Evaluation of First Come First Serve Scheduling in A Geometrically-Constrained Wet Bulk Terminal

Bachelor Business Case – June 2020
Port of Amsterdam

H.E. Eisen, J.E. Van der Lei,
D. Pop, U. Rimša, and H. Tao

E.R. Dugundji
J. Zuidema
H-J. Oost
P. Valk

Vrije Universiteit Amsterdam
Faculty of Science
Department of Mathematics

1 Executive Summary

This paper focuses on evaluating the first come, first serve scheduling principle in the EVOS terminal in the Port of Amsterdam. As the EVOS terminal is a wet bulk terminal that has additional geometrical constraints and limited research is known on scheduling in wet bulk terminals, a creative mathematical approach is required when searching for alternative scheduling methods. Finding an alternative scheduling method is especially relevant when the new sealock comes in operation in 2022 when (more) larger ships are expected to arrive to the port.

The problem of allocating ships to berths is known as the Berth Allocation Problem (BAP) and is widely studied, however not for wet bulk terminals with geometrical constraints as in the EVOS terminal. The BAP for this case is defined as a discrete dynamic berth allocation problem, meaning that one ship can only be assigned to one berth, and vessels have an arrival time. The complete mathematical formulation can be found in Section 6.4.2.

This study has created four types of models, also further explained in Section 6.4.3: optimal first come first serve; no first come first serve; 48 hour relaxation; and complete relaxation with two different objective functions representing the terminal and client perspective. The performance of the model is tested by means of a rolling time window: of each day within the timeframe, a schedule was created for the following two weeks. For each iteration, the objective value of the model is calculated and stored, such that eventually the mean over the entire time period can be compared to the mean value of the historical schedules.

When the outcomes of the four previously described models with different objective functions are compared to the historical values, it is found that the optimal first come, first serve model already shows an improvement compared to the historical situation. Between the first come, first serve and the no first come, first serve model, there are no considerable differences, because the vessels are constrained to be scheduled on/after their arrival time at the port. It can be seen that when relaxation is allowed (in the latter two model types), a considerable efficiency gain is possible.

Moreover, the arrival of larger ships is simulated by taking a percentage $(25, 50, 75)$ of the historical ships and enlarging them to the theoretical maximum sizes that EVOS allows $(285 \times 48 \times 14.05)$, also updating their duration accordingly. A similar simulation is performed as for regular sizes ships, only this time the outcomes are compared to the optimal first come, first serve objective values. The outcomes show that a slight improvement can be made by abandoning the first come, first serve strategy, but (as with regular ships), most realistic efficiency can be gained when relaxing the arrival times of the vessels.

The model as constructed in this paper can be used in real time by using the Estimated Time of Arrival (ETA) as the arrival time of a vessel, and the prediction model to predict the duration of stay. It should however be noted that the more accurate these two factors are, the more realistic the schedule is. An example proof of concept of real time use of this model is displayed in Section 7.

Contents

2 Introduction

In 2022, the new sea lock in IJmuiden is expected to open. This new lock will be five hundred metres long, seventy metres wide and eighteen metres deep and has -after its completion- the title for being the world's largest sea lock. Consequently, this leads to an expected increase of traffic of larger ships from and to the North Sea. Therefore, the Port of Amsterdam (PoA) is interested in exploring alternatives to their current first come, first serve berth allocation policy.

To maximise the usage of the limited number of berths, the relevant departments have to find a balance between the size of the vessels in relation to maneuverability within the port channel and harbour basins. Normally, vessels are served on a first come, first serve (FCFS) basis. However, this may not be the optimal approach. This problem has been studied by logistical researchers and has become known as the Berth Allocation Problem (BAP), which involves assigning available berths to incoming ships subject to the given time and berth layout constraints. Its objectives are usually two in number: to reduce the overall waiting time of ships at the port, and to minimise the probability that an incoming ship's docking request will be refused. According to Imai, Nishimura and Papadimitriou [10], the BAP can have a decisive impact on the efficiency of port operations. However, there are many side constraints involved in BAP, which can hamper the process of allocation.

A special aspect of this case is that EVOS, the targeted terminal, is primarily wet bulk- rather than container-oriented. The latter has been widely studied, but, research on bulk ports is relatively scarce. PoA wants to find better alternatives to the current berth allocation strategy, done on a "first come, first served" basis. Additionally, since larger vessels need to be able to moor, the current layout of the terminals has to be considered besides the temporal aspect of the scheduling, taking into account that the ships need enough space to maneuver in and out of a berth. Consequently, the BAP for the Port of Amsterdam involves both the spatial and the temporal dimensions and requires a creative mathematical solution.

The aim of this research is to investigate if there are other ways for berth allocation that optimise the total berth usage. This is done by solving three subproblems: deriving the duration of stay from historical Automatic Identification System (AIS) signals, using this data to predict how long arriving ships will stay, and determining the berth occupancy in the terminal based on AIS data. Then, four types of final models are used to assign berths to arriving ships: an optimal first come first serve model, a model where only the arrival times are fixed, a model with arrival time relaxation, and lastly a model with complete arrival time relaxation, meaning a vessel can be scheduled anywhere before their estimated arrival time. The outcomes of these models can be used to test if changing the first come, first serve approach would be beneficial for the EVOS terminal. The study on this terminal can then serve as an example case for other terminals in the port: it can show that further efficiency can be achieved by scheduling in advance and can be used as incentive for other terminals to become more inventive.

3 Case Description

To justify the potential value of a data-driven berthing arrangement, instead of simply applying a first come, first serve criterion, a dynamic berth allocation system with comprehensive considerations is tailored to the EVOS terminal in the Afrikahaven. In this work, the newly overall infrastructure comprises three unique sections. As shown in Figure 1, they are request handling and queue management systems, dock allocation and departure estimation systems, and terminal tracking system respectively.

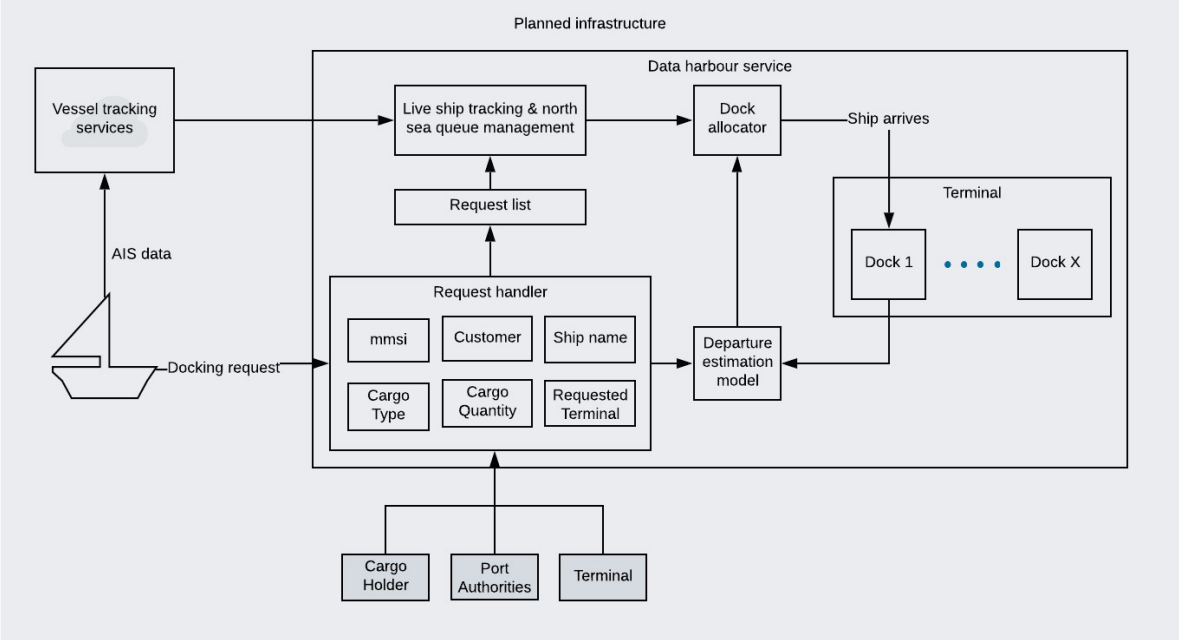


Figure 1: Overall Planned Infrastructure

3.1 Request Handling and Queue Management

The request handling and queue management are a compound control system. Before reaching the sea lock, an incoming vessel will send out a docking request (the equivalent to a Q88), containing information on ship unique ID (MMSI) and name, cargo type and quantity, customer details, and a requested docking terminal. All these raw data are monitored at the request handler under interventions from three independent departments: cargo holders, port authorities, and terminal merchants. Terminals provide loading, unloading and stockpile services for cargo holders, recognised as clients, while port authorities as a whole is in charge of the sea lock queue management covering vessels going to all terminals. Newly arriving vessels are commonly waiting outside the sea lock in a chronological order. However, switches among vessels owned by the same transportation company are also possible. For the sake of complexity, however, these factors are not taken into account in this case. Furthermore, vessels are only allowed to enter the lock when there are berths available. Additionally, the lock queue will proportionally affect the waiting time of vessels going to EVOS terminal. This can be explained with an example. A ship requests to dock at EVOS where an empty berth is available, and is waiting fourth in line to go through the lock. However, it has to wait for the preceding three ships finishing entering the lock first, which might extend the

overall waiting time. Nevertheless, the possible extension will not cause severe delay in reality as the port authorities are responsible for avoiding such conflict by proper interventions.

