Scheduling arrivals in an LNG Receiving Terminal

J.G.A. de Lange

Scheduling arrivals in an LNG Receiving Terminal

by

J.G.A. de Lange

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Wednesday March 8, 2017 at 09:00 AM.

Student number:	4018605	
Project duration:	September 1, 2015 – March 8, 2017	
Thesis committee:	Prof. dr. C. Witteveen,	TU Delft
	Dr. M. M. de Weerdt,	TU Delft, supervisor
	Dr. D. C. Gijswijt,	TU Delft
	M. J. Jansen Msc.,	Systems Navigator B.V.

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft

Abstract

The demand for Liquefied Natural Gas (LNG) has increased rapidly over the last 30 years. LNG is Natural Gas (NG) that is cooled down until it becomes liquid. LNG is easier to transport over long than NG distances and also in larger quantities. Before the LNG can be used it is made gaseous again, for example at an LNG receiving terminal.

In this thesis we create multiple Mixed Integer Linear Programs (MILPs) to solve the problem of creating an Annual Delivery Plan (ADP) for an LNG receiving terminal. An ADP describes the times vessels should arrive at the terminal and when this LNG will be gasified and send-out to the network. We proof this problem is NP-Complete.

First we create three MILPs that can solve for terminals with a single client. We have a discrete time formulation, a formulation that makes use of piecewise linear functions and an event based approach. Experimentation learns us the discrete time and event based formulations are solved the fastest.

We then expand the discrete client formulation to work for multiple clients and also adjust this formulation to work with a rolling horizon approach, which gives up some optimality in return for solving speed. From experimentation and evaluation using a simulation model we learn that the rolling horizon approach is comparable with an heuristic algorithm currently used in practice. While the heuristic is a little faster our solution shows potential to improve the found solutions.

Contents

List of Figures

List of Tables

1 Introduction

The term LNG is used for Liquefied Natural Gas. This is Natural Gas (NG) that is cooled down enough to be converted to a liquid state. This is done to make it easier to transport the NG, also in larger quantities.

The demand for LNG has increased rapidly over the last 30 years, for example in 2015 the global LNG imports increased by 2.5% [7]. The growth is mainly due to the "*flexibility of destination and volumes of gas supplies, supply to regions where pipeline supply is inefficient or impossible, environmental advantages over other fossil fuels, and price competitiveness and efficiency*" [18]. The import and export of Liquefied Natural Gas (LNG) is also projected to increase in the coming decade and even be tripled in the period between 2011 and 2030 [19].

The LNG supply chain is shown in Figure 1.1. Each step is explained below.

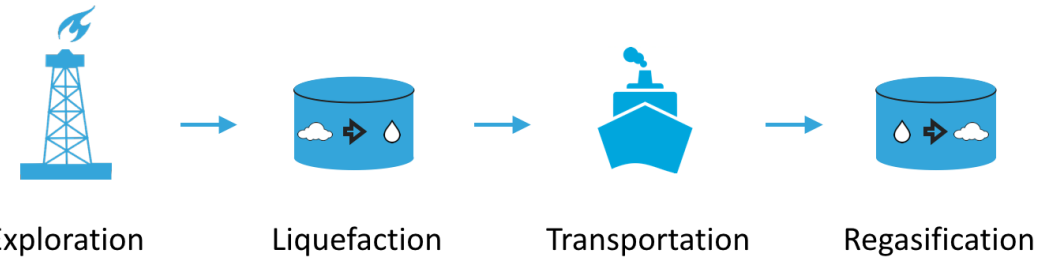


Figure 1.1: Schematic overview of the LNG supply chain

First when a source of NG is found a facility is built to extract the NG from that source.

Once the NG has been collected it can be made into LNG by cooling it to $-162°C$ in a liquefaction facility. At this temperature the gas will become liquid and uses 610 times less space than when in it is in a gaseous state [5]. By using less space it is easier to transfer it over long distances by truck, train or vessel and to store in tanks.

After the transportation, when the LNG arrives at an LNG receiving terminal, it will be stored in tanks in its liquid form. From there it will be reloaded on another truck, train or vessel, or regasified so that it can be delivered to the NG delivery network.

In this thesis several mathematical notations are presented that can be used for making a Annual Delivery Plan (ADP) for a LNG receiving terminal for vessels. Before explaining what an ADP is, a brief overview of how an LNG receiving terminal is operated will first be given.

The operation of a receiving terminal can be reduced to four basic steps: Unloading, storage, regasification and delivery. As schematically shown in Figure 1.2.

The unloading step is when a vessel arrives at the terminal. When it arrives it needs to be berthed at a specific berth where it is unloaded. This is done using a jetty and a network of pipelines from the berth to the tank.

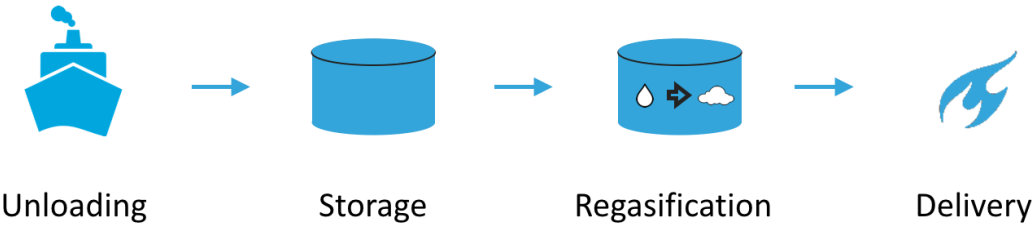


Figure 1.2: Schematic overview of operation in an LNG receiving terminal

Once the LNG has been unloaded from the vessel it will be stored in a tank. Usually a terminal has multiple tanks to store the LNG in, all tanks do not necessarily have to be connected to each of the jetties.

NG is mostly used in its gaseous form, so it needs to be regasified. To do this, a terminal will usually have multiple machines that warm the LNG to a temperature at which it will be gaseous once again. This process is called regasification.

The delivery can be done in two ways. The first is called *send-out*; this is when the LNG is regasified so it can be pumped into any existing network for the distribution of NG. The other possibility is to keep the LNG liquid and reload it to another vessel. This process is called *reloading*.

1.1. ADP

The goal of this thesis is to create an optimised Annual Delivery Plan. An ADP is, as the name already indicates, a planning for a year for when vessels are due to arrive at the terminal and deliver the LNG. It also describes the type of delivery – send-out or reloading – planned during that year.

The ADP is created based on contracts drawn up between the LNG receiving terminal and LNG suppliers (clients). These clients have LNG they want to temporarily store at the terminal and have it from there delivered. In the contracts the amount of LNG, the method of delivery, and the period the contract runs for – a send-out period for send-out contracts and storage period for reloading contracts – are set.

The ADP that is created should fulfil the demands of the contracts while it also honours the constraints of the terminal, e.g. if a terminal has only one berth it is not possible two vessels arrive at the same time. The ADP should also be in line with the objective of the terminal. Typically a objective of a terminal could be to plan the arrivals of vessels in such a way that when vessels that arrive with a delay this should not influence the operation of the terminal.

A good way to optimise terminal utilisation and be sure delays do not influence the operation is lending and borrowing [24]. Lending and borrowing is when a client temporarily borrows LNG from another client of the terminal to fulfil its contracted send-out or reload. However, clients are not keen on lending too much of their LNG, so an objective when creating an ADP is to keep lending and borrowing to a minimum.

1.2. Systems Navigator

"*Systems Navigator is an independent software consultancy firm based in Delft, The Netherlands. Systems Navigator specialises in the design and creation of decision support solutions based on Operations Research technology. [Their] specific expertise is in using a combination of discrete event simulation and optimisation for decision support models that can predict system performance, as well as can be used for operational decision making by means of planning and/or scheduling.*"[1] For this thesis, Systems Navigator operated as an expert for ADP creation systems. They obtained this expertise by creating ADP creation and evaluation systems for multiple terminals in the world.

1.3. Research objectives

The main research objective of this thesis is to design a method to create an optimised Anual Delivery Plan for an LNG receiving terminal within reasonable time. We want a fast solving method since we

[1] http://www.systemsnavigator.com/

want to be able to use the algorithm in an interactive way, i.e. run the algorithms multiple times with different settings in a short time. We also want to do an evaluation of the algorithms currently used by Systems Navigator. To achieve these objectives we pose the following research questions.

RQ 1 *What are the objectives and contraints when creating an ADP for an LNG receiving terminal?*

Before creating an algorithm to solve te problem, first the problem should be analysed to get an idea of what the objectives and constraints are within this problem. This analysis can be used to determine what optimal means within the scope of this thesis and to determine a way of scoring the solutions found by different algorithms so they can be compared and we can determine which algorithm works best.

RQ 2 *Is the problem of creating an ADP an NP-hard problem?*

Before creating algorithms that do not run in polynomial time, first it must be checked if the problem is an NP-hard problem.

RQ 3 *What are appropriate techniques to describe the problem using an (Mixed Integer) Linear Program solver and which of these performs best (on a small scale)?*

In this thesis we decided to use Mixed Integer Linear Programs to solve the problem. We make use of the Gurobi Optimizer[2]. We have chosen for MILPs as solving methods because then the formulations we create can also be used by other solving techniques, e.g. a genetic algorithm with specific mutation rules.

Before creating a formulation for the whole problem we create three formulations in which we do not encode all the constraints. These three formulations make use of different methods to encode time (discrete versus continuous). We then run experiments with these formulation to see with which of these formulation we may be able to solve the whole problem.

RQ 4 *What is a good objective function for the Mixed Integer Linear Program?*

After testing the first formulations we choose the discrete time formulation to extend to allow for multiple clients. With this formulation we run experiments to determine what a good objective function is to use in the solver. A good objective function gives good results in our evaluation in the shortest time possible.

RQ 5 *Will a rolling horizon approach lead to faster results without much loss of optimality?*

Since we expect the solver not to be fast enough we also create a rolling horizon approach to solve the problem of creating an ADP. We want to know if this method solves the problem faster. On the other hand we do not want to lose too much optimality, so we should check the impact of the rolling horizon approach.

RQ 6 *How do the Mixed Integer Linear Program perform compared to the Systems Navigator Heuristic?*

Currently Systems Navigator uses a re-order point (ROP) heuristic in their algorithm (this is explained in section 4.4). We will compare our solutions with this heuristic algorithm to get an idea how this heuristic performs on solution quality compared to the optimal solutions we find. We also use it as a benchmark of how good our methods should perform to be acceptable.

1.4. Background

In this section we give background information on the techniques we use. Next to that we show the context to place our work in. We split up the background in scheduling in LNG – what has already been done on scheduling in LNG terminals – and a short introduction to (Mixed Integer) Linear programming. More background is given throughout the thesis in the sections where it is used.

[2] http://www.gurobi.com

1.4.1. Scheduling in LNG

A lot of work has been done on optimisation of operation in terminals. But the problems solved in literature differ from the creation of an optimal ADP for LNG receiving terminals in that the limiting factor for creating an ADP for multiple clients is the capacity of the tank in the terminal.

Work specifically done in the field of LNG typically does not focus purely on the LNG receiving terminal, which is our focus. There is work on planning the schedule for vessels as well as the route the vessels should take between the liquefaction facility and receiving terminal [1, 6, 14]. In this work the focus typically lays on pathfinding between the terminals. For the receiving terminal the amount a client can store in a tank is limited in such a way that conflicts in tank capacity are averted.

1.4.2. (Mixed Integer) Linear Programming

As stated before in this thesis we use MILP as a solving method for our problem. But before we go into MILP we first look into Linear Programming (LP). LP is the problem of minimising a linear cost function subject to linear equality and inequality constraints [3].

An LP is typically structured as follows:

$$\begin{aligned} &\text{Minimize} && \mathbf{c} \cdot \mathbf{x} \\ &\text{Subject to} && \mathbf{A}\mathbf{x} \geq \mathbf{b} \end{aligned}$$

We have an objective function we want to minimise, in this case the dot product of the cost vector c and variable vector x, subject to given constraints, in this case $Ax \geq b$. We could for example have the following linear program:

$$\begin{aligned} &\text{Minimize} && 2x_1 - && x_2 + && x_3 && \\ &\text{Subject to} && x_1 + && 3x_2 && && \geq 3 \\ & && && x_2 + && 2x_3 && \geq -2 \\ & && && - && x_3 && \geq 2 \end{aligned}$$

In this case we have $\mathbf{c} = (2, -1, 1)$,

$$\mathbf{A} = \begin{bmatrix} 1 & 3 & 0 \\ 0 & 1 & 2 \\ 0 & 0 & -1 \end{bmatrix},$$

and $\mathbf{b} = (3, -2, 2)$.

LPs can be solved in theoretical polynomial time with the ellipsoid method. Theoretical since in many cases this takes a lot longer than the often used Simplex method, which is worst case exponential but in most times is solved rather fast.

When we want to solve discrete optimisation problems LP does not always suffice. That is where Mixed Integer Linear Programming comes in to play. MILP is basically the same as Linear Programming, except that in MILP there are variables that are restricted to take integer values. For example, when we want boolean variables, the variables are restricted to the integer values 0 and 1. MILP is considered to be a strong framework to model discrete optimisation problems. While LPs can be solved in polynomial time, for MILPs that is not possible since the problem is NP-hard [3]. For solving a MILP exact several algorithms are known, while they provide the optimal solution, an exponential number of iterations may be needed. Examples of such algorithms are cutting plane, branch and bound, branch and cut, and dynamic programming methods.

A lot of solvers are available, both commercial and non-commercial, that implement these algorithms. In this thesis we use the Gurobi Optimizer, which uses all kinds of state of the art methods to find the solution as fast as possible. For example the primal and dual simplex algorithms for LPs and the cutting planes and branch-and-cut algorithms for the integer programs. It also makes use of pre-solving methods that for example detect and remove unnecessary variables and constraints so the problem is solved faster.

There is a lot of literature on applying MILP in scheduling problems. For example, when we look at the job shop problem or the flow shop problem we can easily find a MILP to solve it [22]. However

these problems do not translate one-on-one to our problem, as described in section 2.4. The MILPs for these programs typically make use of a discrete representation of time, i.e. we have timeslots in which we can perform certain tasks.

Since we also want to create solutions that make use of a continuous time representation we look into other techniques. One of these techniques is piecewise linear functions. A piecewise linear function consists of multiple linear parts and can, for example, be used to approximate non-linear functions in a linear program while keeping only linear constraints. If the piecewise linear function is convex they can be used in linear programs [3]. When, however, we have a concave function we need solve the problem with integer programmin. We can make use of Special Ordered Sets of type 2 (SOS2) [2] to model this. In literature we could not found anyone who tried to apply piecewise linear functions as way to model the pressure on a tank.

In the field of short term production planning of batch processes there also are a lot of solvers [21, 23]. While they initially worked with discrete time formulations, later also continuous time formulations are created, for example by Ierapetritou et. al [15–17]. The field is often applied to, as the name already states, short term production processes. The creation of an ADP is for a whole year, and therefore not short-term, and the typical scheme of transforming a product A to a product B is not what is happening in an LNG receiving terminal. However, we think looking at our problem of creating an ADP for an LNG receiving terminal can be viewed as a similar problem.

From the literature there does not directly follow an approach to solve the problem of creating an ADP for an LNG receiving terminal. Therefore we will look into how we could apply and extend the existing approaches so it fits our problem and how succesful these adaptations are going to be.

1.5. Thesis outline

The remainder of this thesis is organised as follows. In chapter 2 we present the definition of the problem of creating an ADP for an LNG receiving terminal. In this chapter also a proof the problem is hard is presented. The different methods we use for solving can be found in chapter 3, this chapter covers the single client formulations as well as the multi client formulation and the rolling horizon approach. Experiments and the results for these experiments are discussed in chapter 4. For each experiment we run we first discuss the setup of the experiments followed by analysis of the results for these experiments. Finally, in chapter 5, we summarise the answers to the research questions and present suggestions for future work.

2

Problem definition

In this chapter will go into Research Question 1 *What are the objectives and constraints when creating an ADP for an LNG receiving terminal?*. In this chapter both the output, input and constraints for the problem are defined as well as some goals and objectives a terminal can have.

2.1. Output – The Annual Delivery Plan

The goal of the algorithms presented in this thesis is to create an ADP. An ADP consists of three parts: an arrival list voor the vessels (V_{OUT}), Inventory Positions (IPs) at the beginning of the period ($IP(0)$) and the scheduled send-out pattern(S_{OUT}). A visual representation of an ADP can be found in Figure 2.1. In this image at the top we see the IPs of the tank drawn over time. This line follows directly from the vessel arrivals (in the middle) and the send-out pattern (bottom).

The arrival list is described by a set of vessels $\hat{v}$ each with a vessel type $v \in V$, an arrival time $T^s_{\hat{v}}$ and the load of the vessel $C_{\hat{v}}$.

The send-out pattern is a set containing non-overlapping parts $\hat{so}$ of send-out rate, each with a period ($[T^s_{\hat{so}}, T^f_{\hat{so}}]$) and a send-out rate (per day) $r_{\hat{so}}$.

2.2. Input Parameters

The input for the creation of an ADP for an LNG receiving terminal has two parts. There are the terminal properties and the contract input.

2.2.1. Terminal properties

The ADP created should be applicable to a certain terminal, thus the terminal properties become part of the input.

First of all, there are the tanks in which the LNG is stored in a terminal. A terminal can have multiple tanks. For the purpose of this thesis the tanks will be considered to be linked so that the separate tanks can be viewed as one large tank with capacity c. One should notice that for some terminals this abstraction is not reality, since it might be so that a certain client has a designated tank for himself. The amount of LNG in the tanks is denoted by $IP(t)$ for time t. It is possible to have a non-zero initial inventory $IP(0)$.

All times in both input and output will be given in days (when it is a moment it is from the start of the year).

Next there are the vessel types $v \in V$ that can arrive at this terminal. For all vessels a maximum capacity C_v is given, this is the maximum amount of LNG that a vessel can transport. An unloading time t^u_v is also given, which is the amount of time it takes to unload the vessel into the terminal tanks.

Furthermore, there are the berths $b \in B$ of a terminal. At these berths ships can dock and unload their LNG. It is possible that not all berths b will be able to handle all vessel types.

Lastly, some lending and borrowing restrictions can be added to the terminal. Lending and borrowing is when one client can borrow LNG from another client to achieve the desired send-out for

Figure 2.1: An example ADP. At the top we see the inventory positions of the tank over time, in the middle we see the vessel arrivals, and in the bottom we see the send-out pattern.

both clients while waiting for the arrival of another vessel. First of all, it should be determined whether lending and borrowing is possible at all in the terminal, and then, if it is possible, to what extend.

2.2.2. Contract input

When an ADP is created, this is based on contracts. There are several types of contracts (regasification, reloading, spot-cargo, see chapter 1), but within the scope of this thesis only the regasification contracts are considered.

A regasification contract g is determined by a period $[T_g^s, T_g^f]$, a list of vessels $\hat{v}_g \in \hat{V}_g$ (with a vessel type $v \in V$ and corresponding capacity c_v) and a send-out pattern SO consisting of parts so with a period $[T_{so}^s, T_{so}^f]$ and send-out rate per day r_{so}.

From all contract together also the horizon H follows, which is the maximal T_g^f of all the defined contracts $g \in G$. This value is the maximal value any time reference can take, since otherwise it would be outside the horizon of the instance.

2.3. Terminal goals and objectives

As stated briefly in the introduction goals of the terminal include optimising terminal utilisation and the impact differentiation of a vessel's arrival time has on the rest of the operation. In this section we will discuss different indicator objectives for these two main objectives.

2.3.1. Terminal utilisation

When optimising the terminal utilisation, the goal is to achieve the best usage of the terminal possible. Normally this is achieved by planning the contracts with the clients in such a way that the terminal is used to its maximal capacity throughout the year.

However, it is not always possible to live up to the contracted send-out due to certain circumstances, e.g. insufficient storage capacity or planned maintenance. We can allow for missing send-out to our formulations by adding slack variables (s_t^{SO}) for the desired send-out, representing the missed send-out for time period t. In the objective function we can then add $W^{SO} \sum_{t \in T} s_t^{SO}$ to penalise missing send-out, in which W^{SO} is a weight for missing send-out. Since not achieving the contracted send-out should be

only used as a last resort, the weight should be relatively high. In a multi-client formulation, it also is possible to variate these weights per client.

As already stated in section 1.1 Lending and Borrowing can be introduced to a terminal to increase the total terminal throughput [24]. But, since clients prefer to keep this lending to a minimum, the amount of LNG lent should be kept to a minimum. For this reason Lending and Borrowing could also be added to the objective function. We could add $\sum_{g \in G} \sum_{t \in T} W_g^{bor} bor_{g,t}$ to the objective to achieve this.

Another possible goal related to terminal utilisation is to have the possibility or space to plan more contracts on the ADP created. To be able to measure this multiple indicators can be used, for example free send-out capacity and free tank space. But, since send-out is determined in the input, this will not be an objective of the algorithm.

To optimise for free tank capacity we can create a threshold $IP_{\min}$. If an IP for the terminal or a certain client gets above this treshold we add a penalty on the objective function by adding $W^{IP} \sum t \in T \max(0, IP(t) - IP_{\min})$ to it.

When the objective is to plan more contracts it also might be interesting to include an pre-existing ADP to the input and add penalties for the amount of differentiation from this ADP. This way the already communicated ADP would probably not differ to much from the newly derived ADP.

2.3.2. Impact of differentiation

When an ADP is very sensitive to differentiation of vessel arrival times, this means the whole operation can be influenced by one different arrival time. This should be avoided by creating a “robust” schedule. Since early arrivals can be solved by delaying the vessel, we are more interested in late arrivals.

