Disruption Management: The Vessel Schedule Recovery Problem (VSRP)

Bachelor Thesis
Econometrics and Operational Research

Author:
Melissa Koenen (410887)

Supervisor:
Nemanja Milovanović
Second assessor:
Prof. Dr. Ir. Rommert Dekker

July 2, 2017
Erasmus University Rotterdam
Erasmus School of Economics
Department of Econometrics

Abstract

The Vessel Schedule Recovery Problem (VSRP) is the problem where the proforma schedule of vessels needs to be modified because of a delay. This problem is still scarcely researched and therefore a lot of improvements can be made. As liner shipping companies need to make decisions fast, results must be generated such that they can be used in real-life decision making. A MIP formulation with an underlying time-space network as stated by Brouer et al. (2013) is used to test how the formulation reacts when varying the duration of a shift, the number of vessels and port calls. As the original formulation is only able to handle disruptions by means of changing the speed of a vessel, omitting port calls and swapping port calls, the model is extended such that it can capture the cut-and-run policy as well. This policy enables vessels to leave their port call earlier, before finishing it. The computation times of our implementation react strongly when increasing the number of port calls, however it is more robust when increasing the number of vessels. As there is a trade-off between between cost reduction and time when the duration of the shift is decreased, we advise to not take this duration too small. In general our extension gives on average a relative large cost reduction in case the number of vessels is large.

Contents

1 Introduction

Over the last few decades container shipping has played a tremendous role in the globalisation of the world economy. More than 80% of the total traded volume and around two third of the total traded value is carried by sea (UNCTAD, 2016). Commercial ships are generally divided into the following three modes: *liner*, *industrial* or *tramp*. Liner ships differentiate themselves from other ships as they operate according to an iterative route (*itinerary*), in which they visit ports with a certain time window. The schedule is often predetermined a few years ahead (Christiansen et al., 2013).

Liner ships carry containers as their cargo, as containers have the advantages that they all have the same rectangle shape. A standardized measure for the size/volume of a container is called a TEU (twenty-foot equivalent unit) and a common container is often 1 or 2 TEU large. Liner ships nowadays can easily carry over 14,000 TEU (MAN, 2009), where the OOCL Hong Kong is currently the largest with 21,413 TEU[1]. Containers usually travel in groups, which means that they have the same origin, destination and expected time schedule.

Companies in the liner shipping sector referred to as *carriers* have a need to stay competitive as the earned return is less than 10% on assets (Stopford, 2009). A way to outperform others is by designing a hub-and-spoke network for vessels which is as efficient as possible and by offering short transit times to attract customers. However, offering short transit times means that almost no buffer time is added. Small delays can then easily disrupt the whole schedule as schedules are often intertwined. Container groups can be such delayed, that they can not be delivered on time. Furthermore, it could also happen that a container group is misconnected along its journey. As there often is a weekly service between two consecutive ports on an itinerary, the container group is ultimately delayed as well. Notteboom (2006) states that in exceptional busy periods approximately 70%–80% of the vessel round trips experiences delays in at least one port. These disruptions are commonly caused by port congestions, slacking off at loading/discharging, weather circumstances or on route technical problems.

Carriers nowadays use a variety of actions to cover up for delays and misconnections caused by those disruptions. The applied actions, as discussed by Notteboom (2006), are:

- **Speed up vessel**
  This recovery seems to be a trivial solution, however as the bunker cost (fuel cost) is a cubic function of speed (Alderton, 2004), altering the speed slightly would have a huge impact on the total costs, which is about half of the total operating costs. The speed of a vessel is also limited between a lower and upper limit, therefore speeding up a vessel does not always necessarily imply that there will be no delay.
- **Speed up port call**
  Using more cranes and crew to finish the call earlier. For example, Antwerp is known as a port which can boost their productivity to bring a vessel back on schedule.
- **Cut-and-run**
  A special case of speeding up a port call, which is often used in tide-dependent terminals, where the loading/discharging is stopped earlier (cut) such that that the vessel can leave

[1] `http://www.oocl.com/eng/pressandmedia/pressreleases/2017/Pages/12may17.aspx`
Last accessed at: 23-June-2017.

the port on time (run). In such a case not all appointed containers will be loaded or unloaded to prevent unproductive port time because of the low tide.

- **Omitting a port call**
  When a ship is severely delayed, it could decide to skip a port call. This method is very effective and can generate a large time buffer. However, the container groups which should have been discharged at that particular port will be delayed for their delivery in that port or experience a misconnection.
- **Swapping port calls**
  This is a method commonly used, where the order of port calls is slightly shuffled such that the important container groups are delivered on time.
- **Insert idle vessel**
  These vessels temporarily take over part of the schedule, which can be very costly as such vessels would be idle most of the time.

Several of those actions will lead to the redirection of transport or even to the arranging of new transport. Delayed cargo experiences economic depreciation and dissatisfied customers could thus decide to make claims or use a different carrier in the future, which leads to lost revenues. It is therefore important to reschedule the delayed vessels such that the carrier will not be burdened with too much extra costs.

As it is difficult to oversee all (financial) impacts of handling a disruption in a certain way, it is important to have a good mathematical model which can pinpoint the least costly decision. Stakeholders can sometimes have other opinions about which decision should be taken, therefore carriers need to have strong evidence to convince them of their own decision. However, this model should also be able to give quick results as disruptions can occur suddenly and many other decisions must be taken simultaneously. Therefore the following question plays a role in this thesis:

Is it possible to have an efficient model to handle liner shipping disruptions in several ways, which can be used for real-time decision making?

In Section 2 literature regarding the problem is presented. Section 3 formalizes the problem and Section 4 describes the data. Section 5 describes the methodology used to solve the problem and Section 6 discusses the results. Finally, a conclusion is presented in Section 7.

2 Literature Review

In common literature the disruption management concerning liner shipping is a scarcely researched topic. Only in the last couple of years the problem has been addressed. This work is often heavily inspired by studies in the field of the Aircraft Schedule Recovery Problem (ASRP), as disruption management of aircraft and vessels show some striking resemblances (Christiansen et al., 2013). Therefore, a coherent review will be presented in which first the ASRP is thoroughly discussed. For a general overview concerning the OR used in shipping over the last 30 years, we refer to Meng et al. (2013) and to Christiansen et al. (2013) in specific for the last decade.

When a disruption occurs, there are three kinds of problems which the aircraft company has to deal with: aircraft recovery, crew recovery and passenger recovery. An analogy to vessels can be drawn in which there is vessel and container recovery. Note that crew recovery does not play a role as there is a fixed crew for each container ship.

The problems are often modelled separately. The aircraft recovery is generally considered as the most important to model properly, as the other recoveries follow from the new proposed schedule. Teodorović and Guberinić (1984) were the first to introduce a network model in which the customer delay was minimized for a single fleet. The model is not realistic enough as they only took flight delays into account. Jarrah et al. (1993) later suggested two models, in which one minimizes the total delay cost and the other minimizes the cancellation cost. The results were useful in practice, however the models were not linked and thus no trade-off exists between delay and cancellation. This problem was eventually solved by Yan and Yang (1996), whose model is based on a Lagrangian relaxation. Their model was tested on real-life data of China Airlines.

The work of Yan and Yang (1996) has several times been extended, however Yan and Tu (1997) made it possible to include multiple fleets, where each aircraft type is viewed as a separate commodity. As aircraft companies often want to have reschedules that do not differ much from the original ones, Thengvall et al. (2001) contributed to previous work with the use of so-called protection arcs in a multi-commodity network flow. As this model is still on fleet level instead of on aircraft level, it is more difficult to keep track of the maintenance of a single plane. Dienst et al. (2012) presented a model on aircraft level which is able to capture much more real-life constraints. The model uses a unit action penalty, which punishes actions that divert from the original schedule.

The papers discussed so far have only considered delaying/cancelling a flight, swapping aircraft or using a stand-by aircraft to recover the schedule. Marla et al. (2016) also incorporated flight planning, which enables flight speed changes. Promising results are presented which show that passenger disruptions were reduced by 66%–83%. The work of Marla et al. (2016) is a generalisation of Bratu and Barnhart (2006) and Bratu (2003). The mathematical models suggested by Dienst et al. (2012) and Marla et al. (2016) inspired Dirksen (2011) to construct a similar formulation for the disruption problem of vessels, which he addressed as the Vessel Schedule Recovery Problem (VSRP). The MIP model adopts the idea of the flow-conservation and the set-partitioning constraints and uses a time-space graph as its underlying network. The model is able to make use of the following techniques to recover the schedule: speed up a vessel, omitting a port call or swapping a port call.

The formulation of this model was later refined in Brouer et al. (2013) and tested on four cases of a large carrier. The model is able to give real-time results (< 5 seconds) and produces outcomes comparable or superior to those chosen by operations managers. However, it is important to note that these instances are relatively small, especially as the average number of port calls per vessel is not that large (≈ 4). They then suggested a method to create artificial instances to show how the computational time increases when the number of port calls and the number of vessels change. The computational times seem in both cases to increase exponentially, especially when the number of port calls increase. In both cases building the graph becomes much more complex and time-consuming.