3.2 Berth Allocation and Estimated Departure Time

Dock allocation and departure estimation systems play roles in predicting the durations of stay and departure times of all types of vessels as well as allocating the optimal berth. In this segment, a prediction model and an allocation model are provided separately. Shipping trajectories can be traced by analysing the previously mentioned AIS data, which will then be used to train the prediction model. During this process, several machine learning algorithms are tested. The allocation is considered as a parallel machine scheduling problem, where four machines (four berths in this case) will handle jobs (coming vessels) with different release dates (arrival times). The model is also tailored to apply the spatial restrictions regulated by port authorities. An exact algorithm is used to solve this linear program. Further details of this model is discussed in Section 6.4.

3.3 Terminal Tracking

The terminal specific system keeps tracking berth usage, more specifically, occupancy at the docks, ship arrivals and departures, actual time of arrival (ATA), actual times of departure (ATD) and stores them in a data warehouse. Additionally, the terminal-wise data supports the previous two sections by returning real-time berth usages.

4 Literature Review

The research topic was explored from the perspective of port-scheduling literature, mainly focused on the berth allocation problem (BAP). Bierwirth and Meisel [2] counted 79 new models for berth allocation after 2009, indicating a rising demand for such algorithms. The same authors classified such models in previous research [1] based on four features: spatial attribute, temporal attribute, handling time attribute ,and the performance measure. The spatial attribute describes the layout of the berths, which can be discrete (BAP-D: one boat can dock at a single berth at a time) [4], continuous (BAP-C: boats can moor at random places within the boundaries of the dock as long as they do not overlap with other vessels' positions) [12], or hybrid (one boat is allowed to occupy more than one berth at a time). All these were proven to be NP-hard problems [7].

The temporal attribute concerns the arrival of the boats, and can be static (SBAP: vessels arrived already and can moor at any time as long as there is a berth available) [8], dynamic (DBAP: allows ships to arrive at individual and deterministic times, while work is in progress) [9], cyclic (vessels return to the berth repeatedly after a fixed time), or stochastic (arrival times are defined by stochastic parameters). When it comes to the handling time of the vessels there are five categories: fixed handling times, dependency on the berthing position, dependency on the number of cranes, dependency on the work schedules of the assigned cranes, and combinations of last three [11]. The performance measure concerns the objective functions, which most of the time relate to minimising the time spent by vessels within the port. In terms of the complexity of BAP, heuristic methods are generally used [5] [15] [13].

Accordingly, it was determined that this case can be classified as discrete, dynamic, with fixed handling times, minimising on one hand the sum of all the departure times summed together, for the

client benefit, and on the other the departure time of the last vessel, towards the terminal advantage. Additionally, the terminal on which the analysis is performed, namely EVOS, is a liquid bulk terminal that handles clean petroleum products. One paper that investigated the BAP within bulk port terminals is [16], which also considered dynamic arrivals. However, the spatial aspect was hybrid, which differs from the present research. Moreover, Umang, Bierlair and Vacca addressed the BAP with a different objective, namely minimising the total service completion time, taking into account the cargo types, the conveyors and the pipelines.

5 Data

5.1 Historical AIS Data

The Automatic Identification System (AIS) is a marine tracking system, which is widely used by vessels to avoid collisions and by port authorities to coordinate the traffic within harbours. Accordingly, AIS can be seen as a continuous communication between a vessel and the quay, where the first is repeatedly transmitting its location coordinates to the latter, even when the respective vessel is moored. Additionally, the boat informs the harbour about its speed, orientation, and navigational status. Besides the dynamic information, the quay keeps track of other static variables, such as the ID, the name, the type, and the size of each ship.

The AIS data provided by CWI contains every AIS signal received by their server from May 2017 to May 2020, where every row represents one signal instance. This data is split per day and has nineteen features, including the exact timestamp when the signal took place. Nevertheless, several entries are missing or are filled in with a default value when they are not available. For instance, when the draught value is unknown, a '0' is assigned to the respective entry. This affects the quality of the data and requires careful manipulation.

5.2 AIS Monitor Data

This data is obtained from Port of Amsterdam's internal system and contains information on the berth occupancy in the Afrikahaven from January 2020 - May 2020. This data is also based on the AIS data, but adjusts the entries slightly to deal with inaccuracies in the AIS signal. For example, a duration of stay shorter than ten minutes is not counted as a stay (as it is deemed impossible for a large ship to (un)load within this time). Lastly, if a ship arrives at a berth, turns off its AIS transmitter, and turns it back on after a few days, the system assumes that this ship has stayed there for those days that no signal was sent.

Although this system seems to work well, one of the berths is not included in this dataset because this berth was completed later on and the system is not updated accordingly yet. This data can therefore be used to verify most of the duration of stay derived from the AIS data, but it cannot be used instead of it.

5.3 Besluit Aanwijzing Operationele Ruimte (BAOR)

Another data source made available by the Port of Amsterdam is the BAOR matrix. This matrix depicts multiple ship arrangements within the EVOS terminal of the Port of Amsterdam. Accordingly, each scenario is represented by one row. When it comes to the columns, the BAOR matrix has three, being the width of the ships moored at each of the three berths of the EVOS terminal. Additionally, there is another column that contains the sum of the total width per scenario. In

case the entry of the last column is larger than 111, then the last introduced width is highlighted in red, suggesting that the respective ship arrangement is not possible. The BAOR also specifies that as long as the sum of the total width of the constrained berths ≤ 106, there are no restrictions. Therefore, throughout this case, a maximum width of 106m is used.

5.4 EVOS Data

EVOS has provided data over the period of December 2019 - April 2020 that includes more detailed information on the operation, such as the type and quantity of cargo, the pump start and end times, and also the time that the Notice Of Readiness (NOR) was tendered. The latter is given off by a vessel once it has arrived to the waiting area and it is ready to enter the lock. The EVOS data can be matched up with the AIS data to get a more in-depth view of the entire process. For example, AIS data does not show how much of which cargo a ship carries; yet this can highly influence the duration of stay of a ship.

6 Methods

6.1 Deriving Duration of Stay

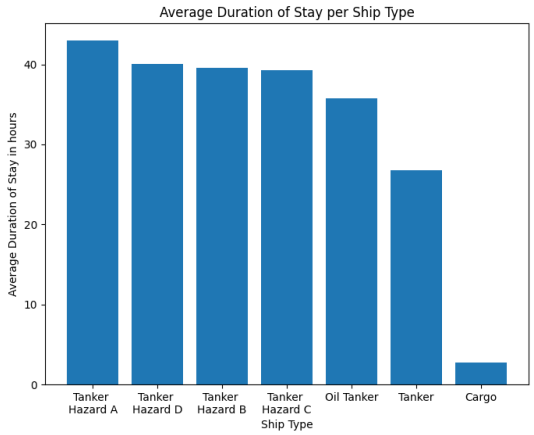
6.1.1 Duration of Stay from Historical AIS Data

A few steps have to be taken to get from the raw AIS data to the duration of stay of a particular ship. The first step in the first approach is to reduce the dataset to only contain the ships located in the EVOS terminal, and to only contain the ship types relevant for this study, such as the tankers and cargo ships. The next step is to group all signals sent out by one vessel (when it is at a certain berth) and look at the difference in time between two signals. If this difference is larger than five days, a ship is considered to have arrived again and the duration of stay is computed for its previous stay. This approach yields results, however, it does not take into account some real-time scenarios. First of all, in ship business, it is common for a ship to arrive at a terminal, discharge, go to another terminal for a day, and then return to that terminal again. Furthermore, it is possible for a ship to arrive at a berth, turn off its AIS signal, and then turn it back on after five days when it is about to leave again. Moreover, AIS signals are not always accurate, especially when a ship is moored. The current approach does not detect this either, which is why it is desirable to use another approach to obtain better results. This new approach also takes into account more data points in a trajectory of a ship than only the ones where it docked, to be sure if a ship has left or if it has turned off its AIS transmitter.

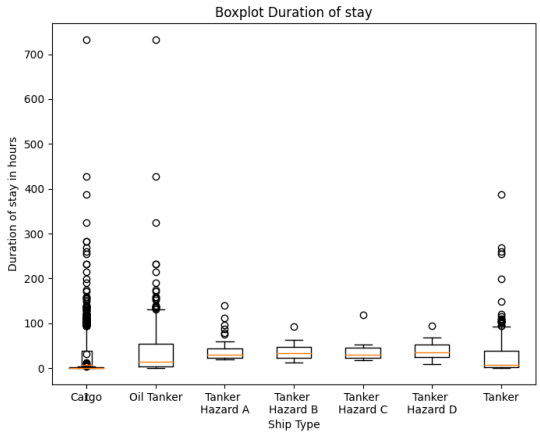
The new approach works as follows: it takes the five most recent AIS signals that a ship has sent out, and checks if the centroid of these locations is at a berth in the EVOS terminal. This method seems to perform well, as the vast majority of entries can be found within a 1.5 hour margin (this means, if a ship arrival or departure is recorded by EVOS at a certain time, the calculated arrival based on AIS is no more than 1.5 hours earlier or later). The entries that do not match up, are mostly entries that do not fall in the 1.5 hour margin or do not meet the search criteria; for this approach to record an entry, a ship has to send out an AIS signal repeatedly and the centroid of the last five points has to be within a berth. This means that if an AIS signal sent by a ship does not come from a berth repeatedly, no entry is recorded. Unfortunately, after saving the results it became clear that in many cases, several separate entries are recorded for a vessel instead of one that spans across the entire stay. Therefore, if there are several entries recorded for one vessel, yet the difference between the end of the last stay and the beginning of a new one is smaller than or equal to the threshold

(in this case set to 2 hours), the two entries are merged into one. An important assumption made is that the vessels that are taken into account in this data (mainly tankers and oil tankers) do not return to the EVOS terminal within two hours of their departure.