A possible way to handle late arrivals is by ensuring there always is at least a certain amount of LNG in the terminal tank. For this, a similar function as with minimising the tank capacities can be added to the objective. This time we add a threshold $IP_{\max}$ below which we add a penalty by adding $W^r \sum_{t \in T} \max(0, IP_{\max} - IP(t))$ to the objective.

2.4. Complexity

Since in this thesis we are planning to use Mixed Integer Linear Programming, which is known to be NP-hard, to solve this problem, we should first determine if the problem we are trying to solve does not have a much simpeler algorithm we could use. Therefore in the introduction we stated Research Question 2 *Is the problem of creating an ADP an NP-hard problem?* In this section we give an overview of simple problem instances and for which instances the problem is hard.

First we split up the instance space in groups with different characteristics. We split up in single client and multiple client instances, one vessel type and multiple vessel types, fixed vessel arrival order per client and totally free vessel arrival order per client, and constant send-out over time and changing send-out. We first look at simple instances we can solve in polynomial time, followed by an overview of known NP-Complete problems that are similar to our problem, and finish of with a complexity proof by showing a reduction from Partition.

2.4.1. Algorithm for simple instances

First we have look at the single client, multi vessel type, free vessel arrival order and changing send-out instances. Note that one vessel type, fixed vessel arrival order and constant send-out are special cases for these instances. Thus if we can solve these instances we can solve all single client instances. Optimising to minimise lending and borrowing with a single client is not possible, because there is no one to lend from.

For these instances and the objective to have as much free tank capacity as posible we can create the following algorithm:

for all $v \in V$ **do**
  t ←time tank is empty based on send-out
  plan vessel v on time t
end for

The order in which the vessels arrive at the terminal does not matter, since eventually all vessels need to be planned in the period. If there are more vessels than is needed, planning more small vessels would be the best plan.

We could also adjust the algorithm to be a little more robust by adding a re-order point, a certain amount of days before we hit the tank bottom or minimal threshold of stored LNG that if we pass it the next vessel should arrive. The algorithm created by Systems Navigator uses a re-order point.

When we want to optimise to have an as full as possible tank at every moment in time, we could simply change step 2. When we plan the next vessel at the moment it will fit we would get the best results.

We have shown an algorithm to solve all single client instances optimally for our objective functions. The algorithm takes $O(n)$ with n the total number of arriving vessels. However, since optimality is a bit arbitrair for the problem of creating an optimised ADP and we have simply some options for objective function, we should mention that solving optimal for another objective function might prove to be hard.

When we look at the multi client instances the algorithm does not work. Since there is limited storage capacity in the tank it might not be possible to fit the vessel contents in the tank at the moment the vessel arrives. The order in which vessels arrive influences whether or not we exceed the tank boundaries. Next to that with more clients it is very likely lending and borrowing becomes necessary because, though two small vessels may fit their contents in the tank, with three vessels this soon is a problem. Notice that when there is enough tank capacity to fit the number of clients times the largest vessel capacity we could use the algorithm described above, but this is not to be expected.

2.4.2. Similar problems

Next we look at known problems that are similar to our problem. For each problem we describe the similarities and why our problem is different and known solution methods cannot be directly used to solve our problem.

One might expect that, when many vessels arrive, berthing all the vessels becomes a problem and the Berth Planning Problem comes to mind. The BPP is NP-Complete [20]. However in most cases the tank capacity is more restrictive than the number of berths in a terminal because unloading multiple vessels at the same time also means all the carried LNG should fit in the tanks at the same time. Next to that in our problem we assume that when there are multiple jetties at a terminal where one can unload the vessels they are far enough appart to fit the vessels. If in the future the problem expands to allow for multiple vessels at a berth the BPP should be considered.

Sequencing with release times and deadlines, problem SS1 in [13], is a known NP-Complete problem. It is known that this problem is also hard when the jobs can only take two lengths [8]. If we want to solve our problem with ordered vessel arrival for multiple clients on one berth, we can convert our instance to an instance of this problem. Release times are determined by when the vessel should fit in the tank and deadlines are the moments the LNG brought by previous vessels for that client would be all sent-out. The deadlines are thus always structured to the release time by the amount of LNG and send-out rate, this might prove to make the problem easier to solve. Next to that with multiple clients the release times should change the moment a vessel has been planned, so the maximum tank capacity is not exceeded.

Our problem also looks, under certain assumptions, like Sequencing to minimise maximum cumulative cost, problem SS7 in [13]. When we have ordered vessel arrivals and one berth in the terminal we could model storage as a resource and add costs for this storage-resource. However, we only have one-sided costs from the arriving vessels (or tasks), which might be easy problems within this category.

Resource constraint scheduling, problem SS10 in [13], is a problem similar to the previous two, but it allows for multiple berths, or processors as they are called in the problem definition. Tank capacity could be modeled as a resource which is used by an arriving vessel. In the normal problem definition only jobs with length of 1 are considered, but in our problem vessels of different sizes arrive at the terminal, which take different times to unload.

Unfortunately the problem definition does not have restoring of resources, which is the case in our problem when LNG is sent-out. So we can not use commonly used practices for solving resource constraint problems directly.

Production Planning, problem SS21 in [13], is a problem of creating a schedule for a production plant. It is based on a production capacity per time unit. It is even hard with equal demands over all time periods, but can be solved in polynomial time if the production capacity is equal [11]. If the cost function is convex and there are no setup costs the problem can be solved in polynomial time, if there are setup costs it is still NP-hard [12].

The problem does not map one-to-one to our problem since it does not limit the inventory amounts,

but it only ensures all the costs do not exceed the a certain bound. The tank limitation is a hard constraint in our problem.

2.4.3. Complexity proof

We finish this section with a proof the general problem of creating an optimised ADP for an LNG receiving terminal is NP-complete. We therefore first show the problem is in NP and then give a reduction from Partition and show a YES-instance in partition is a YES-instance in our problem and vice versa. We close of with some final remarks for this proof. For the proof we use the decision problem of wether it is possible to create a schedule without lending and borrowing and use a discrete version of the problem in which the send-out is handled all at once at the end of the day, we call this problem ADP. Note that this is the same problem where maximal of $k\ m^3$ lending is allowed.

Definition *Partition [13]*
INSTANCE: Finite set A *and a size* $s(a) \in Z^+$ *for each* $a \in A$.
QUESTION: Is there a subset $A' \subseteq A$ *such that* $\sum_{a \in A'} s(a) = \sum_{a \in A - A'} s(a)$*?*

To show ADP is in NP we have to show that given a solution we can verify the solution in polynomial time. This is rather trivial since we can just iterate over the schedule in the solution and check if it exceeds the tank bounds or uses any lending and borrowing throughout the horizon. So ADP is in NP.

The next step is to give a polynomial time reduction from Partition to ADP. This can be done with the following steps:

1. Take an instance of Partition
2. Convert each $a \in A$ to a vessel $v \in V$ with capacity $c_v = s(a)$
3. The horizon of the problem is 2, i.e. we plan for two days. ($H = 2$)
4. The send-out for both days is $S = \sum_{a \in A} s(a)/2$.
5. The tank capacity is S.
6. Start with an empty tank ($IP(0) = 0$).
7. Create the same number of berths as vessels ($|B| = |V|$)

If we have a YES-instance for Partitions there exists a subset A' such that $\sum_{a \in A'} s(a)$ is equal to $\sum_{a \in A} s(a)/2$. By applying above given reduction we get an instance of ADP. In this instance we can create a solution by planning the vessels v corresponding to the items $a \in A'$ on the first day and the other vessels on the next day. Since the sum of the capacities is equal to half the total amount, which is the same as the size of the tank, everything will fit. At the end of the day the whole of the tank is send-out so the tank starts empty on the second day. The deliveries on the second day will also exactly fit in the tank and send-out during the day.

If we have a YES-instance for ADP structured in the same way as in our construction – two days, equal send-out both days, tank capacity equal to half of the sum of all vessel capacities – there exists a schedule which adheres to the constraints of the tanks and send-out. When we do an inverse reduction to Partition we have a solution for Partition by taking the items a corresponding to the vessels planned on the first day and put them in A'.

We have shown that ADP is NP, shown a polynomial reduction from Partition to ADP, and shown that the reduction is correct. We therefore conclude that ADP is NP-Complete. However, there are a few sidenotes to this conclusion. First of all, Partition can be solved in pseudo-polynomial time by dynamic programming [13], and is restricted by the size of the items $a \in A$, which in our case is limited. Next to that the reduction works with only two days with equal send-out and a lot of vessels arriving. This is not typical for our problem.

On the other side, however, the problem definition as used in the reduction is a simplification of the real problem and adding more constraints might make the problem even harder. All in all we conclude there is strong evidence the problem is hard.

3

Mixed Integer Linear Programs

In this chapter we discuss different Mixed Integer Linear Program (MILP) formulations for the problem described in chapter 2. We start with three single client formulations, which are used to determine good ways to model time for this problem. After that we introduce a multi client formulation and a method for a rolling horizon approach. For each method we first introduce the formulation and then discuss the advantages and disadvantages of that specific formulation. In Appendix A the formulations are written out.

The single client formulations assume a terminal which only has one client or multiple clients with no regard for which client brings which amount of LNG. With the multi client formulations the notion of clients is introduced such that each client will send-out their own LNG or lend the needed LNG. In both formulations eventually the amount of LNG brought to the terminal should be equal to the amount that has been sent out.

3.1. Single client formulations

In order to answer Research Question 3 *What are appropriate techniques to describe the problem using an (Mixed Integer) Linear Program solver and which of these performs best (on a small scale)?* we present three formulations for a single client terminal in this section. We first present a discrete time formulation, followed by two different approaches making use of a continuous time scale. The time scale used determines the moments vessels can arrive; with discrete time this only is on each time step, while with a continuous time vessels may arrive at any moment. These formulations are compared on runtime in section 4.2.

3.1.1. Single client Discrete Time

The first MILP formulation is the single client discrete time formulation. This formulation makes use of boolean variables ($A_{v,t}$) to represent the arrival of a certain vessel type $v \in V$ at an integer time $t \in T$. T is the set of all nonnegative values smaller than horizon H.

Each time step, or each day, the inventory positions are updated with the arriving vessels ($D_t = \sum_v A_{v,t} \cdot c_v$) and amount of send-out (S_t) for that day. The tank capacity is checked by ensuring a positive $IP(t)$ which is smaller than the capacity of the tank (c).

The maximum amount of arrivals for a certain vessel type is capped by the amount of vessels of that type. This can be done by adding $\sum_t A_{v,t} \leq \#_v$, with $\#_v$ is the amount of vessels of type v, for $v \in V$ as a constraint. The amount of berths are incorporated into the formulation by adding $\sum_v A_{v,t} \leq |B|$ for $t \in T$. This limits the number of vessel that can arrive for each timestep.

An advantage of this formulation is the simplicity of it. It is relatively easy to read and understand and therefore it is not that hard to extend the formulation with other constraints.

The relative easiness of extending the formulation is due to the fact we plan on whole days, which makes it easy to add constraints on these point. This is also a downside however, since we might lose some optimality, due to the fact that it might be so that the optimal solution would be to plan the vessel halfway through the day. In real life the vessels would be assigned an arrival slot which is most of the time a whole day, so the lost optimality is only theoretical loss of optimality.

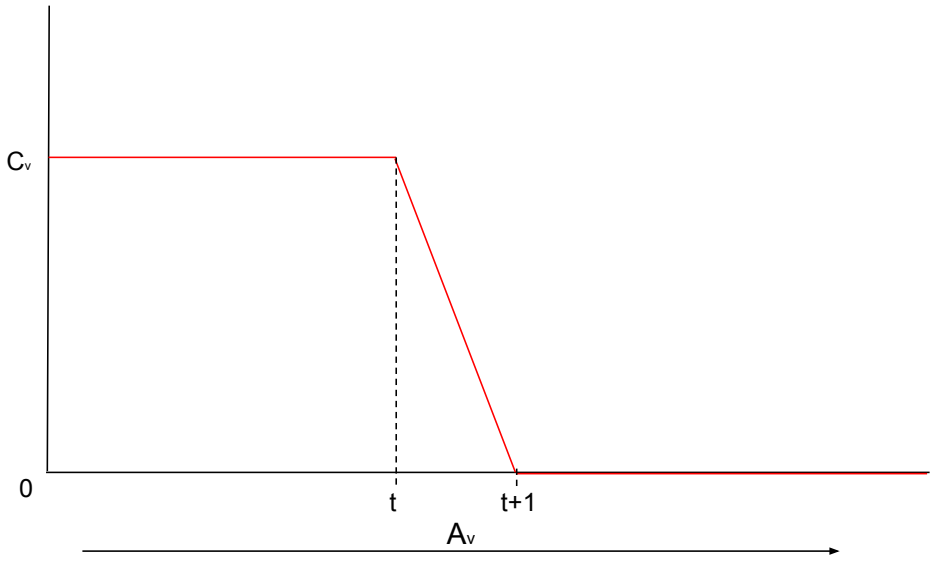


Figure 3.1: $I_v(t)$

Another disadvantage of the formulation is the number of boolean variables needed. We need the number of vessel types ($|V|$) times the number of time steps ($|T|$) boolean variables.

In the current formulation the unloading is fixed to one time slot. Should that be changed to multiple, e.g. when there is an unloading time of two days or we decrease the duration of a time slot, we can achieve that by adding helper variables which will have the value of $A_{v,t-1}$ and sum the helper variables and the original arrival variables together to check for the berth constraints.

3.1.2. Single client Piecewise linear

The second formulation we call the piecewise linear formulation, due to the fact it uses piecewise linear functions. The reason we created this formulation is that we wanted to use a more accurate representation of time. The approach is, as far as we found in literature, not yet used in a similar fashion.

This formulation contains the piecewise linear function $I_v(t)$ that represents the tank load for a vessel v at time t, based on a decision variable A_v which represents the arrival time of vessel v. This amount is C_v if the vessel did already arrive ($A_v \leq t$), between C_v and 0 if the vessel did arrive within the last day ($t < A_v < t + 1$), and 0 if the vessel arrived more than a day ago. This representation is chosen such that the arrival time A_v can be represented using a continuous variable. A piecewise linear function is a function which is composed of multiple linear parts. In Figure 3.1 a drawing of $I_v(t)$ can be found.

A piecewise linear function can not be used in (Gurobi) linear programming directly, but can be represented using SOS2 [2]. SOS2 are ordered sets of non-negative decision variables of which at most two can be non-zero, and when there are two non-zero variables they should be consecutive in their ordering. For a function $f(z)$ it is now possible to write $\sum b_i \lambda_i$ with

$$\sum_i a_i \lambda_i = z \tag{3.1.2.1}$$

$$\sum_i \lambda_i = 1 \tag{3.1.2.2}$$

If the values a_i and b_i are chosen such that $f(z) = b_i$ if $z = a_i$ and λ is a SOS2 we have represented the function $f(z)$ with linear pieces. This can be done to approximate any function. In this case the function $I_v(t)$.

The function $I_v(t)$ is used in the linear program to check the capacity and send-out contraints of the problems at 00:00 each day. So for each vessel v and $t \in \mathbb{N}$ with $t \leq t_{max}$ the SOS2 $\lambda_{v,t}$ is created

and corresponding:

$$a_{v,t,1} = 0 \qquad b_{v,t,1} = C_v$$
$$a_{v,t,2} = t \qquad b_{v,t,2} = C_v$$
$$a_{v,t,3} = t + 1 \qquad b_{v,t,3} = 0$$
$$a_{v,t,4} = t_{max} \qquad b_{v,t,4} = 0$$

For all values t the arrival time of the vessel (A_v) should be the same, and therefore $\sum a_{v,t,i}\lambda_{v,t,i}$ should be equal for all values of t. The value $\sum_v \sum_i \lambda_{v,t,i} b_i$ for a certain value t can be used to check the tank levels.

The main advantage of this formulation is that it allows for a continuous value for the arrival time of a vessel (A_v), which might lead to better results. It is important to notice though that this approach is actually semi-continuous. The tank levels are only checked on integer values for t (each day at 00:00), which may lead to faults in the planning, either because there is too much or not enough LNG in a tank somewhere during the day. This can be improved by adding constraints for checking at a smaller interval, but that leads to a lot more constraints and only reduces the magnitude of the faults.

A disadvantage of this formulation is that the variables and constraints for arrival will be created for each arriving vessel instead of per vessel type, as with the previous formulation. The number of arriving vessels per year is typically significant higher than the number of different vessel types. It would be possible to model the vessels per vessel type, but that would introduce as much new variables and a lot more complexity.

Another big downside of the formulation is the amount of extra variables with SOS2 constraints needed to model piecewise linear functions. This turned out more complex than anticipated beforehand, experimentation shows how much influence this has on the solving time.

Next to that it is also hard to add new constraints to the formulation. For instance, currently there is no constraint on the arrivals with relation to the number of berths in the terminal. Adding such a constraint would be far from trivial with in the formulation.

3.1.3. Single client Event-based

The last of the single client formulations is an event based model, based on work for production plants. It is an adaptation of the work described in [15–17] and most of the formulation is adopted from that work. This formulation is more complex than the previous two and therefore we will discuss this one more thoroughly.

In this approach, instead of pre-defining timeslots with fixed start and end time as with discrete time, we create timeslots with a start and end time which are decision variables. We split up the time slots by event points. During each event point a certain task can be assigned to a certain unit. Each task assigned to a job in an event point gets a starting and ending time, e.g. a vessel arrives at the sixth of March at 19:57 and will be finished exactly a day later. In the same event point the send-out task can be active from the fifth of March until a week later.

Although at first thought it is not trivial to see work created for a production plant is applicable to our problem, this is the case. In our formulation an example of a task and unit is the arrival and unloading of a vessel (task) at a berth (unit). The products produced are stored LNG, produced by a vessel on a berth, and sent-out LNG, produced by the send-out task and unit.

For this event-based formulation we adopt and extend the notation from the papers as follows:

I = Tasks
I^{AR} = Tasks that represent arrivals
I^{SO} = Tasks that represent send-out
i_{SO} = The only task in I^{SO}
I_v^{AR} = Arrival tasks associated with vessel v
I_j = Tasks that can be performed at unit j
I_s = Tasks that process state s
J = Units
J^{AR} = Units used for arrival (berth)
J^{SO} = Units used for send-out
j_{SO} = The only unit in J^{SO}
J_i = Units that can process state i
N = Event points within the time horizon
N_{fixed} = Event points with a fixed time
S = All involved states s

$ST(s)^{\max}$ = Storage size for state s
L = Small value (e.g. 0.1)
H = Time horizon
$\alpha_{i,j}$ = Constant term of processing time of task i on unit j
$\beta_{i,j}$ = Variable term of processing time of task i on unit j
c_i = Capacity of vessel associated with task i
$\rho_{s,i}^{p}$ = Proportion of state s produced by task i
$\rho_{s,i}^{c}$ = Proportion of state s consumed by task i
$r(n)$ = Send-out rate in event slot n

$wv(i,n)$ = Boolean variable that assigns the start of task i at event point n
$yv(j,n)$ = Boolean variable that assigns the usage of unit j at event point n
$B(i,j,n)$ = Amount of material used in task i in unit j at event point n
$ST(s,n)$ = Amount of state s in storage at event point n
$T^{s}(i,j,n)$ = Starting time of task i in unit j at event point n
$T^{s,AR}(n)$ = Latest starting time for an arrival task in event point n
$T^{f}(i,j,n)$ = Finish time of task i in unit j a event point n
$t(n)$ = Only determined for event points $n \in N_{\text{fixed}}$ gives the fixed time for n

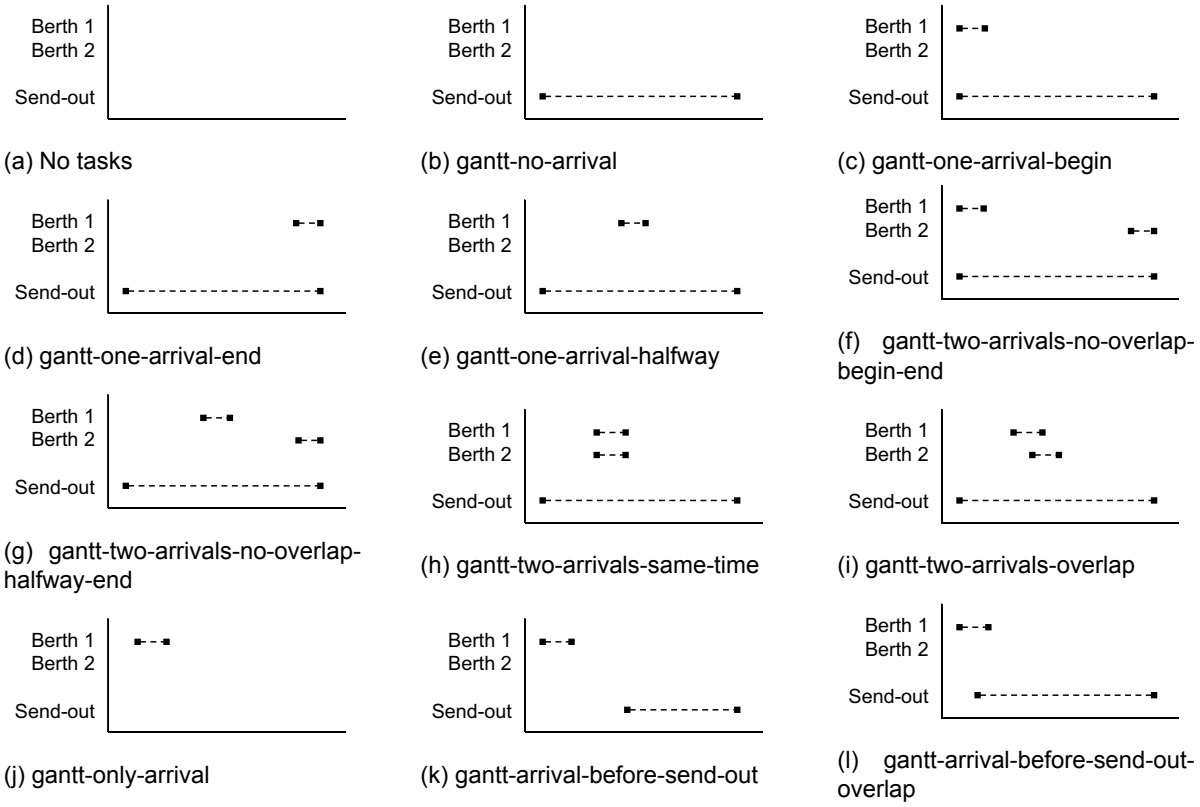
For each vessel an arrival task is created for every berth, e.g. if there are two berths for each vessel there is a `vessel-berth-1` task and a `vessel-berth-2` task. For send-out one task (i_{SO}) is created.