Li et al. (2015) recently investigated the VSRP as well, however their method has a focus on the catch up of a single vessel instead of multiple vessels. A NLP formulation is given for the speeding up of vessels as a recovery possibility and a Dynamic Programming algorithm is presented to also incorporate swapping/omitting port calls. As the aim of this thesis is to handle disruptions, which can corrupt a whole system and multiple vessels, we suggest an extension to the suggested model for the VSRP of Brouer et al. (2013). This extension includes another recovery action: cut-and-run.

3 Problem Definition

The VSRP can be defined as the problem where the proforma schedule is modified because of a delay, such that in the end the costs of the recovery is minimized and the schedule is still valid. Initially given is a set of vessels V and a set of ports P, where each vessel v visits several ports according to a proforma schedule. This proforma schedule consists of an ordered set of port calls, which is denoted by $H_v \subseteq P$. Of each vessel v some basic information is known, such as its design speed, minimum and maximum speed and container capacity.

Furthermore, it is given what the origin, destination, transshipment port(s), final arrival time and size are of each container group $c \in C$, where C is the set of all container groups. As it is known in which port each vessel v will experience its delay, a set of alternative schedules is suggested using several recovery actions. For each port call $h \in H_v$ a set of possible sailings (directed edges) L_h can be determined, where each sailing e represents a different speed. A valid schedule has vessels starting at their initial port and ending at their final destination, such that sailings are bounded by the speed of the vessel and the time. It must also have a plan for the transportation of each container group when there is a delay or a misconnection.

To determine the cost of each reschedule, the bunker cost, port fees and costs of (possible) delays and misconnections of cargo must be known. Define c_e^v as the cost of using a particular sailing e and the port fee of the sailings target port of a vessel v. Define c_c^m and c_c^d as the total cost of a misconnection or delay of a container group c respectively. The chosen optimal recovery schedule will be a valid schedule that minimizes the total costs of the delay and misconnection of cargo, port fees and used sailings.

4 Data

It is in general hard to get data from carriers to test real-life instances, as their data is confidential and they do not want to want to risk exposure of valuable information to their competitors. However, it is possible to generate artificial data to create some test instances. Dirksen (2011) and Brouer et al. (2013) suggested a process to create such instances, in which a grid of η^2 ports is created. The ports on one of the diagonals are considered as the hubs and the others as the spokes, where the distance between the ports is calculated according to the Euclidean distance. The difference between the two kind of ports is that hubs are more likely to be used as transshipment point for container groups.

The next step is to generate vessels, such that each is randomly assigned $\kappa < \eta^2$ ports to visit without replacement. The order in which the ports are drawn determines the route of the vessel. Each ship is assigned a uniform randomly chosen capacity m_v in TEU from the set $M = \{m_v \mid m_v \in [900, 20.000], m_v \in \mathbb{Z}_+\}$ and a design speed s_v in knots per hour from the set $S = \{s_v \mid s_v \in [15, 22], s_v \in \mathbb{R}_+\}$. When the design speed and capacity are known, it is possible to obtain the design consumption u_v. Since there are economies of scale, in which a larger vessel is able to transport a single container cheaper than a smaller vessel, the design consumption must be a non-linear convex function of the capacity. Moreover, the proforma schedule of each vessel can now easily be determined by the distance between each of two successive ports and the design speed of the selected vessel. It is assumed that unloading/loading at each port takes 24 hours, before the vessel can continue to sail to its next port. As the ships are randomly assigned to start their schedule during the week, the problem is constructed without loss of generality. Figure 1 shows an example of a random grid with $\eta = 10$ and 5 vessels, in which each vessel has 5 port calls.

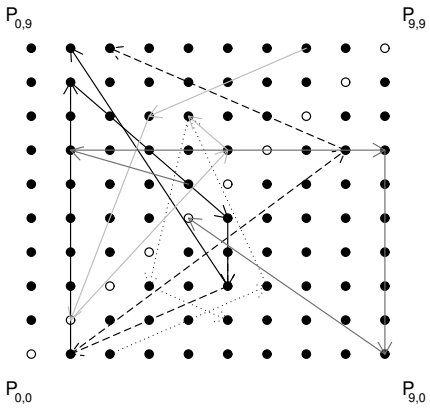


Figure 1: Graphical representation of a random grid with $\eta = 10$, $\kappa = 5$ and 5 vessels. The empty circles on the diagonal represent the hubs, whereas the other circles represent the spokes. The route of different vessels is indicated with distinct lines.

As the ports and routes of the vessels are at this point generated, the container itineraries can be assigned. While assigning the container itineraries, it is assumed that vessels have an unlimited capacity. For each vessel its schedule is then selected in which for each intermediate port with a certain probability π^h_{OTD} an arriving container group of a random number of units is assigned. OTD stands for *On Time Delivery* and h is the type of port (hub or spoke). π^{Hub}_{OTD} is set to 1, whereas π^{Spoke}_{OTD} is set to 0.5. Furthermore, if another vessel would arrive at that same port afterwards according to the proforma schedule, with a probability π^h_{MC} a container group of random number of units is planned to transship between the

vessels. *MC* stands for *Miss-Connected* and h is again the type of port. π_{MC}^{Hub} is set to 1, whereas π_{MC}^{Spoke} is set to 0.5. After the container group is transshipped onto the next ship, its final destination is chosen randomly out of the remaining ports on the route. Note that there is at most one transshipment per container group.

With a certain probability π_D the first port of each vessel will then be delayed. The exponential distribution is used to generate this delay. As larger vessels experience in general more delay, the following linear relationship is used to determine the average delay:

$$d_v = 9.53 + 0.00052356 \cdot m_v,$$

where d_v stands for the average delay and m_v for the capacity of vessel v. Note that the initial delay should not be too large, as the suggested recoveries may be infeasible because of the way the network is generated. To cover up for this delay several discussed recovery actions can be used.

As it known how many containers are shipped in each container group c, the costs of a potential misconnection c_c^m and delay c_c^d of this container group can be calculated. The following formulas are used to determine those costs:

$$c_c^m = c^m n_c,$$
$$c_c^d = c^d n_c,$$

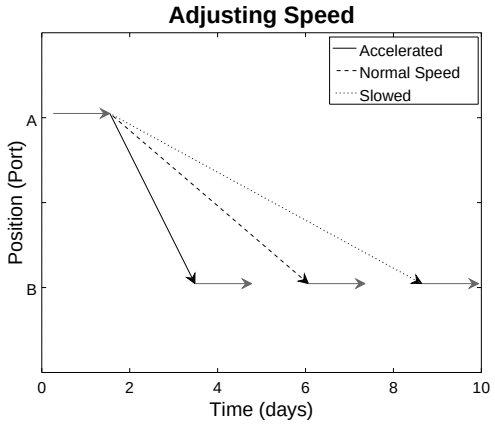
where c^m and c^d stand for the cost of a misconnection and a delay of a single container respectively and n_c for the number of containers in container group c. In a survey of Maersk Line[2], it was stated by a global retailer that 70% of his cargo loses on average 25% of its retail value when it is a week late. Assuming that the average value of a container within one shipment is €30,000, the cost of delay is €7,500 per container. Therefore, it is assumed that $c^m = €1,000$ and $c^d = €1,000$ are reasonable costs to expect when having a misconnection and/or delay of a single container. Note that the costs of a misconnection should always be punished properly relative to the costs of a delay as a misconnection means that a container group is delayed for at least a week (because of the way itineraries are constructed).

[2] `http://www.maerskline.com/en-us/shipping-services/~/media/B39F084693AD4B0DAB7663DAEB477F5A.ashx`
Last accessed at: 11-June-2017.