Once the duration of stay is calculated, the average duration of stay per ship type can be made, as shown in Figure 2a. Figure 2b shows the corresponding box plots per ship type. Figure 3 shows two distribution plots of the generated duration of stay, where Figure 3b is a zoomed in version of Figure 3a. It can be noted that ship types "Oil Tanker" and "Tanker" have many entries around 0, which could be due to the 'supporting' vessels that are only present at a berth to support other (larger) vessels. The stays of these vessels are also registered in the duration of stay when deriving it from AIS data.

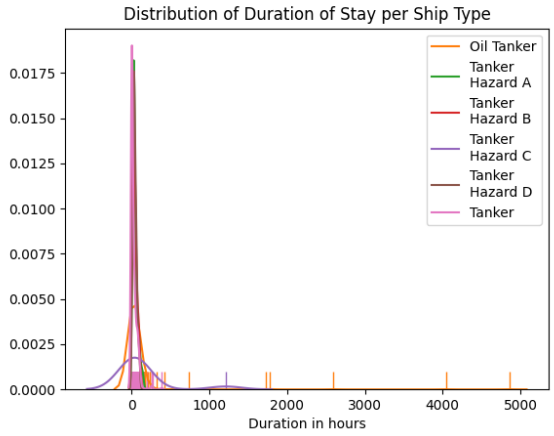


(a) Average Duration of Stay per Ship Type

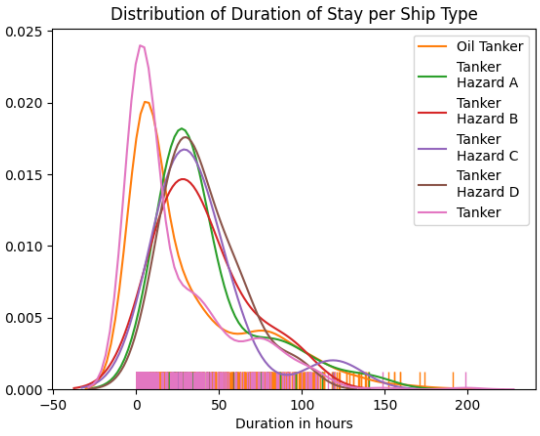


(b) Boxplot of Duration of Stay per Ship Type

Figure 2: Duration of Stay plots based on 2017-2020 AIS Data



(a) Density of Duration



(b) Density of Duration, zoomed in

Figure 3: Density Plots of Duration per Ship Type based on 2017 - 2020 AIS Data

The duration of stay derived can be compared with the AIS Monitor data and the EVOS data. Between the AIS monitor data and the generated duration of stay data, there are some notable

differences. The AIS monitor is more likely to record several entries for one stay of a ship, rather than one long one. Furthermore, the data that EVOS has provided records in a few cases that a ship has left the Jetty, yet in the AIS data that ship leaves much later.

6.1.2 Cleaning EVOS Data

The EVOS data is highly useful in verifying the deduced duration of stay from AIS data and gaining more insights in the vessels that arrive at the EVOS terminal. However, the provided data is rather raw and needs cleaning first. As the data contains more berths than this case focuses on, the data first needs to be filtered such that it only contains the berths relevant for this case. Furthermore, duplicate rows are removed, as well as rows that do not contain full information on a stay (hence rows where the name, arrival, or departure time is missing). Additionally, if a ship carries different products, the (dis)charging of each product is registered as a separate stay. These entries need to be merged as AIS data does not contain any information on the possible different product types that a vessel carries. This is done following the same principle as with the duration of stay, hence under the assumption that a ship returns within two hours to the same berth has not left the berth. In addition to updating the departure time, the number of steps (different kinds of product) is also updated. The latter is relevant for Section 6.1.4.

In order to compare the EVOS data with the generated duration of stay, the so-called masterfile is used to match the vessel name to an IMO, a unique ship identification number. In the process of adding an IMO to each entry, it was discovered that one ship sometimes has several IMOs registered, yet under slightly different spellings of the same name.

6.1.3 Overstay at EVOS

The calculated departure time of ships can be compared with the data provided by EVOS, for example by comparing `last pump end` and the departure as given in the generated duration of stay. It should be noted that once a ship finishes pumping, it is not immediately ready to leave. Therefore, a margin of six hours is taken to mark 'overstay'[1]. This means that only ships that have stayed longer than six hours after the last pump had ended are considered to overstay. For this, only ships that are present in both the EVOS data and the generated duration of stay are considered, hence 95 out of 123 ships in total. The other entries either do not fall within the arrival time boundaries (twelve hours before and twelve hours after the arrival time as given by EVOS), or are registered under different names in the EVOS and AIS data, or are not present in the generated duration of stay (this is possible when the centroid of the AIS signals sent out is continuously not in the berth). When overstay is calculated, it was found that in 52.5% of the cases, overstay occurred. A boxplot of overstay in hours is given in Figure 4.

[1] This margin is based on information given by EVOS.

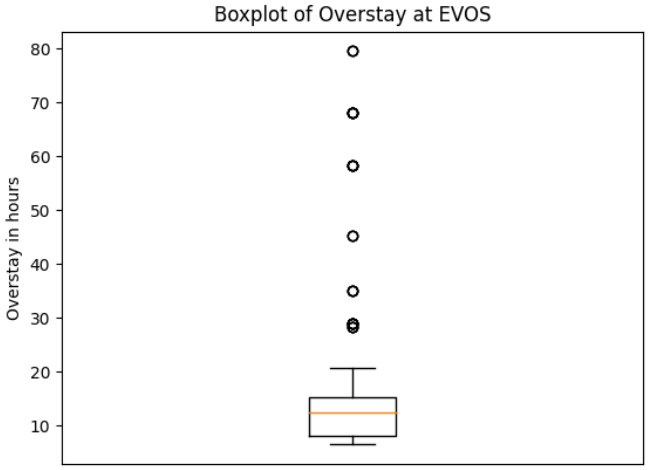


Figure 4: Boxplot of Overstay at Evos

6.1.4 Verifying the Estimation of the Terminal

The terminal (EVOS) uses a rule of thumb to estimate how long a ship will stay docked, namely: 1250 $m^3/h + 1$ extra hour for each step to be loaded. As the dimensions of each ship are known (A+B = length, C+D=width, draught), it can be verified if the rule of thumb used by the terminal is grounded. This is done by first computing the maximum volume of each vessel, then by estimating the duration based on those dimensions. This estimated duration is then compared to two things: the duration according to the deduced AIS duration of stay, and the duration based on pump start and end times. The results of this comparison is displayed in Table 1, where it can be seen that using the average duration per ship type as displayed in Figure 2a are the closest estimation of duration of stay based on the dimensions specified in the AIS data and the total duration of stay of a vessel.

	AIS Duration	Pump Duration	Average Duration
Mean	42.38	29.90	24.68
Median	20.97	14.49	4.43

Table 1: Mean and median difference in hours between estimated duration and actual duration

The main difference between the AIS duration and the Pump duration is that the pump duration is the time between the beginning of the first pump and the end of the last pump, whereas the AIS duration entails the entire stay of the vessel. As finishing pumping in most cases does not imply that a ship is (nearly) ready to leave due to maintenance, or due to waiting until a spot at another terminal is freed up, there is a noticeable difference between the pump duration and the AIS duration.

The results displayed in Table 1 might indicate that the rule of thumb is not valid, yet it should be realised that it is likely based on the product quantity and is used to estimate the operation

duration. The product quantity is specified in the EVOS data and could be used to estimate the duration of the pump time, which might produce a better estimation of the operation duration of a vessel than only using the dimensions as specified in the AIS data.

An important note, however, is that the AIS data does not include any information on the quantity of product that a vessel is actually carrying. The terminal does have this information, and hence would be in a much better position to estimate how long service would take. It should be noted that this rule of thumb is used to calculate how long service of a ship would take, yet that the actual duration of stay also depends on other factors, such as where the vessel goes next and (if it is seagoing) if there is space available in the lock planning. Therefore, although the rule of thumb might hold in terms of service time, it could still be inaccurate due to external factors.

6.2 Predicting Duration of Stay

6.2.1 Data Wrangling

Prior to the modeling phase, the data containing the duration of stay, generated from the AIS dataset, was explored and processed. First, it was discovered that some variables do not have the right type. Thus, they were converted accordingly. Then, the data points containing one of the dimensions equal to zero were filtered out, since it is impossible for a ship to have .

The next step of the wrangling phase involved deriving additional variables. In the beginning, the vessel's continent and country of origin were obtained from the first digit and the first three digits of the Maritime Mobile Service Identity (MMSI) respectively. Moreover, the Inmarsat type of boat was extracted through the last three figures of the MMSI. Besides this, a categorical variable defining the moment of the day when the ship has moored was created based on the mooring time. This feature has four categories, namely morning, afternoon, evening, and night. Also, the days of the week when the ships entered the berth were derived. Finally, the gross tonnage of the boats was obtained from `vesselfinder` by scraping the website, since Port of Amsterdam indicated it might have a large influence over the duration of stay. Unfortunately, the gross tonnage was unavailable for more than a third of the records in the data and, even if the respective feature had a high correlation of 0.83 with dimension A, it was discarded as its imputation based on A would add extra variability to the prediction model.

The last part of the data wrangling addressed the one-hot encoding technique. This was used to transform categorical variables with k categories into k features. The reason for applying this technique is that some regression algorithms are not able to operate with multi-categorical variables.

6.2.2 Prediction Model

During the modeling phase, four Machine Learning models were developed to predict the duration of stay: Linear Regression, Random Forest, Gradient Boosting, and Neural Network. The initial judgment behind building four models was to compare them and determine which one generates the most accurate predictions or, in other words, which has the lowest root-mean-square error (RMSE).