For the arrival tasks the units are the berths, together they produce the amount of “stored LNG” that is available on the vessel. For the send-out task one unit (j_{SO}) is available, this will produce “sent-out LNG”. At the moment of each change in send-out rate the “sent-out LNG” should be delivered as described in the original work and discussed below.

Before introducing the constraints we will first explain how we generate the event points for an instance. We first generate fixed event points and then generate free event points between them.

The fixed event points are generated by taking the start and end time for each of the send-out parts. For each time we store the send-out rate change, i.e. positive r_{SO} at the start point and negative at the end, and for the end time we also save how much send-out should have been sent-out in the given send-out part, this we call the delivery. We then combine same points with the same time by adding the rate changes and deliveries for points with the same time.

The free event points are generated based on the fixed event points. We create five event points before the first event point, these will have a $r(n)$ of zero. Between each set of two fixed event points we dynamically generate a certain number of event points. This number is determined by the total amount of send-out between these points divided by the size of the smallest vessel rounded up then multiplied with 1.5 and again rounded up. For these event points $r(n)$ is equal to the send-out rate from the first fixed event point. We generate this number of free event point between the fixed event points



(a) No tasks

(b) gantt-no-arrival

(c) gantt-one-arrival-begin

(d) gantt-one-arrival-end

(e) gantt-one-arrival-halfway

(f) gantt-two-arrivals-no-overlap-begin-end

(g) gantt-two-arrivals-no-overlap-halfway-end

(h) gantt-two-arrivals-same-time

(i) gantt-two-arrivals-overlap

(j) gantt-only-arrival

(k) gantt-arrival-before-send-out

(l) gantt-arrival-before-send-out-overlap

Figure 3.2: Gantt charts of the key arrival possibilities

so we always have enough event points available to make our planning. At least we need the total send-out divided by the size of the smallest vessel – when we assume only small vessels come on this interval we will need this many event points – and we multiply that by 1.5 so there is more freedom in how to assign the arrivals and possibly allow for extra vessel arrivals if that might be more optimal.

In Figure 3.2 some schematic cases what could happen during a event point are shown for zero, one or two arrivals. For example, in Figure 3.2j we see one vessel arriving at berth 1, and in Figure 3.2f we see a whole period of send-out with an arrival on berth 1 at the beginning of the send-out and an arrival on berth 2 at the end. We reference certain cases during the discussion of the constraints which follows. The figure might also give a better general understanding of how the formulation works internally.

Allocation constraints

$$\sum_{i \in I_j} w(i,n) = yv(j,n) \qquad \forall j \in J, n \in N \tag{3.1.3.1}$$

These constraints ensure that only one task can be active per unit during each event point. If we have $yv(j,n)$ equal to zero than none of the tasks I_j for that unit should not be active. If unit j is active ($yv(j,n) = 1$) there is at most one task that is also active.

Capacity constraints

$$B(i,j,n) = wv(i,n) \cdot c_i \qquad \forall i \in I^{AR}, j \in J_i, n \in N \tag{3.1.3.2}$$

$$B(i,j,n) \leq (T^f(i,j,n) - T^s(i,j,n)) \cdot r_n \qquad \forall i \in I^{SO}, j \in J_i, n \in N \tag{3.1.3.3}$$

The constraints for the amount of material involved in task i on unit j are different for arrival and send-out tasks/units.

For arrival tasks we force that vessels will arrive fully loaded. So the total amount of material involved is equal to the capacity of the vessel. We only count this amount if the task of that vessel type arriving that event point is active ($wv(i,n) = 1$). In the future we can change this constraint to allow for vessels which are not fully loaded, the constraint should then be more like the one in the original papers, e.g. $V_{i,j}^{\min} \leq B(i,j,n) \leq V_{i,j}^{\max}$.

For the send-out task/unit we limit the amount of material involved by the duration of the task on the unit times the send-out rate in that event. Ensuring there are non-zero durations for tasks that are active is done by the duration constraints described below.

Tank constraints

$$ST(s)^{\max} \geq ST(s,n) \qquad \forall s \in S, n \in N \tag{3.1.3.4}$$

$$ST(s)^{\max} \geq ST(s,n) + \sum_{i \in I_s} \rho_{s,i}^{p} \sum_{j \in J_i} B(i,j,n) \qquad \forall n \in N, s = \text{“Stored LNG”} \tag{3.1.3.5}$$

$$ST(s,n) \geq (T^{s,AR}(n) - T^{s}(i_{SO}, j_{SO}, n)) \cdot r_n \qquad \forall n \in n \tag{3.1.3.6}$$

These constraints ensure the limits of the tank are not exceeded. For “send-out LNG” we have a virtual storage with infinite space, for the “stored LNG” the $ST(s)^{\max}$ is equal to the tank capacity c. The first set of constraints only apply to the beginning and end of event points. To ensure we do not get errors (e.g. not enough LNG in the tank or an empty tank) halfway through the event point we add the other two sets of constraints.

First we want to ensure that all LNG delivered within an event point fits into the tank as a whole (Equation 3.1.3.5). This prevents a problem that might occur for instance in the situation shown in Figure 3.2k. In this situation, without the extra constraints, the LNG brought by the vessel might not fit in the tank, but since the LNG can be send-out before the end of the event point (i.e. before the check) this could not be detected.

The other set of constraints (Equation 3.1.3.6)ensures we have enough LNG in the tank to sent-out until the last vessel arrival. We check with the last vessel because otherwise we might have a situation where between multiple vessel arrivals we do not have enough LNG in the tank (e.g. the situation in Figure 3.2f). We assume that the unloading rate of the vessel is much higher than r_n, so from the latest vessel arrival until the end of the event point we will have enough if at the end of the event point we have enough LNG.

Both extra constraint sets are more strict than might be necessary. This is not really a problem because in most cases an event point with multiple arrivals can be split up in multiple different event points. For example the situation in Figure 3.2k can be split up in Figure 3.2j and Figure 3.2b, and the situation in Figure 3.2f can be split up in Figure 3.2c and Figure 3.2d. We can not split up all arrivals in multiple event points however; Figure 3.2h and 3.2i are examples where it would not be possible.

Material balances

$$\begin{aligned} ST(s,n) = {} & ST(s,n-1) - delivery(s,n) \\ & + \sum_{i \in I_s} \rho_{s,i}^{p} \sum_{j \in J_i} B(i,j,n-1) \\ & - \sum_{i \in I_s} \rho_{s,i}^{c} \sum_{j \in J_i} B(i,j,n-1) \qquad \forall s \in S, n \in N \end{aligned} \tag{3.1.3.7}$$

These constraints are used to keep track of how much “stored LNG” is in the tank and how much “sent-out LNG” there is in the virtual sent-out tank. Each event point the ST is updated with the amount of material involved and how much of it is produced or consumed.

The delivery is only non-zero for “sent-out LNG”. Delivery takes place after each send-out change at a fixed-time event point, this way we ensure enough LNG is sent-out at the correct rate.

Duration constraints

$$T^{s,AR}(n) \geq T^{s}(i,j,n) - H(2 - wv(i,n) - yv(j,n)) \qquad \forall i \in I^{AR}, j \in J_i, n \in N \tag{3.1.3.8}$$

$$T^{f}(i,j,n) = T^{s}(i,j,n) + \alpha_{i,j} wv(i,n) + \beta_{i,j} B(i,j,n) \qquad \forall i \in I^{AR}, j \in J_i, n \in N \tag{3.1.3.9}$$

$$L \cdot wv(i,n) \leq T^{f}(i,j,n) - T^{s}(i,j,n) \leq H \cdot wv(i,n) \qquad i \in I^{SO}, j \in J_i, n \in N \tag{3.1.3.10}$$

The first set of constraints is basically a maximum function on $T^s(i,j,n)$ for the arrival tasks that happen within a certain event point n. If there are no arrival tasks active within this task Equation 3.1.3.6 is trivially met, since $T^{s,AR}(n)$ will be negative.

The end times for the arrival task event points are determined by a constant (α) and variable (β) part for a certain vessel on a certain berth. In the experiments ran in chapter 4 $\alpha_{i,j}$ is one day and $\beta_{i,j}$ is zero. For later implementations this could be tweaked to better correspond with how the terminal operates.

The constraint discussed at the introduction of Equation 3.1.3.3 is the last set constraints above. This constraint ensures there is only a duration for the send-out task when the send-out task is active in this event point.

Sequence constraints

$$T^{s}(i,j,n+1) \geq T^{f}(i,j,n) - H(2 - wv(i,n) - yv(j,n)) \qquad i \in I, j \in J_i, n \in N, n \neq N \tag{3.1.3.11}$$

$$T^{s}(i,j,n+1) \geq T^{s}(i,j,n) \qquad i \in I, j \in J_i, n \in N, n \neq N \tag{3.1.3.12}$$

$$T^{f}(i,j,n+1) \geq T^{f}(i,j,n) \qquad i \in I, j \in J_i, n \in N, n \neq N \tag{3.1.3.13}$$

$$T^{s}(i,j,n+1) \geq T^{f}(i',j,n) - H(2 - wv(i',n) - yv(j,n)) \qquad i \in I, i' \in I, i \neq i', j \in J_i, n \in N, n \neq N \tag{3.1.3.14}$$

$$T^{s}(i,j,n+1) \geq T^{f}(i',j',n) - H(2 - wv(i',n) - yv(j',n)) \qquad i \in I, i' \in I, i \neq i', j \in J_i, j' \in J_i, j \neq j', n \in N, n \neq N \tag{3.1.3.15}$$

Vessel arrivals

$$\sum_{n \in N} \sum_{i \in I_v} wv(i,n) = \#_v \qquad \forall v \in V \tag{3.1.3.16}$$

The number of vessels per vessel type that can arrive during the planned period are limited. These constraints limit the number of executed arrival tasks related a vessel type v to the amount of vessels of that type available ($\#_v$). They are equal since we assume all vessels that are available should be planned.

Fixed event points

$$T^{s}(i,j,n) = t(n) \qquad \forall n \in N_{\text{fixed}} \tag{3.1.3.17}$$

$$T^{f}(i,j,n) = t(n) \qquad \forall n \in N_{\text{fixed}} \tag{3.1.3.18}$$

Before the discussion of the constraints the generation of fixed event points is discussed. These constraints ensure the event points are fixed in time.

With this formulation we achieved continuous time arrivals. This should lead to more optimal answers than the discrete time formulations, although it is important to remember this optimality is only theoretical, because eventually vessels will get whole day arrival slots.

Other advantages of this formulation are the configurability of the formulation. For instance the unloading time, which can be configured using the α and β to have standard docking times and maybe even unloading time based on the amount of LNG on the vessel. Another example is the possibility to

model berth restriction for vessels, i.e. a certain vessel can not unload at a certain berth. This can be achieved by not creating the job for unloading the vessel at that berth.

On the downside, it is much harder to really get a grasp on how the formulation is structured exactly. This makes it harder to extend the formulation with extra constraints which are not in line with the ones mentioned as advantage. For example, extending the formulation with multiple clients and lending and borrowing is harder than with the discrete time formulation. Thoughts on how to implement multiple clients and lending and borrowing for this formulation can be found in subsection 5.2.1.

3.2. Multi client formulation

In chapter 2 we introduced many more constraints than modelled in the previous section with the single client formulations. The most important missing constraint is that a terminal has multiple clients. In this section we discuss an extended version of the discrete time formulation which allows for multiple clients and lending and borrowing between these clients.

The vessel types are generated per contract. This means that although the contracts might have access to the same vessel types, the formulation still sees them as different vessel types. The vessel types corresponding to a contract are in the set V_g.

In this formulation we introduce inventory positions per contract ($IP(g,t)$), where each contract corresponds with one client. For each contract we have the same method of updating the inventory positions as with the single client formulation, except we allow for missed send-out in this formulation. This is achieved with slack variables $s^{SO}_{g,t}$ corresponding to the amount of send-out missed for a certain contract in a certain time slot:

$$IP(g,t) = IP(g,t-1) + D_{g,t-1} - (S_{g,t-1} - s^{SO}_{g,t-1}) \qquad \forall g \in G, t \in T$$

Inventory postions for a certain contract are allowed to be negative, the total of inventory positions for all contracts should however always be positive. If the inventory position for a certain contract goes below zero this means the client corresponding to that contract is borrowing LNG from another client. We do not keep track of which of the clients is lending the LNG.

Since clients can borrow LNG from each other we also need a constraint to ensure each of the clients brings enough LNG over the whole year. This constraint also considers possible missed send-out modelled with the slack variable.

For this formulation we introduce multiple objective functions. These objective functions can be added together with different weights as described in chapter 2. The objectives are all to be minimised. We add four parts for the objective function:

$$\Pi_{SO} = \sum_{t\in T}\sum_{g\in G} s^{SO}_{g,t}$$

$$\Pi_{\text{tank}} = \sum_{t\in T}\sum_{g\in G} \frac{\max(0, IP(g,t) - IP_{\min})}{c - IP_{\min}}$$

$$\Pi_{\text{tmax}} = \sum_{t\in T}\sum_{g\in G} \frac{S_{g,t}\cdot \max(0, IP_{\max} - IP(g,t))}{c - IP_{\max}}$$

$$\Pi_{LB} = \sum_{t\in T}\sum_{g\in G} \frac{\max(0, -IP(g,t))}{\min_{v\in V}(c_v)}$$

These objectives are slightly different from what was presented in chapter 2. We decided to normalise the objectives functions to have similar ranges of the objectives for different problem instances.

The first is the amount of missed send-out (Π_{SO}). This objective is simply the sum of all missed send-out over a year. Next to the objective to minimise this, we add the possibility to not allow for any missed send-out at all. This is done by forcing all $s^{SO}_{g,t}$ to be zero.

The second and third are comparable in nature. They both have to do with the amount of LNG in the tank at a certain moment in time. First we have Π_{tank}, this objective is modelled to have a threshold ($IP_{\min}$); if all contract IPs are below this threshold the value of the objective is zero. Otherwise the value will be non-zero normalised against the tank size per contract. This way we penalise solutions where

tank levels go below a certain level and thus get full tanks. Next we have Π_{tmax}, this objective is the inverse of the previous. The solutions are penalised if the contract IPs are below a certain threshold $IP_{\max}$, so we try to keep at least a certain amount of LNG in the tank. We finally multiply with the send-out in the time interval $S_{g,t}$ so going below the threshold is penalised more if the client has more desired send-out, as a bonus it also cancels out the penalty on days where the client does not send-out anything at all.

Lastly we have an objective to minimise the amount of borrowing within the terminal (Π_{LB}). This objective is calculated by summing up all the amounts the IPs go below zero, normalised to the size of the smallest vessel in the instance.

The advantages and disadvantages of this formulation are basically the same of those for the single client discrete time formulation. Although we surrendered a bit of readability in favor of having multiple clients the formulation is still fairly understandable.

3.3. Rolling Horizon

From experimentation we learned that the horizon for which we solve an instance is one of the most important factors that influences the run time for solving. Therefore we introduce an approach in which we run the solver for multiple smaller time steps. In this section we explain how this is achieved.

For the rolling horizon approach we split up the period we want to plan in smaller pieces. We choose pieces of thirty days. We use the formulation from the previous section and run the solver for each of the smaller parts with an overlap of fifteen days and combine them all together. As we explain next in more detail.

For each part we first select the vessels that should be planned within this time. We do this because when we plan for minimal IPs or maximal IPs there is a preference for either small or large vessels, which results in planning vessels according to size. Per contract we pick random available vessels until the vessels in total can bring enough LNG to cover the send-out.

We then run the solver to create a planning for thirty days with fifteen days overlap with the previous part planned. Within this overlap period the arrival of a vessel may at most be one day earlier or later than what was planned before, we do this to reduce the impact of cutting of the period – the solver could “oversee” that planning something later, or earlier, would result in a higher objective value when planning for the whole period. The day-off planning is achieved by adding a constraint to ensure a vessel of that type arrives one day before, on the day itself, or a day after:

$$A_{v,t-1} + A_{v,t} + A_{v,t+1} \geq 1 \qquad \forall v, t \in \text{semi fixed arrivals}$$

where semi fixed arrivals is the set of arrivals planned for the period that overlaps with the previous time window.

After the solver has completed we store the arrivals and IPs for the first fifteen days and repeat the planning step until we have a planning for our whole horizon. The next step will start with non-zero IPs because of what was already planned before.

The biggest advantage of this approach is that we plan many smaller parts of the total ADP, which probably results in lower runtimes. This way we can make plannings for longer periods in a shorter time.

A major disadvantage of this approach is the loss of optimality. With the random vessel selection for each period we can not guarantee we still have the optimal solution, since the order of the vessels is important for that as well. A “wrong” vessel selection might also result in false infeasibility, i.e. the solver can’t find a solution which is feasible for a certain part while actually, with an other vessel selection, there might be one.

Both these problems can be circumvented by running the total algorithm multiple times and selecting the best result from those. This is possible because of the lower runtime. It might still result in a non-optimal solution or false infeasibility, but the chance is reduced.

Currently we fixed the window size to thirty days with an overlap of fifteen days. This is chosen because we know for this window size we can solve the problem fast enough. Running the experiments with varying values for these parameters would be interesting. In subsection 5.2.3 we go further into changes to the way to run the rolling horizon approach.

4

Experiments

In this chapter we discuss the experiments ran with the formulations described in chapter 3. We evaluate how these formulations perform and also touch upon how this relates to real terminal situations.

The experiments are split up in three different parts: the single client formulations, the multi client discrete time formulation, and the rolling-horizon approach. We first describe a general setup used for all experiments followed by for each part a specific setup, some hypotheses, and an analysis of the results and some conclusions.

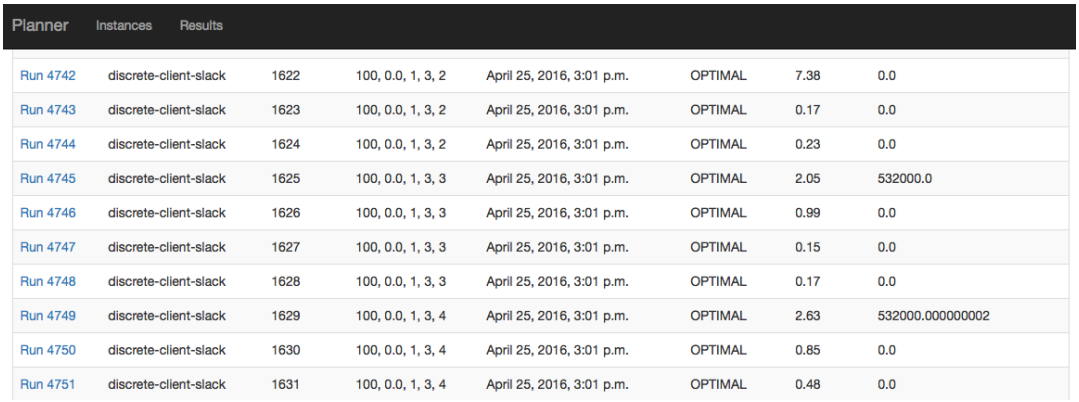
4.1. General setup

All experiments were run on a computer with a Intel(R) Core(TM) i5-3470 CPU @ 3.20GHz, 8 GB RAM, running Debian 8.2. The MILP-solver used is Gurobi 6.5.0.

For the organisation and exploration of we create a web application. In this application the solutions found by Gurobi can be explored. In Figure 4.1 we see the list of runs of Gurobi for the different solving methods we used. In this list we see some basic information for the runs, such as the instance that was solved with some parameters for that instance, which solving method was used, how long it took, and what was the Gurobi run status.

From this list we can navigate to a run detail page where we see more specific information about the run. An example is shown in Figure 4.2. On the detail view we see information of the problem Gurobi solved, such as the number of variables. Next to that we can also see the Gurobi run log, and the solution found. From here we can easily navigate to the Systems Navigator software (SN Web) where we can see the found ADP plotted as shown in Figure 2.1.

