Solving The Liner Shipping Fleet Repositioning Problem with Cargo Flows

Kevin Tierney
Björg Áskelsdóttir
Rune Møller Jensen
David Pisinger

IT University Technical Report Series **TR-2013-165**

ISSN 1600–6100 **January 2013**

ISSN 1600–6100

ISBN 978-87-7949-275-2

Copies may be obtained by contacting:

IT University of Copenhagen
Rued Langgaards Vej 7
DK-2300 Copenhagen S
Denmark

Telephone:	+45 72 18 50 00
Telefax:	+45 72 18 50 01
Web	`www.itu.dk`

Solving The Liner Shipping Fleet Repositioning Problem with Cargo Flows

Kevin Tierney
Björg Áskelsdóttir
Rune Møller Jensen
David Pisinger

Abstract

We solve a central problem in the liner shipping industry called the Liner Shipping Fleet Repositioning Problem (LSFRP). The LSFRP poses a large financial burden on liner shipping firms. During repositioning, vessels are moved between routes in a liner shipping network. Shippers wish to reposition vessels as cheaply as possible without disrupting the cargo flows of the network. The LSFRP is characterized by chains of interacting activities with a multi-commodity flow over paths defined by the activities chosen. Despite its industrial importance, the LSFRP has received little attention in the literature. We introduce a novel mathematical model and a simulated annealing algorithm of the LSFRP with cargo flows that makes use of a carefully constructed graph and evaluate them on real world data from our industrial collaborator. Additionally, we compare our approach's performance against an actual repositioning scenario, one of many undertaken by our industrial collaborator in 2011, in which our simulated annealing algorithm is able to double the profit earned in our industrial collaborator's solution to $15.5 million dollars using only few minutes of CPU time, showing that our algorithm could be used in a decision support system to solve the LSFRP.

1 Introduction

Responsible for transporting over 1.3 billion tons of cargo in 2011 (UNCTAD 2011), liner shipping networks reliably and cheaply connect the world's markets. Vessels are regularly repositioned between services in liner shipping networks to adjust the networks to the world economy and to stay competitive. Since repositioning a single vessel can cost hundreds of thousands of US dollars, optimizing the repositioning activities of vessels is an important problem for the liner shipping industry.

The Liner Shipping Fleet Repositioning Problem (LSFRP) consists of finding sequences of activities that move vessels between services in a liner shipping network while respecting the cargo flows of the network. The LSFRP maximizes the profit earned on the subset of the network affected by the repositioning, balancing sailing costs and port fees against cargo and equipment revenues, while respecting important liner shipping specific constraints dictating the creation of services and movement of cargo. A unique feature of the LSFRP is the state-based nature of the activities in the problem. Many LSFRP activities span multiple physical locations and depend on the location of vessels in order to be performed. Automated planning techniques were used to represent a high-level version of the LSFRP that ignored cargo flows in (Tierney et al., 2012). Cargo flows, however, are an important aspect of the LSFRP that drive decisions on how vessels should be repositioned.

To this end, we present a novel mathematical model of the LSFRP with cargo flows on top of a detailed graph that embeds many LSFRP constraints. We solve our model using CPLEX and with a simulated annealing (SA) approach, and study the performance of our model on real world data from our industrial collaborator. We investigate the scaling performance of our model on an actual repositioning scenario in addition to several constructed scenarios. We provide an overview of our parameter tuning procedures for the SA, as well as a comparison of the SA to a reference solution for our actual repositioning instance. This instance models the decisions available to repositioning coordinators as they planned an actual repositioning at Maersk Line involving 11 vessels in 2011. On this instance, our SA is able to find a solution with a profit between $300,000 and $7 million higher than the reference solution in only a couple minutes, doubling the profit earned in the scenario. It takes a repositioning coordinator several days to find a solution to such a repositioning scenario by hand. Our simulated annealing approach is often able to find the optimal solution or very close to the optimal solution and quickly finds solutions for instances that are too large for CPLEX to solve.

We seek to provide an algorithm capable of functioning within a decision support system (DSS). Within a DSS, the user requires quick answers for planning ship routes. In some cases, the user may need to give feedback to the algorithm, such as not allowing a particular port call or adding extra buffer time on a sailing, so that the plan is real-world feasible. We therefore seek to solve LSFRP instances within a ten minute window.

This paper is structured as follows. We first present a detailed description of the LSFRP in Section 2, including a overview of related work in Section 2.1. Section 3 contains our mathematical model of the LSFRP and graph description. Our SA approach is described in Section 4, followed by a description of our benchmark and a computational evaluation in Section 5. A comparison to our industrial collaborator's repositioning scenario is presented in Section 5.6. Finally, we conclude in Section 6 and present directions for future work. Parts of this paper appeared as an extended abstract in (Tierney and Jensen, 2012).

2 Liner Shipping Fleet Repositioning

Liner shipping networks consist of a set of cyclical routes, called services, that visit ports on a regular, usually weekly, schedule. Liner shipping networks are designed to serve customer's cargo demands, but over time the economy changes and liner shippers must adjust their networks in order to stay competitive. Liner shippers add, remove and modify existing services in their network in order to make changes to the network. Whenever a new service is created, or an existing service is expanded, vessels must be *repositioned* from their current service to the service being added or expanded. Vessel repositioning is expensive due to the cost of fuel (in the region of hundreds of thousands of dollars) and the revenue lost due to cargo flow disruptions. Given that liner shippers around the world reposition hundreds of vessels per year, optimizing vessel movements can significantly reduce the economic and environmental burdens of containerized shipping, and allow shippers to better utilize repositioning vessels to transport cargo.

The aim of the LSFRP is to maximize the profit earned when repositioning a number of vessels from their initial services to a service being added or expanded, called the goal service. We focus on the case where a new service is being added to the network because expanding a service can be seen as a special case of adding a new service, in which vessels are repositioned from the service being expanded to itself along with extra vessels from elsewhere in the network.

Liner shipping services are composed of multiple *slots*, each of which represents a cycle that is assigned to a particular vessel. Each slot is composed of a number of *visitations*, which can be thought of as port calls, i.e. a specific time when a vessel is scheduled to arrive at a port. A vessel that is assigned to a particular slot sequentially sails to each visitation in the slot. Figure 1 shows a schedule of an example service that contains three slots and visits five ports. The service requires three weeks to complete a cycle, and therefore needs three vessels in order to maintain weekly frequency. Each line (black, dark gray, light gray) represents a slot, and each dot is a visitation at a port at a particular time.

Vessel sailing speeds can be adjusted throughout repositioning to balance cost savings with punctuality. The bunker fuel consumption of vessels increases cubically with the speed of the vessel. *Slow steaming*, in which vessels sail near or at their minimum speed, therefore, allows vessels to sail cheaper between two ports than at higher speeds, albeit with a longer duration. We linearize the bunker consumption of each repositioning vessel in order to more easily model the LSFRP.

Phase-out & Phase-in

The repositioning period for each vessel starts at a specific time when the vessel may cease normal operations, that is, it may stop sailing to scheduled visitations and go somewhere else. Each vessel is assigned a different time when it may begin its repositioning, or *phase-out* time. After this time, the vessel may undertake a number of different activities to reach its goal service at low cost. In order to complete the repositioning, each vessel must *phase in* to a

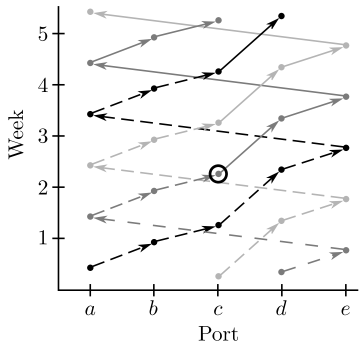


Figure 1: A time-space graph of a service with three vessels, with a latest phase-in requirement of port c in week 3.

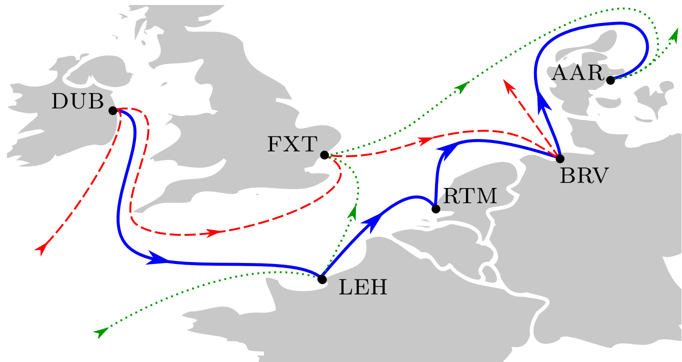


Figure 2: An example repositioning (blue) from a vessel's phase-out service (red, dashed) to its phase-in service (green, dotted).

slot on the goal service before a time set by the repositioning coordinator. After this time, normal operations on the goal service are set to begin, and all scheduled visitations on the service are to be undertaken. In other words, the repositioning of each vessel and optimization of its activities takes place in the period between two fixed times, the vessel's earliest phase-out time and the latest phase-in time of all vessels.

See again Figure 1, which also shows a phase-in service with a phase-in deadline at port c in week 2. The solid lines connect all the visitations that must be undertaken, whereas the dashed lines connect visitations that will only be carried out if they are profitable during the repositioning.

Within a *trade zone*, which is a contiguous geographical area, vessels may sail freely from their initial service to goal service, as well as back from the goal service to the initial service. However, if two ports lie in different trade zones, vessels may only sail between them when going from the initial service to the goal service. Trade zone restrictions ensure cargo is not brought to places that would violate the law, as well as help keep vessels from experiencing unexpected delays. Many countries have laws, called *cabotage restrictions*, that prevent foreign flagged vessels from offering domestic cargo services. Cabotage restrictions are taken into account during network design, but when vessels are repositioned, their altered paths could result in a violation. Additionally, cargo from certain customers may not be brought on ships that visit certain countries (such as military cargo). We can avoid a detailed modeling of such laws by simply restricting vessels to not leave trade zones when transferring between the phase-in and phase-out services.

When a port is visited that is not in the initial or goal service, or is visited out of order, it is called an *inducement*. If a port on the initial or goal service is left off of the repositioning vessel's schedule, it is called an *omission*. Figure 2 shows a vessel's repositioning (solid line) from its initial service (dashed) to its goal service (dotted) within a trade zone. Although FXT is on both the goal and initial services, it is omitted from the repositioning. Note also that the ports RTM and BRV are induced onto the repositioning path. This is only possible because the induced ports are in the same trade zone as LEH and AAR.

Cargo and Equipment

Revenue is earned through delivering cargo and equipment (typically empty containers). We use a detailed view of cargo flows. Cargo is represented as a set of port to port demands with a cargo type, a latest delivery time, an amount of TEU[1] available, and a revenue per TEU delivered. We subtract the cost of loading and unloading each TEU from the revenue to determine the profit per TEU of a particular cargo demand. In contrast to cargo, which can be seen as a multi-commodity flow where each demand is a commodity with a start and end port, equipment can be sent from any

[1]TEU stands for *twenty-foot equivalent unit* and represents a single twenty-foot container.

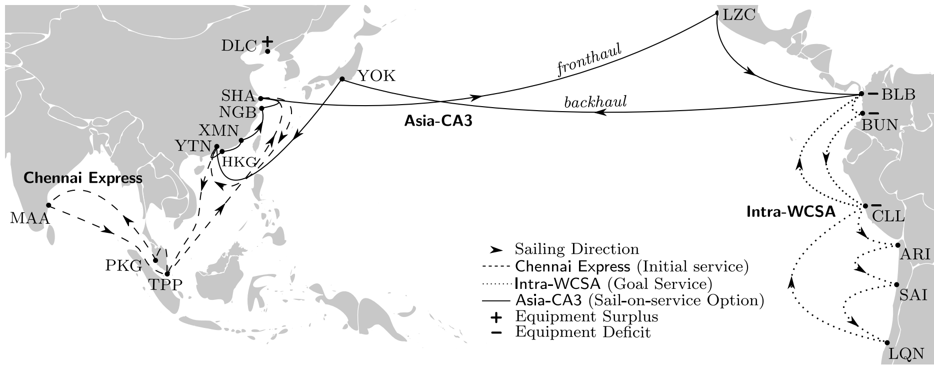


Figure 3: A subset of the case study we performed with our industrial collaborator.

port where it is in surplus to any port where it is in demand. Ports which have an equipment surplus or deficit can be considered to have an infinite amount of supply or demand for a particular type of equipment. This is reasonable since the amount of extra containers on-hand or that are required tends to be much greater than the size of a vessel. Each piece of equipment brought from a port where it is in excess to a port where it is needed earns a small revenue. The revenue earned is an estimation of how much money was saved by bringing the equipment on a repositioning vessel instead of moving the equipment through other, more expensive, means.

We consider both *dry* and *reefer* (refrigerated) cargo. Dry containers are standard containers with no specific handling requirements. Reefer containers, in contrast, must be stowed on a vessel in a location with a plug in order to keep the refrigeration unit running. There are, therefore, different capacities on a vessel for dry and reefer containers. Note that although equipment can consist of both dry and reefer containers, reefer equipment does not require a reefer slot on a vessel, as containers without any cargo do not need any power source. It is still important to differentiate between dry and reefer equipment as they are not interchangeable for customers.

