schedule quality. The look ahead algorithm produced a 12% average improvement in schedule quality over the priority dispatch algorithm, by using the quality function earlier in the process, to allocate jobs immediately to the peak of their quality window, rather than using the quality function only to improve the initial schedule. The GA approach produced further improvements in schedule quality, but at the expense of a significant reduction in speed.

Simplifying the quality function to hard time windows is the simplest and most efficient approach for decreasing the solution time, but does not account for quality at all. Improving the quality with small local changes to the schedule can produce schedules that are within 2% of optimality, with only a small time cost [Lin et al., 2005]. More complex quality optimisation approaches produce better schedules, but with increasingly slower calculation times [Wolfe and Sorensen, 2000], and an approach that finds an exact optimal solution would introduce a much larger time cost.

Optimal approaches may be worthwhile for problems where even a small deviation from the optimum has a significant impact, such as the Bulk Port Cargo Throughput Optimisation Problem (BPCTOP), but close-to-optimal approaches such as that presented by Lin et al. [2005] suffice for applications such as satellite scheduling where a small deviation from the optimum has less impact.

7.2.2 Project Scheduling with Net Present Value

Another scheduling problem with time-dependent action costs is the problem of maximising the net present value (discounted cash flow) in project scheduling, first introduced by Russell [1970]. In standard project scheduling, the objective is usually to minimise makespan (total time). For the alternative objective of maximising net present value, each activity has an associated positive or negative cash flow, which is discounted over time according to a given discount rate.

Project scheduling may also include resource constraints and precedence constraints between actions. The objective function component for each action is monotonic, unlike the satellite scheduling image quality function, which has a plateau of peak quality, and drops off before and after.

The problem of maximising net present value in project scheduling without resource constraints can be solved in polynomial time by transforming to a linear program [Grinold, 1972]. However, the resource-constrained project scheduling problem with net present value (RCPSPNPV) is NP-complete.

The RCPSPNPV can be solved by relaxing the problem to remove resource constraints, and using the resource-unconstrained solution as an upper bound in the search. A review of both the resource-constrained and unconstrained max-NPV problem is presented by Vanhoucke, Demeulemeester, and Herroelen [1999].

Vanhoucke, Demeulemeester, and Herroelen [2001] solved the RCPSPNPV using a depth-first branch-and-bound algorithm with the resource-unconstrained solution as the upper bound. The search algorithm scheduled positive-cashflow activities as early as possible in the process, and negative-cashflow activities as late as possible, thus focussing the search on good quality solutions.

Another technique that has been used to solve this problem is CP with Lazy Clause Generation (LCG), which combines constraint programming with learning where the search has previously failed [Ohrimenko, Stuckey, and Codish, 2009]. LCG was found by Schutt et al. [2012] to be more efficient at solving this problem than basic constraint programming. CP and LCG are very general approaches that can be used for any constrained optimisation problem, which makes CP and LCG promising candidates for generalising to other scheduling or routing problems with time-varying task costs.

7.2.3 Liner Shipping Fleet Repositioning

Another recently introduced scheduling and routing problem with time-varying action costs in maritime transportation is the Liner Shipping Fleet Repositioning Problem (LSFRP) [Tierney et al., 2012a]. The LSFRP concerns itself with the repositioning of container ships in a liner shipping network when routes in the network are added or changed. The LSFRP was introduced in Chapter 2.

The LSFRP is a relatively new problem to the optimisation community, having not been mentioned in any of the most comprehensive surveys of maritime research [Christiansen et al., 2007, 2013; Christiansen, Fagerholt, and Ronen, 2004].

The LSFRP was first solved by Tierney et al. [2012a], who compared two planning approaches and a MIP model for solving the problem. First, the LSFRP was modelled using PDDL 2.1, a modeling language for automated planning problems (see Fox and Long [2003]), and solved using the POPF planner [Coles et al., 2010b]. Although POPF found solutions, it could not solve the LSFRP to optimality. The authors further introduced a MIP based on an activity-flow approach, in which the various activities that can be undertaken by vessels are modeled as nodes, and arcs represent which activities can be ordered after one another. The authors then introduced Linear Temporal Optimisation Planning (LTOP), which uses temporal planning to build optimisation models, and solves these using an optimisation version of partial-order planning based on branch-and-bound. LTOP outperformed both the traditional planning approach and the MIP on a limited dataset of instances.

Both LTOP and MIP were able to find optimal solutions for problem sizes with up to three ships. These are realistic problem sizes similar to those used by shipping lines. As with the BPCTOP, there is a high motivation to find the optimal solutions, even if solving the problem to optimality is more difficult, since the cost of repositioning a single ship can be hundreds of thousands of dollars [Tierney et al., 2012a].

Tierney and Jensen [2012] and Tierney et al. [2013] solve a version of the LSFRP including cargo flows, which the version we address in this paper lacks. In these two papers, the authors sacrifice the flexibility of choosing amongst several phase-in and phase-out opportunities in order to lower the overall problem size enough to include cargo flows. Therefore the problem addressed in this chapter is an equally important, parallel problem to the LSFRP with cargo flows. The version of the LSFRP presented in this chapter is based on activities that avoid disrupting cargo flows, even though they are not taken explicitly into account.

Note that the version of the LSFRP investigated in this thesis (as well as in the above papers) includes restrictions on phase-in actions undertaken by vessels to ensure that vessels phase in to their goal service such that the liner shipping structure of the service is maintained. To maintain the service structure, a periodic (usually weekly) schedule must be created. To ensure this is the case, the models presented in this chapter require that vessels all phase-in to the same port in a sequential, week-by-week fashion. This restriction could be relaxed in practice, which may be worth investigating in future work. All papers on the LSFRP to date have this restriction, with the exception of Tierney et al. [2013] and Tierney and Jensen [2011], which interpret this restriction slightly differently – ships must phase in so as to arrive at a given port on a given date, but they may phase in earlier in the service as long as they arrive at the port on the specified date.

7.2.4 Routing and Scheduling with Soft Time Windows

Chapter 2 included a section on the Vehicle Routing Problem with Soft Time Windows (VRPSTW), which involves time-varying action costs in the form of penalties for early and late delivery of each cargo outside the customer's preferred time window. Since both vehicle and ship routing and scheduling problems with soft time windows were covered in Chapter 2, this section only summarises the types of cost functions considered in each approach.

The VRPSTW has been extensively researched in the field of vehicle routing; however, soft time windows are comparatively rare in the ship routing and scheduling literature. This section only gives a few examples of vehicle routing problems with soft time windows. Many VRPSTWs include penalties for both early and late arrival, for example Figliozzi [2010]; Fan et al. [2011]; Kilby and Verden [2011]. Others only consider penalties for late delivery, eg. Qureshi, Taniguchi, and Yamada [2009]. Some approaches such as Figliozzi [2012] also consider time-varying travel times, which can be used to model different travel times during peak hour or roads with speed limits that vary by time of day.

In ship routing and scheduling, Fagerholt [2001] and Fagerholt [2000] compared several different penalty functions for deliveries outside each customer's preferred time window, which modelled different real-world scenarios, e.g. when a customer gets a fixed discount for a delivery outside the preferred time, or when the discount depends on the lateness of the delivery. Halvorsen-Weare and Fagerholt [2013] considered soft time windows with penalties for both early and late delivery, to enforce a "fairly evenly spread" requirement on a set of deliveries in a maritime inventory routing problem. Karlaftis and Kepaptsoglou [2009] considered penalties for delay only, so the cost function for each action was monotonic. Gatica and Miranda [2011] considered a problem with hard time windows; however their approach could easily be extended to model soft time windows with penalties for early and late delivery, or late delivery only. Their network-based model would also be flexible enough to model cost functions with more complex shapes, though the time discretisation used in their network-based model would limit the accuracy with which cost functions can be modelled.

Christiansen and Fagerholt [2002] considered penalties for ships arriving at port close to weekends, to avoid the risk of long delays caused by some ports closing on weekends, so the problem involved multiple time windows and had a penalty function which increased towards the end of each window. Ship routing and scheduling problems with speed optimisation, such as Norstad, Fagerholt, and Laporte [2011] and the last problem investigated in Chapter 6 also have time-dependent costs for each action, with the cost increasing with sailing speed.

The Bulk Port Cargo Throughput Optimisation Problem (BPCTOP) could also be considered an instance of a ship scheduling problem with soft time windows – the window of peak draft around high tide could be considered the preferred time for each ship, with the draft function modelled as a non-linear penalty function specifying the reduction in cargo amount carried if a ship sails before or after the peak high tide window. The shape of the objective function in the BPCTOP is similar to the quality function for satellite image scheduling, which is the inverse of the cost function for the VRPSTW with penalties for both early and late arrival, since the maximum draft for each ship peaks around high tide, and drops off before and after. However, VRPSTW cost functions are usually modelled as linear outside the preferred time window, whereas the cost function for the BPCTOP is more complex, as it depends on the tide, waves and other environmental conditions, which are not entirely linear, and can be quite complex at ports with complex interactions between the waves, tide and currents.

As with satellite scheduling, the VRPSTW and BPCTOP involve a tradeoff between solution quality and scalability. For handling draft restrictions in maritime optimisation problems, traditional ship scheduling techniques with constant draft are the most scalable, but do not consider time variation in the allowable amount of cargo. Ship scheduling with time windows, such as the approaches presented by Fagerholt [2001] and Christiansen and Fagerholt [2002], could be used to consider time windows of good enough" draft, at the cost of increased complexity. This would allow ships to be scheduled even if they cannot sail at all phases of the tidal cycle. However, this approach would still produce schedules with less than the maximum loading for ships that are large enough to be able to load extra cargo to the peak of the tide. The approaches presented in Chapter 4 produce the best quality schedules, but make even a relatively simple problem at only one port quite complex to solve to optimality. Since each extra centimetre of draft can increase the cargo carried on a Capesize bulk carrier by 130 tonnes [Port Hedland Port Authority, 2011b], and therefore profit for the shipper by up to $10,000 [Curtis, 2012], there is a high incentive to find optimal solutions.

7.2.5 Summary

Each of the problems we reviewed in this section involves time-dependent action costs; however, other constraints as well as the shapes of these objective functions vary between problems. Most problem types include resource constraints and setup times between tasks; however, not all problems include sequence-dependent setup times. Some problems such as ship and vehicle routing problems also include optional tasks, whereas for other problems such as project scheduling, the set of tasks is fixed. Finally, some of the problems also include routing constraints, whereas others are purely scheduling problems. Table 7.1 summarises additional constraints and objective function shapes for problems reviewed in this chapter. Some of these papers were previously discussed in Chapter 2; however, this chapter focusses on their handling of time-varying objectives and related constraints.

Some approaches to dealing with time-dependent action costs, such as Rakke et al. [2011], involve simplifying the objective function by modelling it as constant over time, or by reducing the time-varying action costs to hard "good enough" time windows [Yao et al., 2010]. However, in this chapter, we will focus on approaches that do consider time-dependent objectives, whether by optimising for the time-varying objective directly, such as the approaches used in Chapter 4 to solve the Bulk Cargo Port Throughput Optimisation Problem (BPCTOP), or by solving a simplified constant-time problem initially and then improving the solution with respect to the time-varying objective, such as the approach investigated by Lin et al. [2005] for satellite imaging scheduling.

The choice of an appropriate algorithm needs to be informed by how important it is to get a good quality solution for the given application. In optimising the cargo throughput for a bulk export port, problem sizes are small – there may only be up to 10 ships sailing on a tide – but solution quality is very important, as every centimetre of extra draft for a ship results in a $10,000 increase in profit. In the Liner Shipping Fleet Repositioning Problem (LSFRP), varying the speed of a single vessel can result in savings of tens or even hundreds of thousands of dollars. In satellite image scheduling, however, problems sizes may be very large with thousands of tasks to be scheduled [Wolfe and Sorensen, 2000], but the consequences of a suboptimal solution may be less expensive, so a method that produces solutions within 2% of optimality such as that presented by Lin et al. [2005] is sufficient.

As there are many similarities between scheduling and routing problems involving cost functions that are a sum of time-varying task costs, some techniques may be generalisable from one problem type to others. The next three sections discuss techniques that have been successfully used for one of the problem types discussed in this section that may be generalisable to other problems, and apply these technique to the BPCTOP and LSFRP.

7.3 LSFRP

This section presents a CP model for the Liner Shipping Fleet Repositioning Problem (LSFRP), and compares it against the MIP and automated planning models introduced by Tierney et al. [2012a] on a dataset of 39 instances based on data from an industrial collaborator. This section begins with an overview of the previous approaches, followed by the CP model and experimental results, showing that CP greatly outperforms previous approaches. The LSFRP was described in detail in Chapter 2, so the description of the problem is not repeated here.

Paper	Problem	Objective Function	Seq.Dep.Setups	Routing	Optional
Wolfe and Sorensen [2000]	Satellite	Peaked	Yes	No	Yes
Lin et al. [2005]	Satellite	Peaked	Yes	No	Yes
Wang et al. [2007]	Satellite	TimeWindows	Yes	No	Yes
Yao et al. [2010]	Satellite	TimeWindows	Yes	No	Yes
Schutt et al. [2012]	Project	Monotonic	No	No	No
Grinold [1972]	Project	Monotonic	No	No	No
Vanhoucke, Demeulemeester, and Herroelen [2001]	Project	Monotonic	No	No	No
Chapters 3, 4 and 5	BPCTOP	Peaked	Yes	Yes	Yes
Fagerholt [2001]	ShipSTW	Peaked	No	Yes	No
Christiansen and Fagerholt [2002]	ShipSTW	Monotonic, Multiple windows	Yes	Yes	No
Norstad, Fagerholt, and Laporte [2011]	ShipSpeedOpt	Monotonic	No	Yes	No
Bausch, Brown, and Ronen [1998]	ShipSpeedOpt	Monotonic	No	Yes	No
Brown, Graves, and Ronen [1987]	ShipSpeedOpt	Monotonic	No	Yes	No
Tierney et al. [2012a]	LSFRP	Monotonic	No	Yes	Yes
Tierney and Jensen [2011]	LSFRP	Monotonic	No	Yes	Yes
Figliozzi [2010]	VRPSTW	Peaked	Yes	Yes	No
Figliozzi [2012]	VRPSTW	Peaked	Yes + time dep.	Yes	No
Qureshi, Taniguchi, and Yamada [2009]	VRPSTW	Monotonic (late penalties only)	Yes	Yes	No
Fan et al. [2011]	VRPSTW	Peaked	Yes	Yes	No
Kilby and Verden [2011]	VRPSTW	Peaked	Yes	Yes	Yes

Table 7.1: Characteristics of problems reviewed in the literature.

7.3.1 Automated Planning and LTOP

Automated planning is used to model problems where it is difficult to select and sequence activities in order to achieve specific goals starting from an initial state. Automated planning attempts to find a sequence of activities, which is generally a subset of the overall set of activities, that modify a state representation such that certain goals are achieved.

Activities are represented by actions that can only be applied in states that satisfy their precondition. When applied, actions change a state, thus reassigning some of the state variables representing the state. We refer the reader to Nau, Ghallab, and Traverso [2004] for the details of automated planning. Automated planning offers a distinct advantage over other modeling approaches in the way actions correspond directly to real-world activities, making models easy to understand and interpret for domain experts.