In order for the data to be available in SN Web we also created a tool that can copy the data from our web application to SN Web. This tool creates new scenarios in SN Web from the instances we have generated, and can easily be adapted for future instance generation scripts. Currently scenarios



Run 4742	discrete-client-slack	1622	100, 0.0, 1, 3, 2	April 25, 2016, 3:01 p.m.	OPTIMAL	7.38	0.0
Run 4743	discrete-client-slack	1623	100, 0.0, 1, 3, 2	April 25, 2016, 3:01 p.m.	OPTIMAL	0.17	0.0
Run 4744	discrete-client-slack	1624	100, 0.0, 1, 3, 2	April 25, 2016, 3:01 p.m.	OPTIMAL	0.23	0.0
Run 4745	discrete-client-slack	1625	100, 0.0, 1, 3, 3	April 25, 2016, 3:01 p.m.	OPTIMAL	2.05	532000.0
Run 4746	discrete-client-slack	1626	100, 0.0, 1, 3, 3	April 25, 2016, 3:01 p.m.	OPTIMAL	0.99	0.0
Run 4747	discrete-client-slack	1627	100, 0.0, 1, 3, 3	April 25, 2016, 3:01 p.m.	OPTIMAL	0.15	0.0
Run 4748	discrete-client-slack	1628	100, 0.0, 1, 3, 3	April 25, 2016, 3:01 p.m.	OPTIMAL	0.17	0.0
Run 4749	discrete-client-slack	1629	100, 0.0, 1, 3, 4	April 25, 2016, 3:01 p.m.	OPTIMAL	2.63	532000.000000002
Run 4750	discrete-client-slack	1630	100, 0.0, 1, 3, 4	April 25, 2016, 3:01 p.m.	OPTIMAL	0.85	0.0
Run 4751	discrete-client-slack	1631	100, 0.0, 1, 3, 4	April 25, 2016, 3:01 p.m.	OPTIMAL	0.48	0.0

Figure 4.1: Screenshot of application created to browse generated results. List of all results.

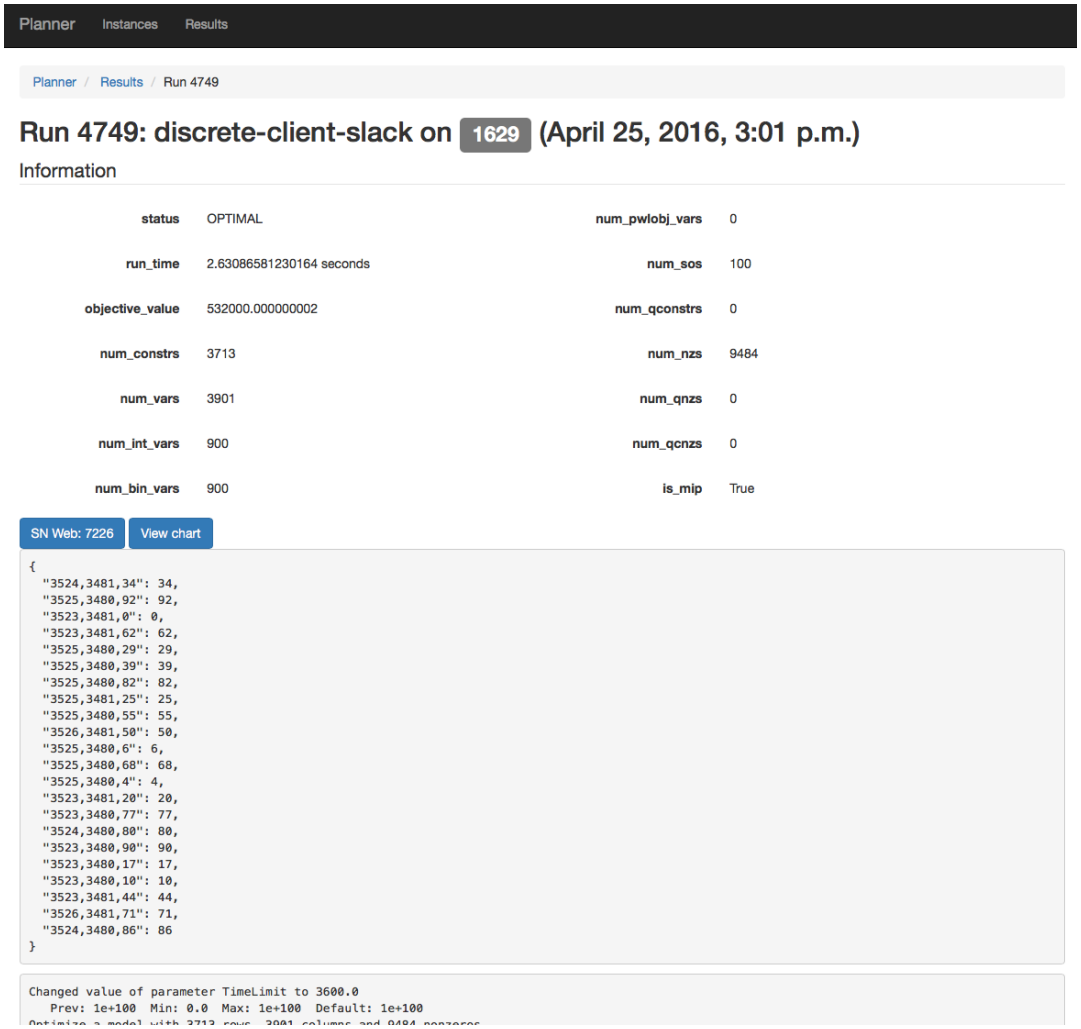


Figure 4.2: Screenshot of application created to browse generated results. Details of a result.

are created by hand in SN Web, so this tool can be used to automate this process in future projects.

The tool runs at the end of all our Gurobi runs to import the found solution. From there we can trigger the simulation software (described in subsection 4.3.2).

The data produced by both solving the problem and the simulation step is combined in R for analysis. This is achieved by querying both the database for the web application (PostgreSQL) and SN Web (SQL Server).

4.2. Single client formulations

The first experiments we run are the single client formulation experiments. In these experiments we evaluate the run time of Gurobi for the different formulations of time, discrete time, event-based, and piecewise, to determine what is a good way to model time in a MILP for the problem of creating an ADP for an LNG receiving terminal. We also look at what parameters, e.g. the number of different vessel types that arrive at a terminal, influences the run time the most. Based on the observations made from these experiments and some general observations on the formulation themselves we can choose which of the formulations we extend further.

4.2.1. The setup

For this experiment we have generated 625 problem instances exploring the effect of different values for four parameters: horizon, number of vessel types, tank ratio, and number of send-out parts. These parameters and their values are explained here.

The horizon (H) is the time period for which the planning should be made. In these experiments this is equal to the duration of the single contract the instance consists of. The values used for the creation of the instances are: 20, 30, 50, 100, 183, and 366 days.

Next is the number of different vessel types that arrive at the terminal during the period for which we are planning. Instances are created for one, two, and three different vessel types. The vessel types used in the instance are than randomly drawn from the following set: *Conventional* (138000 m^3), *Q-Flex* (210000 m^3) and *Q-Max* (266000 m^3). These are based on existing LNG vessels that are currently in use [7] with a variety in sizes. When we have an instance with only one vessel type, this could thus be one of these three.

The tank ratio (r) is used to determine the total tank capacity for the terminal in the instance. The tank capacity is r times the capacity of the largest vessel type in this instance. This way the contents of largest vessel always fit in the tank. Instances are generated with a tank ratio of 1.2, 2, and 5 to see the influence of different sizes for tanks. With a tank ratio of 1.2 there is very little spare space in the tank while with a ratio of 5 there probably never will be space limitations.

The last parameter is the number of send-out parts. A send-out part is a period in which the send-out for a contract is constant. For instance, when we have one send-out part, there is a constant send-out over the whole duration of the contract. With two send-out parts there is one change in send-out somewhere in the duration of the contract. We create instances consisting of one, two, and three parts. From interviews with Systems Navigator we learned that in real life scenarios the send-out changes happen, but not very often. Because of this we want to see the influences of these changes, but not go to high, since this does not happen often. When creating the new parts, each time the largest period is taken and then split up in two. The total send-out over the two new parts are equal to the total send-out of the original part.

The send-out rate for each of the parts is randomly generated and should be between 10000 m^3/day and 20000 m^3/day. When we have determined an initial total send-out we pick random vessels from the selected vessels for this instance until the sum of all vessel capacities exceeds the total send-out. The send-out rate is then adjusted so that the total send-out is equal to the total amount of LNG that is delivered by all vessels.

4.2.2. Hypotheses

As discussed in subsection 3.1.2, while piecewise was designed initially to perform better and faster than the discrete time formulation, and to use continuous time arrivals, the implementation of the piecewise linear constraints led to many extra variables with SOS2 constraints per arriving vessel. We therefore expect the piecewise linear formulation to be solved much slower than the other two formulations. This leads us to the following hypothesis:

	Horizon	# of vessel types 2	Tank ratio	# of send-out parts
Discrete time	++	++	+/-	+/-
Event-based	++	+	+/-	++

Table 4.1: Influence of parameters on run time

Hypothesis 1 *The piecewise linear formulation is the slowest of the single client formulations.*

When comparing the discrete time and event-based single client formulations based on run time we expect them to be similar when compared over all instances. However, we do expect differentiation on run time for the different parameters. In Table 4.1 the expected influence from the different parameters used for creation of the instances on the run time are shown. “++” means a lot of influence, “+” it does influence the run time but other factors are more important, and “+/-” there is no significant relation between this parameter and the run time.

The horizon has a big impact on run time for both formulations, because from this the size of the instance follows; both the total number of vessels that arrive during the duration of the contract as well as the total send-out have a strong relation with the horizon due to the way of creating the instances.

Since the experiments run with the objective to minimise the amount of LNG in the tank, the tank ratio constraint is expected to not be of great influence on the run time on none of the formulations.

For the discrete time formulation we expect that, after horizon, the number of vessel types has the most influence on the run time, since for each vessel type a set of new variables and constraints for the arrivals are created. Also the symmetry between solutions increases with multiple vessels, the order in which different vessel types arrive does not have much influence on the objective function, so when we switch two vessels for the solver it would be almost the same. The number of send-out parts and the tank ratio are not expected to have a strong correlation to the run time.

The run time for the event-based formulation, we expect, is mostly influenced by the number of send-out parts. This is due to that for each send-out part we create a new fixed event point on which the send-out rate is changed. Between each set of fixed event points we create more event points than we expect to need to give the solver some flexibility on when to plan the vessels, this leads to a higher number of total event points. The number of vessel types is also expected to have a correlation with the run time, but this is smaller than the number of send-out parts.

This leads us to the following hypothesis:

Hypothesis 2 *For the single client formulations, discrete time and event-based are overall similar in run time. Discrete time generally is solved slower when there are more different vessel types, while event-based is solved slower when there are more send-out parts.*

4.2.3. Results and conclusions

When analysing the results we learned that the total number of vessels and the total send-out are better to predict the run time than the horizon. Also we learned that the horizons greater than 100 days were unable to be solved within the 1 hour time limit. We therefore only analyse the instances with an horizon smaller or equal to 100, for the piecewise linear formulation we only analyse for 50 and lower.

In Figure 4.3 run time is set out on a logarithmic scale against the the total nummer of vessels, with a jitter (distortion) function on the x-asis to better see the point cloud, for the different formulations each in a different colour, the runs leading to a time limit are not drawn. On a first glance we can see that indeed the piecewise linear formulation is slower than the other two formulations. We also can obtain a general idea that the event-based and discrete time formulation perform simular. We explore both of these hunches in more detail. First we explore the limitations for the piecewise linear formulation, and then we make a more thorough comparison between the event-based and discrete time formulation. This comparison is used to determine which of the formulations to use when expanding to multiple clients and lending and borrowing.

Piecewise linear

With the piecewise linear formulation we only ran the experiments with a horizon up to 50 days. When running for more than 50 days Gurobi crashed frequently. It is suspected that this was caused by excessive memory usage. In Figure 4.4 box plots for the run time with different total amounts of vessels

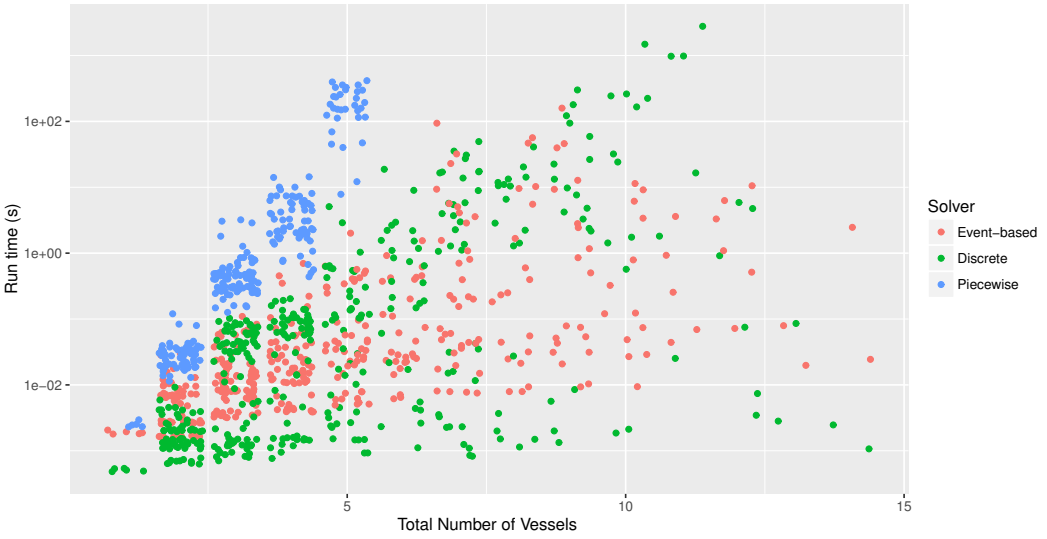


Figure 4.3: Run time for single client formulations

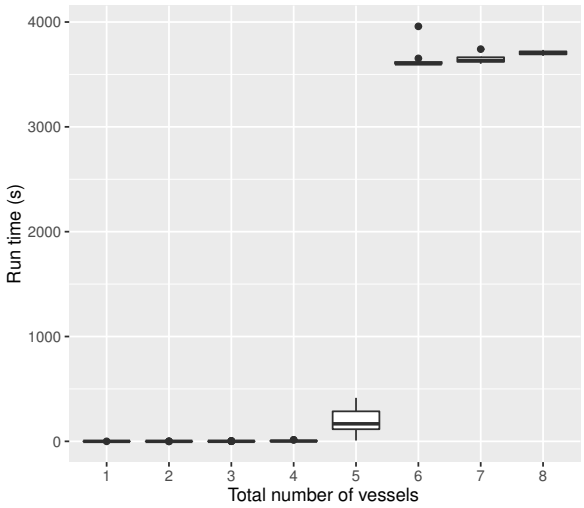


Figure 4.4: Run time for piecewise linear, we cannot solve for more than 5 vessels

are shown. From this plot we can see that instances with up to five vessels can be solved within the timelimit of one hour using the piecewise linear formulation. We can accept Hypothesis 1 en conclude that the piecewise linear formulation is not usable in real life situations, since most terminals have more than five vessels per year, and we can also create a schedule for up to five vessels manually.

Discrete time and event-based

The next step is to analyse the run times of the discrete time and event-based formulations. The results of this analysis is used to determine which of the formulations to use when we expand the formulations with more constraints and options. We first determine if one of the formulations is significantly faster than the other and then analyse which parameters have the most influence on run time for both formulations.

In Table 4.2 we can see how the run times for both formulations are distributed. We see that the mean of the run time for the discrete time formulation is lower than that of the event-based formulation. To determine if the run times for the formulations are statistically significant different we do a paired t-test with

H_0 : The difference in means is equal to 0
H_1 : The difference in means is not equal to 0

Formulation	Min.	1st Qu.	Median	Mean	3d Qu.	Max.
Discrete time	0.000	0.002	0.031	38.850	0.186	3600
Event-based	0.002	0.007	0.020	10.550	0.071	3600

Table 4.2: Distribution of run time (in seconds) for Discrete time and Event-based formulations

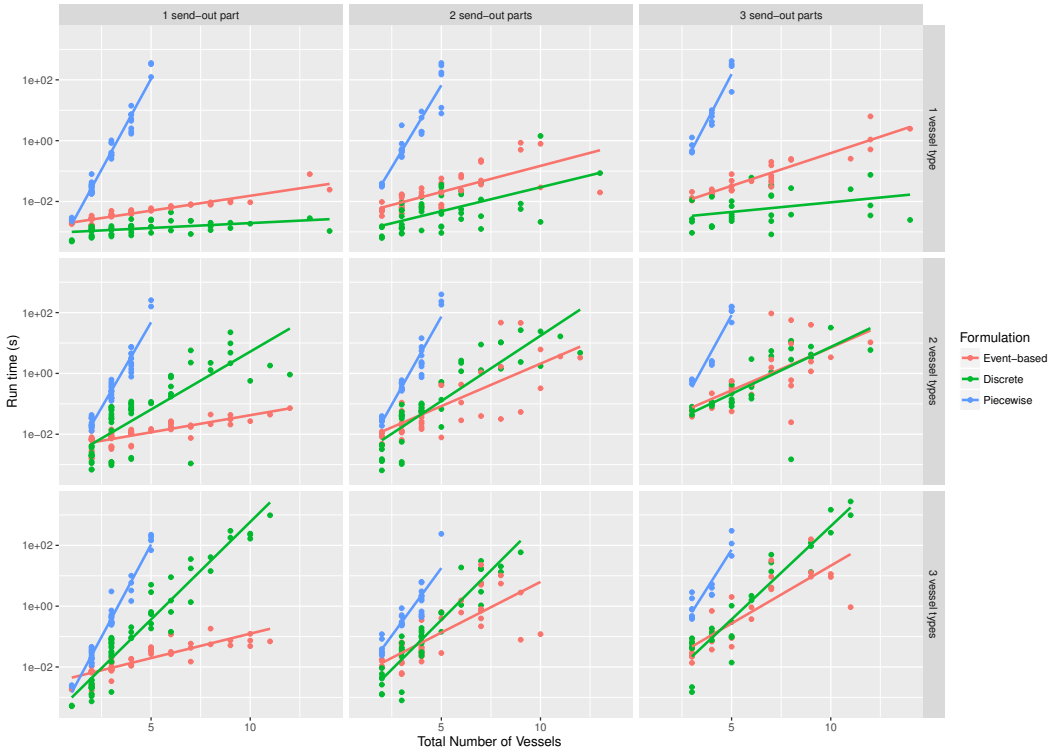


Figure 4.5: Run time for single client formulations, facetted on # send-out parts and # vessel types

This results in a t of 1.7333 and a p-value of 0.08381, which is greater than $\alpha = 0.05$ and thus we cannot reject H_0. We therefore conclude that there is no significant difference in run time between both formulations.

In Figures 4.5 and 4.6 we explore the influence of the different instance creation parameters on the run time. We do this by creating different plots for different number of vessel types, number of send-out parts, and tank ratio. We plot the log of the run time against the number of vessels in the instance. In all the sub-plots we see the piecewise linear formulation is the slowest.

Figure 4.5 shows from left to right increasing number of send-out parts and from top to bottom increasing number of vessel types. When comparing the plots in one row, we see that the discrete time (green) lines are very similar, but the lines for the event-based appear to be more steep from left to right. From this we learn the number of send-out parts influences the run time for the event-based formulation, but has not so much influence on the discrete time formulation. Per column we see it the other way around, more steep lines for discrete time and similar for event-based. Which means more influence of the number of vessel types on the run time for the discrete time formulation than on the event-based formulation.

In Figure 4.6 we see the tank ratio increasing from left to right and again the increasing number of vessel types from top to bottom. The image we get per column is comparable with that of Figure 4.5, we still see the influence of the number of vessel types on the discrete time formulation. In the rows however we see not that much difference, the tank ratio does not seem to influence the run time at all.

Based on the plots in the figures we get the idea the tank ratio does not influence the run time much. But the send-out parts and and the number of vessel types do influence the run time for event-based and discrete time. We will test these ideas further.

To see if the influences of parameters we have seen in the plots described above are true, we fit a

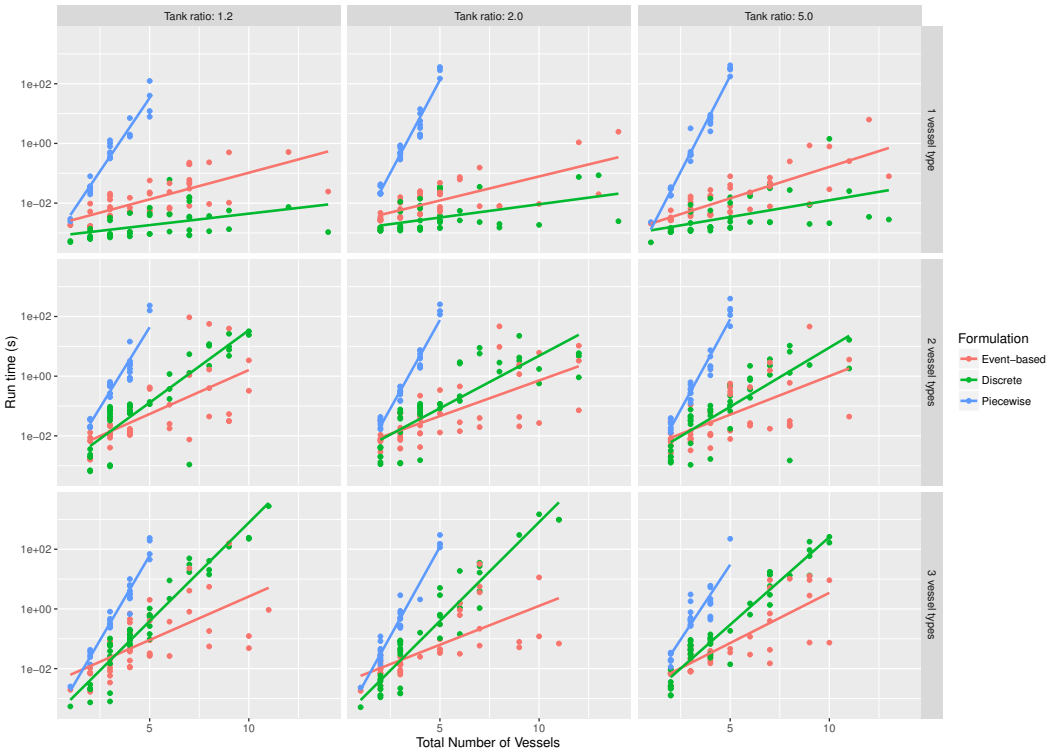


Figure 4.6: Run time for single client formulations, facetted on tank ratio and # vessel types

linear model on the log of the run time. The log of the run time is used because preliminary analysis indicates that the relationships between most parameters and run time appears to be exponential. In Table 4.3 the results for fitting the parameters individually are shown. In the table the R^2 value, t-value, and p-value are shown. R^2 is a measure for how close the data is to the fitting and is between 0 and 1 with values closer to 1 being a better fit. The t and p value are that of a t-test.