Some ports have equipment, but are not on any service visited by repositioning vessels. These ports are called *flexible* ports, and are associated with flexible visitations. The repositioning coordinator may choose the time a vessel arrives at such visitations, if at all. All other visitations are called *inflexible*, because the time a vessel arrives is fixed.

Sail-on-service (SOS) Opportunities

While repositioning, vessels may use certain services to cheaply sail between two parts of the network. These are called *SOS opportunities*. There are two vessels involved in SOS opportunities, referred to as the *repositioning vessel*, which is the vessel under the control of a repositioning coordinator, and the *on-service vessel*, which is the vessel assigned to a slot on the service being offered as an SOS opportunity. Repositioning vessels can use SOS opportunities by replacing the on-service vessel and sailing in its place for a portion of the service. SOS opportunities save significant amounts of money on bunker fuel, since one vessel is sailing where there would have otherwise been two. Using an SOS can even earn money from the *time-charter bonus*, which is money earned by the liner shipper if the on-service vessel is leased.

Consider Figure 3, in which the Asia-CA3 service is offered as a SOS opportunity to the vessel repositioning from Chennai Express to Intra-WCSA. The repositioning vessel can leave the Chennai Express at TPP, and sail to HKG where it picks up the Asia-CA3, replacing the on-service vessel. The repositioning vessel then sails along the Asia-CA3 until it gets to BLB where it can join the Intra-WCSA. Note that no vessel sails on the backhaul of the Asia-CA3, and this is allowed because very little cargo travels on the Asia-CA3 towards Asia.

When a repositioning vessel uses an SOS opportunity, the on-service vessel is either laid-up or leased out, freeing a slot on the service. The repositioning vessel may join the freed slot in any of the *starting visitations* and may leave the slot in one of the *ending visitations*. There are two ways for repositioning vessels to start an SOS: *transshipment* and *parallel sailing*. When starting an SOS by transshipment, all cargo loaded on the on-service vessel is transshipped (moved) to the repositioning vessel. Each TEU transshipped has a fee roughly equal the cost of loading a TEU.

Transshipment is not always possible due to the previously described cabotage restrictions that exist in some countries, or prohibitively expensive. Illegal or expensive transshipments can be avoided using a *parallel sailing*, in which both the repositioning vessel and the on-service vessel visit ports in tandem. The repositioning vessel only loads cargo, and the on-service vessel only discharges cargo. Since two vessels are sailing in tandem during a parallel sailing, fuel consumption is doubled, as are port fees. However, this is still sometimes cheaper than transshipping cargo directly between vessels.

2.1 Literature Review

The LSFRP has received little attention in the literature and was not mentioned in either of the most influential surveys of work in the liner shipping domain (Christiansen et al., 2007, 2004) or container terminals (Stahlbock and Voß, 2008). Although there has been significant work on problems such as the Fleet Deployment Problem (FDP) (e.g., Powell and Perakis (1997)) and the Network Design Problem (NDP) (e.g. Agarwal and Ergun (2008); Álvarez (2009)), these problems deal with strategic decisions related to building the network and assigning vessels to services, rather than the operational problem of finding paths for vessels through the network.

Although tramp shipping problems, such as (Christiansen, 1999; Korsvik et al., 2011), maximize cargo profit while accounting for sailing costs and port fees as in the LSFRP, they lack liner shipping specific constraints, such as sail-on-service opportunities, phase-in requirements and strict visitation times. Airline disruption management (see Clausen et al. (2010); Kohl et al. (2007)), while also relying on time-based graphs, differs from the LSFRP in two key ways. First, airline disruption management requires an exact cover of all flight legs over a planning horizon. The LSFRP has no such requirement over visitations or sailing legs. Second, there are no flexible visitations in airline disruption management.

The liner shipping vessel schedule recovery problem (LSVSRP) (Brouer et al., 2013) focuses on recovering operations after a disruption, such as bad weather or mechanical failure, delays a container vessel. Similar to the LSFRP, the LSVSRP must respect the weekly frequency of services and network cargo flows. However, the two problems differ in that the LSVSRP lacks many cost saving aspects of the LSFRP because it is solved over a much shorter time window with only a single vessel.

Andersen's PhD thesis (Andersen, 2010) discusses a problem similar to the LSFRP, called the Network Transition Problem (NTP). No mathematical model or formal problem description is provided, so it is difficult to exactly ascertain what the NTP solves in comparison to the LSFRP. However, it is clear that the NTP lacks cost saving activities like SOS opportunities, empty equipment flows and slow steaming.

The primary previous work on the LSFRP in the literature is (Tierney et al., 2012) and (Tierney and Jensen, 2012), by the authors. The first work on the LSFRP, (Tierney et al., 2012), solved an abstraction of the LSFRP without cargo/equipment flows and SOS parallel sailings using Linear Temporal Optimization Planning (LTOP), a hybrid of automated planning and linear programming that performs a branch-and-bound search for repositioning solutions. However, LTOP and other automated planning methods are unable to model cargo flows and are thus inapplicable to the LSFRP with cargo flows.

A mathematical model of the LSFRP with cargo and equipment flows is introduced in (Tierney and Jensen, 2012), and CPLEX is used to solve the model. While CPLEX is able to solve a number of instances to optimality, many instances are too large for CPLEX to tackle. This paper extends that work with some model improvements, a public dataset, and a heuristic approach that solves the instances that are too big for CPLEX to solve.

3 Mathematical Model

We model the LSFRP with cargo flows on a graph $G = (V, A)$, where V is the set of nodes and A the set of directed arcs between nodes. Each node in V represents a *visitation* of a vessel at a particular port[2], and each arc in A represents an allowed sailing between two visitations. The graph encompasses all of the activities each vessel may undertake during a fixed *repositioning period*, which is the period from the time the vessel is first allowed to leave its phase-out service until the time when normal operations must begin on the phase-in service. The path of each

[2]We use the terms visitation and node interchangeably.

vessel through the graph represents the activities to be undertaken by that vessel, and we therefore require the paths to be node disjoint to prevent multiple vessels from performing the same activity. This is an important constraint because *i*) container port terminals assign timeslots to vessels, meaning there is not enough room for two vessels, and *ii*) profit from carrying cargo can only be earned a single time, removing any reason for multiple vessels to visit the same node. Note that flexible visitations, i.e. visitations without a prior fixed schedule, can be undertaken by multiple vessels, even simultaneously. For ease of modeling, we therefore replicate flexible visitations for each vessel and consider them as node disjoint. We give more details about this process (and justifications) later. We embed a number of problem constraints and objectives directly in the graph, including sailing costs, sail-on-service opportunities, cabotage restrictions, phase-in/out requirements, and canal fees, which are described in detail in the next section, followed by our MIP model over the graph.

3.1 Graph Description

We give a textual overview of the graph used in our model of the LSFRP with cargo flows. For a detailed formal description, we refer the reader to Appendix A. The visitations in the graph are split into two sets, thus $V = V^i \cup V^f$, where V^i is the set of *inflexible* visitations, i.e. visitations associated with a specific port call time, and V^f is the set of *flexible* visitations, which are assigned a time only if a vessel performs the visitation. The set V^f contains visitations in which a vessel can pick up/deliver equipment or incremental cargo that are not on any phase-out, phase-in, or SOS service. Let S be the set of ships.

The overall structure of the graph involves four types of visitations: phase-out, phase-in, flexible, and SOS visitations. In addition to these visitations, we include a graph sink, τ, which all vessels must reach for a valid repositioning. We let $V' = V \setminus \tau$ be the set of all graph visitations excluding τ. The four types of visitations represent four disjoint sets that make up V'. We now describe the arc structure present in each of the four types of visitations.

Phase-out

Each ship is assigned a particular visitation, $\sigma_s \in V'$, at which the ship $s \in S$ begins its repositioning. This visitation represents the earliest allowed phase-out time for that vessel. A visitation is then created for each subsequent port call of the ship on in its phase-out slot. Each phase-out visitation is connected to the next one with an arc. Note that phase-out visitations do not connect to the phase-out visitations of other ships.

Vessels may leave phase-out nodes to sail to SOS opportunities, flexible nodes, or to a phase-in slot. Thus, arcs are created from each phase-out visitation to each phase-in visitation and SOS start visitation such that sailing between the visitations is temporally feasible (i.e. the starting time of the phase-in/SOS visitation is greater than the end time of the phase-out visitation plus the sailing time). Since flexible nodes have no fixed start and end time, arcs are created to and from all flexible nodes to all phase-outs within the same trade zone. Finally, phase-out visitations have incoming arcs from phase-in visitations in the same trade zone. This allows ships to avoid sailing back and forth between ports when transferring directly between the phase-out and phase-in.

Phase-in

We create visitations for each port call along a phase-in slot, and connect subsequent phase-in visitations to each other. The final visitation in a slot, which represents the time at which regular operations must begin on a service, is connected to the graph sink, τ.

The phase-in graph structure ensures that the goal service has a vessel in each of its slots. An example phase-in graph structure is portrayed in Figure 4, which shows the graph for the service in Figure 1. Each sequence of visitations (colored light gray, dark gray, and black) represents a slot on the goal service. Each visitation is labeled with the port and week that it is visited. The last node in each sequence corresponds to the on-time requirement (node $(c, 2)$) extended to each slot. After each of these visitations, the service begins normal operations, and is no longer under the control of the repositioning coordinator. This graph structure ensures that all vessels perform a legal phase-in, namely that each slot is assigned a single vessel. Each phase-in slot is guaranteed to be assigned a single vessel since there are as many slots as there are vessels (three), the graph sink τ only has a single incoming node from each slot, and the paths of vessels are node disjoint (except for τ).

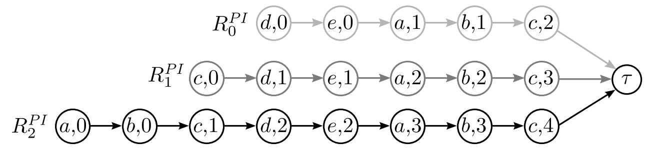


Figure 4: The phase-in graph structure for the service in Figure 1. The sets R_0^{PI}, R_1^{PI}, and R_2^{PI} contain the nodes for each phase-in slot. Each set of nodes ends with a single visitation at port c on weeks 2, 3 and 4, ensuring that the weekly structure of the service is enforced.

Flexible visitations

Flexible visitations are modeled by replicating each flexible visitation for each ship in the model. Flexible visitations are connected to all inflexible and flexible visitations within the same trade zone. Replicating flexible visitations for each vessel avoids requiring special constraints in the MIP model to handle the fact that multiple vessels can visit the same flexible visitation. This is because when a vessel visits a flexible visitation, the visitation must be assigned a time when it can take place. Simply copying the flexible nodes ensures that each flexible node can be scheduled along the path of a vessel with simple constraints. Since our instances generally do not contain many flexible visitations, this duplication does not significantly hinder the solvability of the instances. Note that this opens the possibility that two vessels may visit the same flexible visitation at the same time. We do not consider this to be a problem since flexible visitations are at ports that will probably have the capacity to deal with multiple ships. Since flexible visitations do not have fixed entry and exit times, the time required for a vessel to visit them must be taken into account. The total port stay at a flexible visitation consists of the *piloting time*, which is the time required to maneuver the vessel in to, and out of, a port, and the cargo/equipment (un)loading time.

Sail-on-service

We introduce a number of disjoint sets of graph arcs and graph nodes to represent a special graph structure that models SOS opportunities. We view an SOS as having three types of ports; *entry ports*, where vessels may join the SOS, *through ports*, in which a vessel must already be on the SOS, and *end ports* where a vessel may leave the SOS. The designations of these ports is left to the repositioning coordinator. We make this distinction in case there are circumstances outside the scope of the model that require certain ports to be called if an SOS is used.

Figure 5 shows the graph structure of an example SOS opportunity, o. Vessels may enter the SOS using arcs in the set $\hat{A}_o^{In}$, either through parallel sailing nodes (O_o^P) or transshipment nodes (O_o^{TS}), shown in red and green, respectively. Parallel sailings end in a transshipment, in which cargo is moved to the repositioning vessel. The parallel sailing nodes are connected to the transshipment nodes with the set of arcs $\hat{A}_o^{PTS}$. The set of arcs $\hat{A}_o^{PP}$ contains arcs connecting subsequent ports for parallel sailings. These arcs have twice the sailing cost for each vessel as the other arcs in the SOS graph structure, which have the normal sailing cost between two ports for each vessel. Note that p_3 has no parallel sailing node because transshipment is not allowed in p_4, which is a through port. Ports with cabotage restrictions, such as p_1, do not receive transshipment nodes, as transshipping at that particular port would violate the law. Transshipment nodes connect to through nodes (blue) using the arcs $\hat{A}_o^{TST}$. Once at a through node, a vessel must sail onward to the next through node using the arc set $\hat{A}_o^{TT}$, until it reaches the arc $\hat{a}_o^{TE}$. This arc connects the through nodes to the end nodes, and represents a bottleneck that ensures only one vessel uses a particular SOS. Since the paths of the vessels are node disjoint, two vessels would have to visit the latest node in O_o^T and the earliest node in O_o^E, which is not allowed. Finally, vessels may exit the SOS through the end nodes (orange) in the set O_o^E, using an arc in the set $\hat{A}_o^{Out}$. End nodes are connected by the arcs in the set $\hat{A}_o^{EE}$.