LTOP uses a type of automated planning called partial-order planning [Penberthy and Weld, 1992] to build and search through optimization models that involve continuous time, metric quantities, and a complex mixture of action choices and ordering constraints. LTOP fundamentally diverges from classical automated planning approaches by introducing two sets of modeling variables that decouple the planning problem from the optimization model. Thus, the optimization model is not tightly bound to the semantics of actions. Actions are merely used as handles to optimization components that are combined to complete optimization models using partial-order planning. In other words, LTOP builds optimization models with the help of automated planning, i.e., the focus of LTOP is on the optimization model of a problem, whereas classical planning approaches include optimization components within actions merely as a byproduct of the overall planning process.

LTOP is built on a state variable representation of propositional STRIPS planning [Fikes and Nilsson, 1971]. LTOP utilizes partial-order planning [Penberthy and Weld, 1992], and extends it in several ways. First, an optimization model is associated with each action in the planning domain. This allows for complex objectives and cost interactions that are common in real world optimization problems to be easily modeled. Second, instead of focusing on simply achieving feasibility, LTOP minimizes a cost function. Finally, begin and end times can be associated with actions, making them durative. Such actions can have variable durations that are coupled with a cost function. We refer to Tierney et al. [2012a] for a formalization of the LTOP approach.

7.3.2 LSFRP LTOP Model

This section describes the LTOP model from Tierney et al. [2012a]; we refer readers to Tierney et al. [2012b] for more details about the automated planning PDDL domain that was compared against LTOP, since LTOP greatly outperforms this approach on the LSFRP. The LTOP model consists of several core actions: `phase-out`, `phase-in`, `sail`, `sail-on-service`, `sail-with-equipment` that are based around keeping track of the *state* of each vessel being repositioned. A vessel can be in one of three states:

initial, *transit*, or *goal*, which describe a vessel being on its initial service, performing a repositioning, or having reached its goal service, respectively.

We create phase-out actions at every port call along each vessel's initial service. Phase-out actions may only be applied for a particular vessel when it is in its initial state, meaning it has not yet begun its repositioning. Applying a phase-out action transitions a vessel to a state of transit. Phase-in actions are created at each port on the goal service in each week of the planning period. Vessels must be in a transit state to use a phase-in action. The effect of a phase-in action indicates that a vessel has reached the goal service, which satisfies the goal state of the model.

While a vessel is in transit (i.e., repositioning), it may use any sailing, SoS or sail equipment action to move between ports. Sailing actions are created between all ports in the model, except for sailings with phase-out ports as the destination, as once a vessel leaves the phase-out service it is not allowed to go back. We create SoS actions for each SoS specified by the repositioning coordinator, and equipment sailing opportunities between all ports with equipment surpluses and those with equipment demands.

Our actions describing sailing, SoS opportunities, and equipment sailings are *durative* actions, meaning they take place over a time period that is specified through the preconditions of the action. In the case of SoS actions, the amount of time they require is fixed based on the start port and end port of the action. However, sailing with and without equipment has a variable duration that must be between the minimum and maximum speed of a vessel. Phase-out and phase-in actions are instantaneous actions that have no duration.

We model sailing costs by setting the fixed cost of sailing (and sailing with equipment) actions to the maximum possible cost of sailing between two ports, i.e., sailing at maximum speed. We then subtract an amount from the action cost based on the duration of the sailing, meaning longer sailing subtract more from the cost, making them cheaper.

Hotel costs, i.e. the fixed hourly cost of operating a vessel, are modeled by associating each vessel with two optimization variables representing the begin and end time of the hotel period for that vessel. The hotel period is constrained to be the duration of the vessel between its phase-out and phase-in, minus the duration of any SoS opportunities used. In this way, no extra actions are needed to model the hotel cost, which is a key time-dependent task cost of the LSFRP. Each action that is added to an LTOP plan updates the bounds of the hotel cost, assisting LTOP in its search for the optimal solution.

All repositioning plans contain, at minimum, a phase-out and a phase-in action for each vessel. Sailings, SoS opportunities, and sail equipment actions can be placed between the phase-out and phase-in in order to bring the vessel to the goal service.

7.3.3 MIP Model

Tierney et al. [2012a] also presented a MIP model of the LSFRP and compared it against LTOP, to investigate how LTOP performed against more traditional approaches. The MIP model considers the activities that a vessel may undertake and connects activities based on which ones can feasibly follow one another temporally. The structure of the LSFRP is embedded directly into the graph of the MIP, meaning that it is unable to model general automated planning problems as in Kautz and Walser [1999] and Van Den Briel, Vossen, and Kambhampati [2005]. Note that, unlike LTOP, the MIP is capable of handling negative activity costs. This MIP model is summarised below.

Since the vessel state in fleet repositioning is relatively simple, encompassing where a vessel is and whether it has begun its repositioning or not, there is not an exponential growth in the number of graph nodes as there would be in many planning problems if they were modeled using a graph.

Given a graph $G = (A, T)$, where A is the set of actions (nodes), and T is the set of transitions, with $(a, b) \in T$ iff action b may follow action a, let the decision variable $y_{a,b} \in \{0, 1\}$ indicate whether or not the transition $(a, b) \in T$ is used or not. The auxiliary variable $w_a = \sum_{(a,b)\in T} y_{a,b}$ indicates whether action a is chosen by the model, and $x_a^s, x_a^e \in \mathbb{R}^+$ are action a's start and end time, respectively. Finally, the variables h_v^s and h_v^e are the start and end time of the hotel cost period for vessel v.

Each action $a \in A$ is associated with a fixed cost, $c_a \in \mathbb{R}$, a variable (hourly) cost, $\alpha_a \in \mathbb{R}$, and a minimum and maximum action duration, d_a^{min} and d_a^{max}. For some actions a, there is a fixed time t_a at which they must start, if they are performed. The set of such actions is denoted by $A^t \subseteq A$. The use of a particular action may exclude the use of other actions. These exclusions are specified by $\eta : A \to 2^{|A|}$. There are also n sets of mutually exclusive actions, given by $\mu : \{1, \ldots, n\} \to 2^{|A|}$. We differentiate between phase-out and phase-in actions for each vessel using the sets $A_v^{po}, A_v^{pi} \subseteq A^t$, respectively, and let $A' = A \setminus \cup_{v\in V}(A_v^{po} \cup A_v^{pi})$ be all actions that are not related to the phase-out or phase-in. Finally, let $c_v^H \in \mathbb{R}^+$ represent each vessel's hourly hotel cost.

The upper bound on the difference between the end and start of two actions is given by $M_{a,b}^y$. The upper bound on the start of a vessel's hotel period, and the lower bound on the end of the vessel's hotel period, are given by M_v^s and m_v^e respectively.

Parameters We now summarize the parameters and variables used in our MIP model for easy reference. The model uses the following parameters:

$G = (A, T)$	Graph of nodes A (actions) and arcs T (transitions).
V	Set of vessels.
A_v^{po}	Set of phase-out actions for vessel $v \in V$.
A_v^{pi}	Set of phase-in actions for vessel $v \in V$.
A^t	Set of actions that can only start at fixed times.
t_a	Starting time of action $a \in A^t$.
c_a	Fixed cost of action a.
α_a	Variable cost of action a.
c_v^H	Hotel cost of vessel $v \in V$ per hour.
d_a^{min}, d_a^{max}	Minimum and maximum duration of action a.

$M^y_{a,b}$	Maximum time difference between the start of actions a and b.
M^s_v, m^e_v	Upper and lower bound of the start and end of vessel $v \in V$'s hotel period.
$\mu(a)$	Set of actions that are mutually exclusive with a.
$\eta(a)$	Set of actions that are excluded by a (but not necessarily mutually exclusive).

Variables The model uses the following variables:

$y_{a,b} \in \{0,1\}$	Indicates whether transition $(a,b) \in T$ is used.
$w_a = \sum_{(a,b)\in T} y_{a,b}$	Auxiliary variable indicating whether action a is used or not.
$x^s_a, x^e_a \in \mathbb{R}^+$	The start and end time of action $a \in A$, respectively.
h^s_v, h^e_v	The start and end time of the hotel cost period for vessel $v \in V$.

Objective and constraints The objective and constraints are as follows:

$$\min \sum_{a\in A} (c_a w_a + \alpha_a(x^e_a - x^s_a)) + \sum_{v\in V} c^H_v (h^e_v - h^s_v) \tag{7.1}$$

$$\text{s.t.} \sum_{\{(a,b)\in T \mid b\in A\setminus A^{po}_v\}} y_{a,b} = 1 \qquad \forall a \in \bigcup_{v\in V} A^{po}_v \tag{7.2}$$

$$\sum_{(a,b)\in T} y_{a,b} = \sum_{(b,c)\in T} y_{b,c} \qquad \forall b \in A' \tag{7.3}$$

$$\sum_{(a,b)\in T} y_{a,b} \le 1 \qquad \forall b \in A' \tag{7.4}$$

$$x^e_a - x^s_b \le M^y_{a,b}(1 - y_{a,b}) \qquad \forall (a,b) \in T \tag{7.5}$$

$$x^s_a \le x^e_a \qquad \forall a \in A \tag{7.6}$$

$$d^{min}_a w_a \le x^e_a - x^s_a \le d^{max}_a w_a \qquad \forall a \in A \tag{7.7}$$

$$x^s_a = t_a w_a \qquad \forall a \in A^t \tag{7.8}$$

$$h^s_v + M^s_v w_a \le M^s_v + x^e_a \qquad \forall a \in \bigcup_{v\in V} A^{po}_v \tag{7.9}$$

$$h^e_v + m^e_v w_a \ge m^e_v + x^s_a \qquad \forall a \in \bigcup_{v\in V} A^{pi}_v \tag{7.10}$$

$$\sum_{a\in\mu(i)} w_a \le 1 \qquad \text{for } i = 1,2,\dots n \tag{7.11}$$

$$|\eta(a)| w_a + \sum_{b\in\eta(a)} w_b \le |\eta(a)| \qquad \forall a \in A \tag{7.12}$$

The objective, (7.1), sums the fixed and variable costs of each action that is used along with the hotel cost for each vessel. The single unit flow (i.e., node disjoint) structure of the graph is enforced in constraints (7.2) which start the flow of vessels through the graph, ensuring that every vessel transitions out of its phase-out.

Constraints (7.3) are flow balance constraints that ensure vessels sail traverse path in the graph. Constraints (7.4) limit the incoming number of vessels to any activity to one, meaning that only a single vessel may undertake any particular activity. Constraints (7.5) enforce the ordering of transitions between actions, preventing the end of one action from coming after the start of another if the edge between them is turned on. Action start and end times are ordered by (7.6), and the duration of each action is limited by (7.7). Actions with fixed start times are bound to this time in (7.8). Constraints (7.9) and (7.10) connect the hotel start and end times to the time of the first and last action, respectively. Note that the objective forces h_v^s and h_v^e as close together as possible, and that the big/little-Ms are required because each action has a different start time. The mutual exclusivity of certain sets of actions is enforced in constraints (7.11). Finally, constraints (7.12) prevents actions from being included in the plan if they are excluded by an action that was chosen. These constraints are primarily used to ensure the block phase in structure necessary to have a weekly temporal spacing of vessels. When a phase-in is chosen for a vessel, phase-ins that could not possibly be used with it are disabled. For example, in a 3 vessel problem, any phase-in more than 2 weeks later from a particular phase-in is disabled if a vessel uses that phase-in.

7.3.4 A Novel CP Model

Despite the success of LTOP and MIP in solving LSFRP instances, the question remains as to how these approaches perform versus Constraint Programming techniques. To find out, we create a CP model of the LSFRP, detailed below. This section as well as the later LSFRP results in this chapter are joint work with Kevin Tierney; all LSFRP experiments were carried out by Kevin Tierney.

Let O_v be the set of possible phase-out actions for the vessel v, and let P be the be the set of possible phase-in ports for the new service. The decision variable $\rho \in P$ is the phase-in port for all vessels. The decision variables $w_v \in \{1, \ldots, W\}$ represent the phase-in week for each vessel v, where W is the number of weeks considered in the problem. For each vessel v, let $q_v \in O_v$ be a decision variable specifying the phase-out action (port and time) used for that vessel.

For each vessel v and phase-out action o, the function $t_{out}(v, o)$ specifies the phase-out time for that action. Similarly, $t_{in}(p, w)$ specifies the phase-in time for a vessel phasing in at port p in week w. The function $C(v, o, p, w)$ specifies the cost for vessel v using the phase-out action o, and phasing in at port p in week w, with -1 as a flag that indicates vessel v cannot phase in at port p in week w if it phased out using action q (for example, if action q starts too late for vessel v to reach port p in time). The dependent variable c_v specifies the cost for vessel v when the vessel sails directly from the phase-out port to the phase-in port. For each vessel v, $C_H(v)$ specifies its hourly hotel cost, and h_v is the duration of the hotel cost time period (from the phase-out to the phase-in).

Each SoS opportunity is split into several *SoS actions*, where each SoS action represents starting the SoS at a different port on the SoS service. SoS opportunities save money by allowing vessels to sail for free between two ports, however a cost for transshipping cargo at each side of the SoS is incurred. Let S be the set of available SoS actions and S' be the set of SoS opportunities. The decision variable $s_v \in S$ specifies the SoS action used for each vessel v, with 0 being a flag indicating that vessel v does not use an SoS action. For each SoS action $s \in S$, the function $y : S \to S'$ specifies which SoS opportunity each SoS action belongs to, with $y(s) = 0$ being a flag that specifies that the vessel is not using any SoS opportunity.

In order to use an SoS opportunity, a vessel must sail to the starting port of the SoS opportunity before a deadline, and after using the SoS, it sails from the end port at a pre-determined time to the phase-in port. The function $C^{to}(v, s, o)$ specifies the cost of vessel v using SoS action s, phasing out at phase-out o going to the SoS action, and $C^{from}(v, s, p, w)$ is the cost of vessel v to sail from SoS action s to phase in port p in week w, with -1 as a flag that indicates that this combination of vessel, SoS action, phase-in port and week is infeasible. The dependent variables σ_v^{to} and σ_v^{from} specify the SoS costs for vessel v for sailing to and from the SoS, respectively. The function $A(v, s)$ specifies the cost savings of vessel v using SoS action s, and the dependent variable σ_v^{dur} specifies the SoS cost savings for vessel v on the SoS.

Let Q be the set of *sail-equipment* (SE) opportunities, which are pairs of ports in which one port has an excess of a type of equipment, e.g. empty containers, and the other port has a deficit. Since we do not include a detailed view of cargo flows in this version of the LSFRP, SE opportunities save money by allowing vessels to sail for free between two ports as long as the vessel sails at its slowest speed. The cost then increases linearly as the vessel sails faster. Let the decision variable $e_v \in E$ be the SE opportunity undertaken by vessel v, with $e_v = 0$ indicating that no SE opportunity is used. Let the decision variables d_v^{to}, d_v^{dur} and d_v^{from} be the duration of vessel v sailing to, during, and from an SE opportunity.

The functions $C^{to}(v, e, o)$, $C^{dur}(v, e)$ and $C^{from}(v, e, p, w)$ specify the fixed costs of sailing to, utilizing, and then sailing from SE opportunity e, where v is the vessel, o is the phase-out port/time, p is the phase-in port and w is the phase-in week. Together with the constant α_v, which is the variable sailing cost per hour of vessel v, the hourly cost of sailing can be computed. This is necessary since SE opportunities are not fixed in time and, thus, must be scheduled. Let the dependent variables $\lambda_v^{to}, \lambda_v^{dur}$ and λ_v^{from} be the fixed costs sailing to, on and from an SE opportunity. Additionally, let $\Delta_{min}^{to}(v, e, o)$, $\Delta_{min}^{dur}(v, e)$ and $\Delta_{min}^{from}(v, e, p, w)$ be the minimum sailing time of v before, during and after the SE opportunity and $\Delta_{max}^{to}(v, e, o)$, $\Delta_{max}^{dur}(v, e)$ and $\Delta_{max}^{from}(v, e, p, w)$ be the maximum sailing time of v before, during and after the SE opportunity.