Firstly, the previously processed dataset was randomly split in training and test set, such that the precision of the models could be justly assessed. Unfortunately, during this phase it was discovered that the size of around 1200 instances is extremely small for a Machine Learning method to learn

the underlying patterns from the training data and have enough records left to test the model, considering that it is recommended that the test data has at least 500 records [3]. Accordingly, the goal shifted towards providing a ranking of the four models given multiple samplings, as well as offering insights into the way their prediction power can be improved. In other words, for each sampling of training and test data, the four models were built and their RMSE was compared to see if there is a model that performs generally better.

The first model built was Linear Regression. In order to select the explanatory features, the Backward Elimination technique, also known as the Step-down method, was used. This wrapper method works in the following way: it first computes the Linear Regression considering all the variables, testing each in a t-test. If the largest p-value of a feature is greater than the significance level (0.05), then the variable is removed. The entire algorithm is performed repeatedly until this is no longer the case. Consequently, this method is executed during the training phase to select the best features for the Linear Regression method [6].

The second model is the Random Forest. Since this model allows multi-categorical variables, two variations were formulated: one that includes the MMSI of the vessels and one that does not. First, the models were performed given 1000 trees in the forest as the only hyperparameter. Afterward, the following six hyperparameters were tuned using the Random Search method: the number of trees, the number of features considered for the best split, the minimal size of the terminal nodes, the minimum number of samples required to split an internal node, and the possibility to bootstrap samples when building a tree. The Random Search method works by randomly selecting one element from the given options for each hyperparameter. Next, it splits the training data into five parts and consecutively trains the Random Forest with the given parameters on four parts of the data, while validating the performance on the fifth part. Finally, the five scorings are averaged and stored as a result of a single iteration. After all the 20 iterations are computed, the Random Search method returns the hyperparameters of the model that performed best.

The next Machine Learning model built was the Gradient Boosting. In a similar fashion as the Random Forest, the model was first tested on three default parameters, the number of boosted trees (1000), the number of leaves (1000), and the learning rate (0.05). Afterward, the Random Search tuning method was used again to check the best performing hyperparameters on the validation data. The parameters tuned this time were the learning rate of the model, the number of leaves for base learners, the number of boosted trees, and the maximum tree depth.

The last model is a Neural Network with four layers, including the input and the output ones. The first three layers have 32 neurons each and 'relu' activation. The model has 'adam' activation and mean-square error loss. Before the Neural Network was fed with the train data, this data was first normalised to change the values to a general scale, keeping the ranges between values. Without this technique applied to the data, the learning would have oscillated as a result of the varying gradients. After the creation of this model with a batch size of ten and 100 epochs, the Grid Search method was used to tune these two hyperparameters. The tuning method works in a similar fashion with the Random Search, however, the first scheme tries all the combinations of the given finite set of hyperparameters, validating their prediction power using five-fold cross-validation.

Algorithms Data sampling	Random State 1	Random State 2	Random State 3	Random State 4	Random State 5
Linear Regression with Feature Selection	62.68	346.42	311.81	60.15	58.85
Random Forest (MMSI)	80.95	329.05	329.04	192.85	80.18
Random Forest (MMSI) with hyperparameter tuning	59.88	341.06	316.57	68.25	58.93
Random Forest (/MMSI)	72.11	329.39	325.37	199.72	50.52
Random Forest (/MMSI) with hyperparameter tuning	59.55	340.55	315.71	67.16	56.01
Gradient Boosting	126.05	321.28	346.83	244.31	116.43
Gradient Boosting with hyperparameter tuning	82.7	322.68	322.44	149.27	66.83
Neural Network	136.56	343.83	321.91	118.46	88.95
Neural Network with hyperparameter tuning	61.35	344.89	312.61	68.09	54.29

Table 2: RMSE of each model of the test set

Table 2 presents the RMSE of each model over the test dataset given a different sampling when splitting the data in the training and test sets. It can be seen that there is no clear pattern with regards to which model performs generally better. Moreover, the model with the tuned hyperparameters performed worse than the basic one multiple times. This can be explained by the bad choice of options or their bad sampling. One possible solution to this issue is switching the tuning method to Grid Search, which tries all the combinations, unlike the Random Search, which randomly selects cases. Moreover, given more computation power, several options per hyperparameter can be tried such that the quality of the hyperparameters is ensured.

Another pattern that can be noticed from the table is that the Linear Regression performs remarkably well considering its simple computations compared to the complexity of other models. However, by looking at Figure 5 that displays the estimated durations of stay next to the real values of the test set, it can be seen that Linear Regression allows the prediction of negative values; something that did not occur with other models.

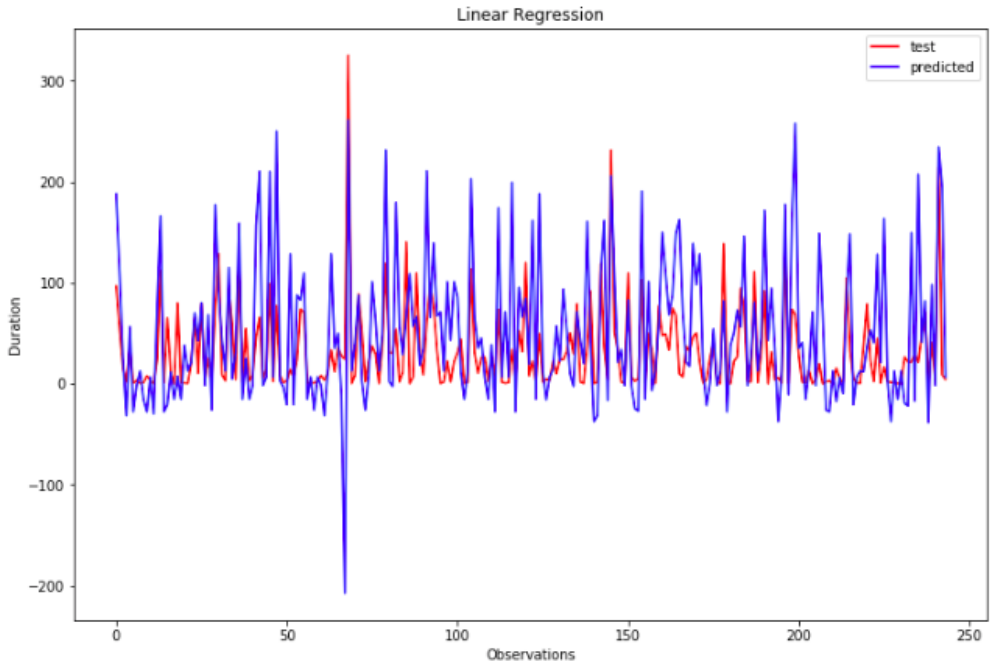


Figure 5: Predictions of the Linear Regression next to the real values

The most important conclusion to be drawn from this analysis is that the lack of data does not allow a stable training and testing environment for a prediction model. Hence, acquiring more data, either by using the algorithm which extracts the duration of stay from the AIS data for terminals other than EVOS, or by using other sources, is the first step that needs to be taken towards building an accurate prediction model. Moreover, adding weather data has the potential to improve the prediction, since it was often noted by the Port of Amsterdam and EVOS employees that weather plays an important role in the duration of stay. When it comes to the technical part of the models, better tuning of hyperparameters is recommended, especially for the Neural Network, which only considers the batch size and the number of epochs. Accordingly, the number of layers, the number of neurons per layer, their activation, and the model optimiser can also be tuned.

6.3 Berth Occupancy

To support the decision-making process of the harbour master, it would be very useful to know what is happening in the harbour at any given time. A great way to present information in a clear and intelligible way is by means of visualisation. The available shapefiles of the harbour and the coordinates of the berths were used to produce Figure 6. This example shows the current and recent locations of a single ship in the EVOS terminal on a certain day. Whenever a new AIS signal comes in, the graph can be updated to reflect the most current situation at the terminal.

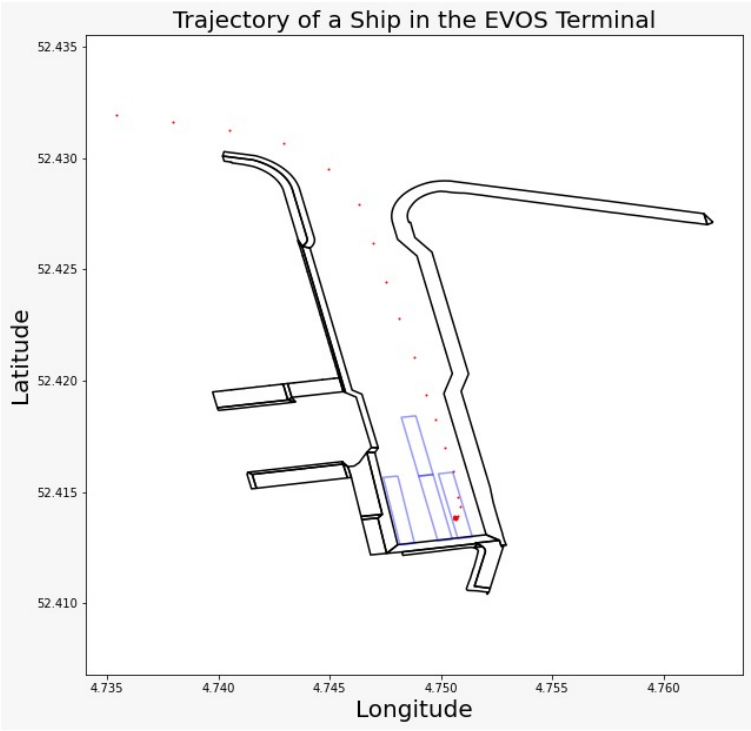


Figure 6: Trajectory of A Ship in The EVOS Terminal

6.4 The Berth Allocation Problem

6.4.1 Scheduling

Generally, scheduling problems are described with a machine that can process jobs (j), where an optimal schedule is to be found for a given objective function. These problems can be written in the form $\alpha|\beta|\gamma$, where α, β and γ represent the characteristics of the machine(s), job(s) and objective function respectively. Characteristics of the machines include how many machines there are and if they are identical or not. If machines are identical, the processing of jobs takes the same amount of time on each machine (parallel machines, P), whereas if machines are not similar, the processing time for a job on one machine differs from the processing time on another machine (unrelated machines R). Regarding jobs, these sometimes have restrictions, for example when they have to be completed before their due date (d), or when they can only be processed after their release date (r). There are also different objective functions, such as minimising the sum of the completion time of each job (C_j), or minimising the maximum completion time, where the completion time is the starting time of a job (s_j) + the processing time of a job (p_j) [14].