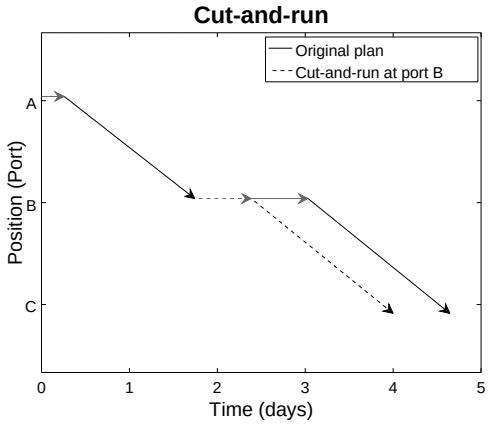
5 Methodology

5.1 Graph Topology

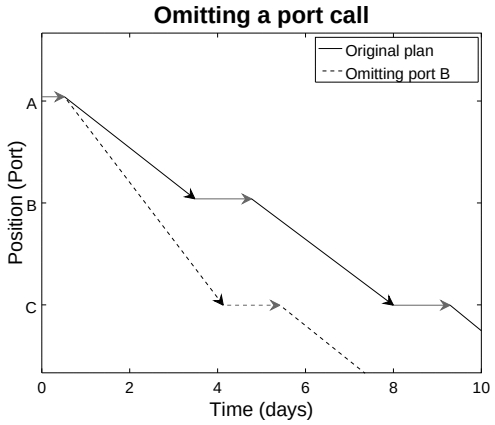
In order to represent the problem properly, a time-space network is used to generate all possible routes a vessel could sail after experiencing an initial delay. In such a network positions are often connected across time, which in the case of the VSRP means that a port call $h \in H_v$ for each vessel vessel $v \in V$ is not only bounded to its physical location, but also to the time in which it is visited. The time horizon consists of discrete timeslots $t \in T$, in which it is assumed that ports are only visited at a beginning of a shift, such that the network is reduced in size. In Figure 2a an example of such a network is shown, where the horizontal axis depicts the time and the vertical axis the geographical position of a vessel. This vessel can travel from port A to B with different speeds, which is shown by the slanted lines. The steepness indicates the speed of the vessel; how steeper, how faster. The horizontal lines indicate the port call duration.



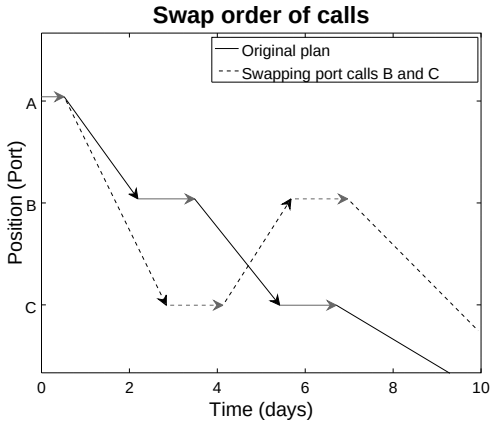
(a) Edges connecting two ports with various sailing speed.



(b) Edges corresponding to applying the cut-and-run policy



(c) Edges corresponding to omitting a port call.



(d) Edges corresponding to changing the order of the port calls.

Figure 2: Implemented recovery options in the space-time network. Port call edges are horizontal grey lines.

More formally, a directed graph $G = (N, E)$ consists of the node set $N = \{p^t \in N \mid p \in P, t \in T\}$. In this set every port p is linked to a time t, such that p^t represents a port p at time t. Each node $n \in N$ has ingoing and outgoing edges, represented by n^- and n^+ respectively. Each vessel $v \in V$ has its own set of nodes it can visit, denoted by $N_v \subseteq N$. N_v consists of at least a source node n_s^v (first port) and a sink node n_t^v (scheduled final destination). Other nodes are created for each port call $h \in H_v$ in a certain time window $\{a_v^h, b_v^h\}$, defining the earliest and latest arrival time. This time window can, for example, be restricted by the minimum and maximum speed of a vessel or in case of a tide-dependent harbour by the time in which the port is opened.

There are in general two types of edges: sailings between two different ports defined by $E_s = \left\{\left(q^t, p^{t'}\right) \mid q^t, p^{t'} \in N, q \neq p, t \leq t'\right\}$ and sailings in which a vessel is being loaded and unloaded in a particular port p, before it can embark again, defined by $E_g = \left\{\left(p^t, p^{t'}\right) \mid p^t, p^{t'} \in N, t < t'\right\}$. The edge set E contains both sets and can thus be defined as $E = E_s \cup E_g$. To reduce the number of edges in the graph and the number of columns in the mathematical model, it is assumed that the port call duration is fixed (24 hours by construction). Therefore, edges of E_g can be included in E_s such that an edge $e \in E_s$ consists partly of the sailing between q^t and p^t and partly of the port call duration between p^t and $p^{t'}$. In Figure 2 the grey port call edges are thus only explicitly shown to give a better understanding of the graphical representation.

Each vessel has its own set of edges, denoted by $E_v \subseteq E_s$. Depending on the properties of vessel v, the cost c_e^v of using a particular edge $e \in E_v$ can be determined. This cost consists of the bunker cost and the port fee of the target port of the sailing. As already mentioned before, the bunker cost depends highly on the speed of a vessel. Alderton (2004) states a formula to estimate the fuel consumption u as a function of the current speed s:

$$u = u_v \left(\frac{s}{s_v}\right)^3,$$

where u_v and s_v stand for the design consumption and design speed of a vessel v respectively. Multiplying u by the fuel cost, which are assumed to be €400 per ton, the sailing cost can be calculated. Port fees generally depend on the size of the vessel's weight. The Port of Gdańsk in Poland charges port fees of a container vessel of 0.22 eur/gt[3], where gt stands for gross tonnage and is a measure for the vessel's weight. These port fees are used as an approximation for the port fees in the model, where it is roughly assumed that 1,000 gt is proportional to 100 TEU. Therefore, it is possible to calculate the port fees for each port of vessel v.

The final edge set E_s depends on the proforma schedule of the vessel and what kind of recovery actions are considered as a possibility to cover up for the delay. The recovery actions used by Brouer et al. (2013) are changing the speed of the vessel, omitting a port call and swapping port calls. Additionally to the actions suggested, the model is extended to incorporate the cut-and-run policy as well. To illustrate the recovery actions, imagine there are some port calls in the order A, B and C. The possible recovery actions then are:

[3] `https://www.portgdansk.pl/shipping/port-authority-tariff`
Last accessed at: 28-May-2017.

- **Speed up vessel**
  As the speed of a vessel v is bounded, there is a limited number of edges which may connect A and B. Define the set of possible sailings $L_h \subset E_v$ for each port call $h \in H_v$ as $L_h = \left\{ \left(A^t, B^{t'} \right) \mid A, B \in H_v, A < B, t \leq b_v^B, a_v^B \leq t' \leq b_v^B, t < t', t < t', \forall t' = a_v^B + D \cdot \delta_B \right\}$, where D is a positive integer denoting the shift and δ_B the duration of a shift a terminal B. In most cases a_v^B is restricted by the maximum speed of v, whereas b_v^B is restricted by the minimum speed. The minimum speed of v is set to $s_v - 5$ and its maximum speed to $s_v + 3$. A graphical representation can be seen in Figure 2a.

- **Cut-and-run**
  The ship leaves its current port earlier, such that not all assigned cargo is loaded and unloaded. Proportional to the time the ship has been in the port, a number of containers is loaded and unloaded. Determine the fraction $\varepsilon = \dfrac{t_{cut}}{t_{normal}}$, where t_{cut} and t_{normal} represent the port time used in the cut-and-run policy and in the normal situation respectively. This fraction splits the container group c each time the policy is used. When speaking of a partial misconnection, it means that the cut-and-run policy has been used and a part of the container group is misconnected. Depending on how many partial misconnections w have taken place, $1 - \varepsilon^w$ part of the container group is misconnected in the end. Generate all possible sailings with the method used before, however, now t_{cut} is used to determine $\left\{ a_v^h, b_v^h \right\}$. t_{normal} is 24 hours by construction and t_{cut} is set to 12 hours. A graphical representation can be seen in Figure 2b.

- **Omitting a port call**
  Omitting port call B after A, coincides with having a sailing between port A and C. Edges should be created similarly to the method proposed before, however now between A and C. A graphical representation can be seen in Figure 2c.

- **Swapping port calls**
  Swapping port calls would mean that port C is directly visited after A and then B is visited after C. First, edges between A and C are created as in the case of omitting a port call, however additional edges should be created between C and B similarly to the method proposed before. It is assumed that port swaps are only allowed between two consecutive ports on the route. A graphical representation can be seen in Figure 2d.

Algorithm 1 shows how graph G_v with the edge set E_v and node set N_v can be constructed for a particular vessel v according to the suggested recovery actions. A possible edge means a sailing $e \in L_h$ for time frame $\{a_v^h, b_v^h\}$, as defined before. If the earliest arrival time for this port is later than the proforma destination time of the last port, there are no possible edges and the algorithm returns to its while-loop. If the latest arrival for this port is later than the proforma destination time of the last port, set the latest arrival time to the destination time.