In this version of the LSFRP, the chaining of SoS and SE opportunities is not allowed, meaning each vessel has the choice of either sailing directly from the phase-out to the phase-in, undertaking an SoS, or performing an SE. The decision variable $r_v \in \{\texttt{SoS}, \texttt{SE}, \texttt{SAIL}\}$ specifies the type of repositioning for each vessel v, where v utilizes an SoS opportunity, SE opportunity, or sails directly from the phase-out to the phase-in, respectively. The CP model is formulated as follows:

$$\min \sum_{v\in V} \big(C_H(v)\,(t_{in}(\rho,w_v) - t_{out}(v,q_v)) + c_v + \sigma_v^{from} + \sigma_v^{dur} + \sigma_v^{to} + \lambda_v^{to} + \lambda_v^{dur} + \lambda_v^{from} + \alpha_v(d_v^{to} + d_v^{dur} + d_v^{from})\big) \tag{7.13}$$

$$\text{s. t.} \quad \texttt{alldifferent}(w_v), v \in V \tag{7.14}$$

$$\max_{v\in V} w_v - \min_{v\in V} w_v = |V| - 1 \tag{7.15}$$

$$\texttt{alldifferent\_except\_0}(s_v), v \in V \tag{7.16}$$

$$\texttt{alldifferent\_except\_0}(y(s_v)), v \in V \tag{7.17}$$

$$r_v = \texttt{SAIL} \to c_v = C(v,q_v,\rho,w_v),\ \ \forall v \in V \tag{7.18}$$

$$r_v = \texttt{SAIL} \to (s_v = 0 \wedge y(s_v) = 0 \wedge \sigma_v^{dur} = 0 \wedge \sigma_v^{from} = 0 \wedge \sigma_v^{to} = 0 \wedge e_v = 0 \wedge \lambda_v^{to} = 0 \wedge \lambda_v^{dur} = 0 \wedge \lambda_v^{from} = 0),\ \ \forall v \in V \tag{7.19}$$

$$r_v = \texttt{SoS} \to s_v > 0 \wedge y(s_v) > 0 \wedge \sigma_v^{dur} = -A(v,s_v) \wedge \sigma_v^{from} = C^{from}(v,s_v,\rho,w_v) \wedge \sigma_v^{to} = C^{to}(v,s_v,q_v),\ \ \forall v \in V \tag{7.20}$$

$$r_v = \texttt{SoS} \to c_v = 0 \wedge e_v = 0 \wedge \lambda_v^{to} = 0 \wedge \lambda_v^{dur} = 0 \wedge \lambda_v^{from} = 0,\ \ \forall v \in V \tag{7.21}$$

$$s_v > 0 \vee y(s_v) > 0 \to r_v = \texttt{SoS},\ \ \forall v \in V \tag{7.22}$$

$$\texttt{alldifferent\_except\_0}(e_v),\ \ \forall v \in V \tag{7.23}$$

$$r_v = \texttt{SE} \to e_v > 0 \wedge \lambda_v^{to} = C^{to}(v,e_v,q_v) \wedge \lambda_v^{dur} = C^{dur}(v,e_v) \wedge \lambda_v^{from} = C^{from}(v,e_v,\rho,w_v), \forall v \in V \tag{7.24}$$

$$r_v = \texttt{SE} \to s_v = 0 \wedge y(s_v) = 0 \wedge \sigma_v^{dur} = 0 \wedge c_v = 0 \wedge \sigma_v^{from} = 0 \wedge \sigma_v^{to} = 0, \forall v \in V \tag{7.25}$$

$$e_v > 0 \to r_v = \texttt{SE} \tag{7.26}$$

$$\Delta_{min}^{to}(v,e_v,q_v) \le d_v^{to} \le \Delta_{max}^{to}(v,e_v,q_v),\ \ \forall v \in V \tag{7.27}$$

$$\Delta_{min}^{dur}(v,e_v) \le d_v^{dur} \le \Delta_{max}^{dur}(v,e_v),\ \ \forall v \in V \tag{7.28}$$

$$\Delta_{min}^{from}(v,e_v,\rho,w_v) \le d_v^{from} \le \Delta_{max}^{from}(v,e_v,\rho,w_v),\ \ \forall v \in V \tag{7.29}$$

$$\sigma_v^{to}, \sigma_v^{from}, c_v \ge 0,\ \ \forall v \in V \tag{7.30}$$

The objective function (7.13) minimises the sum of the hotel costs and repositioning action costs minus the cost savings for SoS actions for the set of vessels. Constraints (7.14) and (7.15) specify that the vessels must all phase in to the new service on different, successive weeks. Constraints (7.16) and (7.17) specify that all vessels using SoS actions must use different actions and action types. `alldifferent_except_0` is a global constraint that requires all elements of an array to be different, except those that have the value 0.

Constraints (7.18) and (7.19) set the costs for a vessel if it uses a `SAIL` repositioning, and ensures that the SoS/SE actions and costs are set to 0, as they are not being used. Constraints (7.20) and (7.21) specify that if vessel v uses an SoS (`SoS`) repositioning action s_v, then its repositioning cost is equal to the costs for sailing to and from that SoS action based on the phase-out action, phase-in port and week, minus the cost savings $A(v, s_v)$ for that SoS action. In addition, the normal repositioning cost c_v and the sail equipment action for that vessel are set to 0. Redundant constraints (7.22) are also added to reinforce that the repositioning type be set correctly when an SoS is chosen.

Constraints (7.23) ensure that no two vessels choose the same SE action (unless they choose no SE action), and constraints (7.24) and (7.25) bind the costs of the sail equipment action to the dependent variables if an SE is chosen, as well as set the costs of a direct sailing and SoS opportunities for each vessel to 0. The redundant constraints (7.26) ensure that the repositioning type of vessel v is correctly set if an SE action is chosen. The minimum and maximum durations of the parts of the SE (sailing to the SE from the phase-out, the SE itself, and sailing from the SE to the phase-in) are set in constraints (7.27), (7.28) and (7.29). Constraint (7.30) requires that all SoS actions and phase-out/phase-in combinations must be valid for each vessel (i.e. transitions with -1 costs must not be used).

7.3.5 LSFRP CP Results

To compare the above CP model for the LSFRP against an earlier MIP model and against the LTOP planner [Tierney et al., 2012a], all three models are used to solve the 11 AC3 problem instances from Tierney et al. [2012a], as well as 27 new instances based on a real-world scenario provided by an industrial collaborator. The problem instances contain up to 9 vessels, with varying SoS and sail-with-equipment opportunities that may be used to reduce repositioning costs.

The LSFRP CP model was formulated in the MiniZinc 1.6 modelling language [Nethercote et al., 2007, 2010], and solved using the G12 finite domain solver [Wallace, 2009]. The CP is compared against a MIP model and the LTOP planner [Tierney et al., 2012a], both using CPLEX 12.4. All problems were solved to optimality. Note that in the CP model for MiniZinc we had to add constraints on the maximum duration of SE actions, as well as a constraint on the maximum sum of the objective, in order to prevent integer overflows. These constraints do not cut off any valid solutions from the search tree. Since MiniZinc does not support floating point objective values, the MiniZinc model is a close approximation of the true objective.

We also used several search annotations within MiniZinc to help guide the solver to a solution. The first is to branch on the type of repositioning, r_v, before other variables. We thereby attempt to first find an SoS option for each vessel, then search through SE options, and finally SAIL options. This search order was the most efficient for the most complex models that include both SoS and SE opportunities. This is because SE constraints are more complex than SoS constraints, so searching

SE options first is more time consuming for models that contain both SoS and SE opportunities.

For instances with SE opportunities, we also add a search annotation to branch on the SE opportunity, e_v, using the "indomain_split" functionality of MiniZinc, which excludes the upper half of a variable's domain. Both annotations use a first failure strategy, meaning the variable the solver branches on is the one with the smallest domain.

Table 7.4 compares the run times of the CP model against the MIP model and LTOP, all of which solve to optimality[1]. We ran the CP model with search annotations using a search order of SAIL/SoS/SE (CP–A), with search annotations and the SoS/SE/SAIL ordering (CP–AO), as well as with only the redundant constraints and using the solver's default search (CP–R), and using the SoS/SE/SAIL ordering with redundant constraints (CP–AOR). We label instances with the format *G_S_O_ce*, where *G* is the name of the goal service in Maersk Line's network, *S* is the number of ships, *O* is the number of SoS opportunities, *c* indicates that there are cabotage restrictions, and *e* indicates whether the instance contains any equipment.

CP–AOR outperforms LTOP on all instances in the dataset, and the MIP on all but one instance in the dataset. This contrasts with the results in Kelareva, Tierney, and Kilby [2013a], where CP–AOR outperformed the MIP on all AC3 instances. This is because we are using CPLEX 12.4, and Kelareva, Tierney, and Kilby [2013a] uses CPLEX 12.3. This shows that using a well-supported solver like CPLEX (or the G12 CP solver) can provide advantages over less developed techniques, like LTOP, in that new versions can provide significantly faster solution times without any model changes. The average solution time required by CP–AOR is also significantly less than LTOP and the MIP both in terms of the arithmetic and geometric means, with CP–AOR requiring, on average, only 59% the time of the MIP, and 46% of the time of LTOP in terms of the arithmetic mean, respectively.

We provide results for the various parameterisations of our CP approach in order to determine the source of its good performance. No configuration on its own is able to dominate the others across all instances. Combining annotations, ordering and redundant constraints into CP–AOR provides better average performance than any single configuration, as well as finds more optimal solutions than any other configuration. Interestingly, several of the best CPU times are not found by CP–AOR, but by individual configurations, such as on FM3_4_4ce, FM3_6_6e, FM3_6_6ce and TP7_6_4. In fact, CP-R finds an optimal solution to TP7_6_4 not found by any other solver, indicating that an instance-specific solving approach using machine learning, like in Kadioglu et al. [2010] could be a good approach on problems like the LSFRP. The implementation of such a system is left for future work.

Our CP model comes with two limitations. The first limitation is the model's lack of flexibility. A natural extension to this model would be to allow for the chaining of SoS and SE opportunities, which is easy to do in both the LTOP and MIP models, due to automated planning's focus on actions, and our MIP model's focus on flows.

[1]Experiments used Intel Core i7-2600K 3.4GHz processors with a maximum of 4GB per execution.

Problem	LTOP	MIP	CP	CP–A	CP–AO	CP–R	CP–AOR
AC3_1_0	0.6	**0.3**	**0.1**	**0.1**	**0.1**	**0.1**	**0.1**
AC3_2_0	27.1	2.0	**0.1**	**0.1**	**0.1**	**0.1**	**0.2**
AC3_3_0	107.1	12.0	**0.2**	**0.2**	**0.2**	**0.2**	**0.3**
AC3_1_1e	2.1	0.9	0.3	0.3	0.3	0.4	**0.2**
AC3_2_2ce	8.1	9.3	-	48.3	12.3	-	**5.3**
AC3_3_2c	111.6	54.9	2.8	**1.3**	**1.3**	2.8	**1.4**
AC3_3_2e	112.4	69.7	657.2	6.8	7.0	1052.5	**6.1**
AC3_3_2ce1	104.4	73.3	999.5	12.2	7.1	83.4	**6.2**
AC3_3_2ce2	99.6	68.0	78.6	15.8	9.7	505.7	**8.2**
AC3_3_2ce3	234.0	**190.1**	-	3067.5	338.6	-	228.9
AC3_3_3	41.0	43.9	7.0	5.0	**4.6**	7.2	**4.8**
FM3_4_0	3002.4	1469.3	**8.2**	9.0	8.5	**8.2**	**8.1**
FM3_4_4	-	712.9	**6.5**	10.8	**6.7**	**6.5**	**6.4**
FM3_4_4c	3031.9	380.2	**6.5**	7.0	**6.7**	**6.4**	**6.4**
FM3_4_4ce	-	514.6	-	**38.1**	68.6	3370.0	55.9
FM3_5_0	-	615.4	23.0	27.4	23.5	23.2	**22.6**
FM3_5_4	-	897.3	**20.4**	25.8	21.1	**20.5**	**20.4**
FM3_5_4e	-	1205.5	-	**120.4**	227.6	-	172.5
FM3_5_6	-	836.4	**21.6**	30.5	25.3	**21.7**	22.6
FM3_6_0	-	-	69.2	80.3	72.4	71.1	**68.6**
FM3_6_1	-	-	89.8	98.3	93.8	93.7	**88.7**
FM3_6_2	-	-	90.3	128.8	93.9	92.7	**89.1**
FM3_6_4	-	-	98.8	111.9	89.5	101.5	**86.1**
FM3_6_6	-	-	90.5	106.3	88.1	94.2	**83.3**
FM3_6_6e	-	-	-	**466.7**	3063.4	-	735.5
FM3_6_6ce	-	2290.3	-	**529.6**	3071.6	-	736.0
TP7_6_0	-	-	468.6	472.7	467.2	469.9	**446.3**
TP7_6_4	-	-	-	-	-	**2041.2**	-
TP7_6_4e	-	-	-	-	-	-	-
TP7_7_0	-	-	-	-	-	-	-
TP7_7_4	-	-	-	-	-	-	-
TP7_7_4e	-	-	-	-	-	-	-
TP7_8_0	-	-	-	-	-	-	-
TP7_8_6	-	-	-	-	-	-	-
TP7_8_6e	-	-	-	-	-	-	-
TP7_9_0	-	-	-	-	-	-	-
TP7_9_3	-	-	-	-	-	-	-
TP7_9_6	-	-	-	-	-	-	-
TP7_9_6e	-	-	-	-	-	-	-
Mean	2549.5	1996.1	1731.8	1246.7	1307.9	1683.9	1182.3
Geo. mean	909.6	553.1	188.2	97.3	92.9	186.8	82.1

Table 7.4: Computation times to optimality in seconds with a timeout of one hour for the CP model with and with annotations (A), repositioning type order (O), and redundant constraints (R) versus LTOP and the MIP.

However, the CP model is structured around exploiting this piece of the problem. Other natural changes, such as allowing vessels to undergo repairs, would also be difficult to implement. The second limitation is that many of the components of the CP model involve pre-computations that multiply the number of phase-out actions with the number of phase-in ports and weeks. Although the model works well on our real world instance, these pre-computations pose an issue for scaling to larger liner shipping services.

7.4 Lazy Clause Generation

Lazy Clause Generation (LCG) [Ohrimenko, Stuckey, and Codish, 2009] or CP with learning has been found to be effective on a number of scheduling problems [Schutt et al., 2012; Feydy and Stuckey, 2009; Chu et al., 2010]. LCG solvers can be used for any problem that is modelled as a constraint programming problem, which makes LCG a highly generalisable technique that is worth investigating for scheduling and routing problems with time-varying action costs. In this section, we compare a CP solver LCG with against a traditional finite domain CP solver for the BPCTOP CP model from Chapter 3, as well as for the LSFRP CP model presented in Section 7.3.

7.4.1 Experimental Results for BPCTOP

The CP model for the BPCTOP with all improvements from Chapter 4 was formulated in the MiniZinc modelling language [Nethercote et al., 2007] and solved using the CPX solver included in G12 2.0 [Wallace, 2009; Feydy and Stuckey, 2009], which uses Lazy Clause Generation. The CPX runtimes were then compared against the G12 2.0 finite domain CP solver, using backtracking search with the fastest variable selection and domain reduction strategies as discussed in Chapter 4. All BPCTOP experiments used an Intel i7-930 quad-core 2.80 GHz processor and 12.0 GB RAM, with a 30-minute cutoff time.