Sailing Cost

The fuel consumption of a ship is a cubic function of the speed of the vessel. We precompute the optimal cost for each inflexible arc using a linearized bunker consumption function, and compute the costs of flexible arcs during optimization in our MIP model. All inflexible arcs in the model are assigned a sailing cost for each ship that is the

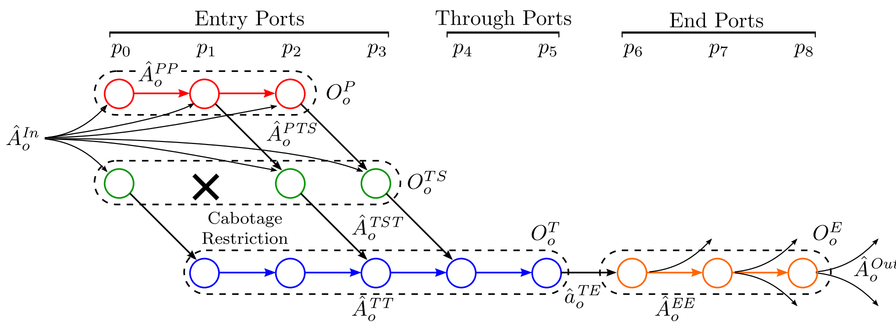


Figure 5: The graph structure of an example SOS opportunity, which contains parallel nodes O_o^P in red, transshipment nodes O_o^{TS} in green, a cabotage restriction at port p_1, through nodes O_o^T in blue, and end nodes O_o^E in orange.

optimal sailing cost given the total duration of the arc. Since ships have a minimum speed, if the duration of the arc is greater than the time required to sail on the arc at a ship's minimum speed, the cost calculated using the minimum speed and the ship simply waits for the remainder of the duration. This is a common practice for liner shippers in order to add buffer to their schedules, thus making the network more robust to disruptions.

3.2 MIP Model

We now define the MIP model that guides the vessels through the graph, and controls the flow of cargo and equipment, using the following parameters and variables to supplement the parameters used to define the graph.

Parameters

T	Set of equipment types. $T = \{dc, rf\}$.
$\sigma_s \in V'$	Starting visitation of vessel $s \in S$.
V^{t+}	Set of visitations with an equipment surplus of type t.
V^{t-}	Set of visitations with an equipment deficit of type t.
V^{t^*}	Set of visitations with an equipment surplus or deficit of type t ($V^{t^*} = V^{t^+} \cup V^{t^-}$).
V^{Goal}	Set of visitations corresponding to visitations on the goal service.
$In(i) \subseteq V'$	Set of visitations with an arc connecting to visitation $i \in V$.
$Out(i) \subseteq V'$	Set of visitations receiving an arc from $i \in V$.
$c_i^{Mv} \in \mathbb{R}^+$	Cost of a TEU move at visitation $i \in V'$.
$f_{s,i}^{Port} \in \mathbb{R}$	Port fee associated with vessel s at visitation $i \in V'$.
$d_{s,i}^{Mv} \in \mathbb{R}$	Move time per TEU for vessel s at visitation $i \in V'$.
$r_t^{Eqp} \in \mathbb{R}^+$	Revenue for each TEU of equipment of type $t \in T$ delivered.
$u_s^t \in \mathbb{R}^+$	Capacity of vessel s for cargo type $t \in T$.
A'	The set of arcs $(i,j) \in A$, where $i,j \in V'$.
$c_{i,j}^s$	Fixed cost of vessel s utilizing arc $(i,j) \in A'$.
$\alpha_{i,j}^s$	Variable hourly cost of vessel $s \in S$ utilizing arc $(i,j) \in A'$.
$\Delta_{i,j,s}^{Min}$	Minimum duration for vessel s to sail on flexible arc (i,j).
$\Delta_{i,j,s}^{Max}$	Maximum duration for vessel s to sail on flexible arc (i,j).
$(o,d,t) \in \Theta$	A demand triplet, where $o \in V', d \subseteq V'$ and $t \in T$ are the origin visitation, destination visitations and the cargo type, respectively.
$a^{(o,d,t)} \in \mathbb{R}^+$	Amount of demand available for the demand triplet.
$r^{(o,d,t)} \in \mathbb{R}^+$	Amount of revenue gained per TEU for the demand triplet.
$\Theta_i^O, (\Theta_i^D) \subseteq \Theta$	Set of demands with an origin (destination) visitation $i \in V$.
Θ_o^{SOS}	Set of demands corresponding to $o \in SOS$.

Variables

$y^s_{i,j} \in \{0,1\}$	Indicates whether vessel s is sailing on arc $(i,j) \in A$.
$t^E_i \in \mathbb{R}^+_0$	Defines the enter time of a vessel at visitation i.
$t^X_i \in \mathbb{R}^+_0$	Defines the exit time of a vessel at visitation i.
$w^s_{i,j} \in \mathbb{R}^+_0$	The duration that vessel $s \in S$ sails on flexible arc $(i,j) \in A^f$.
$x^{(o,d,t)}_{i,j} \in \mathbb{R}^+_0$	Amount of flow of demand triplet $(o,d,t) \in \Theta$ on $(i,j) \in A'$.
$x^t_{i,j} \in \mathbb{R}^+_0$	Amount of equipment of type $t \in T$ flowing on $(i,j) \in A'$.

$$\max\ -\sum_{s\in S}\left(\sum_{(i,j)\in A'} c^s_{i,j} y^s_{i,j} + \sum_{(i,j)\in A^f} \alpha^s_{i,j} w^s_{i,j}\right) \tag{1}$$

$$+ \sum_{(o,d,t)\in\Theta}\left(\sum_{j\in d}\sum_{i\in In(j)} \left(r^{(o,d,t)} - c^{Mv}_o - c^{Mv}_j\right) x^{(o,d,t)}_{i,j}\right) \tag{2}$$

$$+ \sum_{t\in T}\left(\sum_{i\in V^{t+}}\sum_{j\in Out(i)} \left(r^{Eqp}_t - c^{Mv}_i\right) x^t_{i,j} - \sum_{i\in V^{t-}}\sum_{j\in In(i)} c^{Mv}_i x^t_{j,i}\right) \tag{3}$$

$$+ \sum_{j\in V'}\sum_{i\in In(j)}\sum_{s\in S} f^{Port}_{s,j} y^s_{i,j} \tag{4}$$

$$\text{s.t.} \sum_{s\in S}\sum_{i\in In(j)} y^s_{i,j} \le 1 \qquad \forall j \in V' \tag{5}$$

$$\sum_{i\in Out(\sigma_s)} y^s_{\sigma_s,i} = 1 \qquad \forall s \in S \tag{6}$$

$$\sum_{i\in In(\tau)}\sum_{s\in S} y^s_{i,\tau} = |S| \tag{7}$$

$$\sum_{i\in In(j)} y^s_{i,j} - \sum_{i\in Out(j)} y^s_{j,i} = 0 \qquad \forall j \in \{V' \setminus \bigcup_{s\in S}\sigma_s\}, s \in S \tag{8}$$

$$\sum_{(o,d,rf)\in\Theta} x^{(o,d,rf)}_{i,j} - \sum_{s\in S} u^{rf}_k y^s_{i,j} \le 0 \qquad \forall (i,j) \in A' \tag{9}$$

$$\sum_{(o,d,t)\in\Theta} x^{(o,d,t)}_{i,j} + \sum_{t'\in T} x^{t'}_{i,j} - \sum_{s\in S} u^{dc}_s y^s_{i,j} \le 0 \qquad \forall (i,j) \in A' \tag{10}$$

$$\sum_{i\in Out(o)} x^{(o,d,t)}_{o,i} \le a^{(o,d,t)} \sum_{i\in Out(o)}\sum_{s\in S} y^s_{o,i} \qquad \forall (o,d,t) \in \Theta \tag{11}$$

$$\sum_{i\in In(j)} x^{(o,d,t)}_{i,j} - \sum_{k\in Out(j)} x^{(o,d,t)}_{j,k} = 0 \qquad \forall (o,d,t) \in \Theta, j \in V' \setminus (o \cup d) \tag{12}$$

$$\sum_{i\in In(j)} x^t_{i,j} - \sum_{k\in Out(j)} x^t_{j,k} = 0 \qquad \forall t \in T, j \in V' \setminus V^{t^*} \tag{13}$$

$$\Delta^{Min}_{i,j,s} y^s_{i,j} \le w^s_{i,j} \le \Delta^{Max}_{i,j,s} y^s_{i,j} \qquad \forall (i,j) \in A^f, s \in S \tag{14}$$

$$t^E_i = enter(i) \sum_{s\in S}\sum_{j\in In(i)} y^s_{i,j} \qquad \forall i \in V^i \tag{15}$$

$$t^X_i = exit(i) \sum_{s\in S}\sum_{j\in Out(i)} y^s_{i,j} \qquad \forall i \in V^i \tag{16}$$

$$t_i^X + \sum_{s \in S} w_{i,j}^s - t_j^E \leq 0 \qquad \forall (i,j) \in A^f \tag{17}$$

$$\begin{aligned}
&\sum_{(o,d,t) \in \Theta_i^O} \sum_{j \in Out(o)} d_{s,o}^{Mv} x_{o,j}^{(o,d,t)} + \sum_{(o,d,t) \in \Theta_i^D} \sum_{d' \in d} \sum_{j \in In(d')} d_{s,d}^{Mv} x_{j,d'}^{(o,d,t)} \\
&+ \sum_{t \in T} \left(\sum_{i' \in \{V^{t+} \cap i\}} \sum_{j \in Out(i')} d_{s,i'}^{Mv} x_{i',j}^t + \sum_{i' \in \{V^{t-} \cap i\}} \sum_{j \in In(i')} d_{s,j}^{Mv} x_{j,i'}^t \right) \\
&- \ t_i^X + t_i^E + pilot(i) \sum_{j \in In(i)} y_{s,i,j} \leq 0 \qquad \forall i \in V^f, s \in S
\end{aligned} \tag{18}$$

The domains of the variables are as previously described. The objective consists of several components. The sailing cost (1) takes into account the precomputed sailing costs on arcs between inflexible visitations, as well as the variable cost for sailings to and from flexible visitations. Note that the fixed sailing cost on an arc does not only include fuel costs, but can also include canal fees or the time-charter bonus for entering an SOS. The profit from delivering cargo (2) is computed based on the revenue from delivering cargo minus the cost to load and unload the cargo from the vessel. Note that the model can choose how much of a demand to deliver, even choosing to deliver a fractional amount. We can allow this since each demand is an aggregation of cargo between two ports, meaning at most one container between two ports will be fractional. Equipment profit is taken into account in (3). Equipment is handled similar to cargo, except that equipment can flow from any port where it is in supply to any port where it is in demand. Finally, port fees are deducted in (4).

Multiple vessels are prevented from visiting the same visitation in (5). The flow of each vessel from its source node to the graph sink is handled by (6), (7) and (8), with (7) ensuring that all vessels arrive at the sink.

Arcs are assigned capacities if a vessel utilizes the arc in (9), which assigns the reefer container capacity, and in (10), which assigns the total container capacity, respectively. Note that constraints (9) do not take into account empty reefer equipment, since empty containers do not need to be turned on, and can therefore be placed anywhere on the vessel. Cargo is only allowed to flow on arcs with a vessel in (11). The flow of cargo from its source to its destination, through intermediate nodes, is handled by (12). Constraints (13) balance the flow of equipment in to and out of nodes. In contrast to the way cargo is handled, equipment can flow from any port where it is in supply to any port where it is in demand. Since the amount of equipment carried is limited only by the capacity of the vessel, no flow source/sink constraints are required.

Flexible arcs have a duration constrained by the minimum and maximum sailing time of the vessel on the arc in (14). The enter and exit time of a vessel at inflexible ports is handled by (15) and (16), and we note that in practice these constraints are only necessary if one of the outgoing arcs from an inflexible visitation ends at a flexible visitation. Constraints (17) sets the enter time of a visitation to be the duration of a vessel on a flexible arc plus the exit time of the vessel at the start of the arc. Constraints (18) controls the amount of time a vessel spends at a flexible visitation. The first part of the constraint computes the time required to load and unload cargo and equipment, with the final term of the constraint adding the piloting time to the duration only if one of the incoming arcs is enabled (i.e. the flexible visitation is being used).

The model forms a disjoint path problem in which a fractional multicommodity flow is allowed to flow over arcs in the paths, along with a small scheduling component in the flexible nodes. Flexible arcs could be alternatively represented using a discretized approach, however we forego a discretization because of the vast differences in timescales between port activities and sailing activities, which are on the order of hours and days, respectively. In order to achieve such a fine grained view of flexible arc activities, we would require numerous extra arcs and nodes for each flexible node.

3.3 Complexity

We reduce the knapsack problem to the LSFRP with flexible visitations in order to show that the LSFRP is weakly NP-complete. Given n items, each with a profit p_i and a size s_i, and a knapsack with a capacity C, the knapsack problem maximizes the objective $\sum_{i=0}^{n} p_i x_i$ where x_i is a binary variable indicating whether or not item i is in the knapsack, subject to the capacity constraint $\sum_{i=0}^{n} s_i x_i \leq C$.