Algorithm 1: Construction of directed graph G_v

Data: An ordered set of port calls H_v of vessel v
Result: Directed graph G_v, consisting of all possible recovery actions for vessel v
Initialise an empty directed graph G;
Initialise an empty set ζ_v consisting of nodes;
Put the first port call into ζ_v;
while ζ_v *is non-empty* **do**
    Extract node n_v from ζ_v;
    if n_v *is the source port* **then**
        Generate all possible edges between n_v and its following and subsequent port from H_v (if they exist);
    else if n_v *is an intermediate port* **then**
        Set m_v to be the previous visited node of n_v;
        if m_v *is originally scheduled to succeed* n_v **then**
            Generate all possible edges between n_v and the subsequent port of its direct successor m_v from H_v;
        else
            if m_v *is the direct scheduled predecessor of* n_v **then**
                Generate all possible edges between n_v and its following and its subsequent port from H_v (latter only if it exists);
            else
                Generate all possible edges between n_v and its predecessor, its successor and subsequent port from H_v (latter only it it exists);
            end
        end
        For each generated edge, calculate its sailing cost and port cost c_e^v of the destination port of the edge;
        Add the created weighted edges and nodes to G;
        Add the created nodes to ζ_v;
    else
        Continue;
    end
end
return G;

5.2 Mathematical Model

As already briefly mentioned, the MIP formulation of Brouer et al. (2013) is used. First, their formulation is discussed, which uses the three recovery actions: speed up a vessel, omitting a port call and swapping port calls. After this formulation is presented, the model is slightly altered and extended to incorporate the cut-and-run policy as well. For the first formulation the following sets, parameters and decision variables are defined:

Sets:

V	Vessels
P	Ports
T	Discrete timeslots
C	Container groups
H_v	Ordered set of port calls of vessel $v \in V$
W_v	Origin port of vessel $v \in V$
B_c	Origin port for container group $c \in C$
T_c	Destination port for container group $c \in C$
L_h	Feasible sailings to cover a port call $h \in H_v$
N	All nodes, where n^- and n^+ respectively denote outgoing and ingoing edges of node n
N_v	All nodes of vessel $v \in V$, where n_s^v and n_t^v respectively are the source and sink node
E_v	Possible sailings[4] of vessel $v \in V$ among the nodes of N_v
I_c	Ordered set $\left(I_c^1, \ldots, I_c^m\right)$ of intermediate transshipment points for container group $c \in C$. $I_c^i = \left(h_v^i, h_w^i\right) \in (H_v, H_w)$ is a pair of calls for different vessels $(v, w \in V \vert v \neq w)$.
M_c^e	All non-connecting edges $e \in L_h$ that result in misconnection of container group $c \in C$

Parameters:

$c_e^v \in \mathbb{R}_+$	Cost of using edge $e \in E_v$ of vessel $v \in V$
$c_c^d \in \mathbb{R}_+$	Cost of a delay to container group $c \in C$ exceeding a day of planned arrival
$c_c^m \in \mathbb{R}_+$	Cost of one or several misconnections to container group $c \in C$
$O_e^c \in \mathbb{B}$	Binary variable indicating if container group $c \in C$ is delayed, when arriving by edge $e \in L_{T_c}$
$Q_e^c \in \mathbb{B}$	Binary variable indicating if container group $c \in C$, when arriving by edge $e \in L_{B_c}$, as the ship is ready to sail before the container group is present
$S_v^n \in \{-1, 0, 1\}$	Used for flow conservation constraints. Set to -1 when $n = n_s^v$, 1 when $n = n_t^v$ and 0 otherwise

Decision variables:

$x_e \in \mathbb{B}$	1 if edge $e \in E_s$ is used, 0 otherwise
$z_h \in \mathbb{B}$	1 if vessel $v \in V$ omits port call $h \in H_v$, 0 otherwise
$y_c \in \mathbb{B}$	1 if container group $c \in C$ is misconnected, 0 otherwise
$o_c \in \mathbb{B}$	1 if container group $c \in C$ is delayed, 0 otherwise

[4] As discussed in Section 5.1

$$\min \quad \sum_{v \in V} \sum_{h \in H_v} \sum_{e \in L_h} c_e^v x_e + \sum_{c \in C} \left[c_c^m y_c + c_c^d o_c \right] \tag{1}$$

$$\text{s.t.} \quad \sum_{e \in L_h} x_e + z_h = 1 \qquad \forall v \in V,\ h \in H_v \setminus W_v \tag{2}$$

$$z_{W_v} = 0 \tag{3}$$

$$\sum_{e \in n^-} x_e - \sum_{e \in n^+} x_e = S_v^n \qquad \forall v \in V,\ n \in N_v \tag{4}$$

$$y_c \leq o_c \qquad \forall c \in C \tag{5}$$

$$\sum_{e \in L_{T_c}} O_e^c x_e \leq o_c \qquad \forall c \in C \tag{6}$$

$$z_h \leq y_c \qquad \forall c \in C,\ h \in B_c \cup I_c \cup T_c \tag{7}$$

$$x_e + \sum_{\lambda \in M_c^e} x_\lambda \leq 1 + y_c \qquad \forall c \in C,\ e \in \{L_h \mid h \in I_c\} \tag{8}$$

$$\sum_{e \in L_{B_c}} Q_e^c x_e \leq y_c \qquad \forall c \in C \tag{9}$$

$$x_e \in \mathbb{B} \qquad \forall e \in E_s \tag{10}$$

$$z_h \in \mathbb{R}_+ \qquad \forall v \in V,\ h \in H_v \tag{11}$$

$$y_c, o_c \in \mathbb{R}_+ \qquad \forall c \in C \tag{12}$$

The objective (1) minimizes the total costs of operating all vessels and the misconnection and/or delay costs of container groups. Set-partitioning constraints (2) guarantee that the selected ports on the route of a vessel are either visited or omitted. As the first port has by construction no incoming edge, Constraint (3) ensures that this port is not skipped. These constraints in combination with the binary domain of x_e and the positive real domain of z_h, ensure that z_h is binary as well. Constraints (4) ensure the flow-conservation of the vessels for each node. The combination of the Constraints (2), (3) and (4) makes sure that feasible routes are created for each vessel.

As a misconnection automatically means that the container group is delayed, Constraints (5) ensure this relation. Constraints (6) ensure that when a container group c is delayed when arriving too late in his final destination port, o_c is set to 1. As the container can only be shipped to its final destination at a single edge (because of Constraints (2)), the left-hand side will be set to 1 at most. These constraints in combination with the objective and the positive real domain of o_c, ensure that o_c is binary. Constraints (7) make certain that when a port call is omitted on the route of the vessel in which the container group should have loaded, unloaded or transshipped, the container group is misconnected as well. Because of the minimization, the positive real domain of y_c and the forced binary domain of z_h, y_c has a binary domain as well. Constraints (8) are coherence constraints, which detect if a container group is misconnected or not. Constraints (9) guarantee that a misconnection occurs when the vessel leaves the port before the container is even in the port and loaded onto the vessel. It is assumed that a container group arrives in its origin port 24 hours before it is supposed to be loaded according to the proforma schedule.

Note that in comparison to the original formulation of Brouer et al. (2013), Constraints (2) are slightly altered and Constraint (3) is added. It is presumably assumed by the authors that there is an incoming edge of W_v known to enter this first port. Furthermore, the ports in the set for which Constraints (8) hold is altered to I_c instead of $B_c \cup I_c \cup T_c$, as was originally stated. It is outside the scope of this research whether the further transport (in case of T_c) of the container group would be misconnected or not. Especially, when the data is only generated and it is not known when the container group will be collected by another means of transport at its final destination port. In the case of B_c the Constraints (9) are used.

This formulation of the VSRP is in general proven to be $\mathcal{NP}$-complete by Dirksen (2011). When it is assumed that only port omissions are allowed as a recovery action the 0–1 Knapsack Problem (KP) can be reduced to the VSRP. As this KP is a weakly $\mathcal{NP}$-complete problem, the VSRP is in this case weakly $\mathcal{NP}$-complete as well. When it is assumed that only port swaps are allowed as a recovery action the Travelling Salesman Problem (TSP) can be reduced to the VSRP. The TSP is proven to be strongly $\mathcal{NP}$-complete and thus this instance of the VSRP is strongly $\mathcal{NP}$-complete as well.

5.3 Extended Mathematical Model

In order to include the recovery action cut-and-run as well, the existing formulation has to be extended. Define $K = \max_c \{|B_c| + |I_c| + |T_c|\}$ as the maximum number a container group could be split into, which by construction is 3. The following additional sets and decision variables are defined as follows:

Set:

D_c^e	Cut-and-run edges, which split container group c $D_c^e \subset \{L_h \mid h \in B_c \cup I_c \cup T_c\}$
J	Container splits, $j \in \{1, 2, \ldots, K\}$

Decision variables:

f_c	$\in \{0, 1, \ldots, K\}$	Number indicates how many times container group $c \in C$ is split
q_c	$\in \{0, 1, \ldots, 2K+1\}$	Number gives an indication how many times container group $c \in C$ is split and if it supposed to misconnect as a whole
$r_c^{(j)}$	$\in \mathbb{B}$	1 if container group $c \in C$ is split j times, 0 otherwise
$s_c^{(j)}$	$\in \mathbb{B}$	1 if container group $c \in C$ is split j times and has a final delay, 0 otherwise

The orginal objective function (1) is slightly altered to account for partial misconnections:

$$\begin{aligned} \min \quad & \sum_{v \in V} \sum_{h \in H_v} \sum_{e \in L_h} c_e^v x_e \\ & + \sum_{c \in C} \left[c_c^d \left(o_c - \sum_{j=1}^{K} (1 - \varepsilon^j) s_c^{(j)} \right) + c_c^m y_c + \left(c_c^m + c_c^d \right) \sum_{j=1}^{K} (1 - \varepsilon^j) r_c^{(j)} \right] \end{aligned} \tag{13}$$

The new objective (13) still minimizes the total operating costs and the costs of the misconnections and delays, but now it also minimizes the cost of split container groups. In case a total misconnection happens and thus $y_c = 1$, the minimization ensures that $r_c^j = 0$, $\forall j \in J$, therefore the misconnection cost is still only c_c^m. In case there is a partial misconnection and

depending on how many partial misconnections j there are, the cost of a misconnection and a delay is multiplied by the proportional size of the partial missed container group, which is $(1-\varepsilon^j)$. When a partial misconnection happens, the minimization ensures that $y_c = 0$. In case there is a delay of the whole container group and thus $o_c = 1$ and depending on how many partial misconnections j there have been, a certain proportion $1-\varepsilon^j$ of the size of the container group should be subtracted from the costs. If this does not happen, the delay cost of the partial misconnected containers would be counted twice.