For the discrete time formulation total send-out has the highest R^2 and it is therefore the best predictor for the run time. We also see that, as expected, horizon and total number of vessels a have significant predictive value. With the event-based formulation we see that, when used individually, horizon, total number of vessels, and number of parts are the best predictors. These are not quite as good with the discrete-time formulation. The tank ratio does not seem to have a exponential relation with the runtime.

In Table 4.4 the results for fitting a linear model using all parameters together are shown. Here an adjusted R^2 value is shown. The R^2 is adjusted to counteract the fact that adding more parameters always leads to a better fit. Next to the parameters in this table we also see the intercept, which is the intersection with the y-axis. For each parameter we then see the coefficient, the range of values for the parameter, and following from those two the range for the influence on the run time follows. Each coefficient is multiplied by the corresponding parameter and then they are summed together to get a prediction of the run-time.

For the discrete time formulation we see total send-out has the highest coefficient relatively to the range of the parameter followed by number of vessel types. Based on this we confirm the total send-out and the number of vessel types are the most influencing on the run time.

For the event-based formulation the number of send-out parts and total number of vessels have the highest coefficients relatively. Therefore we can also confirm the idea that the number of send-out parts are influencing the run time much.

We have seen that the run times for discrete time and event-based are not significantly different, that the number of vessel types has a big influence on run time for discrete time, and that the number of send-out parts has a big influence on run time for event-based. We therefore accept Hypothesis 2: *For the single client formulations, discrete time and event-based are overall similar in run time. Discrete time generally is solved slower when there are more different vessel types, while event-based is solved*

Parameter	R^2	t-value	p-value
Discrete time			
Horizon	0.3915	16.0627	3.504e−45
Tank ratio	2.010e−05	0.08979	0.9285
# Vessel types	0.2686	12.1345	4.453e−29
# Parts	0.05714	4.9299	1.207e−06
Total # vessels	0.3648	15.1755	2.000e−41
Total send-out	0.6408	26.7489	3.393e−91
Event-based			
Horizon	0.4028	16.4681	6.1379e−47
Tank Ratio	1.6733e−06	−0.0259	0.9793
# Vessel types	0.07677	5.7817	1.4865e−08
# Parts	0.3739	15.4943	8.6362e−43
Total # vessels	0.4193	17.0364	2.2112e−49
Total send-out	0.2664	12.0829	6.9039e−29

Table 4.3: Linear model fitting with parameters individually on run time for single client formulations

Parameter	Coefficient	Range (parameter)	Range (run time)	t-value	p-value
Discrete time (Adjusted $R^2 = 0.7764$)					
(Intercept)	−11.73	NA	NA	−33.906	< 2e−16
Horizon	0.01347	[20, 100]	[0.2694, 1.347]	2.514	0.0123
Tank Ratio	−0.01944	[1.2, 5]	[−0.0972, −0.02333]	−0.385	0.7003
# Vessel types	1.501	[1, 3]	[1.501, 4.503]	12.274	< 2e−16
# Parts	0.8534	[1, 3]	[0.8534, 2.560]	6.856	2.7e−11
Total # vessels	0.1575	[1, 14]	[0.1575, 2.205]	2.545	0.0113
Total send-out	1.720e−06	[266000, 6186000]	[0.4575, 10.64]	11.583	< 2e−16
Event-based (Adjusted $R^2 = 0.7294$)					
(Intercept)	−9.604	NA	NA	−38.272	< 2e−16
Horizon	3.729e−03	[20, 100]	[0.07458, 0.3729]	0.959	0.338198
Tank Ratio	−0.01334	[1.2, 5]	[−.06670, −0.01601]	−0.364	0.715967
# Vessel types	0.6575	[1, 3]	[0.6575, 1.973]	7.408	7.72e−13
# Parts	1.494	[1, 3]	[1.494, 4.482]	16.542	< 2e−16
Total # vessels	0.3089	[1, 14]	[0.3089, 4.325]	6.879	2.34e−11
Total send-out	4.047e−07	[266000, 6186000]	[0.1077, 2.503]	3.757	0.000198

Table 4.4: Linear model fitting with parameters combined on run time for single client formulations

Client 0	Client 1	Client 2	Client 3
2	2	2	2
2	1	2	2
4	1	1	1
2	.5 (short)	.5 (short)	2

Table 4.5: Yearly Quantity per client (in MTPA)

Conventional (% of total)	*Q-Flex* (% of total)	*Q-Max* (% of total)
100	0	0
70	30	0
50	50	0
70	20	10
50	40	10

Table 4.6: Vessel distribution

slower when there are more send-out parts. We should add that, from the parameters related to the problem size (horizon, total number of vessels, and total send-out), the total send-out and the total number of vessels are the biggest influence on the discrete time and event-based formulations.

4.3. Multi client formulation

Next to the single client formulations in section 3.2 a multi client formulation is introduced, which is an extension of the discrete time formulation from the previous section. In this section we discuss the experiments and results for this formulation. For these and following experiments we also introduce a simulation step. In this simulation step we run 50 replications of a simulation model created by Systems Navigator to evaluate how the schedule created by the solver based on the formulation will perform in real life. We will discuss the simulation model further after discussing the instance generation for this experiments.

4.3.1. Instance generation

In the generation of the instances for this experiment more research into real life instances is done. From interviews at Systems Navigator we learned more about the typical features of how terminals currently operate. We talked about the size of the tanks a terminal has, the number of clients a terminal has and typical send-out patterns and the vessels that arrive at a terminal.

A typical terminal has two or three tanks with a volume of 160.000 or 180.000 m^3. Therefore we generate instances with a tank of size 340.000, 500.000 and 540.000 m^3.

Typically a terminal has between one and four clients and most clients have a yearly contract without many send-out changes. The possibility for multiple send-out rates is available, but not often used. Sometimes there are contracts that run for a shorter amount of time. In total a terminal has a send-out rate of between four and eight Million Tonnes Per Annum $\sim 2.320.117m^3$ LNG (MTPA). In Table 4.5 the send-out distributions used in instance generation is shown, the clients in the terminal is increased from one to four. When the total for the terminal is lower than four it is doubled for that instance.

In the first experiment the fleet-mix, the total available vessels, was completely random, although in fact there are not as many *Q-Max*s as there are *Conventional*s. Typically at most 10 percent of the arriving vessels is a *Q-Max*, also the most common vessel is the *Conventional*.

To incorporate this in the instance generation, vessel distributions are predefined for the whole terminal, see Table 4.6. The vessel are generated according to this distribution and than randomly assigned to a client. The total send-out of a client is adjusted to the total amount of LNG delivered to the terminal

The horizons are chosen the same as for the first experiment, being 20, 30, 50, 100, 183 and 366 days.

4.3.2. Simulation model

For determining how well the found solutions would perform when applied in practice we use a simulation model made in Rockwell's Arena by Systems Navigator. This simulation model simulates the operation of the ADP. The simulation model introduces small changes to the original planning to see how the ADP can cope with these variations. For each ADP we generate we will run 50 replications.

The first change in the schedule is that 10 percent of the time the send-out will be up to 5 percent higher, 5 percent it will be 5 percent less, and 5 percent there will be no send-out at all at a certain moment in time. It is important to notice that since the instances are created in such a way that the originally planned send-out can be exactly achieved we might expect to see some achieved send-out losses in the replications where the simulation tries to send-out more than the original plan.

Next to that, the model is configured to have 25 percent of all vessels arrive on time. In the rest of the cases a vessel will arrive between 2 and 10 hours late.

The simulated ADP is scored on Key Perfomance Indicators (KPIs). We will score the ADPs on lending and borrowing, and on missed send-out. For lending and borrowing this will be measured by the total amount LNG borrowed and the maximal amount borrowed on a certain day. For the missed send-out we look at the average percentage and the variance in missed send-out over all replications. The variance should give a measure on how robust a schedule is. If we have a high variance this means small changes in the schedule will have large implications.

4.3.3. Experiments and hypotheses

The experiments will be split up in three parts. First we will look into the run time of the solver on this formulation with an objective function that minimizes the amount of LNG in the tank above a treshold plus the amount of lending and borrowing ($\Pi_{\text{tank}} + 100\Pi_{LB}$) without allowing missed send-out ($s^{SO}_{g,t} = 0$). After that we will run experiments with different objective functions and compare the run time and simulation results so see what are good objective functions. Finally we delve deeper into one of the objective functions to get more insight into the parameter in this objective.

Determining the runtime

The first experiment we run is to get an indication of the solving time for this formulation. We test this with a comparable objective function as the single client discrete time formulation. The objective function is as follows:

$$\Pi_1 = \Pi_{\text{tank}} + 100\Pi_{LB} \quad | \quad s^{SO}_{g,t} = 0 \quad \forall g \in G, t \in T$$

The weight for the lending and borrowing objective is arbitrarily chosen to be larger since it is more important to minimize the amount of borrowing.

Since this formulation is an extension of the one tested in the previous section and uses a comparable objective we expect the solver to be a little slower than with that formulation, due to the fact that adding lending and borrowing complicates the problem a lot. Next to that there will also be more vesels arriving at the terminal. Therefore we come to the following hypothesis:

Hypothesis 3 *The multi client formulation with the objective to minimize $\Pi_{tank} + 100\Pi_{LB}$ will not be able to solve instances with a horizon of 100 days within the 1 hour time limit.*

Comparing objective functions

In the next experiment we will also run the solver with different objective functions to determine what are good objectives for this problem, we evaluate on both run time and simulation results. Next to Π_1 defined before we define the following objective functions:

$$\Pi_2 = \Pi_{SO}$$
$$\Pi_3 = \Pi_{\text{tmax}} + 1000\Pi_{SO}$$
$$\Pi_4 = \Pi_{LB} \quad | \quad s^{SO}_{g,t} = 0 \quad \forall g \in G, t \in T$$

The problem definition for each of the objective functions is the same, but each of the objectives will lead to other results. The first (Π_1) minimises IPs above a certain threshold ($IP_{\min} = 30000$) while also trying to minimize the amount of borrowing within the terminal. We do not allow for missed send-out

when using this formulation. Minimising IPs might collide with minimising lending and borrowing, due to the fact that the latter is ensuring a certain tank level to have enough LNG to lend.

In Π_2 the solver will try to minimize the amount of missed send-out. It will not take in account the individual tank levels per client or the amount of lending and borrowing. This probably results in faster solving times, but the found solutions can be expected to score random in the simulation, since the simulation KPIs are not weighted in the objective function.

Objective function Π_3 minimises the IPs per client below a certain threshold ($IP_{\max} = 30000$) weighted by the amount of send-out by that client, i.e. the solution is punished if the client that had send-out does not have at least $IP_{\max}$ LNG in store. Π_3 does allow for missing send-out, but this has an (arbitrarily chosen) higher weight to mark the importance. The solver probably takes more time to find the optimal solution than with Π_2, but since solutions are punished for not having enough LNG lending and borrowing and empty tanks will happen less often in the simulation.

The last objective function we test in this experiment (Π_4) is designed to purely minimize the amount of lending and borrowing. It does not allow for missed send-out. We expect it to be comparable on run time with Π_3, but expect it will perform less on the simulation due to the fact the solver not trying to keep a small backup of LNG.

Reviewing the above made statements about the different objective function we come to the following two hypotheses:

Hypothesis 4 *Solving with the objective of minimising the amount of missed send-out (Π_2) will be the fastest of the proposed objectives. However, it will lead to solutions with a lot of lending and borrowing and the solutions will not be verry robust.*

Hypothesis 5 *The best simulation results will come from minimising the amount of LNG bellow a certain treshold (Π_3), but solving with this objective function will be relatively slow.*

Since, at least for Π_1, the solver often times out for a horizon bigger than 50 days, we decided to run this experiment only for a horizon of 50 days. This also reduces the amount of variation between the instances, e.g. the amount of send-out does vary less for the same horizon.

4.3.4. Results and conclusions

After running the experiments described in the previous section we can process the results. We first ran an experiment to determine the maximum horizon we want to use for the second experiment. The second experiment is to compare the different objective functions with each other.

In Figure 4.7 we show percentages of run statuses for all runs of the solver with objective function Π_1 split up per horizon. The infeasible instances are instances that have vessels arriving with a larger capacity than the tank capacity. From the image we learn that until a horizon of 50 days we can solve most of the instances within the 1 hour time limit, from 100 days on we get more and more time limits. Therefore we accept Hypothesis 3.

For the rest of the rest of the experiment, as mentioned in the previous section, we only look at a horizon of 50 days. We first do an analysis on the run times for different objective function, followed by the comparison on simulation results.

For the comparison of the run time of the solver with the different objective functions we create Figure 4.8. In this figure we show the run time on a log scale set out against the number of clients. The line is drawn through the average run time per objective function and standard error bars are shown.

From this figure we learn that Π_2 solves indeed, on average, the fastest. The runtime also not that spread out as indicated by a small standard error. For the rest of the objective function the spread, or standard error, is relatively large. Π_1 is more or less constant over the number of clients. Π_3 is the slowest for three and four clients on.

The next step is the comparison on simulation results. These results are based on the average of 50 replications of the simulation model, as described in subsection 4.3.2. We look at missed send-out and total amount borrowed. No instances were infeasible for a horizon of 50 days, so this does not influence any difference between with or without allowing for missed send-out.

In Figure 4.9 the average missed send-out as percentage of the total send-out is shown for each of the objective functions on a log scale. We see Π_1 has the largest range of outcomes, an explanation for this is that when we minimize the the tank contents this makes the schedule a lot more prone to

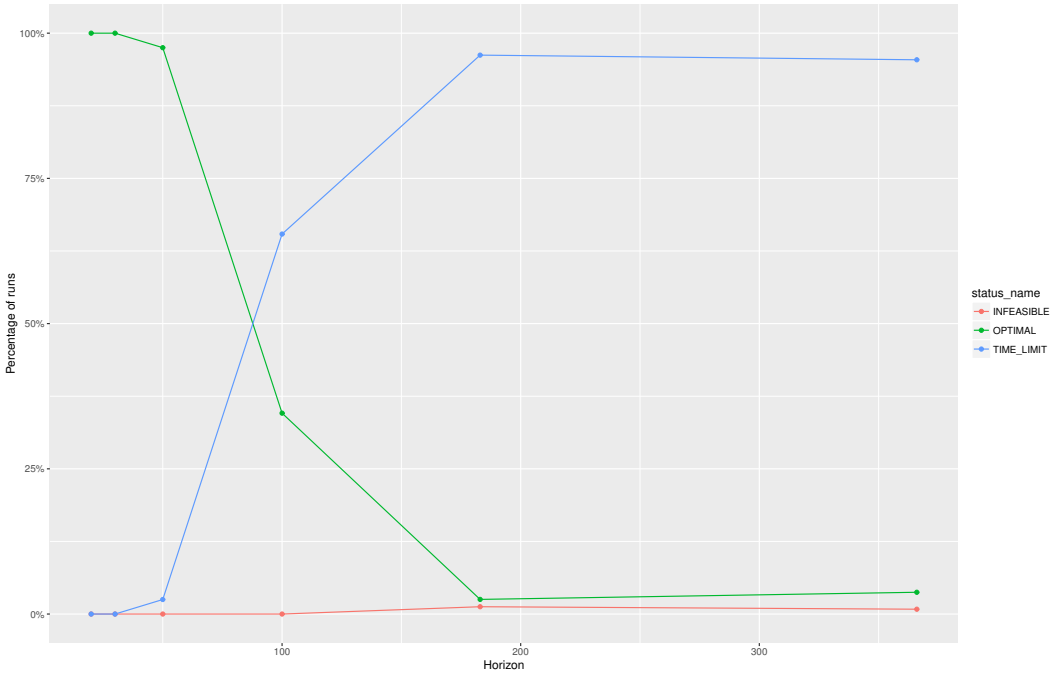


Figure 4.7: Run statuses for runs using the multi client formulation with objective function Π_1

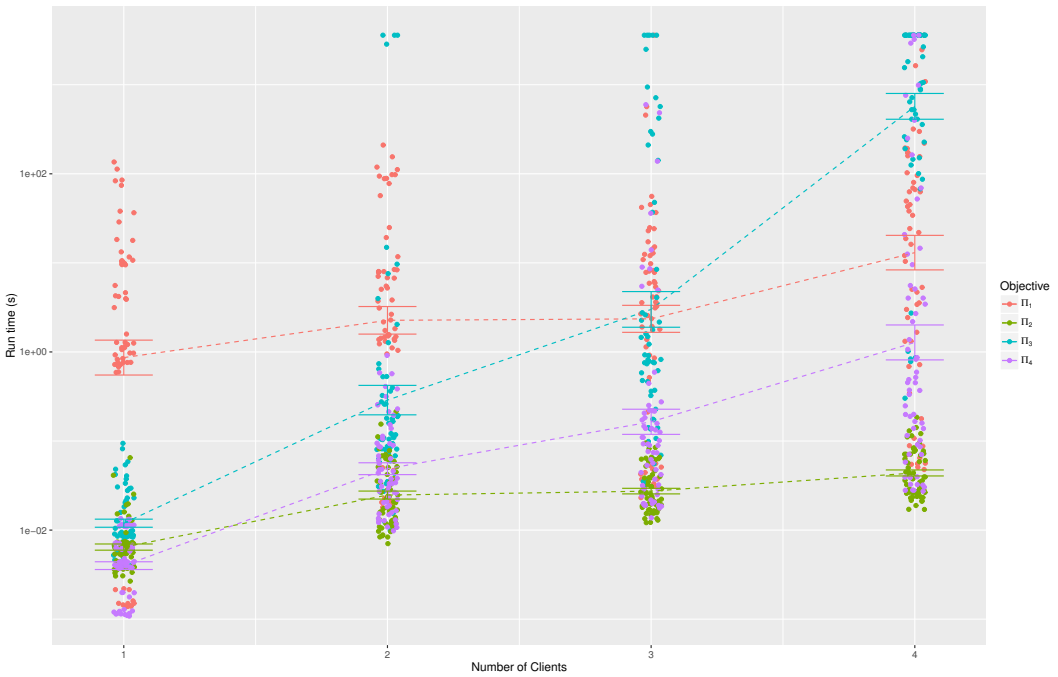


Figure 4.8: Runtime for multi client formulation with different objective functions set out against number of clients

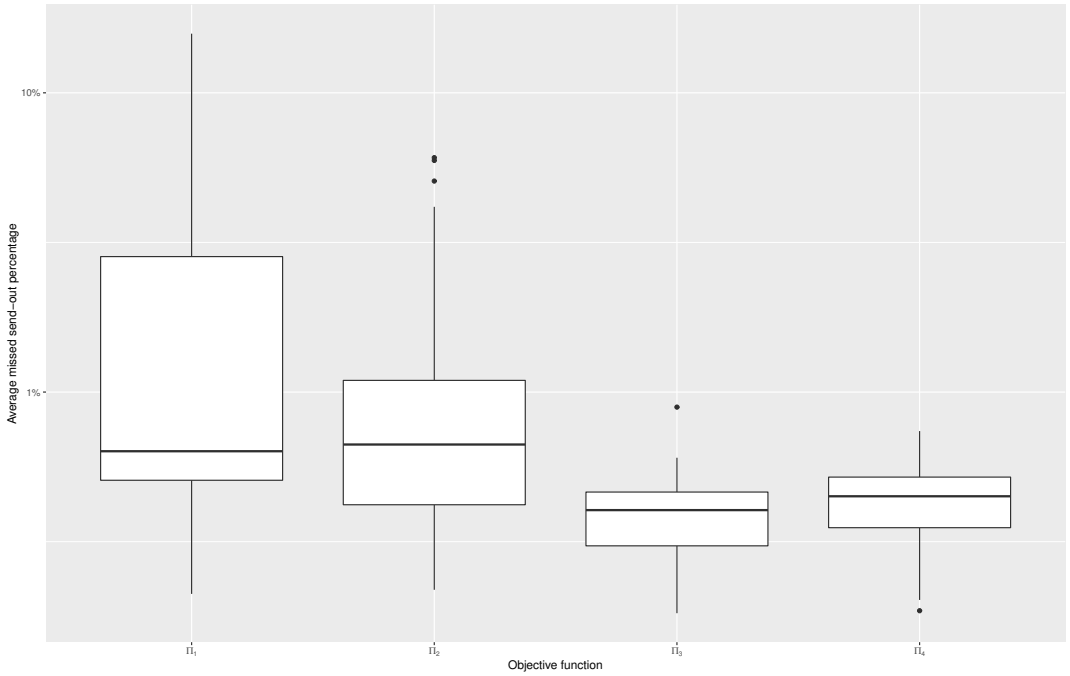


Figure 4.9: Average missed send-out as percentage of total send-out on a log scale for different objective functions in the multi client formulation

delaying vessels. Furthermore we see Π_3 has the lowest average missed send-out, though Π_4 is only slightly higher.