Using the theory as described above, the berth allocation problem can be formulated as a parallel machine scheduling problem. It is defined as $P|r_j|\sum C_j$. P indicates that there are parallel machines; r_j indicates the release dates of the jobs; $\sum C_j$ is the objective function that aims to minimise the sum of the completion times of the jobs. In this scenario, the machines can be thought of as the berths, with four parallel machines representing the four berths from EVOS. The jobs are then the ships, where their duration of stay is their processing time p_j. This model also takes into account the (estimated) arrival times as the berth allocation is dynamically processed: a ship is only ready to dock once it has arrived. Thus the Estimated Time of Arrival (ETA) of a ship can be seen as the release date (r_j) of a ship.

In addition, since the EVOS terminal has a more sophisticated spatial restriction −namely the triangular layout of three of the four berths−, the allocation model stays discrete. This means only one vessel is allowed to dock at one berth at a time. Furthermore, whenever the service of a vessel begins (i.e. whenever it is docked), it cannot be disrupted. To therefore efficiently use the dock, the main aim is to optimise the berths usage rate, which is equivalent to minimising the total completion time of all jobs, or minimising the last completion time over all jobs.

6.4.2 Problem Formulation

The whole berth allocation problem can be denoted as a general $P|r_j|\sum C_j$ (or $P|r_j|C_{max}$) scheduling problem. To solve the problem, it is treated as a linear programming problem and is formulated as follows [16] [17]:

$$\text{Minimise} \quad \sum_{j=1}^{n} C_j \tag{1}$$

$$\text{Or Minimise} \quad C_{max} \tag{2}$$

$$\text{s.t.} \quad \sum_{i=1}^{4} x_{ij} = 1 \qquad (j = 1, 2, ..., n) \tag{3}$$

$$s_j \geq a_j \qquad (j = 1, 2, ..., n) \tag{4}$$

$$s_{j'} \geq s_j \tag{5}$$
$$(j, j' = 1, 2, ..., n \quad \text{s.t. } a_j \leq a_{j'})$$

$$s_j \geq a_j - e_j \qquad (j = 1, 2, ..., n) \tag{6}$$

$$C_{max} \geq s_j + p_j \qquad (j = 1, 2, ..., n) \tag{7}$$

$$s_j \geq (r_i + B)x_{ij} \qquad (j = 1, 2, ..., n; i = 1, 2, 3, 4) \tag{8}$$

$$s_{j'} \geq s_j + p_j - M(1 - I_{ijj'}) \tag{9}$$
$$(j, j' = 1, 2, ..., n \quad \text{s.t. } j \neq j'; i = 1, 2, 3, 4)$$

$$I_{ijj'} + I_{ij'j} \leq \frac{1}{2}(x_{ij} + x_{ij'}) \tag{10}$$
$$(j, j' = 1, 2, ..., n \quad \text{s.t. } j < j'; i = 1, 2, 3, 4)$$

$$I_{ijj'} + I_{ij'j} \geq x_{ij} + x_{ij'} - 1 \tag{11}$$
$$(j, j' = 1, 2, ...n \quad \text{s.t. } j < j'; i = 1, 2, 3, 4$$

$$d_j x_{1j} + d_{j'} x_{2j'} + d_{j''} x_{3j''} \leq 106 \qquad (j, j', j'' = 1, 2, ..., n) \tag{12}$$

$$x_{ij} \in \{0, 1\} \qquad (j = 1, 2, ..., n; i = 1, 2, 3, 4) \tag{13}$$

$$I_{ijj'} \in \{0, 1\} \qquad (j, j' = 1, 2, ..., n \quad \text{s.t. } j \neq j'; i = 1, 2, 3, 4) \tag{14}$$

With variables:
n vessels (i.e., vessels $1, 2, ..., n$)
4 berths (i.e., berths 1, 2, 3, 4), where berth 4 is not part of the triangular berth layout
s_j: the starting processing time of vessel j , $s_j \geq 0$
p_j: the given processing time of vessel j (duration of stay of vessel j), $p_j > 0$
a_j: the arrival time of vessel j (ETA of j), $a_j \geq 0$
C_j: the completion time of vessel j, $C_j > 0$
C_{max}: the last completion time among all vessels, $C_{max} > 0$

d_j: the width of vessel j, $d_j > 0$
e_j: the earlier arrival margin of vessel j in hours, $0 \leq e_j \leq 48$
r_i: the remaining docking time of the currently docked vessel at berth i, $r_i \geq 0$
B: the constant buffer time between each two vessels' berthing, $B > 0$
M: a big constant number

The objective functions:
(1) Minimise the sum of completion times over all ships
(2) Minimise the maximum completion time among all ships

with constraints:
(3) Requires each vessel to be assigned to one berth
(4) Requires that each vessel can start its processing only after it has arrived at the terminal
(5) Requires that vessel j' to be scheduled after j given that $a_j \leq a_{j'}$ (only active when the model enforces a FCFS strategy)
(6) Applies arrival relaxation, meaning vessels can be scheduled up to e_j (48) hours earlier than their (a_j) (only active when a_j relaxation is allowed)
(7) Ensures C_{max} is the last completion time among all vessels (only active when the objective function is (2))
(8) Ensures vessels are scheduled only after the vessels that are currently at the terminal have left
(9) If vessels j, j' are both assigned to berth i and vessel j is processed before vessel j' (i.e. $I_{ijj'} = 1$), then the start time of vessel j' must be no earlier than $s_j + p_j$
(10) & (11) Ensure that one of the $I_{ijj'}$ and $I_{ij'j}$ equals 1 if vessels j and j' are both assigned to berth i. They also ensure that $I_{ijj'} = I_{ij'j} = 0$ if one of vessels j and j' is not assigned to berth i
(12) Ensures the spatial constraint under the regulation of the terminal's triangular berth layout
(13) $x_{ij} = 1$ if vessel j is assigned to berth i; $x_{ij} = 0$ otherwise
(14) $I_{ijj'} = 1$ if vessels j, j' are both assigned to berth i and vessel j is processed before vessel j'; $I_{ijj'} = 0$ otherwise.

In the scheduling, a buffer of two hours is added. The duration of stay in principle already includes the time for maintenance, checking, cleaning, etc., but the buffer is necessary to allow vessels that finish their service to depart the terminal and vessels that will start their service to arrive to the terminal. Furthermore, in order to make this model not EVOS-oriented, constraints (12) can be omitted or replaced by another terminal-specific constraint.

6.4.3 Different Model Types

The model is created such that four types arise:

1. First come, first serve (FCFS) with fixed arrival times
2. No FCFS with fixed arrival times
3. No FCFS with 48 hour relaxation, meaning that ships can be scheduled up to 48 hours earlier than their estimated arrival
4. No FCFS with complete relaxation, meaning that ships can now be scheduled anytime before their estimated arrival.

These types are introduced to display the impact of adjusting the current scheduling policy. Furthermore, two different objective functions are used to represent the two different perspectives: the terminal and the client perspective. The objective function $\sum C_j$ minimises the sum of the completion times over all vessels and would hence fit the client perspective. Since the duration of stay is fixed, vessels will be scheduled as early as possible, meaning the waiting times for vessels overall are minimised. On the other hand, the objective function C_{max} minimises the maximum completion time, which fits the terminal perspective to handle as many vessels as possible in the same amount of time. In both cases of the objective function, the waiting time for individual vessels could in some cases increase significantly, even though the average waiting time decreases.

7 Results

The linear model as described in section 6.4.2 is solved using `PuLP`, a linear solver package in Python. An outline of the code can be found in Appendix D. The results are split up into two scenarios, being the same ships as in the past arriving at the EVOS terminal, and the potentially larger ships arriving at the terminal after the new sea lock is used. The data used here is the derived duration of stay data, matched up with the EVOS data over the time period January 2020 until April 2020. This matching was necessary as the duration of stay data also showed ships that supported larger ships (tugboats), for example with ship-to-ship operations. As the model should only schedule the incoming seagoing vessels, only ships were included that had a recorded stay in both the generated duration of stay and the EVOS data.

For both objective functions, the model runs for a 14-day period as the time frame input where it could test different scheduling possibilities. In order to best simulate the berth allocation and avoid specificities caused by certain time frames, each day from Dec 2019 to April 2020 is used as an entry for the model. Both scenarios (regular sized & larger ships) are tested with four types of model (optimal FCFS, No FCFS, 48hr relaxation, and complete relaxation) for two objective functions ($\sum C_j$ & C_{max}). Detailed plots can be found in Appendix B. In these plots, a sharp decrease of ships berthed can be noted during mid February. This is due to a server error at CWI, meaning there is no AIS data available for this time period (16-27 February 2020). In order to maintain the rolling time window, it was decided to leave those entries in.