Then the following additional constraints are added to the previous mathematical model:

$$\sum_{e \in D_e^c} x_e \leq f_c \qquad \forall c \in C \qquad (14)$$

$$f_c + (K+1)y_c \leq q_c \qquad \forall c \in C \qquad (15)$$

$$q_c - (K-j) \leq \sum_{i=K-j+1}^{K} (i-K+j) r_c^{(i)} + M_j y_c \qquad \forall c \in C,\ j \in J \qquad (16)$$

$$s_c^{(j)} \leq o_c \qquad \forall c \in C,\ j \in J \qquad (17)$$

$$s_c^{(j)} \leq r_c^{(j)} \qquad \forall c \in C,\ j \in J \qquad (18)$$

$$f_c, q_c \in \mathbb{R} \qquad \forall c \in C \qquad (19)$$

$$r_c^{(j)}, s_c^{(j)} \in \mathbb{B} \qquad \forall c \in C, j \in J \qquad (20)$$

$$y_c \in \mathbb{B} \qquad \forall c \in C \qquad (21)$$

Constraints (14) count the number of partial misconnections there have been of each container group. As there is at most one incoming edge in each port (Constraints (2)) and since x_e has a binary domain, it is ensured that $f_c \in \{0, 1, \ldots, K\}$. Constraints (15) keep track what kind of misconnection has occurred, e.g. a total misconnection or a partial misconnection of a container group c. When $y_c = 1$ there has been a complete misconnection and no container in this group can be delivered on time, thus $q_c \geq K+1$. When $y_c = 0$, the number of partial misconnections $q_c = f_c$. As $f_c \in \{0, 1, \ldots, K\}$ and y_c has a binary domain, it follows that $q_c \in \{0, 1, \ldots, 2K+1\}$.

Constraints (16) keep track of the relation between partial misconnections and a total misconnection. In case a potential complete misconnection occurs, it does not matter whether possible partial misconnections have occured as the complete container group is misconnected anyhow. The constraints ensure for the case $q_c = Z > K-j+1$ that at least $y_c = 1$ and therefore $M_j \geq Z-(K-j)$ or $r_c^{(Z)} = 1$ and therefore the constant term multiplied with $r_c^{(Z)}$ is $Z-K+j = Z-(K-j)$. Because of the minimization and the way weights are given to each variable in the summation, $r_c^{(j)}, \forall j \neq Z$ is then set to 0. In the case that $q_c = K-j+1$, it follows that $y_c = 0$ and as it cheaper option not to split containers unnecessarily, only $r_c^{(K-j+1)}$ is set to 1. In the case that $q_c = Z < K-j+1$, it follows that $y_c = 0$ and the left-hand side is $Z-(K-j) \leq 0$. Because of the minimization, it follows that $r_c^{(j)} = 0, \forall j \in \{K-j+1, K-j+2, \ldots, K\}$. The big-M notation is used to ensure that no $r_c^{(j)}$ is set to 1 in case a total misconnection occurs. As q_c can be at most $2K+1$, the left-hand side can be at most $2K+1-(K-j)$. As the value of the M needs in general to be as small as possible, the optimal choice for M_j is :

$$M_j = 2K+1-(K-j) = K+j+1.$$

In the case that both $o_c = 1$ and $r_c^{(j)} = 1$, Constraints (17) and (18) and the minimization ensure that $s_c^{(j)} = 1$.

Constraints (19) give a real domain to both f_c and q_c, however as already stated the model forces the desired domain on both variables. Constraints (20) ensure the binary domain of $r_c^{(j)}$ and $s_c^{(j)}$. As the minimization does no longer enforce y_c to have a binary domain, Constraints (12) are slightly altered such that y_c has a binary domain, which is stated by Constraints (21).

In Appendix A a detailed example of this formulation is given for the case where $K = 3$ and $\varepsilon = 0.5$.

6 Results

As the aim is to obtain real-time results, the cut-off value for which it still reasonable to obtain results is set to 5 minutes. First, we examine how the original formulation and the extended formulation react to changes in δ. Next, we inspect how both formulations react to changes in the number of vessels V and number of port calls κ. Finally, it is investigated in which cases the extended formulation performs significantly better in terms of cost compared to the original formulation. The program has been run on a Toshiba Satellite with Intel Core i5-6200U with 2.30 GHz processor and 8 GB of memory running Windows 10 using IBM ILOG CPLEX 12.7.1 as MIP solver. All computational results are average values based on five runs, where $K = 3$ and $\varepsilon = 0.5$. The exact formulation needed for the extended model in case these specific values of the two parameters are used, is presented in Appendix A.

6.1 Varying Duration of Shift

In order to investigate how the computation time reacts to changes in δ, we set up two instances. The first one has the following specifications: $\eta = 10$, $V = 5$ and $\kappa = 5$. The second instance has these specifications: $\eta = 10$, $V = 10$ and $\kappa = 6$. As one particular example of an instance could be solved faster or slower because of its structure, we generate 10 cases of each instance and report the average value to give a more realistic view. As it can happen that the extended formulation is able to give a solution, whereas the original formulation is infeasible, only cases are used which give feasible solutions in both formulations.

In Table 1 and Table 2 the average number of edges and nodes generated, the total costs and the computational times of the original and the extended formulation in case of the first instance are shown. Table 3 and Table 4 present the same results for the original and extended formulation in case of the second instance.

In general when observing all four tables, it seems that when the duration of a shift δ is decreased, the formulations become slower. When looking at the edges and nodes generated, it is clear why the formulations become so much slower. Especially when δ in Table 1 changes from 0.5 to 0.1677, the number of nodes increases from 3,930 to 43,248, which could partially explain why the total computation time increases from 0.94 to 8.65 seconds. A similar effect for the same decrease in δ is seen for the extension in Table 2, where the number of nodes increases from 6,453 to 85,183 and the total computation time from 4.65 to 101.66 seconds.

Moreover, the extended formulation is in both instances significantly slower than the original. To a certain extent this could be explained by the fact that the average number of generated edges and nodes in the extended formulation is about twice as much as in the original formulation for each value of δ. As its underlying graph is much larger than in the original formulation, it is prone to give memory exceptions more quickly when generating the graph.

Comparing both instances, it also seems that the second instance is in general much slower than the first instance. Again it seems reasonable that the average number of generated edges and nodes are the main cause of this.

Furthermore, it seems that decreasing δ leads to lesser costs. However, this only seems profitable till a certain point, considering the trade-off between the computation time and costs. Especially when looking at the change of $\delta = 1$ to $\delta = 0.5$ in Table 1 and Table

2, as the decrease in cost seems at this point to converge a bit, whereas the computation time increases rapidly. For $\delta = 0.1666$ the cost even went up a bit compared to $\delta = 0.5$, but this could be explained as certain cheaper sailings were possibly not available in this graph. This seems reasonable as the other values of δ are not an exact multiple of 0.1666.

When comparing the decrease in cost for each value of δ for both the normal and extended formulation in both instances, it seems that the extended formulation only provides a relative large decrease in costs in the first instance. In the first instance a general decrease of 7% in the costs is established, whereas the second instance only has a decrease of less than 2%. To see whether the extended formulation provides a large decrease in cost in general, the efficiency of the extended formulation is examined in more detail in Section 6.3.

Table 1: Influence on number of edges and nodes generated, the cost and the computation time in seconds of the original formulation when varying δ in the first instance. All given values are averages of 10 cases. Time (max) denotes the maximum total computation time of the 10 cases. The computation time to load the data is not given as it could easily be determined by the other computation times.