Theorem 1. *The LSFRP with flexible visitations is weakly NP-complete.*

We first note that the LSFRP is clearly in NP, as the total profit can be easily computed from the paths of the vessels through the graph. We initialize an LSFRP with a single vessel and no cargo or equipment demands. The problem instance contains a single phase-out visitation, ω, and a single phase-in visitation, λ. The port fees at both ω and λ are 0, and we let $enter(\omega) = exit(\omega) = 0$ and $enter(\lambda) = exit(\lambda) = C$. In other words, the timespan in which the repositioning must take place is limited to the capacity of the knapsack. For each knapsack item, we create a flexible visitation, f_i, which has a duration of exactly s_i, i.e. $pilot(f_i) = s_i$. The port fee for visiting f_i is $-p_i$, since the LSFRP maximizes profit (i.e. minimizes fees). All flexible nodes, as well as ω and λ, are in a single trade zone. Therefore, the specification of the LSFRP graph ensures that the phase-out node, ω, connects to all flexible nodes, all flexible nodes connect to each other, and all flexible nodes connect to the phase-in node, λ. The sailing time of the vessel between all nodes in the graph is set to 0.

Item i is included in the knapsack solution if and only if the vessel visits f_i during its repositioning. Since the vessel can only visit a single flexible visitation at a time, the duration of each flexible visitation is fixed to the size of the item it represents, and the phase-in visitation is fixed in time to the size of the knapsack, the capacity constraint of the knapsack must be satisfied. Additionally, according to the objective of the LSFRP, only the flexible visitations corresponding to the maximum profit knapsack items will be chosen. Therefore, the LSFRP with flexible visitations is NP-complete. □

We note that flexible ports are not present in every LSFRP problem, and this proof only covers those with flexible ports. This is not to say that LSFRP problems without flexible ports are necessarily polynomial time solvable. Indeed, the LSFRP without flexible ports is likely NP-complete, however this is not trivial to prove and remains an open problem at this time.

4 Simulated Annealing

Since current MIP solvers are unable to scale to the size of the LSFRP faced by industry, as well as unable to solve the problem fast enough to be useful for a repositioning coordinator, we created a heuristic solution procedure for the LSFRP with cargo flows using an SA algorithm (Kirkpatrick et al., 1983). We choose SA over other heuristic approaches due to its fast convergence and ability to find good solutions quickly. Our SA uses a penalized objective in which certain types of infeasible solutions are accepted in order to avoid getting stuck within the search landscape.

Since SA algorithms tend to converge quickly, we use a combined reheating and restart strategy similar to the one used in (Taheri and Zomaya, 2007) to overcome local optima. Reheating involves increasing the temperature of the SA after convergence to a factor of the initial temperature. We combine reheating with full restarts, in that we allow only several reheats before we restart the SA from the initial solution. The idea behind such a restart is that several reheats could put the solution of the SA in a part of the search space that is far away from the global optimum, and continual reheating may not move it in the correct direction. Restarting from the initial solution allows the SA to move in a different direction and find a better solution.

4.1 Initial Solution Heuristics

We introduce several heuristics to generate initial solutions to the LSFRP for use in our SA algorithm.

4.1.1 Direct route heuristic (DRH)

We model the connections between the starting visitation of each vessel with all of the feasible phase-ins in a linear assignment problem. The cost of each vessel/phase-in assignment is equal to the sailing cost of the vessel to the particular phase-in if the sailing is feasible, or infinity if it is not. The direct route heuristic generates a feasible starting solution, but it is rarely, if ever, an optimal solution to the LSFRP. We use the Hungarian algorithm (Kuhn, 1955) to solve the linear assignment problem.

4.1.2 Shortest paths heuristic (SPH)

We generate paths for vessels using a shortest path algorithm that iteratively creates a path for each ship on a graph containing only those visitations that are not visited in any previously generated path. While this ensures that any solution the heuristic generates will be feasible, it may not always be possible to generate any solution at all since all of the feasible phase-in opportunities for a particular ship may already have been assigned to another ship. We order the ships by their first possible phase-out time and generate paths starting with the ship with the latest phase-out. We found that with this ordering, only one out of our 44 instances could not generate a feasible starting solution, whereas with random orderings or with an ascending phase-out time ordering, feasible starting solutions could almost never be generated. If it is not possible to generate a solution, we fall back on DRH, although this never happened in our experiments.

Sailing costs on inflexible arcs as well as port fees are easy to take into account in the shortest path algorithm, however flexible arcs and cargo/equipment revenues pose a challenge. It is not possible to take these components fully into account within a standard shortest path algorithm, since this would require the cost of a particular arc to vary based on the scheduling of flexible arcs. Thus, the heuristic generates solutions that do not re-use visitations between vessels, but are not necessarily temporally feasible on instances with flexible arcs.

Flexible arc handling We allow flexible arcs to be used in the shortest path, even though they represent a scheduling problem that cannot be solved while the shortest path algorithm is running. We ignore temporal feasibility and focus only on the cost of the flexible arc. We define a parameter γ in the range $[0, 1]$ that represents how fast the ship is sailing over its minimum speed, and we assign arcs a sailing cost based on the speed of the vessel. This allows the heuristic to try to take into account some of the costs that would be incurred using a flexible arc.

Cargo handling The profit for each cargo demand in the graph is represented by computing the total possible profit from the demand (cargo revenue less move fees at the origin and destination) and multiplying this value by a scaling parameter ℓ^{Cargo}, which is in the range $[0, 1]$. We then offset the sailing costs and port fees at the origin and destination visitations using this scaled cargo profit. Thus, nodes where lots of cargo originates or is delivered have a high profit and are desirable for the shortest path algorithm to visit. Since cargo can only be delivered if both the origin and destination are on the path of a ship, this heuristic cannot guarantee that the path taken is actually one that has profitable cargo flows.

Equipment handling We perform a similar process of adding profit to visitations with equipment surpluses/deficits. The primary difference is that we do not know how much equipment can be loaded on to the vessel. Thus, we introduce a parameter $\ell^{Eqp} \in \mathbb{Z}^+$ that represents the amount of equipment to load or unload at a visitation. We cannot guarantee the ship will have sufficient capacity to actually load or unload that amount of cargo, but this allows the shortest path algorithm to visit visitations with equipment, which might otherwise be ignored.

Shorted Path Implementation Since we take into account cargo and equipment profits in this heuristic, arcs can have costs or revenues associated with them, meaning the sign of all the arcs is not the same. We therefore use the Bellman-Ford algorithm to sequentially compute the shortest path for each vessel. After a shortest path is computed, we update a list of banned visitations that may not be used again. Any visitation that is banned is considered to have a distance of infinity to and from all visitations, ensuring the shortest path algorithm does not choose it. It is possible that negative cycles are generated by the algorithm. We handle these by clearing the list of banned visitations and starting the algorithm from the first vessel in the vessel ordering using updated cargo and equipment profit parameters. After each failure, we subtract 0.1 from the cargo profit parameter ℓ^{Cargo} and 50 from ℓ^{Eqp}, until these parameters hit 0. When both parameters are 0, the graph is guaranteed to not have any negative cycles since arcs reflect only sailing costs. In practice, this is only necessary on several instances.

4.2 Greedy heuristic (GH)

The greedy heuristic (GH) chooses the most profitable outgoing arc from each visitation based on the same profit calculations and parameters as SPH. Similar to SPH, GH does not allow vessels to visit the same visitation more than once, and does this by storing a list of banned nodes after computing a greedy path for a vessel. The order the paths are generated in is the same as in SPH, as nearly every other ordering we tried resulted in failure of the algorithm to compute a solution. As in SPH, if a solution cannot be generated we fall back on DRH, although this did not happen in our experiments. We also tried creating a greedy heuristic that is not concerned with feasibility, however, we found that the solutions it generates tend to be of poor quality, as many vessels sail to the same phase-in visitation and the SA must then completely construct new routes for those vessels.

4.3 Neighborhoods

We now describe the neighborhood operators for our SA algorithm. At each iteration of our SA, a neighborhood operator is chosen uniformly at random to modify the current solution.

Visitation addition A ship, s, is selected uniformly at random along with an arc (u, v) on the path of s. A new visitation, w, is chosen such that an arc exists from u to w and from w to v, i.e. $w \in out(u)$ and $w \in in(v)$, and w is not already on the path of s. The visitation w is then inserted into the path of s between u and v. If no such visitation w exists, then the neighborhood performs no changes.

Visitation removal A ship, s, is selected uniformly at random along with a visitation on its path, u, such that u is neither the first or last visitation on the path. Visitation u is removed from the path if there exists an arc from the visitation before u to the visitation after u. If no such arc exists, the solution is not changed.

Visitation swap Two ships s and s', $s \neq s'$ are chosen uniformly at random, and a visitation u is selected from the path of s. If a visitation w on the path of s' exists such that swapping u and w is possible, i.e. $In(u) \cap In(w) \neq \emptyset$ and $Out(u) \cap Out(w) \neq \emptyset$, and swapping u and v would not introduce a duplicated node on either path, then u and w are swapped between paths.

Random path completion (RPC) A random ship, s, is selected uniformly at random along with a visitation, u, on its path. All visitations subsequent to u are removed from the path of s, and are replaced with a random path from u to the graph sink. Each visitation added to the random path must not already be on the path of s, to ensure there are no loops over flexible visitations. If it is impossible to finish a random path without containing a loop, the random path is abandoned and the solution is not changed.

Demand destination completion (DDC) A random ship, s, is selected uniformly at random along with a visitation on its path, u, from which demand is loaded. A demand is chosen that could be loaded at u, but cannot be delivered because none of its delivery visitations are on the path of s. The neighborhood attempts to connect the current path to one of the destinations of the cargo using a breadth first search. Then, another breadth first search is started from the destination back to any subsequent visitation on the path, v. If no such path exists, or such paths can only be created by introducing a duplicated node into the vessel's path, then the solution is left unchanged. All nodes in between u and v are deleted from the path and replaced with the nodes from the breadth first searches.

4.4 Objective evaluation

After performing a move and generating a candidate solution, the SA must update its objective value for the solution. We exploit the fact that the paths of vessels are disjoint and only update the objective for the paths that a neighborhood operator changes. Note that since our SA does admit infeasible solutions as described above, it is possible that some cargo is carried multiple times. However, the extra revenue that vessels can gain is significantly less than the penalty for multiple vessels calling the same visitation.

The objective in the LSFRP consists of three components; the easy to calculate fixed costs for inflexible arcs and port fees, the cost of sailing on flexible arcs, and the profit from delivering cargo and equipment. Computing the cost of using flexible arcs requires solving a simple scheduling problem for the vessel, and a linear program determines the cargo and equipment profit. Since the amount of time it takes to load cargo and equipment is taken into account at flexible visitations, the scheduling of flexible arcs and the handling of cargo and equipment must be tackled together in a single linear program, which we solve using CPLEX.

In certain situations it is possible to avoid calling CPLEX and thereby speed up the computation of the objective. If a vessel's path includes no flexible visitations and at no visitation could the amount of cargo (both dry and reefer) loaded on to the vessel exceed its capacity, then we can use a simple greedy algorithm to load cargo onto the vessel. The greedy algorithm works by simply loading all available cargo at all visitations, and then fills the remaining capacity of the vessel with equipment. This will always be the optimal loading of cargo and equipment as long as the profit earned per TEU from carrying equipment is less than the profit per TEU of any demand. In practice this is true, since customer's cargo is preferred over empty containers.

5 Computational Study

We evaluate the performance of our MIP and the SA algorithm across a dataset of real-world and real-world inspired instances. We show that our novel approach to solving the LSFRP with cargo flows is ready for use in a decision support system, as the solution time is fast enough for human interaction (on the order of a few minutes). In addition, our SA algorithm is able to find high quality solutions on our industrial collaborator's reference scenario that earn twice the profit of the reference solution.

5.1 Benchmark

We created a benchmark set of instances containing two real world repositioning scenarios, with three and eleven ships, respectively. The rest of our benchmark set consists of scenarios that never took place, but were crafted using real liner shipping data to examine how our algorithms scale. Since all of our data in the benchmark is confidential information from our industrial collaborator, we have duplicated the confidential instances and perturbed the costs, revenues, amounts of cargo in demands, and randomly deleted/added demands to create a publicly available benchmark. We combine publicly available liner shipping data, such as ship information and port fees, from the ENERPLAN benchmark (Løfstedt et al., 2010) with randomly perturbed data from our collaborator. We perturb all values not already contained in the ENERPLAN benchmark by $\pm 20\%$, as in (Løfstedt et al., 2010), including non-cost/revenue related values such as port productivities, ensuring that no private data is contained in the dataset. We have kept the schedules of the ships in the repositioning scenarios the same, as this is public information.

Table 1 gives information about the instances in both the confidential and public dataset, along with the runtime of our MIP model in CPLEX. The MIP runtimes are computed on AMD Opteron 2425 HE processors with a maximum of 10GB of RAM per process. The table gives the instance ID, the number of ships, $|S|$, the number of nodes in the graph, $|V|$, the number of inflexible arcs, $|A^i|$, the number of flexible arcs, $|A^f|$, the number of demands, $|\Theta|$, the number of ports with equipment surpluses or demands, $|E| = |\cup_{t \in T} V^{t^*}|$, the number of SOS opportunities, $|SOS|$, and finally the runtime of CPLEX 12.4 in seconds with a one hour timeout. We present results for both the confidential and public datasets to show that our public dataset is able to capture the difficult components of the LSFRP, and is thus a viable benchmark for further study of the LSFRP. Note that the instance IDs correspond between the confidential and private instances, with the only difference being the cost structure of the instances, since the schedule information used in the instances is already public.