The CPX solver builds on the G12 finite domain solver, and combines it with a SAT solver. The FD solver controls the search, posting an *explanation* clause to the SAT solver every time a propagator is executed which updates a variable domain or causes failure. Explanation clauses explain the reason for a failure or a variable domain change; however, the implementation of a LCG solver may differ in how explanation clauses are generated, which may affect the efficiency of both the CP and SAT components. After receiving a new explanation clause, the SAT solver performs unit propagation, which may cause further domain changes to be made, after which control passes back to the FD solver. See Ohrimenko, Stuckey, and Codish [2009] for an introduction to Lazy Clause Generation, and Feydy and Stuckey [2009] for an in-depth discussion of the CPX solver.

Both solvers were used to solve the set of problems presented in Chapter 4, extended to contain problems with 4 to 20 ships for each problem type – MIXED_WIDE (MW), MIXED_NARROW (MN), ONEWAY_WIDE (OW) and ONEWAY_NARROW (ON) – both with and without tugs, for a total of $4 \cdot 17 \cdot 2 = 136$ different problem instances.

	No Tugs		With Tug Constraints	
NShips	G12_FD	G12_CPX	G12_FD	G12_CPX
4	0.34	**0.23**	**0.23**	0.33
5	**0.23**	**0.22**	**0.33**	**0.33**
6	0.44	**0.22**	**0.44**	0.67
7	**0.34**	**0.33**	**2.95**	3.60
8	**0.45**	0.87	47.0	**24.4**
9	**4.70**	23.4	184	**65.7**
10	**99.9**	482	>1800	**817**
11	**609**	>1800	–	>1800
12	>1800	–	–	–

Table 7.5: CPU time (s) for CPX solver with LCG vs. G12 finite domain solver on OneWay_Narrow (ON) problems.

Table 7.5 presents calculation times in seconds for CPX and the G12 FD solver, for each number of ships that could be solved to optimality within the 30-minute cutoff time, for the problems in the most tightly constrained (and thus most difficult) OneWay_Narrow (ON) problem set. A dash indicates that the optimal solution was not found within the cutoff time. Problems with 13 or more ships could not be solved within the cutoff time, and are thus omitted.

Table 7.6 shows the number of ships in the largest problem that could be solved within the 30-minute cutoff time for all four problem types, with and without tugs, with runtimes in seconds given in brackets. In both tables, bold font indicates the solver that was faster to solve a given problem.

Table 7.5 shows that CPX scales better than the FD solver for large ON problems with tug constraints. However, CPX is slower to solve ON problems without tugs. This may indicate that CPX finds effective nogoods (areas of the search space with no good solutions) for tug constraints, enabling CPX to avoid searching large areas of the search space for the problem with tug constraints. The FD solver, on the other hand, cannot eliminate those areas of the search space, leading to excessive backtracking resulting from the highly oversubscribed tug problem.

Table 7.6 shows that the difference between the CPX and FD solvers is even greater for Mixed problems. Mixed problems have a mix of inbound and outbound ships, resulting in a less constrained problem, but one that has more complex tug constraints due to needing to consider tugs moving between inbound and outbound ships. This indicates that CPX is very fast at dealing with complex constraints, but the speed difference decreases when the problem is tightly constrained. CPX is able to solve larger problems faster for all problem types except for OneWay_Narrow without tugs – this is the most constrained problem type, using the simplest constraints (no tugs, and no interaction between incoming and outgoing ships).

The slower performance of CPX on the problem without tugs indicates that there may be room for improvement if better explanations are added for constraints that do not involve tugs, such as the sequence-dependent setup times between ships, and the

Problem	FD	CPX
Mixed_Wide (MW)	11 (326)	**16 (1040)**
OneWay_Wide (OW)	11 (1480)	**11 (273)**
Mixed_Narrow (MN)	11 (370)	**16 (1100)**
OneWay_Narrow (ON)	**11 (609)**	10 (482)
Mixed_Wide_Tugs (MWT)	10 (11.5)	**13 (1290)**
OneWay_Wide_Tugs (OWT)	9 (81.5)	**11 (1680)**
Mixed_Narrow_Tugs (MNT)	10 (202)	**12 (1350)**
OneWay_Narrow_Tugs (ONT)	9 (184)	**10 (817)**

Table 7.6: CPX solver vs. G12 finite domain solver: number of ships in the largest problem solved within the cutoff time, with CPU time in seconds in brackets.

propagation of the objective function itself. As the Tugs problem is composed of the NoTugs problem with additional constraints, speeding up the solution time of the NoTugs problem would likely also improve the solution time of the Tugs problem.

7.4.2 Experimental Results for LSFRP

We use the Opturion CPX 1.0 optimizer [Opturion Pty Ltd, 2013] to solve the LSFRP[2]. Table 7.7 shows the time to solve the LSFRP to optimality in seconds for all of the instances in our dataset for both using LCG (CPX–AOR) and without LCG (CP–AOR). We omit LTOP, the MIP and other CP configurations from these results due to the dominance of CP-AOR on the LSFRP. Using LCG results in an average time of 138 seconds less than not using LCG, although the geometric means of both approaches are relatively similar. Of particular note is that LCG is able to solve three instances that timeout when not using LCG, allowing instances with up to 9 ships to be solved. In addition to these three instances, CPX–AOR posts significant runtime gains on 4 instances, although it is significantly slower than not using LCG on 10 instances, with the rest of the instances resulting in (roughly) a tie. LCG seems to be somewhat slower on problems with equipment, which tend to have more time-dependent task costs in their objective than instances without equipment, which could indicate that LCG is useful for problems with limited numbers of time-dependent tasks, or tasks that resemble the hotel costs present in all LSFRP instances.

Lazy Clause Generation (LCG) is a very general technique that has been successfully used to speed up calculation times for many different types of scheduling problems. However, as it is a recent method, its effectiveness for many other problem types has not yet been investigated. Like other complete methods, LCG does not scale well to large problems, but shows promise in its scaling behavior on the LSFRP. However, like traditional CP solvers, LCG solvers can be combined with decomposition techniques or large neighbourhood search to improve scalability Schutt et al. [2012]. Given our positive results on both the BPCTOP and LSFRP, LCG may

[2]This is essentially the same solver as the CPX solver included with G12 that is used for the BPCTOP except for some bug fixes that allow it to work with the LSFRP.

Problem	CP–AOR	CPX–AOR
AC3_1_0	**0.1**	**0.1**
AC3_2_0	**0.2**	**0.3**
AC3_3_0	**0.3**	0.6
AC3_1_1e	**0.2**	**0.2**
AC3_2_2ce	**5.3**	5.8
AC3_3_2c	1.4	**1.0**
AC3_3_2e	**6.1**	11.0
AC3_3_2ce1	**6.2**	14.5
AC3_3_2ce2	**8.2**	14.1
AC3_3_2ce3	228.9	**75.5**
AC3_3_3	4.8	**2.0**
FM3_4_0	**8.1**	13.5
FM3_4_4	**6.4**	8.1
FM3_4_4c	**6.4**	8.3
FM3_4_4ce	**55.9**	125.7
FM3_5_0	**22.6**	25.3
FM3_5_4	**20.4**	28.2
FM3_5_4e	**172.5**	507.7
FM3_5_6	**22.6**	53.9
FM3_6_0	68.6	**44.5**
FM3_6_1	88.7	**80.5**
FM3_6_2	89.1	**82.4**
FM3_6_4	**86.1**	218.6
FM3_6_6	**83.3**	84.4
FM3_6_6e	**735.5**	823.4
FM3_6_6ce	**736.0**	825.2
TP7_6_0	446.3	**63.4**
TP7_6_4	-	-
TP7_6_4e	-	-
TP7_7_0	-	**435.6**
TP7_7_4	-	-
TP7_7_4e	-	-
TP7_8_0	-	**1303.4**
TP7_8_6	-	-
TP7_8_6e	-	-
TP7_9_0	-	**3455.5**
TP7_9_3	-	-
TP7_9_6	-	-
TP7_9_6e	-	-
Mean	1182.3	1043.9
Geo. mean	82.1	83.1

Table 7.7: Computation times to optimality in seconds with a timeout of one hour for the CP model without LCG (CP–AOR) and with LCG (CPX–AOR).

Problem	NShips	Runtime (s)			NShips	Runtime (s)		
Type	(small)	FD	CPX	SIMPLE	(large)	FD	CPX	SIMPLE
MW	11	326	**8.19**	188	16	>1800	**1040**	>1800
OW	11	1480	273	**0.56**	12	>1800	>1800	**26.7**
MN	11	370	**7.64**	180	16	>1800	**1100**	>1800
ON	10	99.9	482	**0.34**	12	>1800	>1800	**19.6**
MWT	10	11.5	17.1	**11.4**	13	>1800	**1290**	>1800
OWT	9	81.5	23.9	**4.60**	11	>1800	**1680**	>1800
MNT	10	202	42.8	**8.21**	12	>1800	**1350**	>1800
ONT	9	184	65.7	**0.69**	10	>1800	817	**262**

Table 7.8: Runtime (s) of CPX vs. the FD solver with a simplified BPCTOP model.

be worth investigating as an approach to dealing with other routing and scheduling problems with time-dependent task costs.

7.5 Solve-and-Improve

Many scheduling and routing approaches initially solve a simplified model, and then use the constraints and objective function of the full problem to improve it. For routing and scheduling problems with time-dependent action costs, removing the time dependence of the objective function is one way to simplify the problem for the first step of a solve-and-improve approach, as used by Lin et al. [2005].

This section investigates simplified models for both the LSFRP and BPCTOP which ignore time-varying action costs, and compares the improvement in runtime against improvements obtained by the CP model for the LSFRP and the LCG solver for the BPCTOP. The runtimes for the simplified models are a lower bound on the runtime of a full solve-and-improve approach, as the improvement step would increase the runtime further.

7.5.1 Results for BPCTOP

A simplified BPCTOP model was implemented by replacing the time-varying objective function by feasible time windows, and solving the problem with the objective of maximising the number of ships scheduled to sail.

Table 7.8 compares the runtimes of the normal and simplified models for the largest problem that was successfully solved by all approaches, as well as for the largest problem solved by any approach. Bold font indicates the fastest runtime, (ie. the approach that results in the largest reduction in runtime).

Table 7.8 shows that, while the simplified model is faster for small problem sizes, for large problem sizes the CPX solver with LCG is faster than the simplified model to solve 5 of the 8 problem types, indicating that CPX scales better than the simplified model for 5 of the 8 problem types. As seen earlier in Table 7.6, CPX is particularly effective on large problems with tugs.

One interesting observation from Table 7.8 is that the simplified model gives the largest runtime improvement for the most tightly constrained ON and ONT problems, allowing problems with one more ship to be solved within the 30-minute cutoff time. The least tightly constrained MW and MWT problems show only a small improvement in runtime from relaxing the time window penalties; and the moderately constrained MN, MNT, OW and OWT problems show moderate improvements. This result is similar to the effect of relaxing hard time windows in a vehicle routing problem, as was investigated by Qureshi, Taniguchi, and Yamada [2009]. Extending the latest delivery time by 10-20 minutes was found to significantly improve schedule costs for tightly constrained problems. For problems with wide time windows, on the other hand, where the time windows did not constrain the problem, relaxing the time windows had little effect on cost and also increased runtime due to increasing the computational complexity of the problem.

7.5.2 Results for LSFRP

A simplified LSFRP model was implemented by fixing all sailing and sail equipment actions to "slow-steaming", i.e. minimum fuel cost with maximum time using the LTOP planner [Tierney et al., 2012a]. A similar simplification was applied to the CP model, by binding all sail equipment actions to their maximum length. Sail actions in the CP model are discretised, so we do not need to change them to simplify the model. However, the hotel cost calculation, an important time-dependent task cost in the LSFRP, cannot be simplified in any reasonable way.

Table 7.9 provides the runtimes in seconds for LTOP and CP versus simplified LTOP and CP approaches, with a timeout of one hour of CPU time[3]. The results show that solve-and-improve is not a particularly effective method within the LTOP framework, with only AC3_3_0 showing any speed improvements. We can therefore conclude that fixing the length and cost of the sail action is not very effective for the LTOP method on the LSFRP, since the problems where the most benefit can be expected from fixing action costs are those in which the optimal answer uses all slow-steaming actions.

Using solve-and-improve with CP–AOR leads to more promising results than with LTOP, although it is unable to offer solutions for large LSFRP instances with equipment. S–CP–AOR is able to find a solution quickly on a number of problems with equipment, which are the only problems for CP where it can make a difference. In particular on AC3_3_2ce3, FM3_6_6e, and FM3_6_6ce solve-and-improve requires only around 10% of the time of CP–AOR. Of course, the improve step still would need to be run.

7.5.3 Summary

A solve-and-improve approach that simplifies away time-varying action costs was found by Lin et al. [2005] to be effective for satellite imaging scheduling. However,

[3]Experiments used Intel Core i7-2600K 3.4GHz processors with a maximum of 4GB per execution.

Problem	LTOP	S–LTOP	CP–AOR	S–CP–AOR
AC3_1_0	0.6	0.5	**0.1**	**0.1**
AC3_2_0	27.1	20.7	**0.2**	**0.1**
AC3_3_0	107.1	66.1	**0.3**	**0.3**
AC3_1_1e	2.1	2.3	**0.2**	**0.2**
AC3_2_2ce	8.1	8.9	5.3	**2.5**
AC3_3_2c	111.6	142.7	**1.4**	**1.3**
AC3_3_2e	112.4	117.4	6.1	**3.1**
AC3_3_2ce1	104.4	115.7	6.2	**3.2**
AC3_3_2ce2	99.6	104.4	8.2	**3.5**
AC3_3_2ce3	234.0	234.0	228.9	**26.9**
AC3_3_3	41.0	42.8	**4.8**	**4.5**
FM3_4_0	3002.4	2931.3	**8.1**	8.8
FM3_4_4	-	-	**6.4**	7.2
FM3_4_4c	3031.9	2939.6	**6.4**	6.9
FM3_4_4ce	-	-	55.9	**8.6**
FM3_5_0	-	-	**22.6**	24.7
FM3_5_4	-	-	**20.4**	22.0
FM3_5_4e	-	-	172.5	**24.3**
FM3_5_6	-	-	**22.6**	24.7
FM3_6_0	-	-	**68.6**	73.5
FM3_6_1	-	-	**88.7**	97.2
FM3_6_2	-	-	**89.1**	96.5
FM3_6_4	-	-	**86.1**	92.9
FM3_6_6	-	-	**83.3**	91.8
FM3_6_6e	-	-	735.5	**78.0**
FM3_6_6ce	-	-	736.0	**78.3**
TP7_6_0	-	-	446.3	-
Mean	2549.5	2572.5	1182.3	1220.0
Geo. mean	909.6	934.5	82.1	61.6

Table 7.9: Computation times to optimality in seconds for CP and LTOP versus simplified approaches (S–LTOP and S–CP–AOR) with optimal windows. Note that we have removed all TP7 instances except for TP7_6_0 due to timeouts.

they did not compare the calculation speed of the complete problem against the simplified problem, so there is no indication of the improvement in calculation speed produced by simplifying away the time-varying quality function. However, the experiments in this section found that simplifying the LSFRP and BPCTOP models by removing the time-varying cost function did not produce significant speed improvement, and that switching to a CP model or a solver with LCG was more effective.