	Original						
δ (h)	E	N	Cost (10^6 euro)	Building graph	Running Cplex	Time (total)	Time (max)
30	77	82	8.353	0.01	0.03	0.04	0.17
10	278	207	7.903	0.01	0.03	0.05	0.20
5	5,846	405	7.796	0.01	0.04	0.05	0.22
2	4,438	1,003	7.743	0.02	0.10	0.12	0.32
1	16,567	1,988	7.732	0.03	0.44	0.47	0.77
0.5	32,937	3,930	7.729	0.04	0.89	0.94	2.89
0.1666	97,789	43,248	7.735	0.35	8.29	8.65	33.68
0.0833	196,854	76,271	7.733	1.15	42.88	44.06	202.54

Table 2: Influence on number of edges and nodes generated, the cost and the computation time in seconds of the extended formulation when varying δ in the first instance. All given values are averages of 10 cases. Time (max) denotes the maximum total computation time of the 10 cases. The computation time to load the data is not given as it could easily be determined by the other computation times. In case a value could not be generated because of a memory exception, it is denoted by: –.

	Extended						
δ (h)	E	N	Cost (10^6 euro)	Building graph	Running Cplex	Time (total)	Time (max)
30	218	198	7,760	0.01	0.05	0.06	0.22
10	846	535	7.320	0.01	0.09	0.10	0.23
5	2,668	1,072	7.225	0.01	0.15	0.16	0.38
2	9,454	1,612	7.152	0.02	0.40	0.43	0.76
1	35,420	3,256	7.151	0.05	1.83	1.89	8.13
0.5	70,354	6,453	7.149	0.12	4.53	4.65	15.14
0.1666	227,061	85,183	7.165	1.74	99.92	101.66	568.62
0.0833	–	–	–	–	–	–	–

Table 3: Influence on number of edges and nodes generated, the cost and the computation time in seconds of the original formulation when varying δ in the second instance. All given values are averages of 10 cases. Time (max) denotes the maximum total computation time of the 10 cases. The computation time to load the data is not given as it could easily be determined by the other computation times. In case a value could not be generated because of a memory exception, it is denoted by: –.

	Original						
δ (h)	E	N	Cost (10^7 euro)	Building graph	Running Cplex	Time (total)	Time (max)
30	336	344	1.636	0.01	0.05	0.07	0.24
10	1,395	998	1.563	0.01	0.07	0.08	0.21
5	4,702	2,082	1.555	0.01	0.16	0.17	0.32
2	25,889	5,283	1.548	0.04	0.73	0.77	1.67
1	97,208	10,470	1.538	0.12	4.07	4.30	18.64
0.5	192,650	20,762	1.516	0.30	12.52	12.83	64.68
0.1666	–	–	–	–	–	–	–

Table 4: Influence on number of edges and nodes generated, the cost and the computation time in seconds of the extended formulation when varying δ in the second instance. All given values are averages of 10 cases. Time (max) denotes the maximum total computation time of the 10 cases. The computation time to load the data is not given as it could easily be determined by the other computation times. In case a value could not be generated because of a memory exception, it is denoted by: –.

	Extended						
δ (h)	E	N	Cost (10^7 euro)	Building graph	Running Cplex	Time (total)	Time (max)
30	1,155	986	1.602	0.01	0.17	0.18	0.42
10	5,416	3,316	1.528	0.02	0.46	0.48	0.78
5	18,491	6,600	1.519	0.03	1.23	1.26	2.49
2	56,813	8,815	1.517	0.08	5.39	5.48	20.21
1	210,500	17,188	1.512	0.24	59.71	59.95	327.92
0.5	–	–	–	–	–	–	–
0.1666	–	–	–	–	–	–	–

6.2 Varying Number of Vessels and Port Calls

To investigate what happens to the computation time when varying the number of vessels and the number of port calls, several instances are created in which $\eta = 10$ and $\delta = 1.0$. Again we generate 10 cases of each instance and report the average value to give a more realistic view. The used cases give feasible solutions for both formulations. Table 5 shows how the computation time reacts when changing the number of vessels or the number of port calls respectively. The top value in a cell represents the computation time in the original formulation, whereas the bottom value represents the computation time in the extended formulation.

In general it seems that the computation time strongly reacts to changes in the number of port calls in both formulations. For example, when κ is increased from 6 to 7 for $V = 5$ the computation time of the original formulation increases from 1.14 to 14.96 seconds. Moreover, increasing κ increases the chance to obtain memory exceptions in Cplex, as the number of nodes and edges grow rapidly. This growth can be explained since the recovery horizon becomes larger and the number of all possible recoveries in the graph grow exponentially. Both formulations are, however, more robust when changing the number of vessels, as the underlying graph does not increase rapidly. For example, when increasing the number of vessels from 20 to 40 for $\kappa = 4$ the computation time in the original formulation increases from 0.55 to 0.93 seconds. Overall, the behaviour of the total computation time in case the number of port calls or the number of vessels increases, seems in line with the findings of Brouer et al. (2013).

Table 5: Total computation time in seconds when varying the number of vessels and number of port calls. All given computation times are averages of 10 cases. The top value in a cell represents the original formulation (Org) and the bottom value the extended formulation (Ext). In case a value could not be generated because of a memory exception, it is denoted by: –.

		V								
κ		5	10	15	20	30	40	60	80	100
4	Org	0.07	0.17	0.50	0.55	0.67	0.93	1.04	3.20	3.44
	Ext	0.28	0.61	1.94	2.13	2.65	9.79	18.27	56.06	259.41
5	Org	0.21	0.97	1.25	1.88	3.07	7.16	43.53	–	–
	Ext	0.80	16.15	11.97	25.21	97.79	272.95	–	–	–
6	Org	1.14	5.39	18.31	22.61	183.26	194.25	–	–	–
	Ext	9.25	156.85	300.85	–	–	–	–	–	–
7	Org	14.96	49.98	125.48	–	–	–	–	–	–
	Ext	17.04	–	–	–	–	–	–	–	–
8	Org	44.91	–	–	–	–	–	–	–	–
	Ext	–	–	–	–	–	–	–	–	–
9	Org	63.73	–	–	–	–	–	–	–	–
	Ext	–	–	–	–	–	–	–	–	–
10	Org	–	–	–	–	–	–	–	–	–
	Ext	–	–	–	–	–	–	–	–	–

When comparing both formulations, it is clear that the extended formulation is slower than the original. This is as expected, as the extended formulation contains the nodes and edges of the original formulation. Especially in instances where the number of vessels is relatively large for the number of port calls, does the computation time between both formulations diverge a lot. For example, the computation time in the original formulation is 3.44 seconds when $\kappa = 4$ and $V = 100$, whereas the extended formulation is solved in 259.41 seconds. This is almost 75 times slower.

6.3 Performance Extended Formulation

From Table 5 it is clear that the extended formulation is much slower than the original formulation for each instance. In some cases the extended formulation is up to 75 times slower than the original formulation. It is interesting to know whether the extended formulation performs much better in terms of costs in these cases, such that it is profitable to wait a bit longer to get a much better solution. Table 6 shows how the total costs react when varying the number of vessels and the number of port calls for the same instances as in Table 5. The top value in a cell represents again the original formulation, whereas the bottom value represents the extended formulation.

It appears that when the number of vessels is increased, the extended formulation gives a relative larger reduction of costs compared to the original formulation. The same statement is not necessarily true when increasing κ. There certainly are reductions in costs, but these are relatively decreasing. However, it is not clear whether this is a systematic effect, as there are less results when increasing κ. On average there is a cost reduction of 9.4% when using the extended formulation.

Table 6: Total costs (10^6 euro) when varying the number of vessels and number of port calls. All given costs are averages of 10 cases. The top value in a cell represents the original formulation (Org) and the bottom value the extended formulation (Ext). In case a value could not be generated because of a memory exception, it is denoted by: –.

		V								
κ		5	10	15	20	30	40	60	80	100
4	Org	7.25	11.88	18.56	39.99	69.79	74.00	155.43	225.90	330.94
	Ext	6.81	11.10	18.19	38.61	58.13	66.48	127.90	180.66	268.59
5	Org	9.66	22.14	30.70	40.74	69.18	99.75	158.64	–	–
	Ext	8.51	20.42	28.50	37.10	61.25	88.97	–	–	–
6	Org	7.82	13.82	29.45	37.62	67.50	83.74	–	–	–
	Ext	7.31	13.27	28.49	–	–	–	–	–	–
7	Org	9.85	20.04	32.14	–	–	–	–	–	–
	Ext	9.45	–	–	–	–	–	–	–	–
8	Org	10.66	–	–	–	–	–	–	–	–
	Ext	–	–	–	–	–	–	–	–	–
9	Org	13.60	–	–	–	–	–	–	–	–
	Ext	–	–	–	–	–	–	–	–	–
10	Org	–	–	–	–	–	–	–	–	–
	Ext	–	–	–	–	–	–	–	–	–