Figure 4.10 shows the average amount of total borrowed LNG as percentage of the total send-out. As well as the amount of LNG that was not send-out because there were no lenders available, so we can take in consideration that amount as well.

We see Π_1, Π_3 and Π_4 have comparable values for the amount borrowed percentage, while Π_2 is a lot higher, which was expected since no consideration for lending and borrowing is in the objective function. If we also take the missed send-out because there were no lenders in consideration we see Π_1 is also not that good, i.e. there is not that much lending and borrowing because there is less send-out due to lenders not being available.

The results for average missed send-out and total amount borrowed show Π_3 is performing the best within the simulation, with Π_4 as a close second. However, if we also consider runtime Π_4 is the best choice. We therefore carefully accept Hypothesis 5 and add to that Π_4 is comparable but faster.

The other two objective functions, Π_1 and Π_2, have worse simulation results. They are comparable on simulation, but not on run time. Since Π_1 is also one of the slower solvers it is not advised to use this objective function. Π_2 however can be useful if finding a solution fast is more important than finding the solution that works best in execution. We accept Hypothesis 4.

4.3.5. Explore best values for $IP_{\max}$

During experimentation we found out Π_3 performs the best in the simulation. Since this objective is dependent on the parameter $IP_{\max}$ we also explore how different values for this parameter influence the outcome of the solver.

Therefore we setup a third experiment in which we will run the solver with Π_3 with $IP_{\max}$ having the values 0, 50.000, 100.000, 150.000, and 200.000. We expect the simulation results to be better with an higher $IP_{\max}$, since when it is higher the solver will try to have a larger buffer per client. This way less lending and borrowing should occur. Also the missing of send-out should happen less.

To get a fair comparison and not have to much influence of factors other than the $IP_{\max}$ we will run this experiment only for the instances with tank size 340.000 m^3.

We pose the following hypothesis:

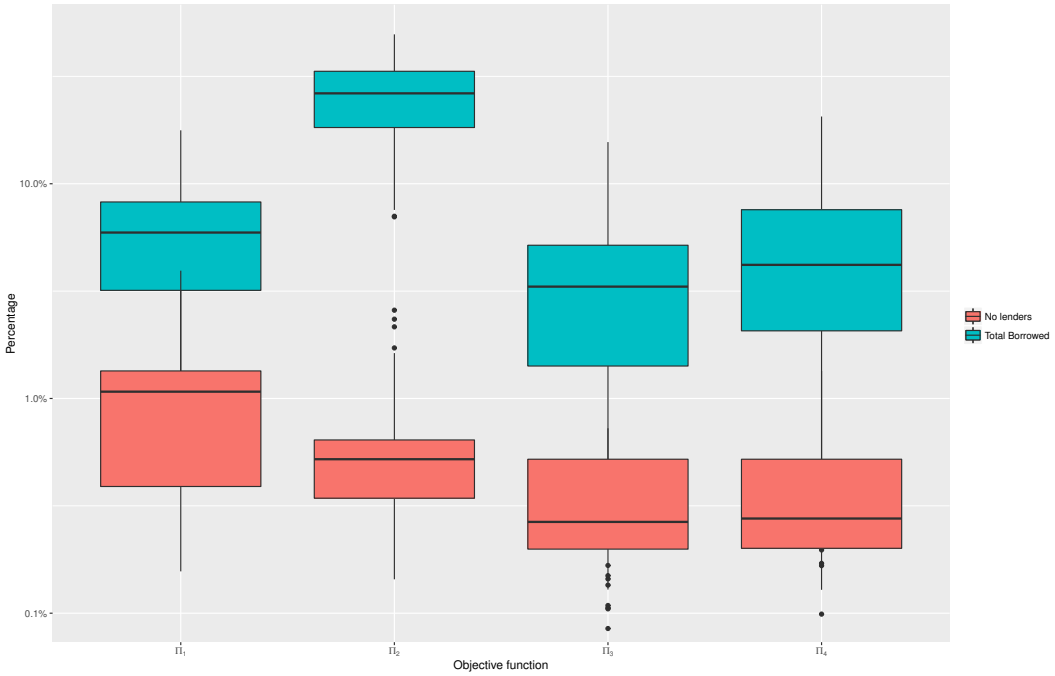


Figure 4.10: Average total borrowed LNG and LNG not send-out because there were no lenders available as percentage of total send-out on a log scale for different objective functions in the multi client formulation

Hypothesis 6 *When solving* Π_3 *with increasing value of* $IP_{\max}$ *the higher values will result to better simulation results.*

We run experiments with the same instances as before. In Figure 4.11 we see boxplots of the average missed send-out for increasing $IP_{\max}$. Note that this figure has a normal y-scale instead of a logarithmic. We see the the average missed send-out is lowest at 100.000 m^3 and from then increases. The differences are not very big.

When we look at Figure 4.12, in which the average borrowed percentage is shown for increasing $IP_{\max}$, we see a similar pattern. Now the lowest point is at 50.000 m^3.

The previous observations are in conflict with Hypothesis 6. We therefore reject this hypothesis. The difference between hypothesis and results is expected to be due to that in formulating the hypothesis the effect that when all clients have a higher $IP_{\max}$ the sum of these thresholds exceeds the tank capacity is ignored. When we try to minimise so that clients do not go below their threshold we face an impossibility since always at least one client goes below its threshold, this effect becomes larger when we exceed the tank capacity more.

4.4. Rolling horizon and heuristic

The last method introduced in section 3.3 is a rolling horizon approach. This approach is introduced to lower the run time for the multi client formulation to allow for longer horizons to be planned. In this section we compare results from this rolling horizon approach with the results from the normal multi client formulation, to see how much speed increase we get and how much optimality is lost.

In order to answer Research Question 6: “How does the Mixed Integer Linear Program perform compared to the Systems Navigator Heuristic?” we also compare the rolling horizon approach with the algorithm written by Systems Navigator. This algorithm makes use of a re-order point (ROP) heuristic and a fixed order of the arriving vessels per client. An ROP is the amount of days before hitting a client tank bottom the next vessel for that client. The order of the vessels is randomised at the start of the run of the algorithm. For the comparison we choose an ROP equal to 50.000 m^3 divided by the send-out rate, e.g. if the send-out rate is 10.000 m^3/day the ROP is 5 days.

For the experiments in this section we use the same instances in as the previous section so we can compare the results, except that for the new experiments we start with randomly determined start

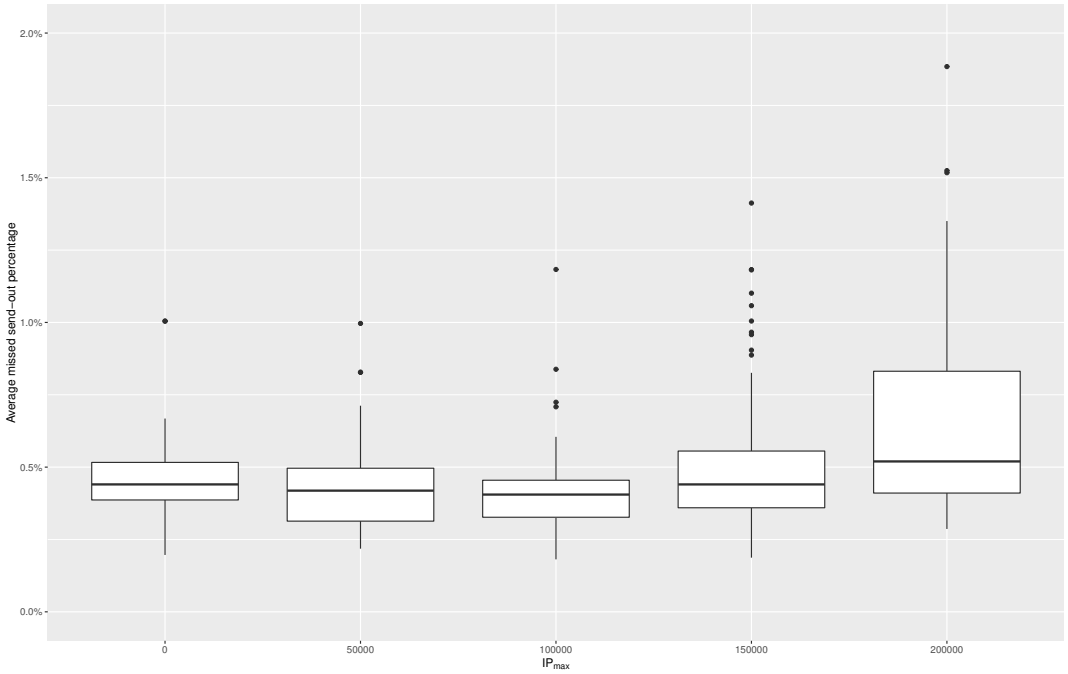


Figure 4.11: Average missed send-out as percentage of total send-out for increasing $IP_{\max}$

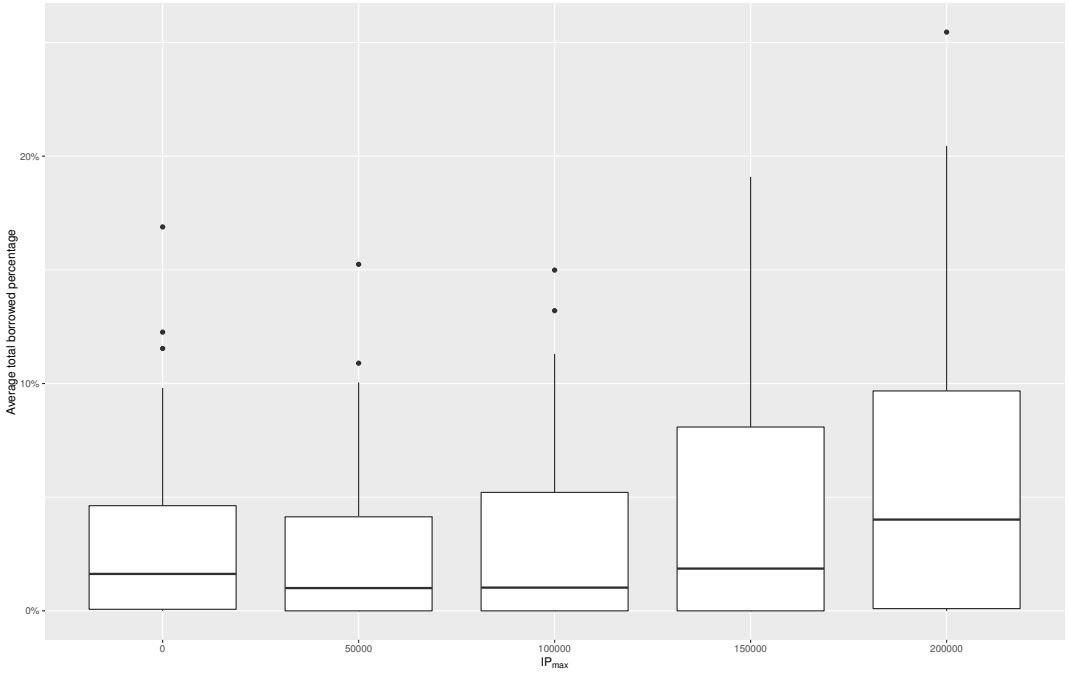


Figure 4.12: Average total borrowed LNG as percentage of total send-out for increasing $IP_{\max}$

inventory positions, which are the same for both the rolling horizon approach and the heuristic. There are 240 instances per horizon and we solve each instance once.

4.4.1. Hypotheses

The experiments for this section run with three different solvers. We use the best performing objective function for the multi client formulation (Π_3), the rolling horizon approach, and the heuristic created by Systems Navigator. These solvers are compared on three different aspects: the runtime, the amount of missed send-out and the amount of borrowing in the simulation.

The rolling horizon approach is applied with a sliding window of 30 days, and an overlapping period of 15 days. E.g. for the instances with a horizon 50 days we have three iterations: days 1 to 30, days 16-45, and days 31-50. Since we run the solver for lower horizon subproblems we use the best performing objective function (Π_3) as objective function in the rolling horizon approach.

The results for Π_3 are the same as in the previous section and because we we only ran experiments for the instances with an horizon of 50 days we only have those results available. For the rolling horizon approach and the heuristic we also use the instances with a horizon of 366 days.

First we compare Π_3 with the rolling horizon approach. We expect the runtime of the rolling horizon approach to be lower, since we have to solve multiple smaller horizons and from previous experiments we have learned the runtime to be exponential in the horizon. On the other hand we expect a decrease of performance in the simulation due to the fact that the order in which the vessels arrive in the rolling horizon approach is not totally free; the available vessel types are selected randomly for each window.

Hypothesis 7 *The rolling horizon approach solves faster than the default multi client formulation with objective function Π_3. Missed send-out and the amount of borrowing in the simulation are higher, but not that much.*

The next comparison is between Π_3 and the heuristic. The heuristic is also expected to be faster than the solver since it only iterates once over all the vessels to plan them. In the simulation the solutions found by de heuristic are expected to perform worse since the order of the vessels is determined beforehand and can not be adjusted during runtime.

Hypothesis 8 *The heuristic gives solutions faster than the best performing multi client solver we have, however, in the simulation the missed send-out and borrowing are significantly higher.*

Next we are interested in the influence of the horizon of an instance on the run time. Since the goal of the work in this thesis is to create an Anual Delivery Plan we should be able to create schedules for at least 366 days in an acceptable timespan.

The rolling horizon approach first makes a schedule for a full window (30 days) and from then it only has to freely plan 15 days. We therefore expect that most of the time is spent on the first period and extending the horizon has some effect on the run time, but not that much.

For the heuristic we expect a linear increase for the run time when changing the horizon. The run time is mostly determined by the total amount of vessels that need to be planned.

Hypothesis 9 *The rolling horizon approach is only slightly slower when changing the horizon from 50 days to 366 days, while the heuristic is linearly slower.*

The last comparison is between the rolling horizon approach and the heuristic. Since for the heuristic the run time is mostly determined by the amount of vessels, and not by for example the amount of clients, we expect the heuristic to be slightly faster than the rolling horizon approach for most instances.

In the simulation we expect the heuristic is outperformed by the rolling horizon approach. In the rolling horizon approach there is some freedom in the order in which vessels arrive where in the heuristic this is totally fixed.

Hypothesis 10 *The heuristic finds solutions in a shorter run time than the rolling horizon approach. However, the simulation results for the rolling horizon approach are better.*

4.4.2. Results and conclusions

After presenting the hypotheses in the previous section in this section we present the results of the experiments. For completeness we also show the results we obtained for the other methods: Π_1, Π_2,

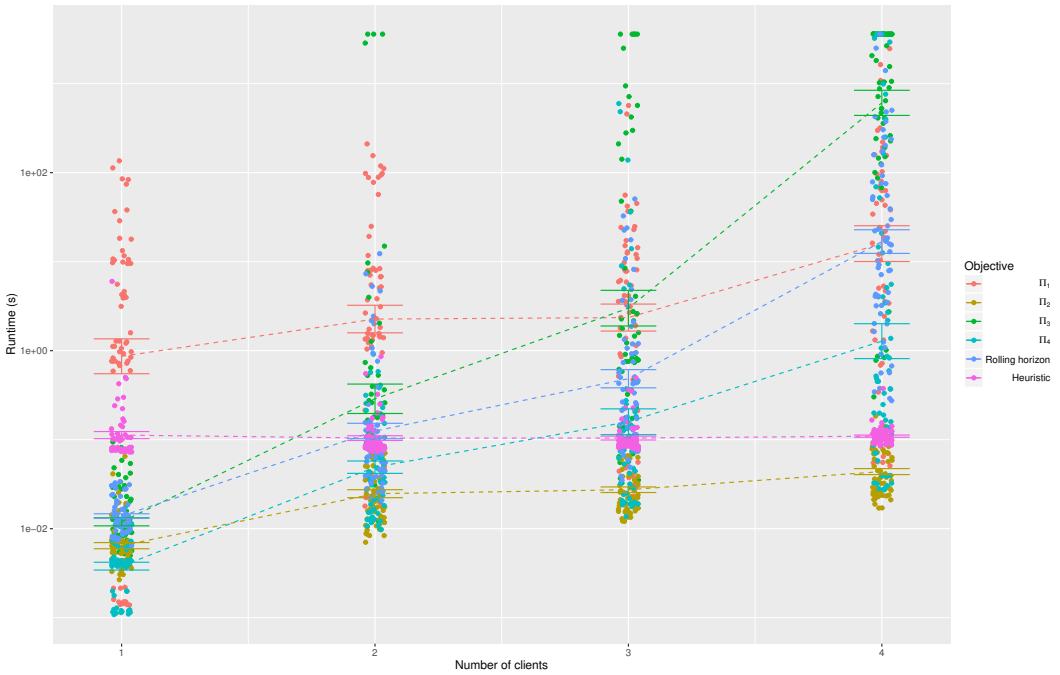


Figure 4.13: Run time on a log scale for instances with a horizon of 50 days, including rolling horizon approach and heuristic

and Π_4. In this section we first look at results for the instances with a horizon of 50 days, followed by an evaluation on what happens to the run time when changing from a horizon of 50 days to a horizon of 366 days and a comparison between the rolling horizon approach and the heuristic on instances with a horizon of 366 days. We finish the section with some last remarks on the validity of the results and future improvements.

In Figure 4.13 the run times for the methods we use are shown on a log scale. If we compare Π_3 (green) with the rolling horizon approach (blue) we see that they are comparable for one client, while for multiple clients the difference between them is increasing. The rolling horizon approach is a lot faster. While the rolling horizon approach is faster than Π_1 and Π_3 it is still not the fastest method available. Π_2 and Π_4 are still faster.

In the same figure we also compare Π_3 to the heuristic (pink) on run time. We see the heuristic is invariant to the number of clients when it comes to run time: all instances are solved in around 0.1 second. Next to that we see that only Π_4, minimising only missed send-out, is faster for each number of clients. A paired t-test learns us the heuristic is significantly faster than the rolling horizon approach.

Figure 4.14 shows the average missed send-out percentage on a log scale during the simulation for all our solving methods. We see the rolling horizon approach has a higher missed send-out than all of the objective functions with the normal multi client formulation. When we look at the heuristic we suspect is worse than the rolling horizon approach; a paired t-test confirms this suspicion: it is statistical significant worse.

In Figure 4.15 boxplots for the average total borrowed LNG and the missed send-out due to the fact no lenders were available, both as percentage of the total send-out, are shown on a log scale. We see the rolling horizon approach performs way worse than the normal multi client method with the same objective function (Π_3). We see all other objective functions except for Π_2 perform better than the rolling horizon approach based on the total amount borrowed. However, for Π_2 the percentage of missed send-out because there are no lenders available is higher.

Looking at the results for the heuristic we see that the median of total amount borrowed is comparable to that of the rolling horizon approach. The larger second quartile does not lead to a statistical significant difference between the two. This also means it is worse than Π_3. It is important to notice that the percentage of missed send-out because no lenders were available for the heuristic is the highest of all the methods we tested. In fact, it is statistical significant worse than the rolling horizon approach.