It is possible that simplifying the cost function was more effective for speeding up solution times for Linear Programming problems such as Lin et al. [2005], rather than for CP or automated planning. However, more work would need to be done to identify the speed improvement produced by simplifying the quality function in Lin et al. [2005].

7.6 Conversion to Vehicle Routing

Of all scheduling and routing problems with time-varying task costs, the Vehicle Routing Problem with Soft Time Windows (VRPSTW) is one of the most thoroughly researched, comprising about 90% of the literature on scheduling and routing problems with soft time windows. Since the vehicle routing domain has been investigated more extensively than other scheduling and routing problems with time-varying task costs, there are several fast, efficient solvers for the VRPSTW in existence.

Many scheduling problems, particularly those without complex side constraints, can be modelled as a VRP, which brings us to the idea that for some scheduling problems, converting them to VRPs and solving using an existing VRP solver may be more efficient than using a generic CP or MIP solver.

This section investigates the approach of converting a scheduling problem with time-varying action costs to a VRPSTW and using an existing VRP solver to solve this problem. We test this approach by modelling the BPCTOP with time-varying draft restrictions as a VRPSTW. The basic BPCTOP without tug availability constraints can be converted easily to a VRPSTW; however, the complex tug constraints are more difficult to convert to vehicle routing constraints. The model, both with and without tugs, is described below.

7.6.1 BPCTOP VRP Model

The BPCTOP without tug constraints is modelled as a VRP as follows:

- Each ship is modelled as a job.
- The channel is modelled as a single vehicle that must be used to complete all jobs.
- The sequence-dependent separation times between ships are modelled as direction-dependent travel times between successive jobs.
- Each job has an earliest time when it can be completed, corresponding to the ship's earliest sailing time.

- The value of each job request corresponds to the weighted draft of the ship.
- For each request, soft time window cost penalties are specified corresponding to the reduction in draft if the ship sails outside its maximum draft window.
- The objective of the VRP is to maximise the total value of all fulfilled requests.

We solve the model using the Indigo VRP solver, which is able to consider a wide variety of side constraints [Kilby and Verden, 2011]. One limitation of the solver is that it is only able to handle soft time windows with linear penalties, whereas the draft function can vary non-linearly outside the peak draft windows. Using linear penalty functions to approximate our draft functions may lead to schedules that are not entirely optimal with respect to the true draft functions. While conversion to VRP may be useful for finding close-to-optimal solutions for large ship scheduling problems, a further improvement step may be needed to optimise the final schedules. For example, we could use the VRP for an approximate solution for large multi-port problems or long time scales, then find optimal schedules for each high tide at each port by solving the CP model with the exact draft function.

The basic port optimisation problem without tug constraints only has one vehicle, and can therefore be considered as a Travelling Salesman Problem which aims to minimise soft time window penalties instead of the total distance. However, a TSP approach would not be generalisable to more complex scheduling problems that have multiple resources. Also, any TSP approach would not be able to handle more complex constraints such as the BPCTOP tug constraints. In this thesis, we only investigate modelling the problem as a VRP, not as a TSP, since the VRP approach would be generalisable to a larger range of problems.

7.6.2 Tug Constraints Model

The BPCTOP constraints on tug availability can be modelled using vehicle routing constraints as follows:

- In order to model the tugs, the VRP requires two commodities – a "ship" commodity, used to fulfil job requests for ship sailings, and a "tug" commodity, used to fulfil tug requirements for each ship.
- Each tug available at the port is modelled as a separate vehicle, able to transport 1 unit of the "tug" commodity, and none of the "ship" commodity. This allows us to model tug turnaround times as travel times from the tug job locations back to the depot.
- The channel is still modelled as a vehicle, able to transport an unlimited amount of the "ship" commodity, and none of the "tug" commodity.
- New locations are now required for each possible duration of tug turnaround times, with travel times from each tug location to the depot representing tug turnaround times.

- For each ship, we add extra requests for each tug required for that ship. Each tug request requires one unit of the tug commodity.
- We add constraints specifying that tug requests for each ship must be serviced at the same time as the corresponding ship request.
- We also add constraints specifying that if a ship request is fulfilled, then all tug requests for that ship must also be fulfilled.

Modelling the tug constraints as a VRP requires a VRP solver that can handle constraints that require two jobs to be completed at the same time, and constraints that specify that two jobs must either both be completed, or both not completed. Unfortunately, at present, Indigo cannot calculate soft time window penalties correctly with these side constraints present, and other existing VRP solvers do not include these types of constraints at all to the authors' knowledge. For the moment, the approach of converting a scheduling problem to a VRP is limited only to simple scheduling problems without complex side constraints. More flexible VRP solvers may be developed in future.

7.6.3 Experimental Results

We converted the most tightly constrained OneWay_Narrow ship scheduling problems from Chapter 4, extended to consider 4-20 ships sailing on a tide, to VRP problems with soft time windows. These problems were then solved using the Indigo VRP solver [Kilby and Verden, 2011]. Indigo does not yet support constraints that require two jobs to be completed at the same time in conjunction with early and late arrival penalties, so we only present results for the problems with no tug constraints for now. Extending Indigo to include the constraints required to support tugs and testing the VRP model with tug constraints is left for future work.

All experiments were run on a Windows 7 machine with an Intel i7-930 quad-core 2.80 GHz processor and 12.0 GB RAM, and with a 30-minute (1800-second) cutoff time. Calculation times for Indigo vs the G12 FD and CPX solvers are shown in Table 7.10.

The Indigo calculations used a Large Neighbourhood Search with two variations of the search parameters: starting from 1000 iterations and increasing the number of iterations by 200 for every ship; or starting from 200 iterations and increasing the number of iterations by 40 for every ship. The calculation times for both sets of iterations are shown in Table 7.10. Up to 15 visits are removed in each LNS iteration and then re-inserted.

Indigo uses Large Neighbourhood Search to find the best solution within the given number of iterations, but it does not produce a proof of optimality for the VRP solution, and may return suboptimal solutions. Also, since Indigo only supports linear time-window penalty functions, it cannot model the non-linear task value functions precisely, which contributes to it finding suboptimal solutions for larger problems where not all ships are able to sail with their peak draft.

NShips	G12_FD	CPX	VRP: 200	VRP: 1K
4	0.34	**0.23**	0.55	2.18
5	**0.23**	**0.22**	1.14	5.15
6	0.44	**0.22**	2.43	11.7
7	**0.34**	**0.33**	4.70	23.0
8	**0.45**	0.87	8.43	41.6
9	**4.70**	23.4	13.8	68.6
10	99.9	482	**21.3**	108
11	609	>1800	**30.7**	151
12	>1800	–	**41.7**	207
13	–	–	**53.6**	276
14	–	–	**65.3**	330
15	–	–	**78.4**	430
16	–	–	**93.5**	480
17	–	–	**118**	548
18	–	–	**130**	668
19	–	–	**151**	737
20	–	–	**170**	851

Table 7.10: Indigo VRP solver vs. G12 FD and CPX solvers, without tug constraints, for OneWay_Narrow problems.

Figure 7.1 shows the cost increase of the optimal and VRP solutions compared to an "ideal" situation with all ships sailing with their peak draft, for problems with up to 11 ships for which an optimal solution was found using the CP model. Since the solver's Large Neighbourhood Search uses randomisation, the solver was run five times for each problem, and the best of the five results is shown in Figure 7.1. Note: the calculation times in Table 7.10 show the mean runtime for the five calculations. The solver failed to find an optimal solution for problems with 10 or more ships, as well as for 8 ships with the smaller number of search iterations.

The VRP modeling of the BPCTOP can be solved much faster than the CP model for large problems with 10 or more ships. Indigo is able to solve problems with up to 20 ships in a short timeframe – 8 ships more than the largest problem solved with any CP solver. However, the results are suboptimal for all problems with 10 or more ships, and no certificate of optimality can be provided by Indigo for any instance.

Increasing the number of iterations does not significantly improve the quality of the schedules – in some cases, the best schedule found over five calculations is better for calculations with fewer search iterations, for example, the case with 11 ships. This indicates that the solver we chose for our experiments may be prone to getting stuck in local minima, since past a certain point, searching with restarts is more effective than simply increasing the number of iterations of Large Neighbourhood Search.

Modelling a scheduling problem as a VRP may be an effective way to find close to optimal solutions for larger scheduling problems than can be solved using optimal methods. However, since VRP solvers only support the types of constraints

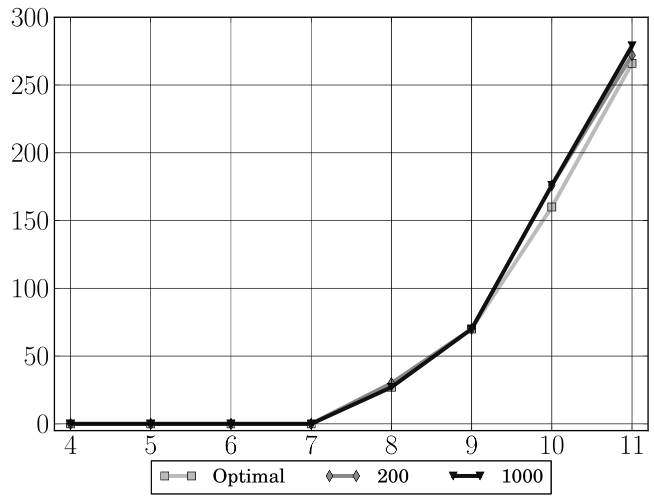


Figure 7.1: Cost increase of optimal and VRP solutions compared to all ships sailing with peak draft.

and penalty functions found in VRP problems, many types of scheduling problems cannot be modelled accurately as a VRP, and a VRP solver will not be able to find optimal solutions to the scheduling problem. However, even for those scheduling problems where the VRP solution is not good enough on its own, it may provide a good starting point for a slower method that can consider complex constraints and objective functions.

7.7 Summary

While scheduling and routing problems have usually been solved using Mixed Integer Programming (MIP) and solve-and-improve approaches, Constraint Programming (CP) with a good choice of model and search strategy, as well as recent techniques such as Lazy Clause Generation (LCG), have been found to be faster for some problem types. This chapter reviewed scheduling and routing problems with time-varying action costs across a number of applications, and applied several approaches that had been successfully used to solve other problems with time-varying action costs to the Liner Shipping Fleet Repositioning Problem (LSFRP), and the Bulk Port Cargo Throughput Optimisation Problem (BPCTOP).

A CP model for the LSFRP was found to be faster than existing MIP and automated planning models by an order of magnitude for some instances, similarly to the findings that a CP model with a good choice of search strategy was faster than a MIP model for the BPCTOP in Chapter 4. The CP models for both problems were also solved with a CP solver that uses Lazy Clause Generation as well as a traditional finite domain CP solver. The LCG solver was faster for 7 out of 8 problem types for the BPCTOP. For LSFRP, the LCG solver was able to solve 3 more instances, as well as achieving a faster average solution time across the entire dataset.

This chapter also compared a solve-and-improve approach, which uses simplified models without time-varying action costs to find initial solutions, against the LTOP and CP models of the LSFRP and the full CP model for the BPCTOP. The speed improvement of removing time-varying costs was less than that obtained by converting the LSFRP to a CP model for all problem instances, and solving the BPCTOP using an LCG solver scaled better than using a finite domain solver for 5 of 8 problem types, particularly for the most challenging problems with complex tug constraints.

The investigation of the LSFRP CP model in this chapter, as well as the BPCTOP CP model in Chapter 4, found that the CP model solution time was highly dependent on having a good choice of model and search strategy. However, with this caveat, we find that CP and LCG are efficient and flexible methods that are able to handle complex side constraints, and may therefore be worth investigating for other scheduling and routing problems that are currently being solved using MIP or solve-and-improve approaches.

This chapter also investigated the approach of converting a scheduling problem with time-varying action costs into a VRPSTW to take advantage of the highly efficient specialised solvers available for the extensively researched VRPSTW. The BPCTOP was modelled as a VRPSTW and solved using the existing Indigo VRP solver. Modelling scheduling problems as a VRP can allow solutions to be found quickly for large problems; however, if the scheduling problem contains complex cost functions or side constraints, VRP solvers may not be able to model them accurately, leading to a reduction in solution quality. However, VRP solvers may be a fast way to solve simple scheduling problems, or to quickly find an approximate initial solution to use as input to a more specialised search algorithm.

Chapter 8

Conclusion

8.1 Motivation

Optimisation problems in maritime transportation are an important field of research, since the majority of the world's goods are transported by sea, so even small improvements in efficiency can translate to multi-million-dollar benefits to the industry. Optimisation problems in maritime transportation are also often challenging to solve due to the size of the search space and the complexity of the constraints.

This thesis investigated one previously under-researched area of maritime transportation – the impact of time-varying restrictions on vessel draft on optimisation problems in the maritime industry. Many ports have restrictions on the draft of ships that can enter the shallow waters surrounding the port, and these restrictions vary with the tide, waves, and other environmental conditions. However, existing maritime transportation research has ignored or oversimplified these draft restrictions, which could lead to ships missing out on opportunities to load more cargo at high tide.

8.2 Contributions

8.2.1 Bulk Port Cargo Throughput Optimisation Problem

This thesis has incorporated time-varying draft restrictions into several problems in maritime transportation. First, we introduced a novel problem not previously investigated in the literature – the Bulk Port Cargo Throughput Optimisation Problem (BPCTOP), which schedules ship sailing times at a port so as to maximise the total cargo carried on the set of ships. The schedule must meet port safety rules on allowable sailing draft for each ship, as well as rules on minimum safe separation time between successive ships, and constraints on the availability of tugs – small boats which are required at some ports to assist ships travelling in and out of the port.

We presented both Constraint Programming (CP) and Mixed Integer Programming (MIP) models for this problem, and investigated a number of search strategies for solving the CP model. We found that the CP model solution time was highly dependent on the search strategy used by the solver – optimal solutions were found much faster by searching initially on draft, rather than time, since this approach cut out poor quality solutions more quickly, and focussed the search on areas of high-quality solutions. A CP solver with a good choice of model and search strategy was found to outperform a MIP solver more traditionally used for this problem type.

We also investigated several potential improvements to the CP model, and found that solver implementation sometimes affected what types of constraints could be solved efficiently. To create efficient CP models, it was often necessary to precompute some input data, or specify the constraints in a way that is more efficiently handled by the CP solver. Some of the improvements that had a strong impact on calculation time for our model could be automated, and thus suggest potential improvements to the CP solver, which are summarised in Section 8.3 – Future Work.

Chapter 4 also considered two logic-based Benders decomposition approaches, but found that one approach used overly complex cuts which did not significantly reduce calculation time compared to solving the complete model, while the other approach used overly simplistic cuts which did not cut out enough of the search space, and thus resulted in too many iterations, even though the individual iterations were fast to calculate.

We compared the results produced by our model against real schedules created by human schedulers at Port Hedland in Western Australia, the world's largest bulk export port, and found that our schedules would allow 2.04Mt more iron ore to be transported on the same set of ships as currently sail from Port Hedland each year, leading to a $154 million increase in profit for shippers, assuming there is sufficient demand and no bottlenecks in production which would constrain the amount of iron ore that can be produced. Even if the amount of cargo available to be transported is constrained by demand or bottlenecks in production, using our more efficient schedules would save $20.4 million per year in shipping costs by allowing more cargo to be loaded onto ships, and thus reducing the number of ships that are required to transport cargo.