The instances range in size from 3 to 11 ships with various SOS and equipment opportunities made available in each instance. The problem becomes rather difficult for the MIP to solve due to the many variables generated from the multi-commodity flow. Most instances have several hundred demands, with the largest confidential instance having 1748 demands and the largest public instance 1423 demands. CPLEX is able to solve 33 out of the 44 instances on both the confidential and the public datasets. Many instances are solved rather quickly, with over 25 confidential instances solved in under two minutes of CPU time and 17 of those instances requiring under ten seconds of computation time. Of the public instances, 28 are solvable in two minutes and 17 in under 10 seconds. The MIP approach does not scale

ID	$\lvert S\rvert$	$\lvert V\rvert$	$\lvert A^i\rvert$	$\lvert A^f\rvert$	$\lvert\Theta\rvert$	$\lvert E\rvert$	$\lvert SOS\rvert$	MIP
repo1c	3	30	94	0	27	0	1	0.04
repo2c	3	30	94	0	27	0	2	0.04
repo3c	3	37	113	0	25	0	2	0.03
repo4c	3	40	143	0	21	0	3	0.03
repo5c	3	47	208	0	24	0	3	0.05
repo6c	3	47	208	0	24	0	3	0.06
repo7c	3	53	146	0	51	0	4	0.06
repo8c	3	104	1015	121	67	6	3	1.92
repo9c	3	104	1015	121	67	9	3	1.91
repo10c	4	58	389	0	150	0	0	16.08
repo11c	4	62	389	40	150	6	0	14.50
repo12c	4	74	469	0	174	0	2	72.91
repo13c	4	80	492	0	186	0	4	231.47
repo14c	4	80	492	0	186	24	4	182.06
repo15c	5	68	237	0	214	0	0	0.39
repo16c	5	103	296	0	396	0	5	0.95
repo17c	6	100	955	0	85	0	0	5.41
repo18c	6	133	1138	0	101	0	9	6.72
repo19c	6	133	1138	0	101	33	9	5.68
repo20c	6	140	1558	0	97	0	4	313.55
repo21c	6	140	1558	0	97	13	4	47.78
repo22c	6	140	1558	0	97	37	4	39.51
repo23c	6	152	1597	162	103	71	9	19.79
repo24c	7	75	395	0	196	0	3	2.30
repo25c	7	77	406	0	195	0	0	2.69
repo26c	7	77	406	0	195	16	0	1.96
repo27c	7	78	502	0	237	0	0	94.48
repo28c	7	89	537	0	241	0	4	174.55
repo29c	7	89	537	0	241	19	4	186.38
repo30c	8	126	1154	0	165	0	0	2075.82
repo31c	8	126	1300	0	312	0	0	99.02
repo32c	8	140	1262	0	188	0	3	487.93
repo33c	8	209	2211	453	213	50	3	548.11
repo34c	9	304	9863	0	435	0	0	Time
repo35c	9	357	11289	38	1075	118	4	Mem
repo36c	9	364	11078	0	1280	0	4	Mem
repo37c	9	371	10416	0	1270	114	7	Mem
repo38c	9	373	11979	38	1280	126	4	Mem
repo39c	9	379	10660	0	1371	0	7	Mem
repo40c	9	379	10660	0	1371	118	7	Mem
repo41c	10	249	7654	0	473	0	0	Time
repo42c	11	275	5562	0	1748	0	5	Time
repo43c	11	320	11391	0	1285	0	0	Mem
repo44c	11	328	11853	0	1403	0	4	Mem

(a) Confidential dataset.

ID	$\lvert S\rvert$	$\lvert V\rvert$	$\lvert A^i\rvert$	$\lvert A^f\rvert$	$\lvert\Theta\rvert$	$\lvert E\rvert$	$\lvert SOS\rvert$	MIP
repo1p	3	36	150	0	28	0	1	0.06
repo2p	3	36	150	0	28	0	2	0.06
repo3p	3	38	151	0	24	0	2	0.04
repo4p	3	42	185	0	20	0	3	0.04
repo5p	3	51	270	0	22	0	3	0.07
repo6p	3	51	270	0	22	0	3	0.08
repo7p	3	54	196	0	46	0	4	0.08
repo8p	3	108	1185	126	50	6	3	1.89
repo9p	3	108	1185	126	50	10	3	1.82
repo10p	4	58	499	0	125	0	0	74.85
repo11p	4	62	499	38	125	6	0	38.17
repo12p	4	74	603	0	145	0	2	106.63
repo13p	4	80	632	0	155	0	4	99.81
repo14p	4	80	632	0	155	24	4	97.15
repo15p	5	71	355	0	173	0	0	0.47
repo16p	5	106	420	0	320	0	5	1.08
repo17p	6	102	1198	0	75	0	0	4.64
repo18p	6	135	1439	0	87	0	9	6.79
repo19p	6	135	1439	0	87	33	9	8.18
repo20p	6	142	1865	0	80	0	4	13.84
repo21p	6	142	1865	0	80	13	4	23.04
repo22p	6	142	1865	0	80	37	4	17.67
repo23p	6	153	1598	159	89	71	9	19.58
repo24p	7	75	482	0	154	0	3	2.23
repo25p	7	77	496	0	156	0	0	3.19
repo26p	7	77	496	0	156	16	0	2.05
repo27p	7	79	571	0	188	0	0	1394.44
repo28p	7	90	618	0	189	0	4	1099.87
repo29p	7	90	618	0	189	19	4	1183.01
repo30p	8	126	1450	0	265	0	0	307.12
repo31p	8	130	1362	0	152	0	0	57.40
repo32p	8	144	1501	0	170	0	3	65.51
repo33p	8	212	2227	433	179	50	3	139.99
repo34p	9	304	10577	0	344	0	0	Time
repo35p	9	357	11284	35	874	118	4	Mem
repo36p	9	364	11972	0	1048	0	4	Mem
repo37p	9	371	11371	0	1023	114	7	Mem
repo38p	9	373	11972	35	1048	126	4	Mem
repo39p	9	379	11666	0	1109	0	7	Mem
repo40p	9	379	11666	0	1109	118	7	Mem
repo41p	10	249	8051	0	375	0	0	Time
repo42p	11	279	6596	0	1423	0	5	Time
repo43p	11	320	13058	0	1013	0	0	Mem
repo44p	11	328	13705	0	1108	0	4	Mem

(b) Public dataset.

Table 1: Instance information and solution time to optimality with CPLEX 12.4. The columns describe the number of ships, visitations, inflexible arcs, flexible arcs, demands, ports with equipment demands or surpluses, number of SOS opportunities, and the solution time with CPLEX (in seconds). Instance repo42c represents our industrial collaborator's scenario.

past 8 ships on either dataset, as the graph sizes start to become large, and the number of variables increases as well. Many of these instances are unsolvable purely due to running out of memory, which gives hope for branch and price approaches. We encounter memory issues on instances that have many arcs and cargo/equipment demands, since a variable must be created for each demand on each arc.

In addition to solving our MIP model to optimality, we attempted to solve to a 5% and 10% gap, however the results of Table 1 remain unchanged; instances that cannot be solved to optimality still cannot be solved to a 10% gap, and instances that can be solved are not solved significantly faster as a result. Since we aim to find good solutions to the LSFRP within ten minutes in order to create a decision support system with our algorithms that can be used in industry, heuristic methods are required.

5.2 SA Implementation

We implemented the SA algorithm described in Section 4 in C++11. The implementation relies on CPLEX 12.4 for computing components of the objective function, as well as Google OR Tools (Google, 2012) for computing the assignment problem in the DRH heuristic. Our implementation is able to process over 700,000 SA iterations per second on smaller instances, where an iteration is an application of a neighborhood operator to the current solution and an update of the objective function, and 7,100 iterations per second on our largest instance, repo44c.

5.2.1 Parameter Tuning

Our SA algorithm has many different parameters which can affect its performance, and in order to ensure a fair comparison of our SA against the MIP model, we perform parameter tuning on the SA. There are many suggestions in the literature for parameter settings of the components of SA algorithms (Johnson et al., 1989; Hoos and Stützle, 2004). We have used these guidelines in our parameter tuning procedure, but are ultimately relying on the performance of our SA on a training set of instances to determine which parameters are the best for the LSFRP. In order to avoid overtuning our algorithm to the instances of the LSFRP that we present, we tune our SA algorithm on a training set of 15 instances from the confidential dataset and validate the performance of the parameters on the entire set of instances. The instances were chosen at random from instances in which the SA algorithm using the GH heuristic was unable to find the optimal solution. The training set consists of 15 instances, which is a little over one third of our dataset, which is a standard amount in the machine learning and parameter tuning literature (Ansotegui et al., 2009). The instances comprising our training set are: repo15c, repo17c, repo18c, repo19c, repo21c, repo23c, repo32c, repo35c, repo36c, repo37c, repo38c, repo39c, repo41c, repo42c, and repo43c.

There are 13 parameters to tune in total. The feasible shortest paths heuristic has three parameters, γ, ℓ^{Cargo} and ℓ^{Eqp}, which describe the cost factor to use when estimating flexible arc costs, the amount of cargo profit to earn at the origin and destination visitations of a demand, and the amount of equipment to "load" in the heuristic, respectively. The SA algorithm has seven parameters, α, t^{Init}, t^{Max}, r^{Itrs}, r^{Reheat}, β, $r^{Restart}$, which are the geometric temperature decrease factor, the starting SA temperature, the SA convergence temperature, the maximum number of non-improving iterations before convergence, the number of non-improving reheats before convergence, the reheating factor, and the number of reheats before resetting the current solution to the starting solution, respectively. Finally, there are three parameters that control the penalization of infeasible solutions in the objective. The parameters p, p^{PI}, and p^{T} are the penalization of multiple vessels utilizing the same visitation in their paths, the penalty for two vessels using the same phase-in visitation, and the penalty for a vessel's path being temporally infeasible. Note that p^{PI} is describing a case of p which we penalize separately because it tends to be a more difficult infeasibility for the SA to fix, and thus needs a higher penalization to ensure its repair.

We tune the SA algorithm by hand, running each parameter configuration ten times on each training instance, each time with a different random seed. Running each parameter configuration multiple times is necessary due to the stochastic nature of the SA algorithm. In order to avoid a combinatorial explosion of parameter settings, we assume that parameters are independent of each other. While this is clearly not a true assumption, it is the only way to perform parameter tuning across so many parameters without spending extraordinary amounts of CPU time. We then choose the best parameter value for each parameter based on the total profit earned by using each parameter.

Table 2 gives the discretized parameter domains we used during hand tuning, as well as the best value for each parameter for each initial heuristic with all SA neighborhoods enabled. We determined which parameter was the best by computing the total profit earned by each parameter across the training set.

Category	Parameter	Discretized domain values.	DRH	SPH	GH
GH/SPH	γ	0.0, 0.1, 0.25, 0.5, 0.75	0.25	0.25	0.25
	ℓ^{Cargo}	0.0, 0.1, 0.25, 0.5, 0.75, 0.95	0.95	0.95	0.95
	ℓ^{Eqp}	0, 10, 50, 100, 250, 500, 1000	100	100	100
SA	α	0.997, 0.998, 0.999, 0.9999, 0.99999, 0.999999	0.999999	0.999999	0.999999
	t^{Init}	1×10^4, 5×10^4, 1×10^5, 5×10^5, 1×10^6	1×10^6	1×10^6	5×10^5
	t^{Max}	1×10^{-8}, 1×10^{-10}, 1×10^{-12}, 1×10^{-15}	1×10^{-8}	1×10^{-8}	1×10^{-8}
	r^{Itrs}	500, 1000, 2500, 5000, 7500, 1×10^4	7500	10000	7500
	r^{Reheat}	1, 5, 10, 20	20	20	20
	β	0.1, 0.25, 0.5, 0.75	0.75	0.75	0.75
	$r^{Restart}$	1, 5, 10, 20	10	10	20
Penalization	p	1×10^5, 2×10^5, 5×10^5, 7×10^5, 1×10^6, 2×10^6, 5×10^6, 7×10^6, 1×10^7	7×10^6	7×10^6	1×10^6
	p^{PI}	1×10^5, 2×10^5, 5×10^5, 7×10^5, 1×10^6, 2×10^6, 5×10^6, 7×10^6, 1×10^7	7×10^6	5×10^6	5×10^6
	p^T	1×10^5, 2×10^5, 5×10^5, 7×10^5, 1×10^6, 2×10^6, 5×10^6, 7×10^6, 1×10^7	1×10^5	1×10^5	7×10^5

Table 2: The discretized parameter domains used in hand tuning are given with parameters classified into several categories. The best parameter for each initial solution heuristic as determined through parameter tuning are given on the right side of the table.

	DRH	SPH	GH
# Best Obj.	35	37	7
# Worst Obj.	3	7	37
Obj. Average	-2.05×10^6	-2.98×10^6	-1.42×10^7
Obj. Median	-3.51×10^7	-3.51×10^7	-5.25×10^7

Table 3: Starting solution statistics for all three heuristics on the confidential dataset.