The rolling horizon approach solves faster than the default multi client formulation with objective

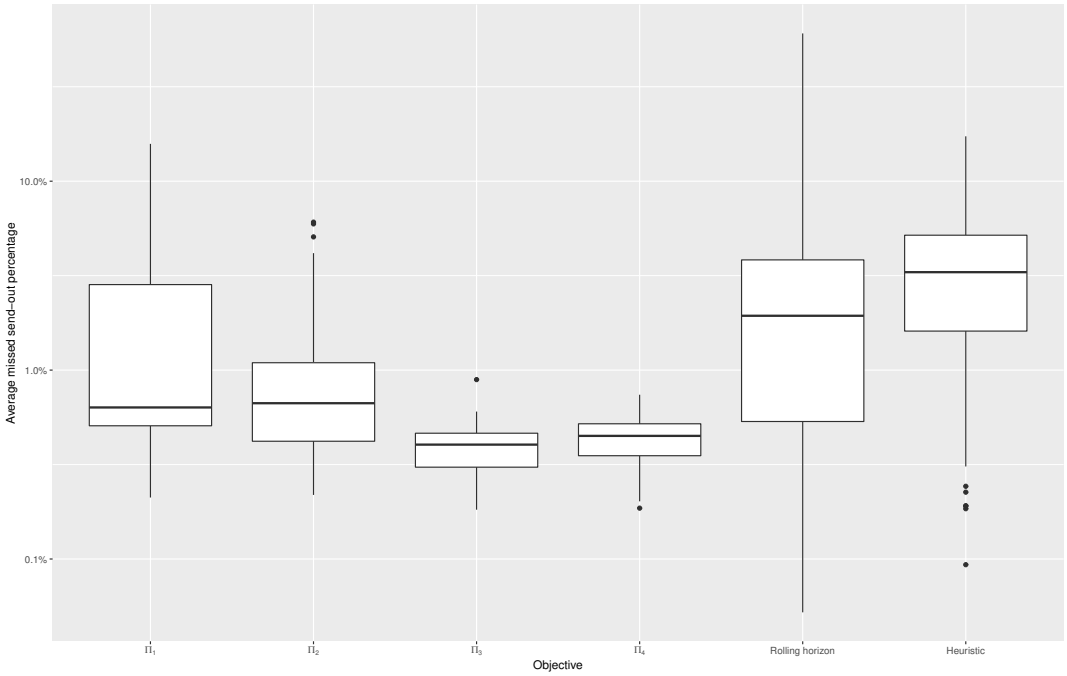


Figure 4.14: Average missed send-out as percentage of total send-out on a log scale for instances with a horizon of 50 days, including rolling horizon approach and heuristic

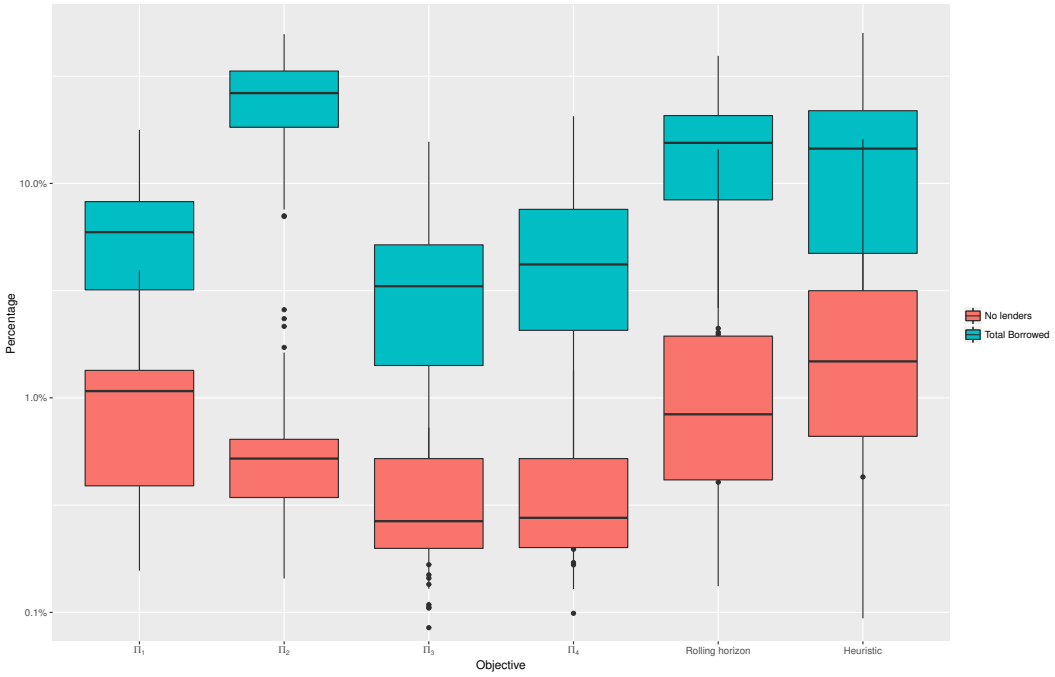


Figure 4.15: Average total borrowed LNG and LNG not send-out because there were no lenders available as percentage of total send-out on a log scale for instances with a horizon of 50 days, including rolling horizon approach and heuristic

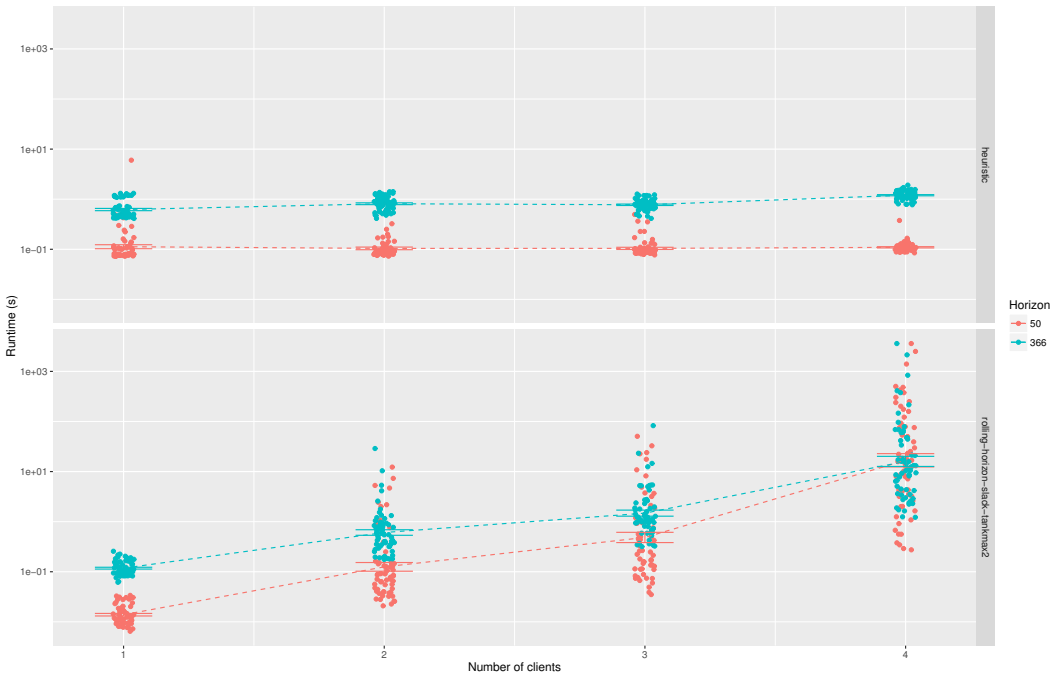


Figure 4.16: Run time on a log scale for instances with a horizon of 50 days and 366 days for rolling horizon approach and heuristic

function Π_3. Missed send-out and the amount of borrowing in the simulation are higher, but not that much.

When we combine the results described above we reject Hypothesis 7 in which was stated that the simulation results for the rolling horizon are not that much worse. While the run time is indeed lower, we have seen the results for both missed send-out and the amount of borrowing are worse than initially expected.

We accept Hypothesis 8. The heuristic is indeed faster than Π_3 and the results obtained through simulation are worse.

The next Hypothesis is on the effects of a higher horizon on the run time. Figure 4.16 shows two plots, both on a logarithmic scale, for the run time against the number of clients for instances with a horizon of 50 and 366 days. The top half of the image shows the run time for the heuristic and the bottom half shows the runtime for the rolling horizon approach.

For the heuristic we see that also for the instance with a horizon of 366 days the run time for this method is not dependent on the number of clients. When we look at Table 4.7 we can see that, when we divide the mean of the run time by the horizon, it is almost the same, i.e. the run time per day to plan is equal, so the run time is linear in the horizon. This is as expected, since the heuristic iterates once over all the vessels that need to be planned, and the number of vessels is also linear dependent on the horizon.

The means of the run time for instances with 50 and 366 days and an increasing number of clients are closer when the amount of clients is higher, the mean run time for 366 days is even lower than that for 50 days with four clients. This could come from the randomness in both instance generation – the instances for 366 days could be easier for this method than those for 50 days – and the method itself – the order in which vessel types are selected influence the outcome.

The mean run time per day planned, mean divided by horizon, for 50 days is almost a factor 10 larger than the mean for 366 days. However, the median run times divided by horizon are more or less the same. This is in line with the expectations. In the results we also see most of the run time is spent on planning the first time window which effectively has more days to plan since there are no vessels planned for the first half.

Therefore we accept Hypothesis 9. We have seen the run time for the heuristic is indeed linear in the horizon, while the rolling horizon approach is sublinear.

Lastly we compare the rolling horizon approach on instances with a horizon of 366 days. We do

Method	Horizon	Median	Mean	Median / Horizon	Mean / Horizon
Heuristic	50	0.094	0.143	0.00188	0.00287
Heuristic	366	0.865	0.889	0.00236	0.00242
Rolling Horizon	50	0.139	49.240	0.00278	0.98470
Rolling Horizon	366	0.885	37.860	0.00242	0.10340

Table 4.7: Table runtimes in seconds

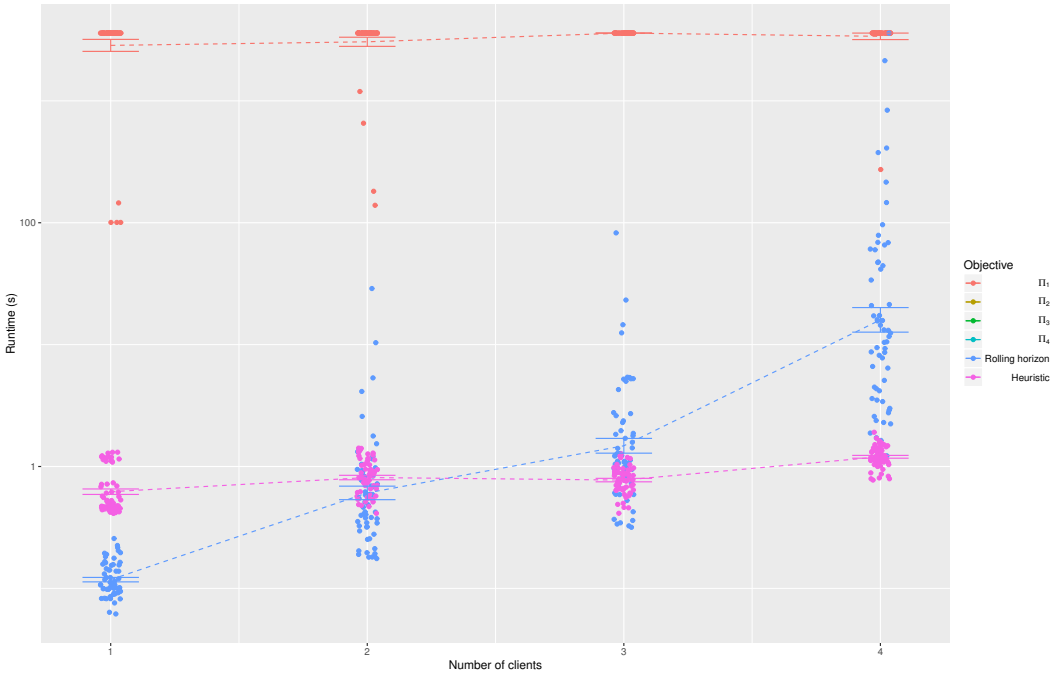


Figure 4.17: Runtime on a log scale for instances with a horizon of 366 days

this with Figure 4.17, Figure 4.18 and Figure 4.19. These figures do not show results for Π_2, Π_3, and Π_4 since there are no results for these objective functions.

The figure for runtime, Figure 4.17, sketches more or less the same image as the figure with the instances with a horizon of 50 days. The differences are that the multi client formulation with objective function Π_1 times out in almost all the cases and the rolling horizon approach is faster more often. So, while the influence of the horizon was worse on the heuristic than the rolling horizon approach, the rolling horizon is still slower for many instances. A paired t-test learns us the rolling horizon approach is statistically significant slower than the heuristic.

In Figure 4.18 the average missed send-out as percentage of the total send-out is shown. There is less missed send-out than for the instances with 50 day horizons for all three methods. We see the rolling horizon approach is still a little bit better than the heuristic – it is lower in 83% of the cases – but the difference is not statistical significant as it was with the instances with a horizon of 50 days.

Figure 4.19 shows the average total amount borrowed and the average LNG not send-out because no lenders were available as percentage of the total send-out. Compared to the same plot with the instances with a 50 day horizon we see Π_1 is more or less the same while the rolling horizon and heuristic both perform better on this aspect. We see the rolling horizon approach and the heuristic perform almost the same on these instances and there is no statistical significant difference between the two. The rolling horizon approach has a little less missed send-out due to the fact that no lenders were available, while the heuristic has a little less borrowing.

We have seen the rolling horizon approach is statistically significant slower than the heuristic for instances with a horizon of 50 and 366 days. The simulation showed the rolling horizon is statistical significant better than the heuristic for instances with a horizon of 50 days, but for instances with a horizon of 366 days this is no longer true. We therefore conclude that, based on the experiments we performed, the heuristic is a little better purely based on the run time. The rolling horizon does seem

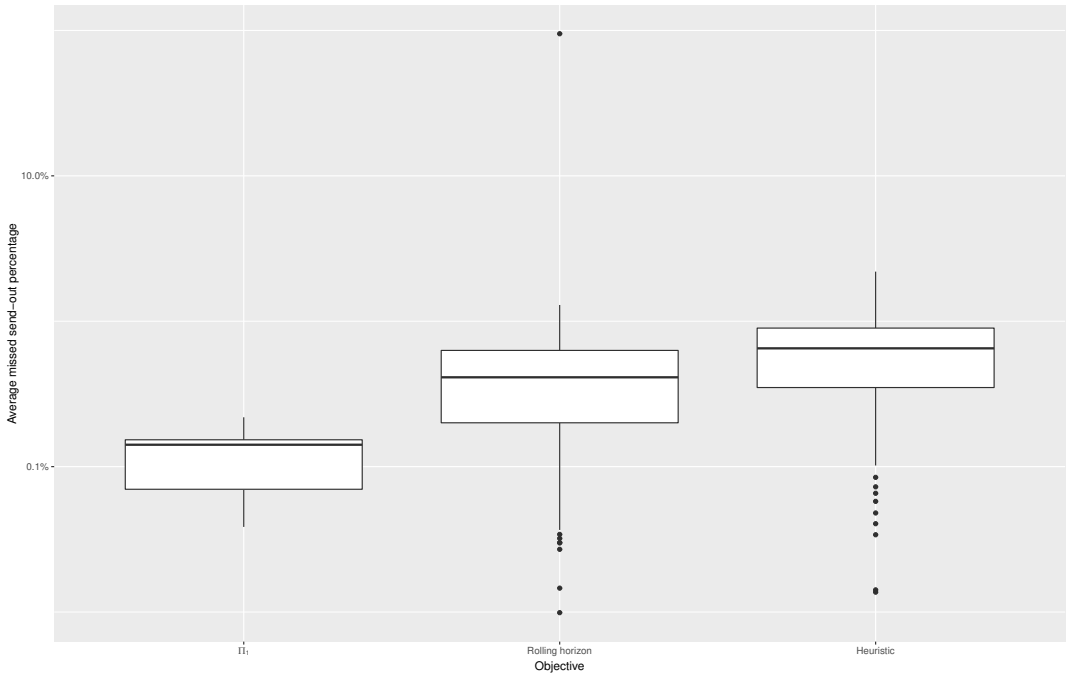


Figure 4.18: Average missed send-out as percentage of total send-out on a log scale for instances with a horizon of 366 days

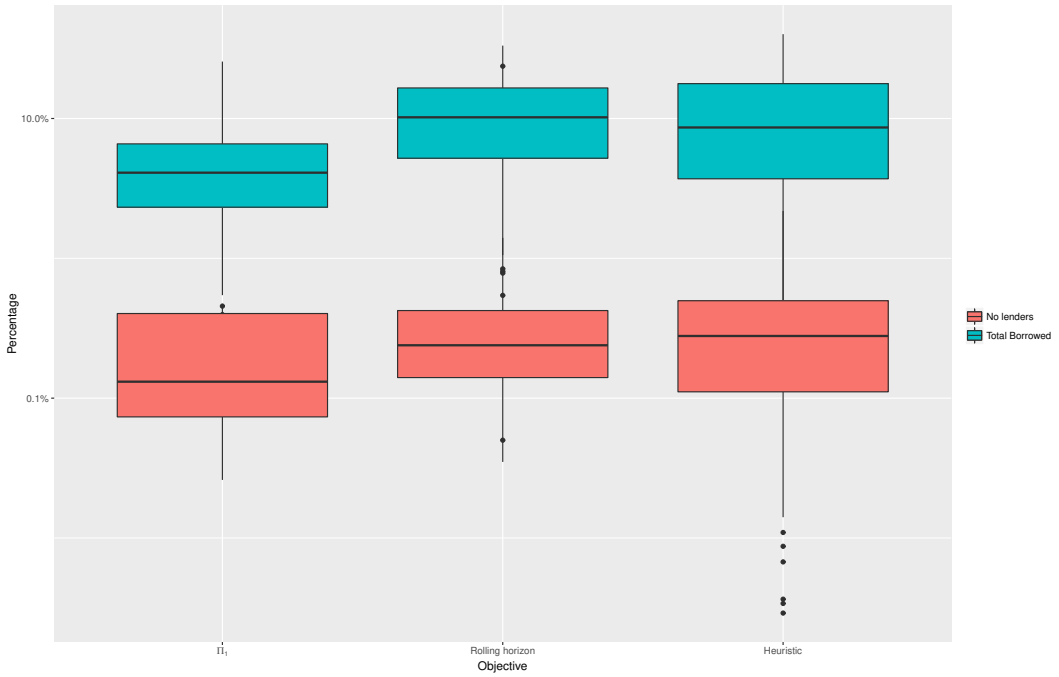


Figure 4.19: Average total borrowed LNG and LNG not send-out because there were no lenders available as percentage of total send-out on a log scale for instances with a horizon of 366 days

promising however and could prove to be better with a little more tweaking.

After manual inspection of the solutions found by the running horizon approach and the heuristic and some discussion with experts at Systems Navigator we found that some of the solutions found by the rolling horizon approach were not totally in line with what is desired in operation, e.g. the rolling horizon approach often plans two of the same vessel types for the same client directly after each other. Possible solutions to improve this are discussed in subsection 5.2.2.

Some final remarks should be made for the results for the rolling horizon approach and the heuristic. First, the results presented in this section are based on one run of the methods. This may have influenced the results due to the fact both methods are based on a fixed order in which the different vessel types for the clients should arrive, which is generated randomly. This effect is compensated in our experiments by solving for multiple somewhat similar instances, but might still have influenced the outcome.

Another remark is that the rolling horizon is only tested with one objective function and for one size of the sliding window. Changing the configuration for the rolling horizon approach might improve the results presented in this section. More on extra experiments can be found in subsection 5.2.3.

5 Conclusion

In this chapter we conclude our work. We first summarise the conclusions by answering the research questions and then pose some directions for future work.

5.1. Research Questions

In the introduction we posed research questions we answered over the course of this thesis. In this section we summarise the answers and refer to the corresponding chapters for the whole answer.

RQ 1 *What are the objectives and contraints when creating an ADP for an LNG receiving terminal?*

In chapter 2 the problem definition was stated. We defined the desired output and the different input parameter. The input consists of properties of the terminal and contract definitions we have to create the ADP for.

Next to that some goals and objectives are given. These goals and objectives are split up in terminal utilisation and impact of differentiation. Terminal utilisation includes minimising the amount of missed send-out, minimising lending and borrowing, and maximising free space in the tank. With impact of differentiation we mean that we create an ADP that is “robust”, for this for this we introduce an objective that maximises the inventory positions.

RQ 2 *Is the problem of creating an ADP an NP-hard problem?*

In section 2.4 we have looked at the problem of creating an ADP and compared it to different known NP-complete problems. We have shown different comparable problems and shown where they differ and cannot be mapped one-on-one.

We have also given a proof that the problem is NP-complete by creating a reduction from Partition and proofing this reduction is correct. Although Partition can be solved in pseudo-polynomial time, the proof, together with the other comparable problems, give a really strong indication the typical instances for this problem are also hard to solve.

RQ 3 *What are appropriate techniques to describe the problem using an (Mixed Integer) Linear Program solver and which of these performs best (on a small scale)?*

In order to answer this question we introduced three formulations in section 3.1, each using a different method to model time. We created a discrete time formulation – using boolean arrival variables for each vessel type and discrete time step – a piecewise linear formulation – that uses piecewise linear functions to describe the impact on the tank level – and an event-based formulation – that uses event points and allows for arrivals on a continuous time scale.

In section 4.2 we discussed the experiments using the single client formulations, we evaluate on run time. We have given the setup, including the way instances are generated, some hypotheses and finally the results for the experiments. We learned the piecewise linear formulation is not usable in a production environment and recommend not to use it. The event-based and discrete time formulation both seem

promising. After analysis we learned that not one of the two formulations is statistical significant faster than the other and we did an analysis on the factors that influence the run time for these instances the most. For both formulations this is the total number of vessels and the total send-out, next to that the number of send-out changes influences the run time for the event-based formulation a lot and the number of vessel types is an important factor for the discrete time formulation.

RQ 4 *What is a good objective function for the Mixed Integer Linear Program?*

In order to answer this question we extended the discrete single client formulation to allow for multiple clients in section 3.2. In this section also four parts for objective functions are introduced, based on the objectives discussed in the problem definition. We later combined the objective functions for evaluation by the experiments. We created four different objective functions which can roughly be summarised as follows: Minimise inventory positions and lending and borrowing (Π_1), minimise the amount of missed send-out (Π_2), minimise tank level below threshold (Π_3), and minimise lending and borrowing (Π_4).

The experiments performed with this multi client formulation ran with newly created instances that better represent the real life situation. A simulation step for evaluation of the found solutions is also introduced. We evaluate the different objective functions based on run time and this simulation. The simulation gives results for amount of missed send-out and the amount of lending and borrowing.

From the experiments we learned that purely based on run time minimising on missed send-out (Π_2) is the best choice as objective function. On the other hand, however, the results from the simulation are worse than the other objective functions. Minimising the inventory positions (Π_1) is also not very good in the simulation step and it takes a long time to solve.

For the best solutions one should either use minimise the IPs going below a certain threshold (Π_3) or minimise lending and borrowing (Π_4). These objective functions lead to the best performing solutions in the simulation, though they take more time to solve. Π_4 is the faster of the two by far, while Π_3 is a little better in the simulation step.