A crucial step is applied to ensure a fair comparison with historical data and show the results of our models are at least as good as FCFS. Instead of simply summing up the completion times of all ships newly scheduled within the given time frame, only ships actually berthed during this time period according to the historical data are again scheduled by the model within the time frame. For example, a ship with an ETA on the last day of the 14-day period can according to the model be scheduled at this day, yet historically this ship actually berthed a day later, outside of our time frame. Consequently, this resulted in a substantial deviation to the objective value. Therefore, only scheduling the same ships that docked in the past could avoid such a problem.

Another crucial step is that for the simulation of historical schedules, the NOR is used as arrival time instead of the ETA, as the ETA from AIS data is highly unreliable. This step, in addition to the step described previously, is essential for a fair comparison between historical values and the optimal values according to the model.

7.1 Simulation with Regular Sized Ships

For regular sized ships, the average of summed completion times is taken as the judging criteria of the model's performance, which is in a format of percentage with reference to historical AIS data (shown in Section 7.1). An inspection of changes in individual waiting times is also performed. Some vessels are scheduled earlier than historical allocation and some are rescheduled later consequently. In the corresponding model, the overall individual waiting time is synergistically varied with the objective function.

Obj. Func.	Historical AIS	Optimal FCFS	No FCFS	48hr Relaxation	Complete Relaxation
$\sum C_j$	100%	79%	79%	64%	39%
C_{max}	100%	88%	87%	75%	42%

Table 3: Percentage of Average Objective Value Compared to Historical AIS Objective Value (lower is better)

Comparison with historical FCFS berthing already shows an outperformance with the new, optimal FCFS model, with 21% and 12% improvements in $\sum C_j$ and C_{max} respectively. No FCFS model has almost the same objective value as optimal FCFS does. This is because both models have fixed arrival times and vessels can only be docked after they have arrived. As a result, not many new schedules could be rearranged. In contrast, no FCFS models with 48 hours and complete relaxation could save significant time in both objective functions.

7.2 Simulation with Larger Ships

To simulate the arrival of larger ships, the same input is taken as for the regular sized ships, only a portion of 25%, 50% and 75% of the ships are enlarged (with a random seed of 8) to the theoretical maximum sizes that the EVOS terminal can handle (meaning dimensions of $285 \times 48 \times 14.05$). The duration of stay for these ships is set according to the rule of thumb as provided by EVOS: $1250m^3/h+$ 1 extra hour for each step to be loaded (in this case 2 hours to have an extra margin).

Since larger ships will only arrive after the new sea lock has become operational, performances shown in Table 4 and Table 5 are compared with the optimal FCFS model, which is initialised as 100%. The No FCFS model generates similar results as optimal FCFS because of the same reason as mentioned in 7.1. Likewise, smaller objective values are obtained in no FCFS models with relaxations in all scenarios, meaning abandoning the FCFS principle and allowing ships to come earlier could shorten the overall berthing time.

% of Ships Enlarged	Optimal FCFS	No FCFS	48hr Relaxation	Complete Relaxation
25	100%	99.8%	83.7%	58.9%
50	100%	97.0%	84.5%	65.0%
75	100%	98.0%	87.6%	70.3%

Table 4: Percentage of Average Objective Value $\sum C_j$ with Reference to Optimal FCFS Model (lower is better)

% of Ships Enlarged	Optimal FCFS	No FCFS	48hr Relaxation	Complete Relaxation
25	100%	100%	88.4%	55.7%
50	100%	100%	88.0%	62.5%
75	100%	100%	90.5%	66.1%

Table 5: Percentage of Average Objective Value C_{max} with Reference to Optimal FCFS Model (lower is better)

7.3 A Real Time Model

As the results have demonstrated there are theoretical improvements that can be made when optimizing berth scheduling using our model. However, in order to show that this is also applicable in day to day activities a real time more user-centric model had to be designed. In addition, the Port of Amsterdam had expressed that the model should not aim to replace the harbour master, but provide valuable assistance in the decision making process. Therefore, it was essential to firstly understand the needs and responsibilities of our target user: the harbour master.

In order to build up this knowledge, collaboration with port authorities during the entire development process was necessary and provided a clear and concise list of requirements for our model. This meant that our model had to have these features:

- Real time overview of incoming ships
- Real time overview of terminal usage
- Real time scheduling options (Assistance in berth allocation)

Based on these requirements a web based dashboard as displayed in Figure 7 was designed and built using Flask, a micro web framework written in Python. Flask was chosen with future development in consideration, since it is modular and expandable. The dashboard provides information on incoming ships and the current situation using tables. This is further assisted by the use of embedded real time maps from vesselfinder that give a geographical perspective and live tracking for incoming ships and already docked vessels. Moreover, the dashboard contains a widget that displays the current width of the geometrically constrained berths in the terminal. This widget also functions as a calculator that will turn red when the width given as input + the current total width of the geometrically constrained berths exceeds the maximum width. This widget automatically resets after three seconds, ready for new input. Lastly, our model is used to provide different schedules based on the different model types (1-3) discussed in Section 6.4.3. Consequently, this provides the harbour master with an overview and some recommendations that aim to help and streamline the process of managing terminal operations.

The three schedules that are displayed are all optimal schedules, but choosing the right schedule for the right situation depends on the set of rules that hold. If scheduling is only allowed under a FCFS policy, the optimal FCFS schedule is the best. If no FCFS policy is set and vessels cannot arrive earlier than their ETA, the No FCFS model is optimal. Lastly, if vessels can arrive earlier than their ETA, the 48 hour relaxation model is optimal. The fourth model (complete relaxation) is excluded from the options, as this is the least realistic model.

It is important to note that the dashboard only imitates a real time model and can thus be seen as a proof of concept. The dashboard provides all the technical front facing functionalities, however currently depends on historical data as a placeholder and therefore cannot be called real time. In order to provide a real time experience a significant amount of data engineering and integration work with existing port systems would still need to be carried out. Furthermore, in real time, the model would rely on the prediction of duration of stay of vessels and the ETA as communicated to the Port by the vessels.

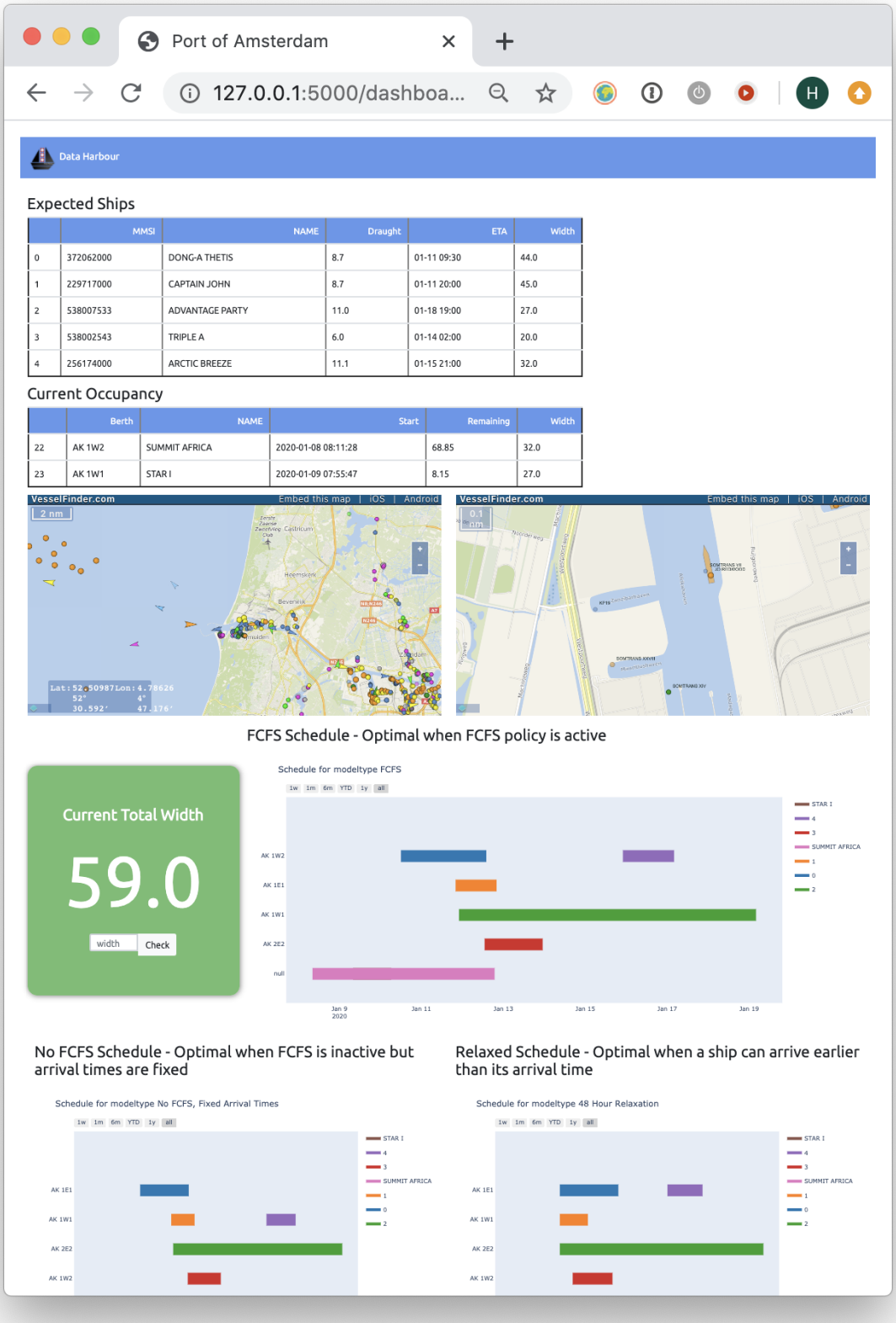


Figure 7: Preview of the Dashboard

8 Discussion

8.1 Difficulties

During the research it became evident that the problem was more complex than originally expected. There are multiple external factors which are hard to incorporate in the allocation model. Sometimes, tugboats might not be available, making it impossible for vessels to follow the schedule. Also, weather can have a drastic impact on efficiency. Furthermore, the planning of the EVOS terminal is highly dependent on the planning of the sea lock at IJmuiden. Additionally, when oil companies have multiple ships in the EVOS queue they can decide to switch them around. To fully disrupt the schedule, they can even decide to postpone entering the harbour when oil prices are lower than desired. Lastly, some parts of the data like the ETAs are entered by hand, which makes them unreliable and therefore unfit for use. Many of these obstacles could in the future be overcome if more data is collected and incorporated in the allocation model. More data would also greatly improve the quality of the prediction model.