5.3 Initial Solution Heuristics Comparison

We compare the performance of the initial solution heuristics introduced in Section 4.1 across our dataset using the tuned parameters from Section 5.2.1. Table 3 describes the performance of the starting heuristics across the dataset. The DRH and SPH heuristics perform the best, achieving the best initial objective out of the three heuristics on 35 and 37 of the instances, respectively (i.e. on many instances both heuristics return the same solution). The average objective found by SPH is $924,000 better than DRH's objective, and both SPH and DRH have average objectives that are a full order of magnitude better than GH's. In order to determine the significance of the difference in the means, we use a one-way ANOVA test (see, e.g., (Tabachnick and Fidell, 2012)) with the null hypothesis that the means of SPH, DRH and GH on the private dataset are not different from each other. We reject the null hypothesis with $p = 0.005$. However, comparing the results of SPH and DRH directly with a t-test gives a p value of 0.057, slightly above a statistically significant level, meaning we cannot claim DRH to be significantly better than SPH at generating initial solutions.

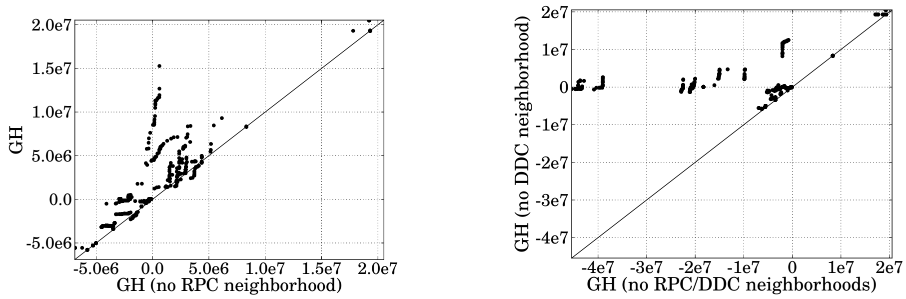
Despite the differences in performance between the heuristics in creating an LSFRP solution, the answers tend to be relatively similar once the SA algorithm runs. This speaks to the strength of our SA algorithm, in that it is able to find high quality solutions despite having initial solutions of varying quality.

5.4 Neighborhood Analysis

We perform an analysis of two of the neighborhoods from Section 4.3, the RPC neighborhood and the DDC neighborhood, in order to determine if they are beneficial to the SA algorithm. We do not analyze the visitation addition, removal and swapping neighborhoods because they are basic building blocks that any LS would need to be successful.

5.4.1 Random Path Completion

We parameter tune the SA algorithm with and without the RPC neighborhood in order to determine whether the random paths generated are beneficial to the SA. We perform this experiment both with and without the DDC neighborhood, and solve each instance using 25 different seeds. Figures 6a and 6b show the performance of the SA algorithm using



(a) RPC vs. No RPC with DDC neighborhood

(b) RPC vs. No RPC without DDC neighborhood

Figure 6: Effectiveness of the RPC neighborhood with and without the DDC neighborhood using the GH initial solution heuristic.

an initial solution generated by GH with the RPC neighborhood vs. not using the RPC neighborhood, both using the DDC neighborhood and without using DDC, respectively. We only show data using the GH initial heuristic since the performance of all three heuristics is similar after optimization. Points above the line $y = x$ in the scatter plot indicate better performance for the RPC neighborhood. The RPC neighborhood's usefulness is clear both with and without the DDC neighborhood. Not using the RPC neighborhood outperforms using the neighborhood only on several instances, and in many cases, the RPC neighborhood is able to find solutions that are orders of magnitude better than without the neighborhood.

With the DDC neighborhood, the average objective of GH with the RPC neighborhood is 2.21×10^6, versus an average objective of only 1.12×10^6 without the RPC neighborhood. When looking at only the best objective found out of all 25 seeds for each instance, using RPC still greatly outperforms not using RPC with an average objective of 2.76×10^6 against 1.52×10^6 for not using RPC. A t-test confirms the statistical significance of our findings, with $p < 1 \times 10^{-4}$, allowing us to reject the null hypothesis that the RPC neighborhood does not improve the solution quality. The difference in objective quality becomes even more pronounced when the DDC neighborhood is turned off. In this case, using the RPC neighborhood has an average objective of 1.67×10^6, but turning off RPC results in only -4.58×10^6.

We conclude that the RPC neighborhood is an important mechanism for the SA to explore new paths in the graph and avoid getting stuck in a local optimum. In contrast to the visitation addition, removal and swap operators, which can help refine a solution, the RPC neighborhood creates large changes in solutions that are critical to good performance from our SA algorithm.

5.4.2 Demand Destination Completion

We also test the effectiveness of the DDC neighborhood in order to see how much the neighborhood benefits the solution. Figure 7 plots the performance of the SA algorithm using initial solutions generated by GH with the DDC neighborhood vs. without the DDC neighborhood on each instance in the confidential dataset with 25 different seeds per instance. We hand tuned parameters for the SA for both with and without the neighborhood for fairness of comparison. Points above the line $y = x$ indicate a higher profit for the DDC neighborhood, whereas points below the line indicate that turning the neighborhood off provides a higher profit. The benefit of the DDC neighborhood is clearly demonstrated by the plot, with the majority of the instances lying above the line. Indeed, a t-test confirms the statistical significance of the result, allowing us to reject the null hypothesis that the mean of the SA performance with the DDC neighborhood is the same as without the neighborhood with a significance level of $p < 1 \times 10^{-4}$. The average objective across all instances and seeds with the neighborhood is 2.21×10^6, and without the neighborhood is 1.67×10^6, an improvement of a half million dollars.

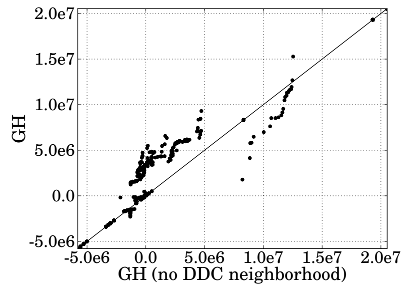


Figure 7: Performance of the SA algorithm with and without the DDC neighborhood using the GH initial solution heuristic.

5.5 SA vs. MIP

We compare the objective value of the SA algorithm to the optimal value found by the MIP to determine whether our SA is finding good solutions. A simple optimality gap comparison between our SA and MIP would not adequately show the performance of the SA, since sometimes the optimal objective is positive, but the SA only finds a negative solution. Such situations result in relative optimality gaps that are meaningless. We therefore compute a score showing how much of the gap between the best starting solution of the SA and the optimal MIP solution the SA closes. Our score is computed as follows:

$$score(inst) = \frac{opt(inst) - SA(inst)}{opt(inst) - base(inst)},$$

where $SA(inst)$ is the objective found by the SA algorithm, $base(inst)$ is the objective of the best starting solution and $opt(inst)$ is the optimal objective for the instance, if one is known. The closer the score is to 0, the better the SA performance. Note that in cases where the baseline is the same as the optimal solution (i.e. the starting heuristic found the optimal solution), we set the score to 0 for clarity.

Table 4 shows the objectives of the solutions found by the hand tuned SA algorithm with the three starting heuristics versus the optimal solution found by the MIP on the confidential dataset. We also show the performance of running our MIP for 10 seconds, and if no solution is found, switching to the SA algorithm with DRH. This sequential approach is able to combine the optimal performance of the MIP on small instances with the SA's ability to solve even large instances. The SA algorithm is able to find high quality solutions across the dataset, scaling easily to large instances. With all three initial solution heuristics, the SA finds the optimal solution on six instances, and has a score below 0.1 on 57% of the instances where the optimal solution is known. The average scores of each initial solution heuristic are very similar, showing that our SA handles a variety of starting solutions. Our sequential approach (MIP+GH) achieves an average score that is 58% that of GH alone, at the expense of only 10 seconds of CPU time. Since the MIP rarely finds a solution within the first 10 seconds unless the problem is solved in the first 10 seconds, we do not pass any information from the MIP to the SA algorithm. These results indicate that our SA algorithm is successful at finding near optimal objectives on many instances. In addition, our simple sequential approach is able to increase the number of optimal solutions found, while at the same time maintaining the ability to scale to large instances, an important quality for decision support systems.

Table 5 shows the same data as Table 4 for the public dataset. A similar picture emerges, in which our SA algorithm is able to find good solutions across the entire dataset, with the starting solution heuristic being relatively irrelevant. As before GH+MIP performs the best, with an average score of only 0.059, which is nearly half that of DRH, SPH, or GH alone. Note that the objective values tend to be a bit lower than in the confidential data set. This is due to a combination of different ship fuel consumption profiles and a different demand structure than in the confidential dataset. Despite these small differences, the performance of the SA is comparable to its performance on the confidential dataset, indicating that our publicly available data is a good representation of the actual LSFRP.

We now show some details of the performance of the SA on individual instances.

On instance repo30c (Figure 8), the SA is unable to find the optimal solution, but comes close, and is able to find

			DRH		SPH		GH		GH+MIP	
ID	SA Baseline	Optimal	Obj.	Score	Obj.	Score	Obj.	Score	Obj.	Score
repo1c	-55.88	-33.91	-33.91	0.000	-33.91	0.000	-33.91	0.000	-33.91	0.000
repo2c	-55.88	-33.91	-33.91	0.000	-33.91	0.000	-33.91	0.000	-33.91	0.000
repo3c	-71.13	-55.58	-55.58	0.000	-55.58	0.000	-55.58	0.000	-55.58	0.000
repo4c	-47.70	-6.30	-8.11	0.044	-8.11	0.044	-8.11	0.044	-6.30	0.000
repo5c	-7.68	0.44	0.44	0.000	0.44	0.000	0.44	0.000	0.44	0.000
repo6c	-7.68	0.44	0.44	0.000	0.44	0.000	0.44	0.000	0.44	0.000
repo7c	-68.95	83.20	83.20	0.000	83.20	0.000	83.20	0.000	83.20	0.000
repo8c	-7.68	0.44	0.44	0.000	0.44	0.000	0.44	0.000	0.44	0.000
repo9c	-7.68	0.44	-1.21	0.203	0.44	0.000	0.44	0.000	0.44	0.000
repo10c	-13.43	205.76	193.06	0.058	193.06	0.058	193.06	0.058	193.06	0.058
repo11c	39.73	205.76	193.06	0.076	193.06	0.076	193.06	0.076	193.06	0.076
repo12c	-13.43	210.34	205.11	0.023	205.11	0.023	205.11	0.023	205.11	0.023
repo13c	-13.43	210.56	205.34	0.023	205.34	0.023	205.34	0.023	205.34	0.023
repo14c	-13.43	210.56	205.34	0.023	205.34	0.023	205.34	0.023	205.34	0.023
repo15c	-52.80	4.91	0.50	0.076	0.50	0.076	0.50	0.076	4.91	0.000
repo16c	-183.10	4.91	0.50	0.023	0.50	0.023	0.50	0.023	4.91	0.000
repo17c	-52.46	-16.64	-30.39	0.384	-30.39	0.384	-30.39	0.384	-16.64	0.000
repo18c	-52.46	-13.38	-27.13	0.352	-27.13	0.352	-27.13	0.352	-13.38	0.000
repo19c	-52.46	-13.38	-30.39	0.435	-30.39	0.435	-30.39	0.435	-13.38	0.000
repo20c	-44.07	-20.15	-29.90	0.408	-29.90	0.408	-29.90	0.408	-29.90	0.408
repo21c	-44.07	-20.15	-29.90	0.408	-29.90	0.408	-29.90	0.408	-29.90	0.408
repo22c	-48.29	-20.15	-31.72	0.411	-31.72	0.411	-31.72	0.411	-31.72	0.411
repo23c	-32.56	14.07	-15.76	0.640	-15.76	0.640	-15.03	0.624	-15.03	0.624
repo24c	-87.45	-46.30	-52.86	0.160	-52.86	0.160	-52.86	0.160	-46.30	0.000
repo25c	-87.75	-41.07	-50.17	0.195	-50.17	0.195	-50.17	0.195	-41.07	0.000
repo26c	-95.49	-41.07	-57.91	0.309	-57.91	0.309	-57.91	0.309	-41.07	0.000
repo27c	-35.42	2.89	-2.26	0.135	-2.26	0.135	-2.26	0.135	-2.26	0.135
repo28c	-37.32	2.67	-2.48	0.129	-2.48	0.129	-2.48	0.129	-2.48	0.129
repo29c	-38.43	2.67	-2.48	0.125	-2.48	0.125	-2.48	0.125	-2.48	0.125
repo30c	-53.41	0.62	-4.10	0.087	-4.10	0.087	-4.10	0.087	-4.10	0.087
repo31c	-45.80	3.55	2.55	0.020	2.55	0.020	2.55	0.020	2.55	0.020
repo32c	-56.54	1.57	-2.88	0.077	-2.88	0.077	-2.88	0.077	-2.88	0.077
repo33c	-51.95	2.38	-2.06	0.082	-2.06	0.082	-2.06	0.082	-2.06	0.082
repo34c	-286.30	-	5.08	-	5.08	-	5.08	-	5.08	-
repo35c	-1812.71	-	56.66	-	63.25	-	63.43	-	63.43	-
repo36c	-666.46	-	48.42	-	48.42	-	49.76	-	49.76	-
repo37c	-586.78	-	40.97	-	42.21	-	40.97	-	40.97	-
repo38c	-684.21	-	56.66	-	56.66	-	65.69	-	65.69	-
repo39c	-432.91	-	48.42	-	48.42	-	49.71	-	49.71	-
repo40c	-452.54	-	47.71	-	42.99	-	47.94	-	47.94	-
repo41c	-274.47	-	-9.66	-	-9.66	-	-14.52	-	-14.52	-
repo42c	-124.61	-	155.04	-	139.81	-	152.76	-	152.76	-
repo43c	-223.81	-	82.21	-	69.69	-	71.36	-	71.36	-
repo44c	-274.65	-	94.01	-	71.87	-	93.05	-	93.05	-
⊘	-166.26	24.43	27.51	0.149	26.48	0.143	27.56	0.142	29.55	0.082
σ	306.33	81.26	75.25	0.168	74.17	0.170	75.20	0.168	73.66	0.151

Table 4: Comparison of MIP optimal solution profit to the best SA objective on the confidential dataset, in thousands, for each starting heuristic, along with the score of each SA solution. The average and standard deviation of the SA objectives and scores are provided at the bottom of the table. The final columns give the performance of running the MIP for 10 seconds and then switching to GH if no solution is found.