The method to solve the BPCTOP has now been incorporated into a commercial system for scheduling ship sailing times at a port – DUKC® Optimiser – which was discussed in Chapter 5. This chapter also compared optimal schedules against constant-draft approaches such as those used in the maritime transportation literature, as well as simple algorithms similar to those used by human schedulers at ports. These comparisons illustrate why such approaches may fail to find optimal schedules even for relatively small problem sizes.

8.2.2 Ship Speed Optimisation with Time-Varying Draft Restrictions

Moving on beyond the BPCTOP, this thesis also investigated the impact of considering time-varying draft restrictions in problems involving ship speed optimisation. Fuel costs can comprise up to 75% of the total cost of a voyage, and fuel usage is a cubic function of ship speed, so ship speed optimisation has been investigated more frequently in recent years as a means of reducing shipping costs.

This thesis investigated four problems of increasing complexity, all involving time-varying restrictions on ship draft at waypoints as well as ship speed optimisation decisions. The first problem was the Speed Optimisation Problem with time-varying draft restrictions, for a single ship sailing along a fixed route. This approach built on previous work by Fagerholt, Laporte, and Norstad [2010] and Norstad, Fagerholt, and Laporte [2011], who compared an approach that discretised arrival times at each waypoint against a Recursive Smoothing Algorithm which used features of the problem to efficiently find optimal solutions. We extended both approaches to handle time-varying draft restrictions, including the associated multiple time windows at each high tide instead of the single time window considered by Norstad, Fagerholt, and Laporte [2011]. We also extended this algorithm to work for non-monotonic cost functions, since the cost function considered in our approach included the opportunity cost of not being able to use the ship to transport other cargo, whereas the approach of Norstad, Fagerholt, and Laporte [2011] only minimised fuel costs, without considering this opportunity cost.

The initial Speed Optimisation Problem only found optimal speeds for a single ship with a fixed draft sailing along a fixed route. This problem is of limited use on its own, but it can be used as a subproblem to assist in solving larger problems. The first larger problem we considered was the problem of finding the optimal draft and set of speeds for a single ship sailing along a fixed route. This problem was solved by simply discretising the drafts, optimising speeds for each draft, and selecting the optimal draft that balanced fuel costs, opportunity costs of the ship being in use for a longer period of time, and the opportunity cost of carrying less cargo than the maximum feasible amount.

The next problem we considered was the Multi-Ship Speed Optimisation Problem with Time-Varying Draft which optimised drafts and speeds for a set of ships sailing along a fixed route, close together in time. This problem considered variable ship speeds, variable draft, time-varying draft restrictions, and resource conflicts between waypoints caused by multiple ships trying to sail past a shallow waypoint at the same time, where the waypoint has safety restrictions on the separation time between successive ships. The conflicts at waypoints were solved using a simplified version of the BPCTOP as a subproblem, to minimise costs over the set of ships.

Finally, the most complex problem with speed optimisation and time-varying draft restrictions investigated in this thesis was the Tramp Ship Routing and Scheduling Problem with Speed Optimisation (TSRSPSO), introduced by Norstad, Fagerholt, and Laporte [2011], extended to consider time-varying draft restrictions as well as resource conflicts between ships at waypoints. This was solved using a similar approach to that presented by Norstad, Fagerholt, and Laporte [2011], by creating a set of initial solutions to the routing problem, evaluating them by solving the Speed Optimisation Problem for each ship's route, then iteratively improving the solutions using local neighbourhood search.

We extended this approach by using the Speed Optimisation Problem with Time-Varying Draft, and adding waypoint conflict resolution. Our approach produced schedules with 22.59% higher profit compared to the traditional approaches of sailing at service speed and ignoring draft restrictions in scheduling and waiting at waypoints for the tide to rise if required. The speed optimisation approach introduced by Norstad, Fagerholt, and Laporte [2011] resulted in a 15.34% improvement in profit on its own, without considering draft restrictions, so draft restrictions account for the remaining difference. The 15.34% improvement is similar to the results observed by Norstad, Fagerholt, and Laporte [2011].

8.2.3 Scheduling and Routing with Time-Varying Action Costs

The final contribution of this thesis is to consider more generally the problem of scheduling and routing with time-varying action costs.

It is clear that time-varying draft restrictions are valuable to consider in maritime optimisation approaches, since accurate consideration of draft restrictions allows ships to load more cargo by sailing at high tide. However, time-varying draft restrictions significantly increase the complexity of the problem, so we also investigated other scheduling and routing problems in related fields that have time-varying action costs or values, and applied a number of approaches that were found to be effective for these problems to the BPCTOP as well as another problem in maritime transportation – the Liner Shipping Fleet Repositioning Problem (LSFRP). The LSFRP investigations were conducted by Kevin Tierney.

We developed a CP model for the LSFRP in joint work with Kevin Tierney, and compared it against existing MIP and automated planning approaches to find that the CP model could be solved to optimality faster on average than existing approaches. We also solved both the BPCTOP and LSFRP problem sets with a CP solver that uses Lazy Clause Generation to learn areas of the search space where the search previously failed. The LCG solver was able to find solutions faster than traditional finite domain CP solvers, so Lazy Clause Generation may be worth testing for other scheduling and routing problems.

We investigated a solve-and-improve approach traditionally used for solving problems with complex cost functions by simplifying the cost functions for the BPCTOP to windows of "good enough" quality, and for the LSFRP by setting the sailing speeds of each ship to the minimum value initially. However, we found that this approach did not significantly reduce the time required to solve the problem compared to the impact of switching from a MIP/planning model to a CP model for the LSFRP, or switching from a finite domain solver to a LCG solver for the BPCTOP.

Finally, we converted the BPCTOP to a Vehicle Routing Problem, to investigate whether a fast VRP solver that was able to handle soft time window constraints would solve this problem faster than more general CP and MIP solvers. We found that the VRP solver was able to find solutions for large problems quickly; however, the solver's ability to model complex side constraints limited the accuracy of our VRP model, so the VRP solver was unable to find optimal solutions. Converting a scheduling problem to a VRP and solving it using a VRP solver is unlikely to produce optimal solutions for problems with complex side constraints; however, it may be useful as a method of finding "good enough" solutions quickly, perhaps to use as a starting point for a more complex optimal method.

8.3 Future Work

The largest and most complex area of future research arising from this thesis is to extend more problems in maritime transportation to consider time-varying draft restrictions. Given the prevalence of draft-restricted ports worldwide, there are likely to be many more problems where accurate consideration of draft restrictions would allow ships to load more cargo, leading to significant benefits to industry. Further investigation of such problems is likely to be not only an interesting area of academic research, but may lead to significant commercial opportunities. This thesis introduced time-varying draft restrictions into a cargo routing problem with one cargo carried per ship; one natural extension of this problem would be to consider multiple cargoes carried per ship, as the order of deliveries would also affect the draft of the ship at each port, similarly to the Traveling Salesman Problem with draft limits considered by Rakke et al. [2012]. Maritime inventory routing problems with time-varying draft restrictions would also be worth investigating. The Berth Allocation Problem would also benefit from considering draft restrictions, since draft restrictions may limit feasible ship arrival times and amounts of cargo carried. Different berths may also have their own draft restrictions in addition to the port as a whole.

Another large area of future research for all the problems presented in this thesis is the consideration of uncertainty. Ship routing and scheduling problems with time-varying draft involve uncertainty arising from several factors, including environmental conditions that affect allowable draft; equipment breakdowns and ship loading delays. There has been some research on ship scheduling with uncertainty, for example Christiansen and Fagerholt [2002]; Hwang, Visoldilokpun, and Rosenberger [2008]. However, uncertainty in draft constraints or the amount of cargo that can be carried on each ship has not yet been investigated. In this thesis, we account for uncertainty by using a conservative method for calculating the allowable draft; however, more thorough investigation into handling uncertainty, robust schedules, and recovering from disruptions for this problem would be very valuable.

As this thesis introduced a number of new problems in maritime transportation, there are also a number of potential directions for future research into algorithms for solving those problems in particular. A few of these are summarised below.

Potential Improvements for the BPCTOP

- *Integration with land-side supply chain optimisation:* many export ports have constraints in the supply chain which limit the amount of cargo that may be available for loading at any point in time. This may also limit the benefit of increased drafts if there is insufficient cargo available to load the ship fully. Extending the BPCTOP to combine it with mining supply chain optimisation problems would allow shippers to make the best use of increased draft even when mining supply chain constraints limit the amount of cargo available.

- *Continuous-time formulation:* the models presented in this thesis for the BPCTOP used time-indexed formulations to model the allowable draft for each ship by specifying it at every 5-minute interval. A continuous-time formulation, such as the concept of event points proposed by Ierapetritou and Floudas [1998a] and Ierapetritou and Floudas [1998b], may reduce the problem size resulting in significant speedup. However, since the allowable draft for each ship varies with time and does not have a simple algebraic formulation, modelling this problem with a continuous-time formulation is a non-trivial task. One possible approach could be to start with a simple linear approximation to the draft function for each ship and to iteratively increase the resolution; however, other approaches may also be worth investigating.
- *The Benders decomposition approach* introduced in Chapter 4 may be worth investigating further, as there may be alternative cuts that balance complexity of calculation with effectiveness of removing large enough areas of the search space.
- *Faster close-to-optimal approaches*, eg. local search with a good heuristic, are worth investigating for the BPCTOP, as such approaches are likely to produce significant speedups in search and may produce results that are close to optimal. Local search techniques have been used successfully in other ship routing and scheduling problems, for example by Brønmo et al. [2007] and Gunnarson, Rönnqvist, and Carlsson [2006].

Potential Improvements in Speed Optimisation

- *Considering variable draft* for each leg of each ship's route could reduce shipping costs further. The approach presented in this thesis used a simple approximation to set the draft for each ship to one that would get reasonably wide windows. Allowing the draft to vary independently for each cargo would make the problem significantly more complex to solve, but could significantly improve results. One approach to varying the draft of ships for each cargo could be to introduce a new local search operator which would change the amount of a single cargo on a ship by fixed increments.
- *Considering ocean currents* in travel time calculations for each leg of each ship's route. The models presented in this thesis assume that the travel time for each leg of each voyage is constant. However, in reality, travel times may be affected by ocean currents which vary over time. Extending the models to consider the effect of ocean currents on travel time would further improve the results.
- *Global optimisation approach:* the approach used to handle resource conflicts between ships at waypoints is to first optimise speeds for each ship independently, then resolve conflicts for waypoints one at a time, adjusting ship speeds to and from adjacent waypoints. However, this approach may get stuck in local

minima, and better quality schedules could be obtained by a global optimisation approach that considers the entire multi-ship routing, speed optimisation and waypoint conflict resolution problem as a whole. Such a global approach would, however, be very complex to solve.

- Faster methods for subproblems could significantly improve the calculation speed of the combined routing and scheduling problem. For example, a dynamic programming approach could be worth investigating for the Speed Optimisation Problem with a single ship. Speed improvement methods for the waypoint conflict resolution problem could arise from further research into fast algorithms for solving the BPCTOP, discussed above.

Potential Improvements in Generalised Approaches

- *Other optimisation problems may benefit from considering time-dependent task costs,* as this would increase their realism and thereby their relevance to industry. For example, adding time-varying draft restrictions to other maritime transportation problems, or adding detailed traffic congestion modelling to problems like inter-terminal transportation delay reduction at container terminals Tierney, Voß, and Stahlbock [2013]. A number of vehicle routing applications could benefit from a view of time-dependent travel durations as in Malandraki and Daskin [1992] or the pollution routing problem [Bektaş and Laporte, 2011; Franceschetti et al., 2012]. Although adding time-dependent task costs to these problems would likely increase the solving difficulty, the benefits of more closely matching real-world processes is worth the trade-off.
- *Constraint Programming approaches may be worth investigating for other problems,* since we found that CP solvers, particularly those with Lazy Clause Generation, could solve some problems more efficiently than traditional MIP approaches. However, CP model solution time was highly dependent on the choice of model and search strategy, so creating effective CP models for other problems may require substantial research.
- *Further investigation into modelling scheduling problems as Vehicle Routing Problems* could be beneficial, particularly as VRP solvers become able to deal with a wider variety of complex side constraints.

8.4 Summary

Optimisation problems based on real-world scenarios often have complex constraints or objectives. Modelling these accurately make an optimisation problem more closely match the real world, but this can also make the problem more difficult to solve. This thesis investigated a particular example where time-varying objectives and constraints have previously been simplified away – draft restrictions in maritime transportation, which allow ships to carry more cargo at high tide when there is more water for the ship to sail in.

Existing optimisation problems in maritime transportation either ignored draft restrictions entirely, or modelled them very simply, as a constant limit rather than varying every few minutes with the height of the tide. However, the results presented in this thesis showed that accurately modelling draft restrictions in maritime transportation problems can significantly reduce shipping costs or allow more cargo to be transported on the same set of ships for problems involving draft-restricted ports. Comparisons against both human schedulers at a real port and against existing academic approaches showed that consideration of time-varying draft restrictions has large potential benefits to industry.

Finally, this thesis investigated approaches to routing and scheduling with time-varying action costs in related fields, and found that some approaches such as Constraint Programming and Lazy Clause Generation could solve some problems faster than Mixed Integer Programming, which is traditionally used to solve such problems.

We hope that this research will inspire more optimisation researchers to consider accurate modelling of complex constraints and objective functions in their work to make their models more accurately reflect reality and produce better quality solutions, both in maritime transportation problems which may benefit from considering time-varying draft restrictions, and in related fields.