8.2 Other Objective Functions

In addition to the two objective functions discussed in this research paper, other objective functions were considered and could be investigated in the future. In particular, minimising the sum of the maximum completion times of each berth could be interesting to examine. Instead of only looking at the maximum completion time of a single berth, all berths would constitute to the final objective value. Ultimately, a combination of all three mentioned objective functions might generate the best schedules. However, this would require more computational power as well as add complexity to the model, as objective values would be harder to interpret.

8.3 Further Research

A further implementation of applying weather analysis is of great importance as the weather condition highly affects the speed of ships. Extreme weather like storms will extend the duration of stay of berthed vessels, nevertheless, following wind may speed up a ship's journey and therefore arrive earlier than the estimated time. Additionally, incorporating market analysis of relevant cargo types also has significant impact. It occurs frequently that oil tankers have arrived on time at the waiting area, yet they decide to wait there for longer until the oil prices have a desirable rate, which can disrupt the scheduling. Thus, taking the market factor into consideration could contribute to a higher quality of the simulation model. On the other hand, further research on improving the accuracy of the ETA could result in a more robust schedule. Lastly, further research on predicting the duration of stay (accompanied with more data) is necessary to produce more accurate predictions. The more accurate these two factors are, the more accurate the real-time model will be.

The dashboard could be extended by several features, including the addition of a line graph that displays the total width over time in the Gantt charts that display the schedule. Furthermore, instead of a current width displaying widget, the "total allowed width" could be displayed, showing the harbour master how much space is still available.

9 Conclusion

This research has shown that AIS data is a very powerful source for spatial analysis. In this case it is first used to derive the berthing time of vessels staying at the Port, after which a prediction model was made to predict how long the ships stay at a certain berth. Moreover, AIS data could especially be of use when tracking inland vessels, something that could be of great importance for improving efficiency in terminals.

Furthermore, it has become clear that the current first come, first serve berth allocation strategy works, but can be improved. Especially when larger ships arrive at the port −something expected to happen when the new lock becomes operational−, a more efficient way of scheduling is necessary to make more efficient use of the terminal. The first come, first serve model as constructed in this case can be useful when a first come, first serve policy still holds, but this case has also shown alternatives for when there is no FCFS policy, or when the arrival times of ships is relaxed.

When larger ships arrive at the port, the first come, first serve allocation strategy will not be the most efficient berth allocation strategy anymore. Efficiency can be gained by abandoning the first come, first serve policy, yet realistically most efficiency can be gained by also allowing ships to be scheduled before their estimated arrival time. Although it might seem unrealistic to let ships arrive before their estimated arrival times, this also highlights the importance of having a reliable ETA: the more reliable the ETA of a vessel is, the more reliable the allocation model works. Similarly, the importance of good prediction of duration of stay is essential to having a reliable real time allocation model.

The main aim of this research was to investigate if there are other ways for berth allocation that optimise the total berth usage. Furthermore, a main goal was to create a model that *supports* the harbour master in taking berth allocation decisions. Therefore, the created dashboard shows the different schedules generated by the model in a way where the final decision of allocation is always made by the harbour master.

10 Acknowledgements

We would like to thank Joost Zuidema, Pieter Valk, and Hendrik-Jan Oost from Port of Amsterdam for their guidance, and for answering all of our questions. Being an international team, we sometimes had some difficulties getting our points across, but eventually all worked out very well.

Furthermore, we would like to thank Paul Dekker from EVOS for taking the time to answer our questions and for providing us with data about the EVOS terminal.

Lastly, we would like to thank our supervisor Elenna Dugundji for all of her time and dedication to helping us out and brainstorming with us. This has helped us tremendously and has ensured we kept on track.

References

[1] Christian Bierwirth and Frank Meisel. "A survey of berth allocation and quay crane scheduling problems in container terminals". In: *European Journal of Operational Research* 202.3 (2010), pp. 615–627. DOI: doi.org/10.1016/j.ejor.2009.05.031.

[2] Christian Bierwirth and Frank Meisel. "A follow-up survey of berth allocation and quay crane scheduling problems in container terminals". In: *European Journal of Operational Research* 244.3 (2015), pp. 675–689. DOI: doi.org/10.1016/j.ejor.2014.12.030.

[3] Peter Bloem. *Lecture 3, Methodology for Model Evaluation*. Machine Learning (X_400154), Vrije Universiteit Amsterdam. 2020.

[4] Gerald G Brown, Siriphong Lawphongpanich, and Katie Podolak Thurman. "Optimizing ship berthing". In: *Naval Research Logistics (NRL)* 41.1 (1994), pp. 1–15. DOI: 10.1002/1520-6750(199402)41:1<1::AID-NAV3220410102>3.0.CO;2-L.

[5] D Clements et al. "Heuristic optimization: A hybrid AI/OR approach". In: *Workshop on Industrial Constraint-Directed Scheduling*. 1997. URL: https://www2.isye.gatech.edu/~ms79/publications/cp97.pdf.

[6] Dennis Dobler. *Lecture 12, Multiple Linear Regression*. Statistical Data Analysis (X_401029), Vrije Universiteit Amsterdam. 2019.

[7] Michael R Garey and David S Johnson. *Computers and intractability*. Vol. 174. freeman San Francisco, 1979. ISBN: 0-7167-1044-7.

[8] Akio Imai, Ken'Ichiro Nagaiwa, and Chan Weng Tat. "Efficient planning of berth allocation for container terminals in Asia". In: *Journal of Advanced transportation* 31.1 (1997), pp. 75–94. DOI: doi.org/10.1002/atr.5670310107.

[9] Akio Imai, Etsuko Nishimura, and Stratos Papadimitriou. "The dynamic berth allocation problem for a container port". In: *Transportation Research Part B: Methodological* 35.4 (2001), pp. 401–417. DOI: 10.1016/S0191-2615(99)00057-0.

[10] Akio Imai, Etsuko Nishimura, and Stratos Papadimitriou. "Berth allocation with service priority". In: *Transportation Research Part B: Methodological* 37.5 (2003), pp. 437–457. DOI: 10.1016/S0191-2615(02)00023-1.

[11] Nataša Kovač. "Metaheuristic approaches for the berth allocation problem". In: *Yugoslav Journal of Operations Research* 27.3 (2017), pp. 265–289. DOI: 10.2298/YJOR160518001K.

[12] Andrew Lim. "The berth planning problem". In: *Operations research letters* 22.2-3 (1998), pp. 105–110. DOI: doi.org/10.1016/S0167-6377(98)00010-8.

[13] Andrew Lim et al. "Crane scheduling with spatial constraints". In: *Naval Research Logistics (NRL)* 51.3 (2004), pp. 386–406. DOI: 10.1002/nav.10123.

[14] Rene A. Sitters and Joaquim Gromichio. *Lecture 4, Scheduling*. Combinatorial Optimization (X_401067), Vrije Universiteit Amsterdam. 2020.

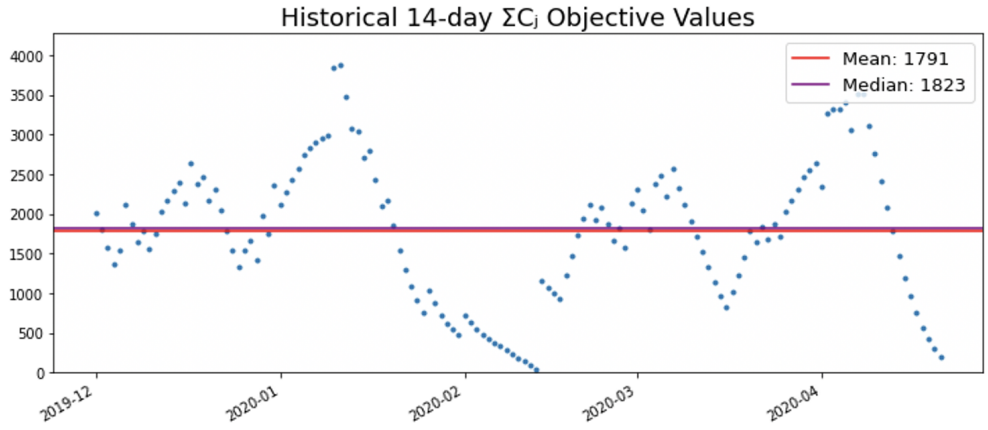
[15] Tristan B Smith and John M Pyle. "An effective algorithm for project scheduling with arbitrary temporal constraints". In: *AAAI*. Vol. 4. 2004, pp. 544–549. URL: https://www.aaai.org/Library/AAAI/2004/aaai04-087.php.

[16] Nitish Umang, Michel Bierlaire, and Ilaria Vacca. "Exact and heuristic methods to solve the berth allocation problem in bulk ports". In: *Transportation Research Part E: Logistics and Transportation Review* 54 (2013), pp. 14–31. DOI: 10.1016/j.tre.2013.03.003.