6.3.1 Example

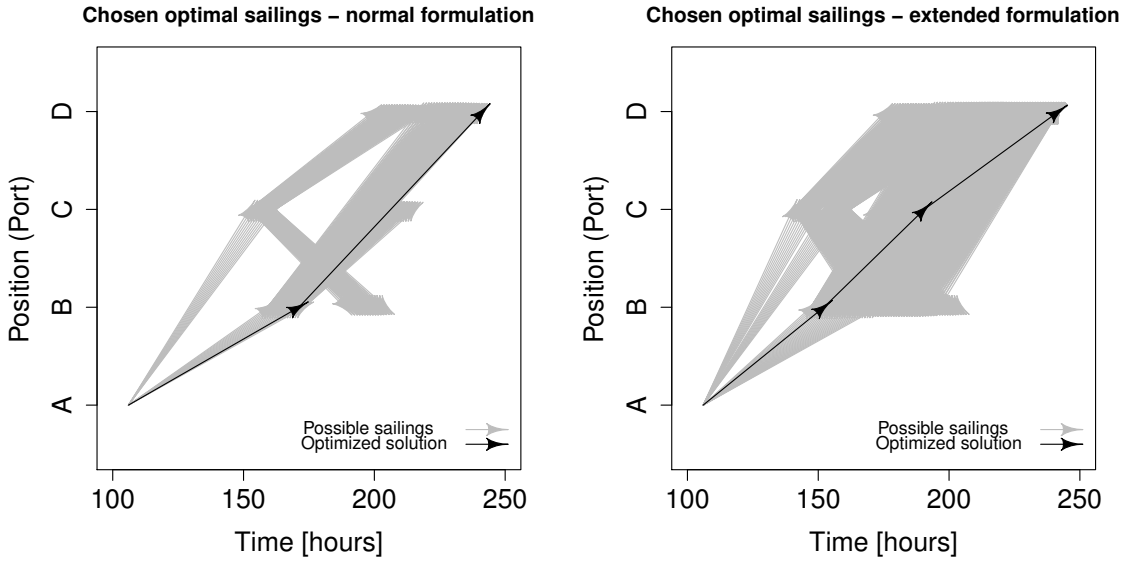
In order to see in what kind of situations the extended formulation performs much better than the original, we constructed a specific example, where $\kappa = 4$, $V = 1$, $\eta = 10$, $K = 3$ and $\varepsilon = 0.5$. This single vessel picks up three container groups at its origin port A and it delivers one container group at each port call. Its proforma schedule is in the order A, B, C and D. At port B it has to deliver 1,000 containers, at port C 200 containers and at D 4,000 containers. Originally the vessel was supposed to arrive at A at 96 hours, however there has been a delay of 10 hours. In Table 7 and Table 8 it is shown how the example performs in both formulations. Figure 3 shows a time-space representation of the problem.

Table 7: Overview of some aspects of the example, where E are the number of edges and N the number of nodes. The costs are reported in euros and the computation time in seconds.

	E	N	Constraints	Total cost	Sailing cost	Time (total)
Original	580	113	82	1,478,229	139,432	0.05
Extended	3,170	301	186	1,139,432	78,229	0.23

Table 8: Overview of some aspects of the example. Container impact is reported in units. Split indicates how many times there has been a partial misconnection.

	Delay		Misconnection	
	Total	Split	Total	Split
Original	1,200	-	200	-
Extended	500	1	500	1



(a) All possible sailings when using the original formulation.

(b) All possible sailings when using the extended formulation.

Figure 3: Time-space network representation of the example.

From Table 7 it is clear that the extended formulation performs much better than the original formulation, as the cost reduction is almost 23%. Table 8 and Figure 3 give more insight in the final solution. In the extended formulation the container group with 1,000 containers has been split into two groups at port B. Because of the time savings it potentially gained at this port, it could deliver the other two container groups on time. The original formulation was not able to go to port C at all, as the cost of a delay of the container group with 4,000 units at port D is too high. However, in both formulations the vessel is not able to deliver the first container group in time. Therefore, in this situation it is the best choice to have a partial misconnection there, as delay costs are already inflicted. With the time it gained, the vessel did not need to sail on its maximum speed as in the original formulation, what caused the sailing costs to decrease as well.

The figure also gives an idea how the number of nodes and edges behave when implementing the extended formulation. In comparison to Figure 3a is the number of possible sailings much higher in Figure 3b.

The reason why there is such a large cost reduction in this particular example, is because it would have experienced a delay of its first delivery anyway and because it needs to deliver a large container group somewhere else on its itinerary. In such cases the use of the extended formulation seems especially profitable.

7 Conclusion

In this thesis it was investigated whether it is possible to have a model which is able to handle liner shipping disruptions in real-time decision making. To test this, a formulation of Brouer et al. (2013) was used, such that disruptions could be handled by speeding up vessels, omitting port calls or swapping port calls. The original model was extended in this thesis to also handle disruptions by means of the cut-and-run method, such that a vessel does not finish its entire port call before embarking again. To answer how both formulations perform in several instances, it was researched how they react when varying the duration of the shift, the number of vessels and the number of port calls.

It seems that decreasing the duration of the shift, such that there are more timeslots for the vessel to arrive at its next destination, is profitable till a certain point. Considering the trade-off between time and cost decrease, we advise to use $\delta = 1$ or $\delta = 0.5$. Furthermore, when the number of port calls in which a vessel is allowed to recover is too large, both formulations have memory issues. For a large number of vessels (20) the extended formulation is already not able to give results when the number of port calls is 6, whereas the original formulation still gives results for cases which include 40 vessels. In case the number of port calls is 8 or 9, only the original formulation is able to perform, however for a relative small number of vessels (5). The model is more robust when increasing the number of vessels. In case the number of port calls is only 4, both formulations are able to give results in an instance with 100 vessels in still a reasonable amount of time (< 5 minutes).

In general the extended formulation is slower than the original formulation. In some cases the extended formulation is up to 75 times slower than the original. However, on average it can give cost reductions of around 9% compared to the original formulation. Especially in instances where the number of vessels (> 40) is large and the number of port calls not too large (< 6), does the extended formulation on average give a relative large cost reduction. As in those instances the extended formulation is still solvable in reasonable time, we advise to use the extended formulation in case the number of port calls is not too large (< 6), whereas we recommend to use the original formulation when the number of port calls is larger. Next, we advise to use the extended formulation in case there is a real severe delay. In this case the original formulation is more likely to not be able to give a feasible solution because of the way the graph is constructed.

However, there are some remarks regarding both models. The considered sailing costs possibly do not capture all costs. It is generally known that the engine of a vessel performs best when it sails on its design speed. In case it sails slower than its design speed, it gives strain on the machinery. This could give rise to bigger engine claims and maintenance costs (Meyer et al., 2012). Another remark is that the current formulation as used in this thesis only allows port swaps with a direct successor. This may not be the most effective option, as it is generated data and the swapped ports are not within a designated geographical area. However, in cases where real data is used, this could be a better option, as itineraries are created such that a successor of a certain port is in its designated geographical area.

For future research we would recommend to try to reduce the size of the underlying time-space network without changing its optimal solution. Reducing its size would certainly decrease the computation time significantly.

References

Alderton, P. (2004). *Reeds sea transport: Operation and economics*. A&C Black.

Bratu, S. (2003). Airline passenger on-time schedule reliability: Analysis, algorithms and optimization decision models. *Massachusetts Institute of Technology, PhD Thesis*.

Bratu, S. and Barnhart, C. (2006). Flight operations recovery: New approaches considering passenger recovery. *Journal of Scheduling*, 9(3):279–298.

Brouer, B., Dirksen, J., Pisinger, D., Plum, C., and Vaaben, B. (2013). The vessel schedule recovery problem (vsrp)–a mip model for handling disruptions in liner shipping. *European Journal of Operational Research*, 224(2):362–374.

Christiansen, M., Fagerholt, K., Nygreen, B., and Ronen, D. (2013). Ship routing and scheduling in the new millennium. *European Journal of Operational Research*, 228(3):467–483.

Dienst, D., Røpke, S., and Vaaben, B. (2012). Realistic models and computational results for disruption management in the airline industry. *Report from DTU Management Engineering*.

Dirksen, J. (2011). Disruption management in liner shipping: Introducing the vessel schedule recovery problem. Master's thesis, Technical University of Denmark, DTU, DK-2800 Kgs. Lyngby, Denmark.

Jarrah, A., Yu, G., Krishnamurthy, N., and Rakshit, A. (1993). A decision support framework for airline flight cancellations and delays. *Transportation Science*, 27(3):266–280.

Li, C., Qi, X., and Lee, C. (2015). Disruption recovery for a vessel in liner shipping. *Transportation Science*, 49(4):900–921.

MAN (2009). Propulsion trends in container vessels, two-stroke engines. `http://marine.man.eu/docs/librariesprovider6/technical-papers/propulsion-trends-in-container-vessels.pdf?sfvrsn=20/`. [Online; accessed 27-May-2017].

Marla, L., Vaaben, B., and Barnhart, C. (2016). Integrated disruption management and flight planning to trade off delays and fuel burn. *Transportation Science*.