All scripts used for generating test data and running the experiments described in this thesis will be made available at http://users.cecs.anu.edu.au/~pjk/ShipSched/

Bibliography

Achterberg, T. 2009. SCIP: solving constraint integer programs. *Mathematical Programming Computation* 1(1):1–41. (cited on page 151)

Agerschou, H.; Dand, I.; Ernst, T.; Ghoos, H.; Jensen, O.; Korsgaard, J.; Land, J.; McKay, T.; Oumeraci, H.; Petersen, J.; Runge-Schmidt, L.; and Svendsen, H. 2004. *Planning and Design of Ports and Marine Terminals, 2nd Ed.* John Wiley and Sons. (cited on page 21)

Agnetis, A.; de Pascale, G.; Detti, P.; and Vicino, A. 2013. Load scheduling for household energy consumption optimization. *IEEE Transactions on Smart Grid* 4(4):2364–2373. (cited on page 152)

Al-Khayyal, F., and Hwang, S.-J. 2007. Inventory constrained maritime routing and scheduling for multi-commodity liquid bulk part i: applications and model. *European Journal of Operational Research* 176:106–130. (cited on pages 22, 23, and 33)

Álvarez, J.; Longva, T.; and Engebrethsen, E. 2010. A methodology to assess vessel berthing and speed optimization policies. *Maritime Economics & Logistics* 12(4):327–346. (cited on pages 23, 24, 27, 28, and 34)

Amir, E., and Englehart, B. 2003. Factored planning. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI'03)*, 929–935. (cited on pages 19 and 20)

Barrass, C. B. 2004. *Ship design and performance for masters and mates*. Elsevier. chapter 17: Ship squat in open water and in confined channels, 148–163. (cited on pages xv and 13)

Bausch, D.; Brown, G.; and Ronen, D. 1998. Scheduling short-term marine transport of bulk products. *Maritime Policy and Management* 25(4):335–348. (cited on pages 22, 26, 31, 33, 89, and 159)

Bektaş, T., and Laporte, G. 2011. The pollution-routing problem. *Transportation Research Part B: Methodological* 45(8):1232–1250. (cited on page 189)

Benders, J. F. 1962. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik* 4:238–252. (cited on pages 20 and 76)

BHP Billiton. 2011. Outer harbour development implementation of best practicable approach to dredging program rev 0. http://www.bhpbilliton.

com/home/aboutus/regulatory/Documents/IronOreOuterHarbourDevelopment/Appendix14BestPracticableDredgingApproach_rev0.pdf. (cited on page 12)

Blum, A. L., and Furst, M. L. 1997. Fast planning through planning graph analysis. *Artificial Intelligence* 90(1-2):279–298. (cited on page 19)

Bonet, B., and Geffner, H. 2001. Planning as heuristic search. *Artificial Intelligence* 129(1-2):5–33. (cited on page 19)

Brafman, R. I., and Domshlak, C. 2006. Factored planning: How, when, and when not. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI'06)*, 809–814. (cited on pages 19 and 20)

Brønmo, G.; Christiansen, M.; Fagerholt, K.; and Nygreen, B. 2007. A multi-start local search heuristic for ship scheduling - a computational study. *Computers & Operations Research* 34(3):900–917. (cited on pages 22, 33, 143, 145, and 188)

Brønmo, G.; Christiansen, M.; and Nygreen, B. 2007. Ship routing and scheduling with flexible cargo sizes. *Journal of the Operational Research Society* 58:1167–1177. (cited on pages 22 and 33)

Brønmo, G.; Nygreen, B.; and Lysgaard, J. 2010. Column generation approaches to ship scheduling with flexible cargo sizes. *European Journal of Operational Research* 200(1):139–150. (cited on pages 22 and 33)

Brown, G.; Kline, J.; Rosenthal, R.; and Washburn, A. 2007. Steaming on convex hulls. *Interfaces* 37(4):342–352. (cited on page 27)

Brown, G.; Graves, G.; and Ronen, D. 1987. Scheduling ocean transportation of crude oil. *Management Science* 33(3):335–346. (cited on pages 22, 26, 31, 33, and 159)

Bunkerworld. 2012. Bunkerworld prices - bunker fuel Singapore, Rotterdam, Houston, Fujairah. http://www.bunkerworld.com/prices. (cited on page 100)

Castellini, C.; Giunchiglia, E.; and Tacchella, A. 2003. SAT-based planning in complex domains: Concurrency, constraints and nondeterminism. *Artificial Intelligence* 147(1-2):85–117. (cited on page 19)

Christiansen, M., and Fagerholt, K. 2002. Robust ship scheduling with multiple time windows. *Naval Research Logistics* 49(6):611–625. (cited on pages 22, 26, 33, 152, 156, 157, 159, and 187)

Christiansen, M.; Fagerholt, K.; Nygreen, B.; and Ronen, D. 2007. Chapter 4: Maritime transportation. In Barnhart, C., and Laporte, G., eds., *Transportation*, volume 14 of *Handbooks in Operations Research and Management Science*. Elsevier. 189–284. (cited on pages 21, 26, and 155)

Christiansen, M.; Fagerholt, K.; Flatberg, T.; Haugen, Ø.; Kloster, O.; and Lunda, E. 2011. Maritime inventory routing with multiple products: A case study from the cement industry. *European Journal of Operational Research* 208(1):86–94. (cited on pages 2, 23, 31, 33, and 89)

Christiansen, M.; Fagerholt, K.; Nygreen, B.; and Ronen, D. 2013. Ship routing and scheduling in the new millennium. *European Journal of Operational Research* 228(3):467–483. (cited on pages 1, 21, 26, and 155)

Christiansen, M.; Fagerholt, K.; and Ronen, D. 2004. Ship routing and scheduling: status and perspectives. *Transportation Science* 38(1):1–18. (cited on pages 1 and 155)

Chu, G.; de la Banda, M.; Mears, C.; and Stuckey, P. 2010. Symmetries and lazy clause generation. In *Proceedings of the 16th International Conference on Principles and Practice of Constraint Programming (CP'10) Doctoral Programme*, 43–48. (cited on pages 19 and 170)

Clarkson Research. 2012. Sources and methods for the Shipping Intelligence Weekly. www.clarksons.net. (cited on page 99)

CoalSpot.com. 2012. Panamax freight rates under pressure from low demand. http://www.coalspot.com/. (cited on page 100)

Coles, A.; Coles, A.; Fox, M.; and Long, D. 2010a. Forward-chaining partial-order planning. In *Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS'10)*, 42–49. (cited on page 19)

Coles, A. J.; Coles, A. I.; Fox, M.; and Long, D. 2010b. Forward-chaining partial-order planning. In *Proceedings of the Twentieth International Conference on Automated Planning and Scheduling (ICAPS-10)*, 42–49. (cited on page 155)

Consensus Group. 2013. NASSCOM Innovation Student Awards winners announced - 2 universities win awards. http://www.consensus.com.au/nasscomawards.html. (cited on pages 4 and 85)

Corbett, J.; Wang, H.; and Winebrake, J. 2009. The effectiveness and costs of speed reductions on emissions from international shipping. *Transportation Research Part D* 14:593–598. (cited on page 27)

Corbett, A. 2009. Safety push to cut traffic at Newcastle. *TradeWinds*. (cited on page 28)

Cordeau, J.-F.; Laporte, G.; Legato, P.; and Moccia, L. 2005. Models and tabu search heuristics for the berth allocation problem. *Transportation Science* 39(4):526–538. (cited on pages 23, 24, 31, and 34)

Curtis, B. 2012. Impact study: estimating cost savings. Personal communication. (cited on pages 96, 97, 101, and 157)

Dantzig, G. B., and Wolfe, P. 1960. Decomposition principle for linear programs. *Operations Research* 8:101–111. (cited on pages 17 and 20)

DryShips Inc. 2012. Baltic dry index daily market report. http://www.dryships.com/index.htm. (cited on page 100)

Du, Y.; Chen, Q.; Quan, X.; Long, L.; and Fung, Y. 2011. Berth allocation considering fuel consumption and vessel emissions. *Transportation Research Part E* 47:1021–1037. (cited on pages 23, 24, 27, and 34)

Endresen, Ø.; Sørgøard, E.; Bakke, J.; and Isaksen, I. 2004. Substantiation of a lower estimate for the bunker inventory: Comment on "Updated emissions from ocean shipping" by James J. Corbett and Horst W. Koehler. *Journal of Geophysical Research* 109:D23302. (cited on page 101)

Fagerholt, K., and Christiansen, M. 2000. A combined ship scheduling and allocation problem. *The Journal of the Operational Research Society* 51(7):834–842. (cited on page 33)

Fagerholt, K.; Laporte, G.; and Norstad, I. 2010. Reducing fuel emissions by optimizing speed on shipping routes. *Journal of the Operational Research Society* 61:523–529. (cited on pages 23, 28, 29, 34, 99, 102, 103, 104, 105, 106, 107, 133, 141, 142, 152, and 185)

Fagerholt, K. 2000. Evaluating the trade-off between the level of customer service and transportation costs in a ship scheduling problem. *Maritime Policy & Management* 27(2):145–153. (cited on pages 22, 25, 33, and 156)

Fagerholt, K. 2001. Ship scheduling with soft time windows: an optimisation based approach. *European Journal of Operational Research* 131(3):559–571. (cited on pages 22, 25, 26, 32, 33, 152, 156, 157, and 159)

Fagerholt, K. 2004. A computer-based decision support system for vessel fleet scheduling - experience and future research. *Decision Support Systems* 37(1):35–47. (cited on pages 1, 2, 22, 31, and 33)

Fan, X.-C.; Li, N.-L.; Zhang, B.-Y.; and Liu, Z.-H. 2011. Research on vehicle routing problem with soft time windows based on tabu search algorithm. In *IEEE 18th International Conference on Industrial Engineering and Engineering Management (IE&EM)*, 420–423. (cited on pages 25, 34, 156, and 159)

Feydy, T., and Stuckey, P. 2009. Lazy clause generation reengineered. In *Proceedings of the 15th International Conference on Principles and Practice of Constraint Programming (CP'09), LNCS*, volume 5732, 352–366. (cited on pages 18, 152, and 170)

Figliozzi, M. 2010. An iterative route construction and improvement algorithm for the vehicle routing problem with soft time windows. *Transportation Research Part C: Emerging Technologies* 18(5):668–679. (cited on pages 24, 25, 34, 156, and 159)

Figliozzi, M. 2012. The time dependent vehicle routing problem with time windows: Benchmark problems, an efficient solution algorithm, and solution characteristics. *Transportation Research Part E: Logistics and Transportation Review* 48(3):616–636. (cited on pages 24, 25, 34, 152, 156, and 159)

Fikes, R., and Nilsson, N. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial intelligence* 2(3-4):189–208. (cited on page 160)

Fisher, M., and Rosenwein, M. 1989. An interactive optimization system for bulk-cargo ship scheduling. *Naval Research Logistics* 36(1):27–42. (cited on pages 22, 31, and 33)

Fox, M., and Long, D. 2003. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research* 20(1):61–124. (cited on page 155)

Franceschetti, A.; Van Woensel, T.; Honhon, D.; Bektaş, T.; and Laporte, G. 2012. The timedependent pollution routing problem. Technical report, Industrial engineering and innovation Sciences, Technology University of Eindhoven, Eindhoven. (cited on page 189)

Gatica, R. A., and Miranda, P. A. 2011. A time based discretization approach for ship routing and scheduling with variable speed. *Networks and Spacial Economics* 11(3):465–485. (cited on pages 22, 28, 33, and 156)

Ghallab, M.; Nau, D.; and Traverso, P. 2004. *Automated planning: theory & practice*. Morgan Kaufmann. (cited on page 20)

Grinold, R. 1972. The payment scheduling problem. *Naval Logistics Research Quarterly* 19(1):123–136. (cited on pages 154 and 159)

Gunnarson, H.; Rönnqvist, M.; and Carlsson, D. 2006. A combined terminal location and ship routing problem. *Journal of the Operational Research Society* 57(8):928–938. (cited on page 188)

Halvorsen-Weare, E. E., and Fagerholt, K. 2013. Routing and scheduling in a liquefied natural gas shipping problem with inventory and berth constraints. *Annals of Operations Research* 203:167–186. (cited on pages 23, 26, 32, 33, and 156)

Hoffmann, J. 2001. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research* 14:253–302. (cited on page 19)

Hooker, J. N., and Ottosson, G. 2003. Logic-based Benders decomposition. *Mathematical Programming* 96:33–60. (cited on pages 20 and 77)

Hooker, J. N. 2007. Planning and scheduling by logic-based Benders decomposition. *Operations Research* 55:588–602. (cited on page 20)

Hwang, H.; Visoldilokpun, S.; and Rosenberger, J. M. 2008. A branch-and-price-and-cut method for ship scheduling with limited risk. *Transportation Science* 42(3):336–351. (cited on pages 33 and 187)

Hwang, S. J. 2005. *Inventory Constrained Maritime Routing and Scheduling for Multi-Commodity Liquid Bulk*. Ph.D. Dissertation, Georgia Institute of Technology. (cited on pages 22, 23, and 33)

Ierapetritou, M. G., and Floudas, C. A. 1998a. Effective continuous-time formulation for short-term scheduling. 1. multipurpose batch processes. *Industrial & Engineering Chemistry Research* 37(11):4341–4359. (cited on page 188)

Ierapetritou, M. G., and Floudas, C. A. 1998b. Effective continuous-time formulation for short-term scheduling. 2. continuous and semicontinuous processes. *Industrial & Engineering Chemistry Research* 37(11):4360–4374. (cited on page 188)

Index Mundi. 2013. Iron ore monthly price. http://www.indexmundi.com/commodities/?commodity=iron-ore. (cited on pages 89 and 97)

Kadioglu, S.; Malitsky, Y.; Sellmann, M.; and Tierney, K. 2010. ISAC – Instance-Specific Algorithm Configuration. In Coelho, H.; Studer, R.; and Wooldridge, M., eds., *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI-10)*, volume 215 of *Frontiers in Intelligence and Applications*, 751–756. (cited on page 168)

Kao, C., and Lee, H. 1996. Discrete time parallel-machine scheduling: a case of ship scheduling. *Engineering Optimization* 26(4):287–294. (cited on pages 23, 24, 32, and 34)

Karlaftis, M. G., and Kepaptsoglou, K. 2009. Containership routing with time deadlines and simultaneous deliveries and pick-ups. *Transportation Research Part E: Logistics and Transportation Review* 45(1):210–221. (cited on pages 25 and 156)

Kautz, H., and Walser, J. 1999. State-space planning by integer optimization. In *Proceedings of the National Conference on Artificial Intelligence*, 526–533. (cited on page 162)

Kelareva, E.; Brand, S.; Kilby, P.; Thiébaux, S.; and Wallace, M. 2012a. CP and MIP methods for ship scheduling with time-varying draft. In *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS'12)*, 110–118. (cited on page 7)

Kelareva, E.; Kilby, P.; Thiébaux, S.; and Wallace, M. 2012b. Ship scheduling with time-varying draft. In *5th International Workshop on Freight Transportation and Logistics (ODYSSEUS'12) Extended Abstracts*, 103–106. (cited on page 7)

Kelareva, E.; Kilby, P.; Thiébaux, S.; and Wallace, M. 2013. Ship speed optimisation with time-varying draft restrictions. In *8th Triennial Symposium on Transportation Analysis (TRISTAN VIII)*. (cited on page 8)

Kelareva, E.; Tierney, K.; and Kilby, P. 2013a. CP methods for scheduling and routing with time-dependent task costs. In Gomes, C., and Sellmann, M., eds., *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, volume 7874 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg. 111–127. (cited on pages 8, 152, and 168)

Kelareva, E.; Tierney, K.; and Kilby, P. 2013b. CP methods for scheduling and routing with time-dependent task costs. *European Journal of Computational Optimization (submitted)*. (cited on pages 8 and 152)

Kelareva, E. 2011. The "DUKC Optimiser" ship scheduling system. In *21st International Conference on Automated Planning and Scheduling (ICAPS'11) System Demonstrations*. (cited on pages 7 and 85)

Kelareva, E. 2012. DUKC Optimiser: Maximising cargo throughput at a bulk export port. In *22nd International Conference on Automated Planning and Scheduling (ICAPS'12) System Demonstrations*. (cited on page 8)

Kilby, P., and Verden, A. 2011. Flexible routing combing constraint programming, large neighbourhood search, and feature-based insertion. In Schill, K.; Scholz-Reiter, B.; and Frommberger, L., eds., *Proceedings 2nd Workshop on Artificial Intelligence and Logistics (AILOG'11)*, 43–49. (cited on pages 25, 34, 151, 156, 159, 178, and 179)

Kim, S.-H., and Lee, K.-K. 1997. An optimization-based decision support system for ship scheduling. *Computers and Industrial Engineering* 33(3-4):689–692. (cited on pages 22 and 33)

Kobayashi, K., and Kubo, M. 2010. Optimization of oil tanker schedules by decomposition, column generation, and time-space network techniques. *Japan Journal of Industrial and Applied Mathematics* 27:161–173. (cited on pages 22 and 33)

Kohl, N.; Larsen, A.; Larsen, J.; Ross, A.; and Tiourine, S. 2007. Airline disruption management âĂŤ perspectives, experiences and outlook. *Journal of Air Transport Management* 13(3):149âĂŞ162. (cited on page 152)

Korsvik, J. E., and Fagerholt, K. 2010. A tabu search heuristic for ship routing and scheduling with flexible cargo quantities. *Journal of Heuristics* 16:117–137. (cited on pages 22 and 33)

Korsvik, J. E.; Fagerholt, K.; and Laporte, G. 2010. A tabu search heuristic for ship routing and scheduling. *Journal of the Operational Research Society* 61:594–603. (cited on pages 31, 33, and 89)