			DRH		SPH		GH		GH+MIP	
ID	SA Baseline	Optimal	Obj.	Score	Obj.	Score	Obj.	Score	Obj.	Score
repo1p	-52.84	-39.83	-39.83	0.000	-39.83	0.000	-39.83	0.000	-39.83	0.000
repo2p	-52.84	-39.83	-39.83	0.000	-39.83	0.000	-39.83	0.000	-39.83	0.000
repo3p	-87.80	-61.77	-61.77	0.000	-61.77	0.000	-61.77	0.000	-61.77	0.000
repo4p	-55.52	-46.62	-46.62	0.000	-46.62	0.000	-46.62	0.000	-46.62	0.000
repo5p	-49.82	-8.21	-8.21	0.000	-8.21	0.000	-8.21	0.000	-8.21	0.000
repo6p	-49.82	-8.21	-8.21	0.000	-8.21	0.000	-8.21	0.000	-8.21	0.000
repo7p	-87.80	-11.49	-11.49	0.000	-11.49	0.000	-11.49	0.000	-11.49	0.000
repo8p	-49.82	-8.21	-11.54	0.080	-8.21	0.000	-11.54	0.080	-8.21	0.000
repo9p	-49.82	-8.21	-11.54	0.080	-8.21	0.000	-12.44	0.102	-8.21	0.000
repo10p	-7.78	137.61	135.12	0.017	135.12	0.017	135.12	0.017	135.12	0.017
repo11p	-7.78	137.61	135.12	0.017	135.12	0.017	135.12	0.017	135.12	0.017
repo12p	-7.78	138.55	136.07	0.017	136.07	0.017	136.07	0.017	136.07	0.017
repo13p	-7.78	138.86	136.07	0.019	136.07	0.019	136.07	0.019	136.07	0.019
repo14p	-7.78	138.86	136.07	0.019	136.07	0.019	136.07	0.019	136.07	0.019
repo15p	-119.14	-36.59	-36.59	0.000	-36.59	0.000	-36.59	0.000	-36.59	0.000
repo16p	-228.99	-36.59	-36.59	0.000	-36.59	0.000	-36.59	0.000	-36.59	0.000
repo17p	-32.64	-9.36	-21.90	0.539	-21.36	0.516	-21.36	0.516	-9.36	0.000
repo18p	-27.07	5.22	-11.34	0.513	-11.34	0.513	-11.30	0.512	5.22	0.000
repo19p	-36.74	5.22	-21.44	0.635	-21.44	0.635	-21.44	0.635	5.22	0.000
repo20p	-34.08	-11.85	-20.16	0.374	-20.16	0.374	-20.16	0.374	-20.16	0.374
repo21p	-34.08	-11.85	-20.16	0.374	-20.16	0.374	-20.16	0.374	-20.16	0.374
repo22p	-41.58	-11.85	-23.39	0.388	-23.39	0.388	-23.39	0.388	-23.39	0.388
repo23p	-34.80	5.22	-22.68	0.697	-21.98	0.680	-22.72	0.698	-22.72	0.698
repo24p	-62.96	-53.89	-53.89	0.000	-53.89	0.000	-53.89	0.000	-53.89	0.000
repo25p	-66.33	-53.13	-53.89	0.058	-53.89	0.058	-53.89	0.058	-53.13	0.000
repo26p	-66.33	-53.13	-53.89	0.058	-53.89	0.058	-53.89	0.058	-53.13	0.000
repo27p	-51.56	-28.20	-28.53	0.014	-28.53	0.014	-28.53	0.014	-28.53	0.014
repo28p	-50.83	-32.13	-32.14	0.001	-32.14	0.001	-32.14	0.001	-32.14	0.001
repo29p	-49.36	-32.13	-32.14	0.001	-32.14	0.001	-32.14	0.001	-32.14	0.001
repo30p	-45.86	5.72	5.06	0.013	5.06	0.013	5.06	0.013	5.06	0.013
repo31p	-48.21	-12.08	-12.08	0.000	-12.08	0.000	-12.08	0.000	-12.08	0.000
repo32p	-44.50	-10.92	-10.92	0.000	-10.92	0.000	-10.92	0.000	-10.92	0.000
repo33p	-44.34	-10.92	-10.92	0.000	-10.92	0.000	-10.92	0.000	-10.92	0.000
repo34p	-355.56	-	-23.83	-	-23.83	-	-23.83	-	-23.83	-
repo35p	-1353.80	-	-24.02	-	-23.83	-	-24.02	-	-24.02	-
repo36p	-799.76	-	-23.83	-	-23.83	-	-23.83	-	-23.83	-
repo37p	-933.23	-	-23.83	-	-23.83	-	-23.83	-	-23.83	-
repo38p	-818.55	-	-24.05	-	-23.83	-	-23.83	-	-23.83	-
repo39p	-639.55	-	-23.83	-	-23.83	-	-23.83	-	-23.83	-
repo40p	-659.15	-	-23.83	-	-23.83	-	-23.83	-	-23.83	-
repo41p	-294.08	-	-75.10	-	-74.88	-	-70.92	-	-70.92	-
repo42p	-214.73	-	94.88	-	95.10	-	88.25	-	88.25	-
repo43p	-245.58	-	39.91	-	39.00	-	38.54	-	38.54	-
repo44p	-325.07	-	39.51	-	41.81	-	44.31	-	44.31	-
$\oslash$	-189.40	2.30	-2.87	0.119	-2.64	0.113	-2.85	0.119	-1.39	0.059
σ	295.48	60.33	56.57	0.208	56.56	0.207	56.26	0.207	56.07	0.156

Table 5: Comparison of MIP optimal solution profit to the best SA objective on the public dataset, in thousands, for each starting heuristic, along with the score of each SA solution. The average and standard deviation of the SA objectives and scores are provided at the bottom of the table. The final columns give the performance of running the MIP for 10 seconds and then switching to GH if no solution is found.

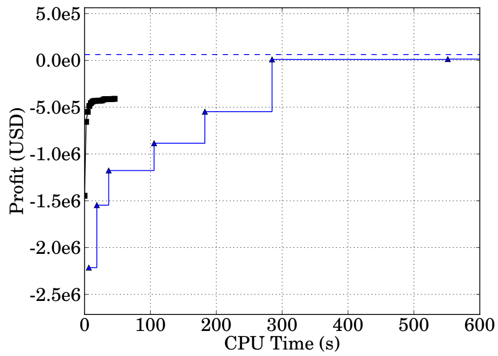
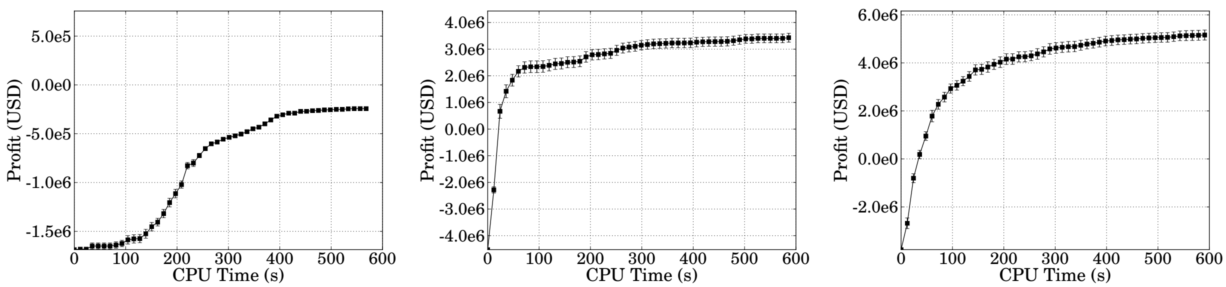


Figure 8: SA algorithm with the SPH initial solution (black, squares) vs. the MIP (blue, triangles) on instance repo30c.



(a) Instance repo33c

(b) Instance repo36c

(c) Instance repo44c

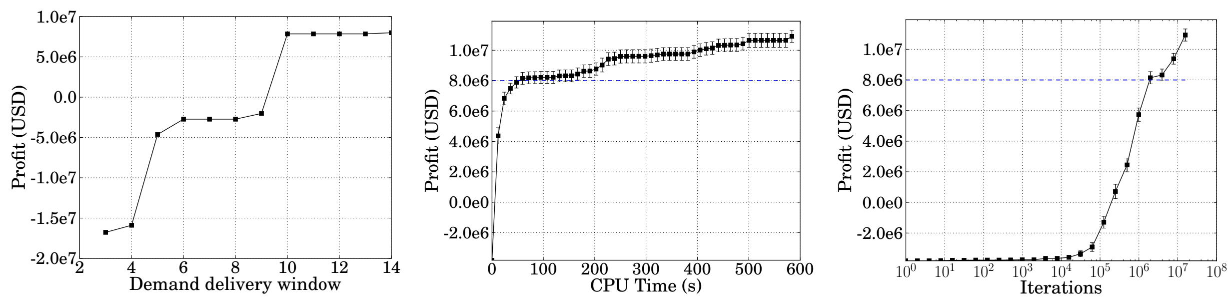
Figure 9: Plots of the average solution quality of the SA (black, squares) algorithm with the SPH initial solution versus MIP solutions (blue, triangles) when available.

high quality solutions significantly faster than the MIP. Given enough patience the MIP eventually does overcome the SA, which converges around time 60. In fact, on instance repo30c, the MIP takes 5 times as long as the SA to find a solution as good as the SA's. In addition, the SA is able to achieve an average objective of -5×10^5 with low standard error in just 6 seconds, compared to 200 for the MIP.

Figure 9 shows a comparison between the SA algorithm with the SPH initial solution during optimization (black line with squares). The solution quality of the MIP is shown in blue with triangles, with the dashed line denoting the optimal solution value. Figures 9a, 9b and 9c show the average SA performance on instances repo33c, repo36c and repo44c, respectively, with error bars representing the standard error across all 25 seeds. These instances have no comparison to the MIP because the MIP is unable to find any integer solutions on these problems. On both instances repo36c and repo44c, the SA is able to quickly find a good solution, which is then slowly improved until the time limit is reached, whereas on instance repo33c, the SA is unable to improve the objective significantly until later in the optimization. Most instances resemble repo36c and repo44c, with the SA improving significantly early, and then leveling out. This behavior is good for a decision support system where a user may sometimes be in a rush to have an answer to a repositioning instance, or is continually interacting with the system and wants quick feedback.

5.6 Reference Scenario Comparison

We foresee our SA algorithm being used as part of a decision support system for the LSFRP. In order to test whether or not the SA algorithm would be useful in such a system, we compare the results of our algorithm with a reference scenario from our industrial collaborator. The scenario, instance repo42c, encompasses 11 vessels originating from 3 initial services. The vessels seek to create a new service that visits the east coast of South America, Spain and the Middle East. The vessels have a single SOS each week that can be used from Tanjung Pelepas, Malaysia, to Jebel Ali,



(a) Reference solution profit. (b) SA CPU Time (c) SA Iterations

Figure 10: The profit of the reference solution with varying demand window sizes and the CPU time and number of iterations used to solve the reference scenario (instance repo42c) with the GH initial heuristic.

United Arab Emirates.

Since the reference solution to the scenario faced by our industrial partner was created in advance of the repositioning happening (as one would expect), the people who made it were at a disadvantage compared to our algorithm, which has a more complete view of the opportunities available during the full repositioning period. In order to counter act this unfairness, we calculate the profit of the reference solution under varying relaxations of restrictions present in our model.

Figure 10a shows the total profit earned by the reference solution as the size of the *demand delivery window* is increased. This window determines what visitations may be used to deliver cargo. Our real-world data only contains when demands were delivered, not the deadline for delivery. Thus, in our model, we use a value of ± 3 days for the demand delivery window, which means that any visitation within three days of the delivery date is used as a feasible delivery visitation for a demand. By relaxing this demand window to larger values, we allow the reference solution more flexibility as to where cargo gets delivered, which counter-acts the uncertainty planners had when creating the solution. The reference solution profit peaks at $7,996,908 when the demand delivery window is relaxed to greater than 10 days.