Meng, Q., Wang, S., Andersson, H., and Thun, K. (2013). Containership routing and scheduling in liner shipping: overview and future research directions. *Transportation Science*, 48(2):265–280.

Meyer, J., Stahlbock, R., and Voß, S. (2012). Slow steaming in container shipping. In *System Science (HICSS), 2012 45th Hawaii International Conference on*, pages 1306–1314. IEEE.

Notteboom, T. E. (2006). The time factor in liner shipping services. *Maritime Economics & Logistics*, 8(1):19–39.

Stopford, M. (2009). *Maritime economics 3e*. Routledge.

Teodorović, D. and Guberinić, S. (1984). Optimal dispatching strategy on an airline network after a schedule perturbation. *European Journal of Operational Research*, 15(2):178–182.

Thengvall, B., Yu, G., and Bard, J. (2001). Multiple fleet aircraft schedule recovery following hub closures. *Transportation Research Part A: Policy and Practice*, 35(4):289–308.

UNCTAD (2016). Review of maritime transport 2016. `http://unctad.org/en/PublicationsLibrary/rmt2016_en.pdf`. [Online; accessed 27-May-2017].

Yan, S. and Tu, Y. (1997). Multifleet routing and multistop flight scheduling for schedule perturbation. *European Journal of Operational Research*, 103(1):155–169.

Yan, S. and Yang, D. (1996). A decision support framework for handling schedule perturbation. *Transportation Research Part B: Methodological*, 30(6):405–419.

A Appendix

This appendix describes an example of the extended formulation in which the cut-and-run policy is considered a recovery option as well. In this example, we set $K = 3$ and $\varepsilon = 0.5$. The set J is not explicitly mentioned in this case, as all decision variables and constraints are written down. The following formulation is used to generate the results as discussed in Section 6.

Set:

D_c^e	Cut-and-run edges, which split container group c. $D_c^e \subset \{L_h \mid h \in B_c \cup I_c \cup T_c\}$

Decision variables:

f_c	$\in \{0, 1, 2, 3\}$	Number indicates how many times container group $c \in C$ is split
q_c	$\in \{0, 1, \ldots, 7\}$	Number gives an indication how many times container group $c \in C$ is split and if it supposed to misconnect as a whole
$r_c^{(1)}$	$\in \mathbb{B}$	1 if container group $c \in C$ is split once, 0 otherwise
$r_c^{(2)}$	$\in \mathbb{B}$	1 if container group $c \in C$ is split twice, 0 otherwise
$r_c^{(3)}$	$\in \mathbb{B}$	1 if container group $c \in C$ is split thrice, 0 otherwise
$s_c^{(1)}$	$\in \mathbb{B}$	1 if container group $c \in C$ is split once and has a final delay, 0 otherwise
$s_c^{(2)}$	$\in \mathbb{B}$	1 if container group $c \in C$ is split twice and has a final delay, 0 otherwise
$s_c^{(3)}$	$\in \mathbb{B}$	1 if container group $c \in C$ is split thrice and has a final delay, 0 otherwise

The original objective (1) is reformulated to account for partial misconnections:

$$\min \quad \sum_{v \in V} \sum_{h \in H_v} \sum_{e \in L_h} c_e^v x_e \tag{22}$$
$$+ \sum_{c \in C} \left[c_c^d \left(o_c - \frac{1}{2} s_c^{(1)} - \frac{3}{4} s_c^{(2)} - \frac{7}{8} s_c^{(3)} \right) + c_c^m y_c + \left(c_c^m + c_c^d \right) \left(\frac{1}{2} r_c^{(1)} + \frac{3}{4} r_c^{(2)} + \frac{7}{8} r_c^{(3)} \right) \right]$$

The new objective (22) minimizes once more the total operating costs and the costs of the misconnections and delays, however now it also minimizes the cost of split container groups. In case a total misconnection happens and thus $y_c = 1$, the minimzation ensures that $r_c^{(1)}$, $r_c^{(2)}$ and $r_c^{(3)}$ are 0. Therefore, the misconnection cost is still only c_c^m. In case there is a partial misconnection and depending on how many partial misconnections there are, the cost of a misconnection and a delay is multiplied by the size of the partial missed container group. In case there is a delay of (what is left of) a container group is, and thus $o_c = 1$, and depending on how many times the container group has been partially misconnected, a certain proportion should be subtracted from o_c otherwise the delay cost of the partial misconnected containers would be counted twice.

The following additional constraints are added to the original mathematical model:

$$\sum_{e \in D_e^c} x_e \leq f_c \qquad \forall c \in C \qquad (23)$$

$$f_c + 4y_c \leq q_c \qquad \forall c \in C \qquad (24)$$

$$q_c - 2 \leq r_c^{(3)} + 5y_c \qquad \forall c \in C \qquad (25)$$

$$q_c - 1 \leq r_c^{(2)} + 2r_c^{(3)} + 6y_c \qquad \forall c \in C \qquad (26)$$

$$q_c \leq r_c^{(1)} + 2r_c^{(2)} + 3r_c^{(3)} + 7y_c \qquad \forall c \in C \qquad (27)$$

$$s_c^{(1)} \leq o_c \qquad \forall c \in C \qquad (28)$$

$$s_c^{(1)} \leq r_c^{(1)} \qquad \forall c \in C \qquad (29)$$

$$s_c^{(2)} \leq o_c \qquad \forall c \in C \qquad (30)$$

$$s_c^{(2)} \leq r_c^{(2)} \qquad \forall c \in C \qquad (31)$$

$$s_c^{(3)} \leq o_c \qquad \forall c \in C \qquad (32)$$

$$s_c^{(3)} \leq r_c^{(3)} \qquad \forall c \in C \qquad (33)$$

$$f_c, q_c \in \mathbb{R} \qquad \forall c \in C \qquad (34)$$

$$r_c^{(1)}, r_c^{(2)}, r_c^{(3)}, s_c^{(1)}, s_c^{(2)}, s_c^{(3)} \in \mathbb{B} \qquad \forall c \in C \qquad (35)$$

$$y_c \in \mathbb{B} \qquad \forall c \in C \qquad (36)$$

Constraints (23) count the number of misconnections there have been of each container group. As there is at most one incoming edge in each port (Constraints (2)) and since x_e has a binary domain, it is ensured that $f_c \in \{0, 1, 2, 3\}$. Constraints (24) keep track what kind of misconnection has occurred, e.g. a total misconnection or a partial misconnection of a container group c. When $f_c \geq 4$ there has been a complete misconnection and no container in this group can be delivered on time. When $f_c = 3$ the container group is three times partially misconnected, therefore only 12.5% of the cargo can be delivered without experiencing a misconnection. When $f_c = 2$ the container group has been split twice, thus only 25% of the group can be delivered without experiencing a misconnection. When $f_c = 1$ the container group is split once, such that half of the cargo does not experience a misconnection. In the case that $f_c = 0$, no misconnection has occurred at all. As $f_c \in \{0, 1, 2, 3\}$ and y_c has a binary domain, $q_c \in \{0, 1, \ldots, 7\}$.

Constraints (25) ensure that when there are three partial misconnections ($q_c = 3$), the variable $r_c^{(3)}$ is set to 1. In case $q_c \geq 4$, $r_c^{(3)}$ is set to 0 as $y_c = 1$ and since there is a minimization. The general idea of these constraints is also applied to the Constraints (26) and (27). Constraints (26) ensure that when there are two partial misconnections ($q_c = 2$), the variable $r_c^{(2)}$ is set to 1. The case of $q_c \geq 4$ is similar to the one above and thus $r_c^{(2)}$ is set to 0. In case $q_c = 3$, $r_c^{(2)}$ is set to 0 as well, as it is cheaper to only set $r_c^{(3)}$ to 1. Constraints (27) ensure that when there is one partial misconnection ($q_c = 1$), the variable $r_c^{(1)}$ is set to 1. The case of $q_c \geq 4$ and $q_c = 3$ are similar to the ones above, thus $r_c^{(1)}$ is set to 0. In case $q_c = 2$, $r_c^{(1)}$ is set to 0 as well, as it is cheaper to only set $r_c^{(2)}$ to 1.

Constraints (28) & (29), (30) & (31) and (32) & (33) apply all the same idea, therefore only the first pair is explained in more detail. Only in the case there is a supposed delay of the whole container group $o_c = 1$ and the container group is split once $r_c^{(1)} = 1$, $s_c^{(1)}$ is able to be 1. Because of the minimization $s_c^{(1)}$ is then set to 1. Constraints (34) give a real domain to f_c

and q_c, however as already discusses the model forces the desired domain on both variables. Constraints (35) force a binary domain on $r_c^{(1)}$, $r_c^{(2)}$, $r_c^{(3)}$, $s_c^{(1)}$, $s_c^{(2)}$ and $s_c^{(3)}$. As the structure of the minimization is changed, y_c is no longer forced to be binary, therefore Constraints (12) are slightly altered such that y_c has a binary domain, which is stated by Constraints (36).