Korsvik, J. E.; Fagerholt, K.; and Laporte, G. 2011. A large neighbourhood search heuristic for ship routing and scheduling with split loads. *Computers & Operations Research* 38:474–483. (cited on pages 22 and 33)

Land, A. H., and Doig, A. G. 1960. An automatic method of solving discrete programming problems. *Econometrica* 28(3):497–520. (cited on page 16)

Lee, Y., and Chen, C.-Y. 2009. An optimization heuristic for the berth scheduling problem. *European Journal of Operational Research* 196(2):500–508. (cited on pages 23, 24, 31, and 34)

Lemaître, M.; Verfaillie, G.; Jouhaud, F.; Lachiver, J.-M.; and Bataille, N. 2002. Selecting and scheduling observations of agile satellites. *Aerospace Science and Technology* 6:367–381. (cited on page 153)

Lin, W.; Liao, D.; Liu, C.; and Lee, Y. 2005. Daily imaging scheduling of an Earth observation satellite. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions* 35(2):213–223. (cited on pages 152, 153, 154, 158, 159, 174, 175, and 177)

Linstad, H.; Asbjørnslett, B.; and Strømman, A. 2011. Reductions in greenhouse gas emissions and cost by shipping at lower speeds. *Energy Policy* 2011(39):3456–3464. (cited on page 27)

Løfstedt, B.; Alvarez, J.; Plum, C.; Pisinger, D.; and Sigurd, M. 2010. An integer programming model and benchmark suite for liner shipping network design. Technical Report 19, DTU Management. (cited on page 27)

Lougee-Heimer, R. 2003. The Common Optimization INterface for Operations Research: Promoting open-source software in the operations research community. *IBM Journal of Research and Development* 47(1):57–66. (cited on page 68)

Malandraki, C., and Daskin, M. S. 1992. Time dependent vehicle routing problems: Formulations, properties and heuristic algorithms. *Transportation Science* 26(3):185. (cited on page 189)

Marriott, K., and Stuckey, P. J. 1998. *Programming with constraints: an introduction*. MIT Press. (cited on page 18)

Meng, Q., and Wang, S. 2011. Optimal operating strategy for a long-haul liner service route. *European Journal of Operational Research* 215:105–114. (cited on pages 23, 27, and 34)

Nau, D.; Ghallab, M.; and Traverso, P. 2004. *Automated Planning: Theory & Practice*. Morgan Kaufmann Publishers Inc. (cited on page 160)

Nethercote, N.; Stuckey, P.; Becket, R.; Brand, S.; Duck, G.; and Tack, G. 2007. MiniZinc: Towards a standard CP modelling language. In Bessière, C., ed., *Principles and Practice of Constraint Programming - CP 2007*, volume 4741 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg. 529–543. (cited on pages 58, 88, 167, and 170)

Nethercote, N.; Marriott, K.; Rafeh, R.; Wallace, M.; and de la Banda, M. 2010. Specification of Zinc and MiniZinc. (cited on pages 58 and 167)

Newcastle Port Corporation. 2010. Vessel arrival system letter and rules. http://www.newportcorp.com.au/client_images/912600.pdf. (cited on page 28)

Norstad, I.; Fagerholt, K.; and Laporte, G. 2011. Tramp ship routing and scheduling with speed optimization. *Transportation Research* 19:853–865. (cited on pages 2, 4, 5, 22, 28, 29, 33, 99, 102, 103, 105, 106, 107, 108, 109, 111, 119, 121, 140, 141, 142, 143, 148, 149, 152, 157, 159, 185, and 186)

Notteboom, T. E., and Vernimmen, B. 2009. The effect of high fuel costs on liner service configuration in container shipping. *Journal of Transport Geography* 17:325–337. (cited on page 27)

O'Brien, T. 2002. Experience using dynamic underkeel clearance systems. In *Proceedings of the PIANC 30th International Navigational Congress*, 1793–1804. (cited on pages 12 and 13)

Ohrimenko, O.; Stuckey, P.; and Codish, M. 2009. Propagation via lazy clause generation. *Constraints* 14(3):357–391. (cited on pages 18, 19, 151, 154, and 170)

OMC International. 2009. DUKC helps Port Hedland set ship loading record. http://www.omc-international.com/images/stories/press/omc-20090810-news-in-wa.pdf. (cited on page 58)

OMC International. 2013. OMC International public website. http://www.omc-international.com.au. (cited on page 38)

Opturion Pty Ltd. 2013. Opturion CPX User's Guide, Version 1.0. (cited on page 172)

Penberthy, J., and Weld, D. 1992. UCPOP: A sound, complete, partial order planner for ADL. In *Proceedings of the 3rd International Conference on Knowledge Representation and Reasoning*. (cited on page 160)

Port Hedland Port Authority. 2011a. 2009/10 cargo statistics and port information. http://www.phpa.com.au/docs/CargoStatisticsReport.pdf. (cited on pages 42, 89, and 96)

Port Hedland Port Authority. 2011b. Dynamic under keel clearance system. http://www.phpa.com.au/dukc_information.asp. (cited on pages 42, 89, 93, 96, and 157)

Port Hedland Port Authority. 2012. Port Hedland Port Authority 2011/12 cargo statistics & port information. http://www.phpa.com.au/About-the-Port/Statistics/Cargo-statistics-and-port-information/PDF-File/CargoStatisticsReport2012.aspx. (cited on page 37)

Port Hedland Port Authority. 2013a. Current fees. http://www.phpa.com.au/Port-operations/Port-charges/Current-fees.aspx. (cited on page 11)

Port Hedland Port Authority. 2013b. DUKC System. http://www.phpa.com.au/Port-operations/Dynamic-under-keel-clearance-(DUKC)-system.aspx. (cited on page 38)

Port Hedland Port Authority. 2013c. PHPA port user guidelines and procedures: PR – MO004. http://www.phpa.com.au/Port-operations/Port-user-guidelines-and-procedures/PDF-Files/MO_PR_PortUsers.aspx. (cited on pages 11, 39, 48, and 56)

Port Hedland Port Authority. 2013d. Port profile and handbook. http://www.phpa.com.au/About-the-Port/Port-profile-and-handbook.aspx. (cited on pages 41 and 83)

Ports Australia. 2012. Bulk cargo (mass tonnes) for 2011/2012. http://www.portsaustralia.com.au/tradestats. (cited on page 147)

Psaraftis, H., and Kontovas, C. 2009. Ship emissions: Logistics and other tradeoffs. In *10th International Marine Design Conference (IMDC'09)*. (cited on page 99)

Psaraftis, H. N., and Kontovas, C. A. 2013. Speed models for energy-efficient maritime transportation: A taxonomy and survey. *Transportation Research Part C: Emerging Technologies* 26:331–351. (cited on pages 26 and 29)

Psaraftis, H. N. 2012. A ship pickup and delivery model with multiple commodities, variable speeds, cargo inventory costs and freight rates. In *5th International Workshop on Freight Transportation and Logistics (ODYSSEUS'12) Extended Abstracts*, 467–470. (cited on pages 22, 28, 29, and 33)

Qureshi, A.; Taniguchi, E.; and Yamada, T. 2009. An exact solution approach for vehicle routing and scheduling problems with soft time windows. *Transportation Research Part E: Logistics and Transportation Review* 45(6):960–977. (cited on pages 24, 25, 34, 156, 159, and 175)

Rakke, J. G.; Stålhane, M.; Moe, C. R.; Christiansen, M.; Andersson, H.; Fagerholt, K.; and Norstad, I. 2011. A rolling horizon heuristic for creating a liquefied natural gas annual delivery program. *Transportation Research Part C* 19:896–911. (cited on pages 2, 23, 33, and 158)

Rakke, J.; Christiansen, M.; Fagerholt, K.; and Laporte, G. 2012. The Traveling Salesman Problem with draft limits. *Computers & Operations Research* 39:2161–2167. (cited on pages 23, 31, 32, 34, 103, and 187)

Rao, L.; Liu, X.; Xie, L.; and Liu, W. 2010. Minimizing electricity cost: Optimization of distributed internet data centers in a multi-electricity-market environment. In *INFOCOM*, 1145–1153. (cited on page 153)

Rintanen, J. 2010. Heuristics for planning with SAT. In *Principles and Practice of Constraint Programming - CP 2010*, volume 6308 of *Lecture Notes in Computer Science*. Springer. 414–428. (cited on page 19)

Rodrigue, J.-P.; Comtois, C.; and Slack, B. 2013. *The Geography of Transport Systems, Third Edition*. Routledge. (cited on page 10)

Ronen, D. 1982. The effect of oil price on the optimal speed of ships. *Journal of the Operational Research Society* 33:1035–1040. (cited on pages 23, 26, 34, and 99)

Ronen, D. 2011. The effect of oil price on containership speed and fleet size. *Journal of the Operational Research Society* 62:211–216. (cited on pages 26, 27, 34, and 101)

Rosenberger, J.; Johnson, E.; and Nemhauser, G. 2003. Rerouting aircraft for airline recovery. *Transportation Science* 37(4):408–421. (cited on page 152)

Rossi, F.; Van Beek, P.; and Walsh, T. 2006. *Handbook of constraint programming*. Elsevier Science. (cited on page 18)

Russell, A. 1970. Cash flows in networks. *Management Science* 16(5):357–373. (cited on pages 152 and 154)

Ryder, S., and Chappell, D. 1980. Optimal speed and ship size for the liner trades. *Maritime Policy & Management* 7(1):55–57. (cited on page 99)

Schrijver, A. 1986. *Theory of Linear and Integer Programming*. Wiley and Sons. (cited on page 17)

Schutt, A.; Chu, G.; Stuckey, P.; and Wallace, M. 2012. Maximising the net present value for resource-constrained project scheduling. In *Proceedings of the 3rd International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR'06)*, volume 7298 of *LNCS*. 362–378. (cited on pages 18, 152, 154, 159, 170, and 172)

Sexton, T., and Choi, Y. 1985. Pickup and delivery of partial loads with soft time windows. *American Journal of Mathematical and Management Science* 6:369–398. (cited on page 152)

Shaw, P. 1997. A new local search algorithm providing high quality solutions to vehicle routing problems. Working paper, University of Strathclyde, Glasgow, Scotland. (cited on page 25)

Song, J.-H., and Furman, K. 2010. A maritime inventory routing problem: Practical approach. *Computers & Operations Research*. (cited on pages 23, 31, and 33)

Stahlbock, R., and Voss, S. 2008. Operations research at container terminals: a literature update. *OR Spectrum* 30:1–52. (cited on page 21)

Stuckey, P.; de la Banda, M.; Maher, M.; Marriott, K.; Slaney, J.; Somogyi, Z.; Wallace, M.; and Walsh, T. 2005. The G12 Project: Mapping solver independent models to efficient solutions. In Gabbrielli, M., and Gupta, G., eds., *Logic Programming*, volume 3668 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg. 9–13. (cited on pages 59, 68, 85, and 88)

Tengku-Adnan, T.; Sier, D.; and Ibrahim, R. N. 2009. Performance of ship queuing rules at coal export terminals. In *Proceedings of the 2009 IEEE International Conference on Industrial Engineering and Engineering Management*, 1795–1799. (cited on page 47)

Tierney, K., and Jensen, R. 2011. Liner shipping fleet repositioning. In *International Conference on Computational Logistics (ICCL'11)*. (cited on pages 156 and 159)

Tierney, K., and Jensen, R. M. 2012. The liner shipping fleet repositioning problem with cargo flows. In Hu, H.; Shi, X.; Stahlbock, R.; and Voß, S., eds., *Computational Logistics*, volume 7555 of *LNCS*. 1–16. (cited on pages 34 and 155)

Tierney, K.; Coles, A.; Coles, A.; Kroer, C.; Britt, A.; and Jensen, R. 2012a. Automated planning for liner shipping fleet repositioning. In *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS'12)*, 279–287. (cited on pages xv, 20, 27, 29, 30, 34, 155, 158, 159, 160, 161, 167, and 175)

Tierney, K.; Coles, A.; Coles, A.; and Jensen, R. 2012b. A PDDL domain of the liner shipping fleet repositioning problem. Technical Report TR-2012-152, IT University of Copenhagen. (cited on page 160)

Tierney, K.; Áskelsdóttir, B.; Jensen, R.; and Pisinger, D. 2013. Solving the liner shipping fleet repositioning problem with cargo flows. *Under Revision at Transportation Science*. (cited on pages 34, 155, and 156)

Tierney, K.; Voß, S.; and Stahlbock, R. 2013. A mathematical model of inter-terminal transportation. *To Appear in the European Journal of Operational Research*. (cited on page 189)

UNCTAD. 2011. Review of maritime transport. United Nations, New York and Geneva. (cited on page 1)

UNCTAD. 2012. Review of maritime transport. United Nations, New York and Geneva. (cited on page 10)

University of Melbourne. 2011. MiniZinc Challenge 2011. http://www.g12.cs.mu.oz.au/minizinc/challenge2011/challenge.html. (cited on page 56)

Van Den Briel, M.; Vossen, T.; and Kambhampati, S. 2005. Reviving integer programming approaches for AI planning: A branch-and-cut framework. In *Proceedings of the 15th International Conference on Automated Planning and Scheduling (ICAPS-05)*, 310–319. (cited on page 162)

Vanhoucke, M.; Demeulemeester, E.; and Herroelen, W. 1999. On maximising the net present value of a project under resource constraints. Research Report 9915, Department of Applied Economics, Katholieke Universiteit Leuven, Belgium. (cited on page 154)

Vanhoucke, M.; Demeulemeester, E.; and Herroelen, W. 2001. On maximizing the net present value of a project under renewable resource constraints. *Management Science* 47:1113–1121. (cited on pages 154 and 159)

Vernimmen, B.; Dullaert, W.; and Engelen, S. 2007. Schedule unreliability in liner shipping: Origins and consequences for the hinterland supply chain. *Maritime Economics & Logistics* 9:193–213. (cited on page 99)

Vilhelmsen, C.; Lusby, R.; and Larsen, J. 2013. Routing and scheduling in tramp shipping - integrating bunker optimization. Technical Report. (cited on pages 29, 32, and 33)

Wallace, M. 2009. G12 - towards the separation of problem modelling and problem solving. In *Proceedings of the 6th International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR'09)*, volume 5547 of *LNCS*. 8–10. (cited on pages 167 and 170)

Wang, J.; Jing, N.; Li, J.; and Chen, H. 2007. A multi-objective imaging scheduling approach for Earth observing satellites. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation (GECCO'07)*, 2211–2218. (cited on pages 153 and 159)

Wijnolst, N., and Wergeland, T. 1997. *Shipping*. Delft University Press. (cited on page 101)

Wolfe, W., and Sorensen, S. 2000. Three scheduling algorithms applied to the Earth observing systems domain. *Management Science* 46(1):148–168. (cited on pages 152, 153, 154, 158, and 159)

Wolsey, L. A. 1998. *Integer Programming*. Wiley and Sons. (cited on page 17)

Yan, W.; Bian, Z.; Chang, D.; and Huang, Y. 2009. An improved particle swarm optimization algorithm for solving dynamic tugboat scheduling problem. In *Proceedings of the 2nd International Conference on Power Electronics and Intelligent Transportation System (PEITS'09)*, 433 – 437. (cited on page 48)

Yao, F.; Li, J.; Bai, B.; and He, R. 2010. Earth observation satellites scheduling based on decomposition optimization algorithm. *International Journal of Image, Graphics and Signal Processing* 1:10–18. (cited on pages 153, 158, and 159)