Instance 42c represents this scenario, and Figures 10b and 10c show the profit of the incumbent solution of the SA algorithm using the initial solution generated by DRH in terms of the CPU time and the number of SA iterations, respectively. Error bars display the standard error across all 25 runs of the instance. The dashed blue line shows the profit of the reference solution with a 14 day demand delivery window. The SA algorithm with the DRH initial solution is able to overcome the reference solution profit in under 100 seconds, and converges at an average value of $10.9 million. Note that SPH and GH have similar performance, and are thus not shown in the figure. The best solution found by SA with DRH has an objective of $15.5 million, a full $7 million higher than the best profit in the reference solution. The majority of this extra profit comes from starting the goal service earlier than the reference solution. Even when the goal service is not started earlier than in the reference solution, the SA algorithm is able to achieve an objective of $8.3 million, which is still over $300,000 more than in the reference solution. In this case, rather than flowing more cargo, costs are saved by omitting unprofitable visitations.

6 Conclusion

We presented a novel model of an important real-world problem, the LSFRP, and solved it using a MIP model on a constraint embedded graph and an SA approach. Our model takes into account all of the key aspects of the LSFRP, including liner shipping service construction constraints, cargo flows, empty equipment repositioning, cabotage restrictions, and sail-on-service opportunities, and maximizes the profit earned during repositioning. We evaluated our MIP and SA approaches, showing that not only does the SA scale to real-world sized problems, but it is also able to find a solution with a significantly higher profit than that of the reference solution from our industrial collaborator.

Our modeling techniques, especially our graph construction, could be applicable to other liner shipping problems, such as an extension to the Vessel Schedule Recovery Problem (Brouer et al., 2013) if SOS opportunities and flexible visitations were included. Additionally, our SA approach, in particular our demand destination completion heuristic and initial solution heuristics, could be useful in other dual-layer flow problems, in which a multi-commodity flow is directed through the graph by the paths of vehicles.

For future work, we intend to use a branch and price framework to overcome scaling issues in our MIP model in order to solve large instances to optimality. Given the large amounts of money involved in the LSFRP, optimal solutions to repositioning problems can give liner shippers a critical edge over their competition.

References

R. Agarwal and Ö. Ergun. Ship scheduling and network design for cargo routing in liner shipping. *Transportation Science*, 42(2):175–196, 2008.

J.F. Álvarez. Joint routing and deployment of a fleet of container vessels. *Maritime Economics and Logistics*, 11(2): 186–208, June 2009.

M.W. Andersen. *Service Network Design and Management in Liner Container Shipping Applications*. PhD thesis, Technical University of Denmark, Department of Transport, 2010.

C. Ansotegui, M. Sellmann, and K. Tierney. A Gender-Based Genetic Algorithm for the Automatic Configuration of Algorithms. In Ian P. Gent, editor, *Principles and Practice of Constraint Programming (CP 2009)*, volume 5732 of *LNCS*, pages 142–157. Springer, 2009.

B.D. Brouer, J. Dirksen, D. Pisinger, C.E.M Plum, and B. Vaaben. The Vessel Schedule Recovery Problem (VSRP) – A MIP model for handling disruptions in liner shipping. *European Journal of Operational Research*, 224(2): 362–374, 2013. ISSN 0377-2217.

M. Christiansen. Decomposition of a combined inventory and time constrained ship routing problem. *Transportation Science*, 33(1):3–16, 1999.

M. Christiansen, K. Fagerholt, and D. Ronen. Ship routing and scheduling: Status and perspectives. *Transportation Science*, 38(1):1–18, 2004.

M. Christiansen, K. Fagerholt, B. Nygreen, and D. Ronen. Maritime transportation. *Handbooks in operations research and management science*, 14:189–284, 2007.

J. Clausen, A. Larsen, J. Larsen, and N.J. Rezanova. Disruption management in the airline industry–concepts, models and methods. *Computers & Operations Research*, 37(5):809–821, 2010.

T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. Introduction to algorithms (third edition), 2009.

Google. Google OR-Tools. http://code.google.com/p/or-tools/, 2012.

H.H. Hoos and T. Stützle. *Stochastic local search: Foundations & applications*. Morgan Kaufmann, 2004.

D.S. Johnson, C.R. Aragon, L.A. McGeoch, and C. Schevon. Optimization by Simulated Annealing: An Experimental Evaluation; Part I, Graph Partitioning. *Operations Research*, 37(6):865–892, 1989.

S. Kirkpatrick, CD Gelatt, and MP Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

N. Kohl, A. Larsen, J. Larsen, A. Ross, and S. Tiourine. Airline disruption management–perspectives, experiences and outlook. *Journal of Air Transport Management*, 13(3):149–162, 2007.

J.E. Korsvik, K. Fagerholt, and G. Laporte. A large neighbourhood search heuristic for ship routing and scheduling with split loads. *Computers & Operations Research*, 38(2):474 – 483, 2011.

H.W. Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

B. Løfstedt, J.F. Alvarez, C.E.M. Plum, D. Pisinger, and M.M. Sigurd. An integer programming model and benchmark suite for liner shipping network design. Technical Report 19, DTU Management, 2010.

B.J. Powell and A.N. Perakis. Fleet deployment optimization for liner shipping: An integer programming model. *Maritime Policy and Management*, 24(2):183–192, Spring 1997.

R. Stahlbock and S. Voß. Operations research at container terminals: a literature update. *OR Spectrum*, 30(1):1–52, 2008.

B.G. Tabachnick and L.S. Fidell. *Using multivariate statistics*. Pearson, 2012.

J. Taheri and A.Y. Zomaya. A simulated annealing approach for mobile location management. *Computer communications*, 30(4):714–730, 2007.

K. Tierney and R. M. Jensen. The Liner Shipping Fleet Repositioning Problem with Cargo Flows. In Hao Hu, Xiaoning Shi, Robert Stahlbock, and Stefan Voß, editors, *Computational Logistics*, volume 7555 of *Lecture Notes in Computer Science 7555*, pages 1–16. Springer, 2012.

K. Tierney, A.J. Coles, A.I. Coles, C. Kroer, A.M Britt, and R.M. Jensen. Automated planning for liner shipping fleet repositioning. In L. McCluskey, B. Williams, J.R. Silva, and B. Bonet, editors, *Proceedings of the 22nd International Conference on Automated Planning and Scheduling*, pages 279–287, 2012.

United Nations Conference on Trade and Development (UNCTAD). Review of maritime transport, 2011.

A Graph Formalization

In this appendix, we provide the details of our graph structure. The following parameters are used to define the graph.

S	Set of ships, indexed by s.
L	Set of phase-in slots, where $\lvert L\rvert = \lvert S\rvert$, indexed by ℓ.
SOS	The set of SOS slots, indexed by o.
R_ℓ^{PI}	Set of visitations of phase-in slot $\ell \in L$.
R_s^{PO}	Set of phase-out visitations of vessel $s \in S$.
$O_o^{\{P,TS,T,E\}}$	Sets of parallel, transshipment, transit, and end visitations, with $o \in SOS$.
V^R	Set of non-SOS inflexible visitations, $V^R = \bigcup_{\ell \in l} r_\ell^{pi} \bigcup_{s \in s} r_s^{po}$.
TZ	Set of trade zones.
$z_i \in TZ$	Trade zone of visitation $i \in V$.
$enter(i) \in \mathbb{R}^+$	Time a vessel begins inflexible visitation $i \in V^i$.
$exit(i) \in \mathbb{R}^+$	Time a vessel ends inflexible visitation $i \in V^i$.
$\tau \in V$	Graph sink, which is not an actual visitation.
$V' = V \setminus \tau$	Set of nodes without the graph sink.
$\Delta_{i,j}$	Minimum time required for any ship to sail from visitation i to j.
$A^{SD}(R)$	Set of arcs connecting subsequent visitations in the visitation set R.
A^{PO}	Set of arcs connecting phase-out slots to phase-in slots.
A^{PI}	Set of arcs from phase-in visitations to same trade zone phase-out visitations.
A^τ	Set of arcs from the phase-in visitations to the graph sink.
A^f	Set of arcs connecting flexible visitations to other visitations.
$\hat{A}_o^{In}$	Set of arcs connecting to the start nodes of $o \in SOS$.
$\hat{A}_o^{Out}$	Set of arcs extending from the end nodes of $o \in SOS$.
$\hat{A}_o^{PTS}$	Set of arcs connecting the parallel nodes to transshipment nodes of $o \in SOS$.

$\hat{A}_o^{TST}$	Set of arcs connecting transshipment nodes to transit nodes of $o \in SOS$.
$\hat{A}_o^{TT}$	Set of arcs between transit nodes of $o \in SOS$.
$\hat{A}_o^{EE}$	Set of arcs between sequential end nodes of $o \in SOS$.
$\hat{a}_o^{TE}$	Arc from the latest transit node in $o \in SOS$ to its earliest end node.

We define the set of inflexible nodes as $V^i = \bigcup_{\ell \in l} r_\ell^{pi} \bigcup_{s \in s} r_s^{po} \bigcup_{o \in SOS} (O_o^P \cup O_o^T \cup O_o^{TS} \cup O_o^E)$. The set of flexible visitations, V^f, contains all visitations that have equipment surpluses/deficits such that $V^f \cap V^i = \emptyset$. In order to formally define the set of arcs contained in the graph, let $\mathit{follows}(i,j) \in \mathbb{B}$ return *true* if and only if visitation j is scheduled on any service to immediately follow visitation i, with $i, j \in V^i$. In addition, we let $\mathit{can\text{-}sail}(i,j) \in \mathbb{B}$ be *true* if and only if $\mathit{enter}(j) \geq \mathit{exit}(i) + \Delta_{i,j}$, where $i, j \in V'$. This indicates whether or not it is possible to sail between two visitations at the fastest speed of the fastest vessel in the model. Note that all of the arc sets are disjoint. We now formally define all of the previously mentioned sets of arcs.

$$
\begin{aligned}
A^{SD}(R) &= \{(i,j) \mid i,j \in R \wedge \mathit{follows}(i,j)\}, R \in \bigcup_{s \in S} \{R_s^{PO}\} \bigcup_{\ell \in L} \{R_\ell^{PI}\} \\
A^{PO} &= \{(i,j) \mid i \in \bigcup_{s \in S} R_s^{PO} \wedge j \in \bigcup_{\ell \in L} R_\ell^{PI} \wedge \mathit{can\text{-}sail}(i,j)\} \\
A^{PI} &= \{(i,j) \mid i \in \bigcup_{\ell \in L} R_\ell^{PI} \wedge j \in \bigcup_{s \in S} R_s^{PO} \wedge z_i = z_j \wedge \mathit{can\text{-}sail}(i,j)\} \\
A^{\tau} &= \{(i,\tau) \mid i \in \bigcup_{\ell \in L} \operatorname*{argmax}_{i' \in R_\ell^{PI}} \{\mathit{exit}(i')\}\} \\
A^{f} &= \{(i,j) \mid \ \big((i \in V^f \wedge j \in V^R) \vee (i \in V^R \wedge j \in V^f)\big) \wedge z_i = z_j\} \\
\hat{A}_o^{In} &= \{(i,j) \mid i \in \bigcup_{s \in S} R_s^{PO} \wedge j \in (O_o^P \cup O_o^{TS}) \wedge \mathit{can\text{-}sail}(i,j)\} \\
&\qquad \bigcup \{(i,j) \mid i \in V^f \wedge j \in (O_o^P \cup O_o^{TS}) \wedge z_i = z_j \wedge \mathit{can\text{-}sail}(i,j)\} \\
\hat{A}_o^{Out} &= \{(i,j) \mid i \in O_o^E \wedge j \in \left(\bigcup_{\ell \in L} R_\ell^{PI} \bigcup_{o' \in \{SOS \setminus o\}} (O_{o'}^P \cup O_{o'}^{TS}) \right) \wedge \mathit{can\text{-}sail}(i,j)\} \\
&\qquad \bigcup \{(i,j) \mid i \in O_o^E \wedge j \in V^f \wedge z_i = z_j \wedge \mathit{can\text{-}sail}(i,j)\} \\
\hat{A}_o^{PTS} &= \{(i,j) \mid i \in O_o^P \wedge j \in O_o^{TS} \wedge \mathit{follows}(i,j)\} \\
\hat{A}_o^{TST} &= \{(i,j) \mid i \in O_o^{TS} \wedge j \in O_o^T \wedge \mathit{follows}(i,j)\} \\
\hat{A}_o^{TT} &= \{(i,j) \mid i,j \in O_o^T \wedge \mathit{follows}(i,j)\} \\
\hat{A}_o^{EE} &= \{(i,j) \mid i,j \in O_o^E \wedge \mathit{follows}(i,j)\} \\
\hat{a}_o^{TE} &= (\operatorname*{argmax}_{i \in O_o^T} \{\mathit{exit}(i)\}, \operatorname*{argmin}_{j \in O_o^E} \{\mathit{enter}(j)\})
\end{aligned}
$$

The set of all arcs in the graph, A, is therefore defined by

$$
\begin{aligned}
A = &\bigcup_{s \in S} \left(A^{SD}(R_s^{PO})\right) \bigcup_{\ell \in L} \left(A^{SD}(R_\ell^{PI})\right) \cup A^{PI} \cup A^f \cup A^\tau \\
&\bigcup_{o \in SOS} \left(\hat{A}_o^{In} \cup \hat{A}_o^{Out} \cup A_o^{ST} \cup \hat{A}_o^{TT} \cup \hat{A}_o^{EE} \cup \hat{a}_o^{TE}\right).
\end{aligned}
$$