To see if we can further improve the solutions found by Π_3 we also did an experiment to see what the influence of the used threshold is. We found that the best threshold is 50.000 m^3 for our instances.

RQ 5 *Will a rolling horizon approach lead to faster results without much loss of optimality?*

We found that solving instances for a whole year cannot be done within our time limit of one hour. And while one might say that this is not really a problem since we need a plan for a whole year and one only needs to run the solver once, this is not entirely true; the solving solution is also ought to be used in an interactive way, i.e. try some settings, run the solver, see the results, and then try again for some other settings. Because of the interactive use we sought for a way to speed up solving for a whole year and we therefore introduced a rolling horizon approach in section 3.3.

In the rolling horizon approach we used the best performing objective function (Π_3) and we compared the solutions found to the solutions found with the normal multi client formulation with this same objective function. We have seen the rolling horizon approach solved significantly faster than the multi client formulation. We have even shown that for the rolling horizon approach the run time is sublinear to the horizon of the instance, since the bulk of the solving time is in solving the first time window we solve.

However, we also learned that in the simulation step the solutions found by the rolling horizon approach perform significantly worse than the solutions found by the normal multi client formulation. It is expected this is due to the semi-fixed order of the arriving vessels that is produced randomly for the rolling horizon approach.

RQ 6 *How do the Mixed Integer Linear Program perform compared to the Systems Navigator Heuristic?*

The work in this thesis started of with the question of Systems Navigator if the problem of creating an ADP for an LNG receiving terminal can be optimised in a time efficient way. They already had a heuristic algorithm in use, so we also wanted to know how this heuristic algorithm performs. The algorithm makes use of a reorder point. This is the number of days before the tank is empty a new vessel should be scheduled, in our experiments we chose a reorder point equal to 50.000 m^3. To see the perfomance of the heuristic we compared it to both the multi client formulation and the rolling horizon approach.

First of all, we learned that overall the heuristic finds a solution for the instances faster than most of the multi client formulations, only minimising the amount of missed send-out (Π_2) is always faster. We have also seen, with a paired t-test, that the heuristic is statistically significant faster than the rolling horizon approach, and learned the run time of the heuristic is linear in the horizon (as expected).

When we look at the results from the simulation, we see the solutions found by the heuristic are not near optimal. In the comparison on instances with a horizon of 50 days we see that both the multi client formulation and the rolling horizon approach are significantly better than the heuristic. However, with the instances with a horizon of 366 days we see that there is no statistically significant difference between the simulation results of the rolling horizon approach and the heuristic algorithm.

5.2. Future Work

Over the course of this thesis on multiple occasions we have indicated suggestions for future work. In this section we present these suggestions and expand on them a little more. The suggestions are split up in extension of formulations, improvement of solutions, and further experimentation.

5.2.1. Extension of formulations

First we present the suggestions for improvement of the formulations used in this thesis. In the real life problem we have more constraints on the problem than we have modelled in our formulations.

For example, in subsection 2.2.1 we introduced that not al berths can handle all vessel types, while this cannot be found in any of our formulations. For the event based formulation this is easily added, we could only add unloading tasks for the appropriate combinations of vessel types and berths. The discrete time formulation should be extended further to plan the vessel arrivals on a specific berth before such a constraint can be added.

Another example introduced in the same section is lending and borrowing restrictions. In our multi client formulation we allow lending and borrowing without restrictions. However, in practice there are contractual restrictions on the lending and borrowing. These restrictions are restrictions on whom can borrow from whom and the amount that can be borrowed at a given time.

Other known constraints that are not incorporated include: planned maintenance of a terminal, blackout periods – periods for which we know beforehand the network will not be available – and a more specific tank model. For an extended tank model one could think for example of modeling the different tanks as separate storage entities, i.e. we model the LNG to be stored in a specific tank, separate tanks reserved for a certain client, and taking boil-off gas – basically gas that is lost during storage – into account.

More on the internal workings of a terminal and possible influences on scheduling can be found in [4, 25]. Both theses are specifically for the Gate Terminal, but most operations are found similar in other terminals.

Furthermore the event-based formulation should be extended to allow for multiple clients and lending and borrowing. The formulation looked promising from the experiments in the single client situation. To allow for multiple clients one could model the LNG per client, e.g. stored LNG for client one would be modeled as “stored-LNG for client 1”. For lending and borrowing we could create a new product (“client A borrowed LNG from client B”) and a corresponding task could be performed that consumes LNG for client A and creates the same amount of both LNG for client B and the new product and is instant, i.e. it does not take time. A comparable task for giving back LNG should also be created. At the end of the horizon a constraint could be added to force there is no borrowing debt between clients.

5.2.2. Improve solutions

The next suggestions for future work are to improve the methods presented in this thesis. This is both to improve the actual solutions, so they are more in accord with what terminals and terminal clients would like to see in the ADP, and so they are more in line with how the scheduling tools planners like to use typically work.

From manual inspection and comparison between our solutions and the solutions found by the heuristic we learned there were some improvements possible for how the solutions should look like. The most obvious improvement found this way is that vessels of the same client should not be planned back-to-back (i.e. directly after each other). To prevent this in our solutions we could add a new objective to our objective functions that punishes vessels of the same client coming closely after each

other. Of course some leeway should be given when a client has a high send-out and therefore needs the vessels to come closely to each other.

Such an objective could also improve solving times. Currently different solutions can look very similar to the solver. This is due to the fact that when we switch two vessels in the ADP this does not necessarily affect the objective. Adding an objective to force time between two arriving vessels might reduce symmetry in the linear program.

Another improvement would be to allow to find solutions based on existing ADPs. In this situation a plan already exists – this plan could be made by hand, the heuristic, or one of our methods – and we want to for example add an extra contract or improve the plan a little. An extra objective function could be added to minimise the changes to the input plan.

5.2.3. Experiments

There are some suggestions for extra experiments. The suggestions in this section are only for experiments with the rolling horizon approach and the heuristic. For the multi client we could also for example run experiments with other objective functions, but we will not go into that.

For the rolling horizon we currently only ran experiments with a sliding window of 30 days and overlap of 15 days. It would be interesting to run experiments to determine the optimal values for these parameters. It would also be interesting to examine the influence of running the first iteration of the rolling horizon approach with only half the window size on both run time and the simulation.

Both the heuristic and the rolling horizon approach depend on some randomness to determine the order of arriving vessels. In the rolling approach this is a little less fixed, but we still expect this is the biggest influence in the fact that the simulation performed worse. Since both approaches are relatively fast they could be executed multiple times in a row and the best solution can be chosen. This could be based on the value of the objective function, but it could also be based on the results of the simulation. When depending on the simulation it is important to keep in mind that running the simulation also takes some time – about 3 minutes in our setup. Another possibility is to show the different solutions found and let the planner of a terminal decide which is the best solution.

Another interesting extra experiment is to perform a usertest to rate the different methods with planners in a terminal. In such a usertest the solutions of the different methods are shown to the user and we let the user rank these solutions compared to each other.

5.2.4. Other solving methods

We finish of with some suggestions for other methods of solving the problem.

First we have the field of dynamic optimisation. This field consists of problems in which there is a optimisation problem where the objective function changes over time. The changes are known beforehand. An example of such a problem is the Dynamic Knapsack Problem [26], in which the knapsack capacity, or the profit or weight for an item, changes over time. We could see our problem as a knapsack with increasing size. In that case the knapsack grows with the amount of send-out each time step. However, the problem does not currently incorporate a minimum weight in the knapsack at any moment in time, so this should be added. Also, the objective function should be extended to be able to minimise lending and borrowing. These dynamic optimisation problems are often solved using Genetic Algorithms.

We could also, as briefly stated in the introduction, use Genetic Algorithms as a solving method for our problem. We could the formulations presented in this thesis to test feasibility of found solutions and the objectives as fitness function.

Another possible interesting way to tackle this problem is Mixed Initiative Planning. In Mixed Initiative Planning a traditional human planner teams up with a computer system that performs as an assistant to help planning [9]. In this situation both the human and computer both perform tasks they are good in. For example the computer can quickly calculate the tank levels for each time step, and a human has an intuition on what might be good schedules.

Ferguson et al. [10] introduce a framework to represent plans and arguments about these plans so communication about the plans is possibile. Later this framework was implemented in a prototype for a simple scheduling problem [9]. The underlying TRAINS system has since then been improved and could be an interesting starting point for a mixed initiative planning solution for the problem of creating an ADP in LNG receiving terminals.

A

Formulations

A.1. Single client Discrete Time

$$T = \{i \in N | i \leq H\}$$
$$IP(t) = \text{Tank level at time } t$$
$$S_t = \text{Send-out at time } t$$
$$D_t = \text{Delivery at time } t$$
$$A_{v,t} = \text{Vessel } v \text{ arrives at time } t$$
$$c_v = \text{Capacity for vessel } v$$
$$c = \text{Capacity of tank}$$
$$\#_v = \text{Number of vessels of type } v$$
$$B = \text{Berths}$$

Minimize

$$\sum_t IP(t)$$

Subject to

$$IP(0) = 0$$
$$IP(t) = IP(t-1) + D_{t-1} - S_{t-1} \qquad t \in T - \{0\}$$
$$IP(t) \leq c \qquad t \in T$$
$$D_t = \sum_v A_{v,t} * c_v \qquad t \in T$$
$$\sum_v A_{v,t} \leq |B| \qquad t \in T$$
$$\sum_t A_{v,t} \leq \#_v \qquad v \in V$$
$$A_{v,t} \in \{0,1\} \qquad v \in V, t \in T$$

A.2. Single client Piecewise linear

c_v = Capacity of vessel v
c = Capacity of terminal
A_v = Arrival time of vessel v
$I_v(t)$ = Amount of LNG in tank at time t
I_t = Total amount of LNG in tank at time t
S_t = Amount of LNG sent-out during time slot t

Minimize $\sum_t I_t$

Subject to

$$IP_v(t) = \begin{cases} c_v & A_v \leq t \\ c_v - (t - A_v) * c_v & t < A_v < t + 1 \\ 0 & \text{otherwise} \end{cases} \qquad v \in V$$

$$IP(t) = \sum_v IP_v(t) - \sum_{x \leq t+1} S_x \qquad t \in T$$

$$IP(t) \leq c \qquad t \in T$$

A.3. Single client event-based

$$\sum_{i \in I_j} w(i,n) = yv(j,n) \qquad \forall j \in J, n \in N$$

$$B(i,j,n) = wv(i,n) \cdot c_i \qquad \forall i \in I^{AR}, j \in J_i, n \in N$$

$$B(i,j,n) \leq (T^f(i,j,n) - T^s(i,j,n)) \cdot r_n \qquad \forall i \in I^{SO}, j \in J_i, n \in N$$

$$ST(s)^{\max} \geq ST(s,n) \qquad \forall s \in S, n \in N$$

$$ST(s)^{\max} \geq ST(s,n) + \sum_{i \in I_s} \rho^p_{s,i} \sum_{j \in J_i} B(i,j,n) \qquad \forall n \in N, s = \text{“Stored LNG”}$$

$$ST(s,n) \geq (T^{s,AR}(n) - T^s(i_{SO}, j_{SO}, n)) \cdot r_n \qquad \forall n \in n$$

$$\begin{aligned} ST(s,n) = {} & ST(s,n-1) - delivery(s,n) \\ & + \sum_{i \in I_s} \rho^p_{s,i} \sum_{j \in J_i} B(i,j,n-1) \\ & - \sum_{i \in I_s} \rho^c_{s,i} \sum_{j \in J_i} B(i,j,n-1) \end{aligned} \qquad \forall s \in S, n \in N$$

$$T^{s,AR}(n) \geq T^s(i,j,n) - H(2 - wv(i,n) - yv(j,n)) \qquad \forall i \in I^{AR}, j \in J_i, n \in N$$

$$T^f(i,j,n) = T^s(i,j,n) + \alpha_{i,j} wv(i,n) + \beta_{i,j} B(i,j,n) \qquad \forall i \in I^{AR}, j \in J_i, n \in N$$

$$L \cdot wv(i,n) \leq T^f(i,j,n) - T^s(i,j,n) \leq H \cdot wv(i,n) \qquad i \in I^{SO}, j \in J_i, n \in N$$

$$T^s(i,j,n+1) \geq T^f(i,j,n) - H(2 - wv(i,n) - yv(j,n)) \qquad i \in I, j \in J_i, n \in N, n \neq N$$

$$T^s(i,j,n+1) \geq T^s(i,j,n) \qquad i \in I, j \in J_i, n \in N, n \neq N$$

$$T^f(i,j,n+1) \geq T^f(i,j,n) \qquad i \in I, j \in J_i, n \in N, n \neq N$$

$$T^s(i,j,n+1) \geq T^f(i',j,n) - H(2 - wv(i',n) - yv(j,n)) \qquad i \in I, i' \in I, i \neq i', j \in J_i, n \in N, n \neq N$$

$$T^s(i,j,n+1) \geq T^f(i',j',n) - H(2 - wv(i',n) - yv(j',n)) \qquad i \in I, i' \in I, i \neq i', j \in J_i, j' \in J_i, j \neq j', n \in N, n \neq N$$

$$\sum_{n \in N} \sum_{i \in I_v} wv(i,n) = \#_v \qquad \forall v \in V$$

A.4. Multi client discrete time

$$
\begin{aligned}
IP(g,0) &= 0 && \forall g \in G \\
IP(0) &= 0 && \\
IP(g,t) &= IP(g,t-1) + D_{g,t-1} - (S_{g,t-1} - s^{SO}_{g,t-1}) && \forall g \in G, t \in T \\
IP(t) &= \sum_{g \in G} IP(g,t) && \forall t \in T \\
0 &\leq IP(t) \leq c && \forall t \in T \\
-c &\leq IP(g,t) \leq c && \forall g \in G, t \in T \\
D_{g,t} &= \sum_{v \in V_g} c_v A_{v,t} && \forall g \in G, t \in T \\
\#_v &\geq \sum_{t \in T} A_{v,t} && \forall v \in V \\
|B| &\geq \sum_{v \in V} A_{v,t} && \forall t \in T \\
\sum_{t \in T} D_{g,t} &\geq \sum_{t \in T} \left(S_{g,t} - s^{SO}_{g,t}\right) && \forall g \in G
\end{aligned}
$$

$$
\begin{aligned}
\text{tank min var}_{g,t} &\geq 0 && \forall g \in G, t \in T \\
\text{tank min var}_{g,t} &\geq IP(g,t) - IP_{\min} && \forall g \in G, t \in T \\
\text{tank max var}_{g,t} &\geq 0 && \forall g \in G, t \in T \\
\text{tank max var}_{g,t} &\geq IP_{\max} - IP(g,t) && \forall g \in G, t \in T \\
bor_{g,t} &\geq 0 && \forall g \in G, t \in T \\
bor_{g,t} &\geq -IP(g,t) && \forall g \in G, t \in T
\end{aligned}
$$

$$
\begin{aligned}
\Pi_{SO} &= \sum_{t \in T} \sum_{g \in G} s^{SO}_{g,t} \\
\Pi_{\text{tank}} &= \sum_{t \in T} \sum_{g \in G} \frac{\text{tank min var}_{g,t}}{c - IP_{\min}} \\
\Pi_{\text{tmax}} &= \sum_{t \in T} \sum_{g \in G} \frac{S_{g,t} \cdot \text{tank max var}_{g,t}}{c - IP_{\max}} \\
\Pi_{LB} &= \sum_{t \in T} \sum_{g \in G} \frac{bor_{g,t}}{\min_{v \in V}(c_v)}
\end{aligned}
$$

Bibliography

[1] H Andersson, M Christiansen, and K Fagerholt. *Transportation Planning and Inventory Management in the LNG Supply Chain*, pages 427–439. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. ISBN 978-3-642-12067-1. doi: 10.1007/978-3-642-12067-1_24. URL http://dx.doi.org/10.1007/978-3-642-12067-1_24.

[2] E. M. L. Beale and J. A. Tomlin. Special Facilities in a General Mathematical Programming System for Non-convex Problems Using Ordered Sets of Variables. *Operations Research*, 69(JANUARY 1969):447–454, 1970.

[3] D Bertsimas and J Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 3rd edition, 1997. ISBN 1886529191.

[4] J Caljé. Lng receiving terminals - designing a decision support tool to investigate business cases from a logistical and gas quality perspective. Master's thesis, TU Delft, The Netherlands, 2008.

[5] H P Cheung. Planning and scheduling under uncertainty. Master's thesis, Erasmus University Rotterdam, The Netherlands, 2013.

[6] Marielle Christiansen, Kjetil Fagerholt, and David Ronen. Ship Routing and Scheduling: Status and Perspectives. *Transportation Science*, feb 2004. URL http://pubsonline.informs.org/doi/abs/10.1287/trsc.1030.0036.

[7] D Dispenza. The lng industry, giignl annual report 2016 edition. Technical report, GIIGNL, 2016.

[8] J A Elffers. Scheduling with release times and deadlines. Master's thesis, TU Delft, The Netherlands, 2014.

[9] G Ferguson, J F Allen, B W Miller, et al. Trains-95: Towards a mixed-initiative planning assistant. In *AIPS*, pages 70–77, 1996.

[10] George Ferguson and Jf Allen. Arguing about Plans: Plan Representation and Reasoning for Mixed-initiative Planning. *Aips*, 2010:43–48, 1994. URL http://www.aaai.org/Papers/AIPS/1994/AIPS94-008.pdf.

[11] M Florian and M Klein. Deterministic production planning with concave costs and capacity constraints. *Management Science*, 18(1):12–20, 1971.

[12] M Florian, J K Lenstra, and A H G Rinnooy Kan. Deterministic production planning: Algorithms and complexity. *Management science*, 26(7):669–679, 1980.

[13] M R Garey and D S Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979. ISBN 0716710447.

[14] R. Gronhaug, Marielle Christiansen, Guy Desaulniers, and Jacques Desrosiers. A Branch-and-Price Method for a Liquefied Natural Gas Inventory Routing Problem. *Transportation Science*, 44 (3):400–415, 2010. ISSN 0041-1655. doi: 10.1287/trsc.1100.0317. URL http://transci.journal.informs.org/cgi/doi/10.1287/trsc.1100.0317.

[15] M. G. Ierapetritou and C. a. Floudas. Effective Continuous-Time Formulation for Short-Term Scheduling. 1. Multipurpose Batch Processes. *Industrial & Engineering Chemistry Research*, 37 (11):4341–4359, 1998. ISSN 0888-5885. doi: 10.1021/ie970927g. URL http://pubs.acs.org/doi/abs/10.1021/ie970927g.

[16] M G Ierapetritou and C A Floudas. Effective Continuous-Time Formulation for Short-Term Scheduling . 2 . Continuous and Semicontinuous Processes. *Industrial & Engineering Chemistry Research*, 37:4360–4374, 1998.

[17] M G Ierapetritou, T S Hené, and C a Floudas. Effective Continuous-Time Formulation for Short-Term Scheduling . 3 . Multiple Intermediate Due Dates. *Industrial & Engineering Chemistry Research*, 38:3446–3461, 1999.

[18] IGU. Life Cycle Assessment of LNG. (June):43, 2015.

[19] Satish Kumar, Hyouk-tae Kwon, Kwang-ho Choi, Jae Hyun, Wonsub Lim, and Il Moon. Current status and future projections of LNG demand and supplies : A global prospective. *Energy Policy*, 39(7):4097–4104, 2011. ISSN 0301-4215. doi: 10.1016/j.enpol.2011.03.067. URL http://dx.doi.org/10.1016/j.enpol.2011.03.067.

[20] A Lim. The berth planning problem. *Operations Research Letters*, 22(2–3):105–110, 1998. ISSN 0167-6377. doi: http://dx.doi.org/10.1016/S0167-6377(98)00010-8. URL http://www.sciencedirect.com/science/article/pii/S0167637798000108.

[21] C C Pantelides. Unified frameworks for optimal process planning and scheduling. In *Proceedings on the second conference on foundations of computer aided operations*, pages 253–274. Cache Publications New York, 1994.

[22] M L Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Springer Publishing Company, Incorporated, 4th edition, 2012. ISBN 9781461419860.

[23] G V Reklaitis. *Overview of Scheduling and Planning of Batch Process Operations*, pages 660–705. Springer Berlin Heidelberg, Berlin, Heidelberg, 1996. ISBN 9783642609725. doi: 10.1007/978-3-642-60972-5_27.

[24] J Smits. The impact of a good inventory management policy. Technical report, Systems Navigator B.V. URL http://www.systemsnavigator.com/sn{_}website/whitepapers/Whitepaper-Theimpactofagoodinventorymanagementpolicy.pdf.

[25] S H van Leeuwen. Registration of stock control at gate - designing a registration system for the stock control of an lng terminal. Master's thesis, TU Delft, The Netherlands, 2006.

[26] Shengxiang Yang and Xin Yao. *Evolutionary Computation for Dynamic Optimization Problems*, volume 490. 2013. ISBN 978-3-642-38415-8. doi: 10.1007/978-3-642-38416-5. URL http://link.springer.com/10.1007/978-3-642-38416-5.

Acronyms

ADP Annual Delivery Plan. 1, 2, 7, *Glossary:* Annual Delivery Plan

IP Inventory Position. 7

KPI Key Perfomance Indicator. 32

LNG Liquefied Natural Gas. 1–3, 7

MILP Mixed Integer Linear Program. 3, 13, 25, 37, 46

MTPA Million Tonnes Per Annum $\sim 2.320.117 m^3$ LNG. 31

NG Natural Gas. 1, 2

ROP re-order point. 3, 37

SOS2 Special Ordered Sets of type 2. 5, 14, 15, 25