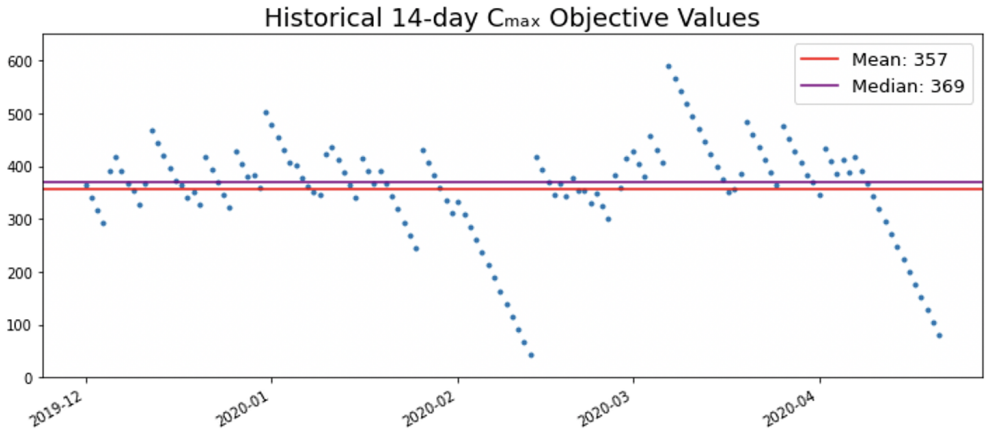
[17] Dongsheng Xu, Chung-Lun Li, and Joseph Y.-T. Leung. "Berth allocation with time-dependent physical limitations on vessels". In: *European Journal of Operational Research* 216.1 (2012), pp. 47–56. DOI: 10.1016/j.ejor.2011.07.012.

Appendices

A Objective Values of Historical Data



(a) Historical Objective Values of function $\sum C_j$



(b) Historical Objective Values of function C_{max}

Figure 8: Historical Objective Values of Two Functions

B Objective Values of Models – Regular Sized Ships

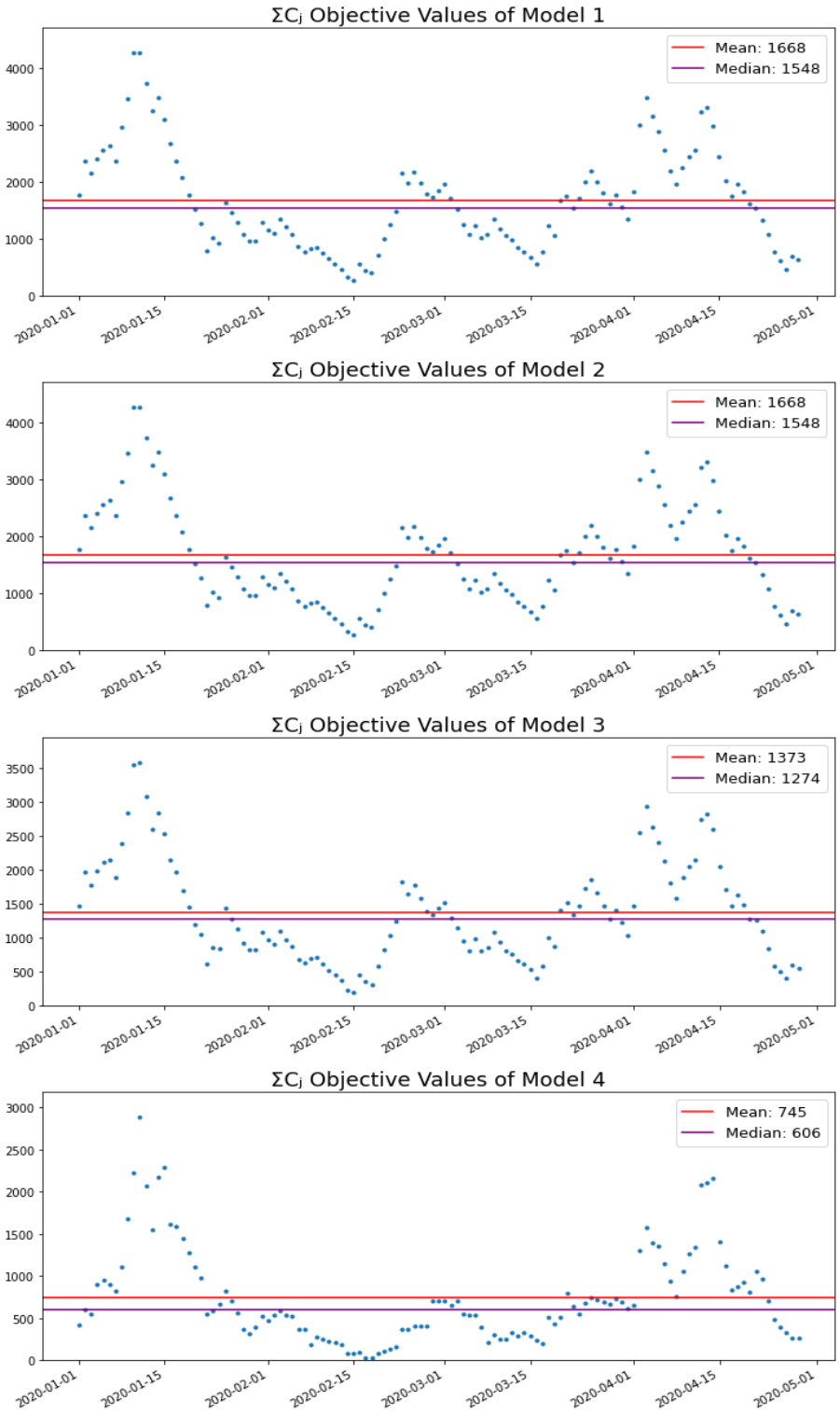


Figure 9: Objective Values of function $\sum C_j$ the Four Modeltypes

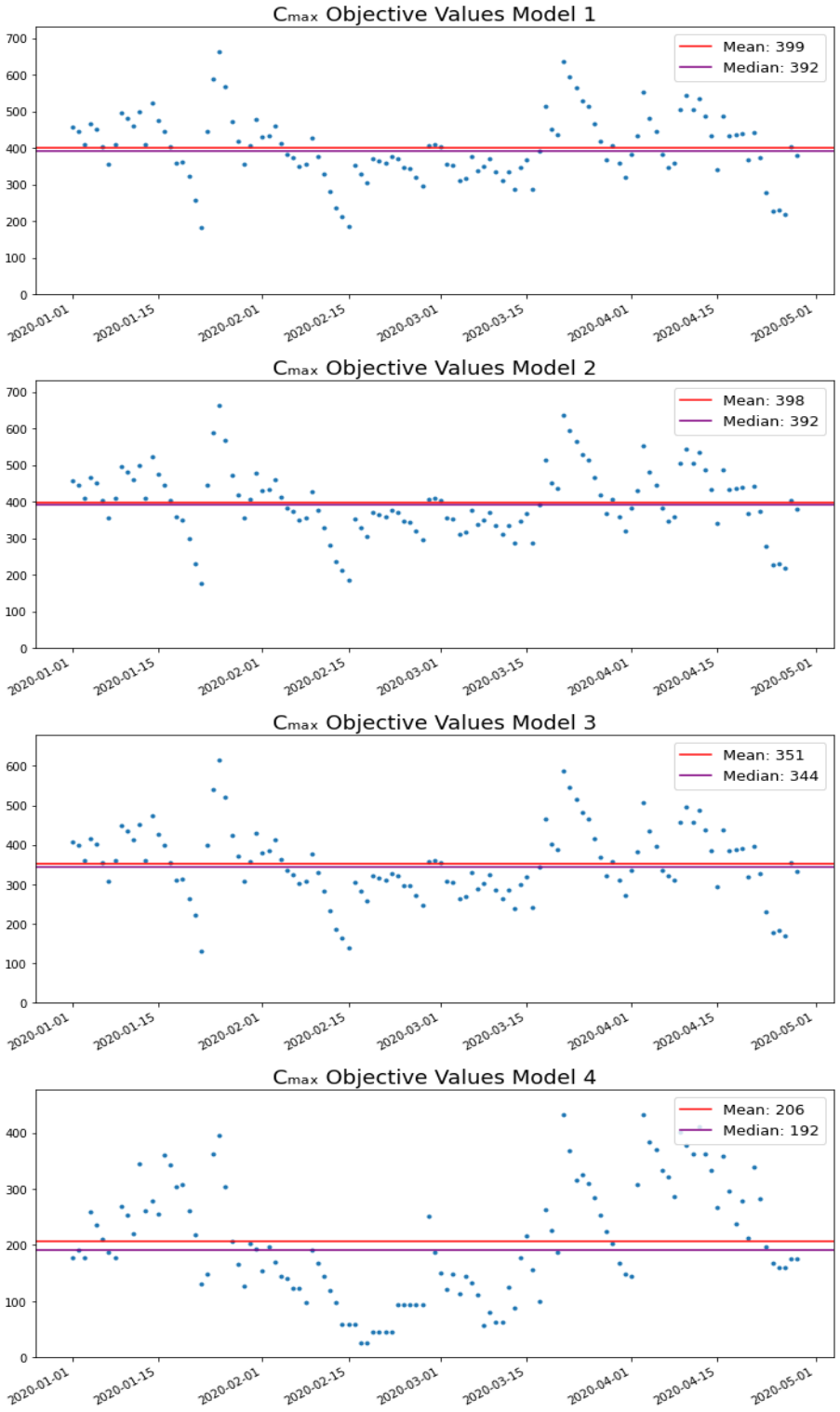


Figure 10: Objective Values of function C_{max} the Four Modeltypes

C Objective Values of Models - Larger Ships

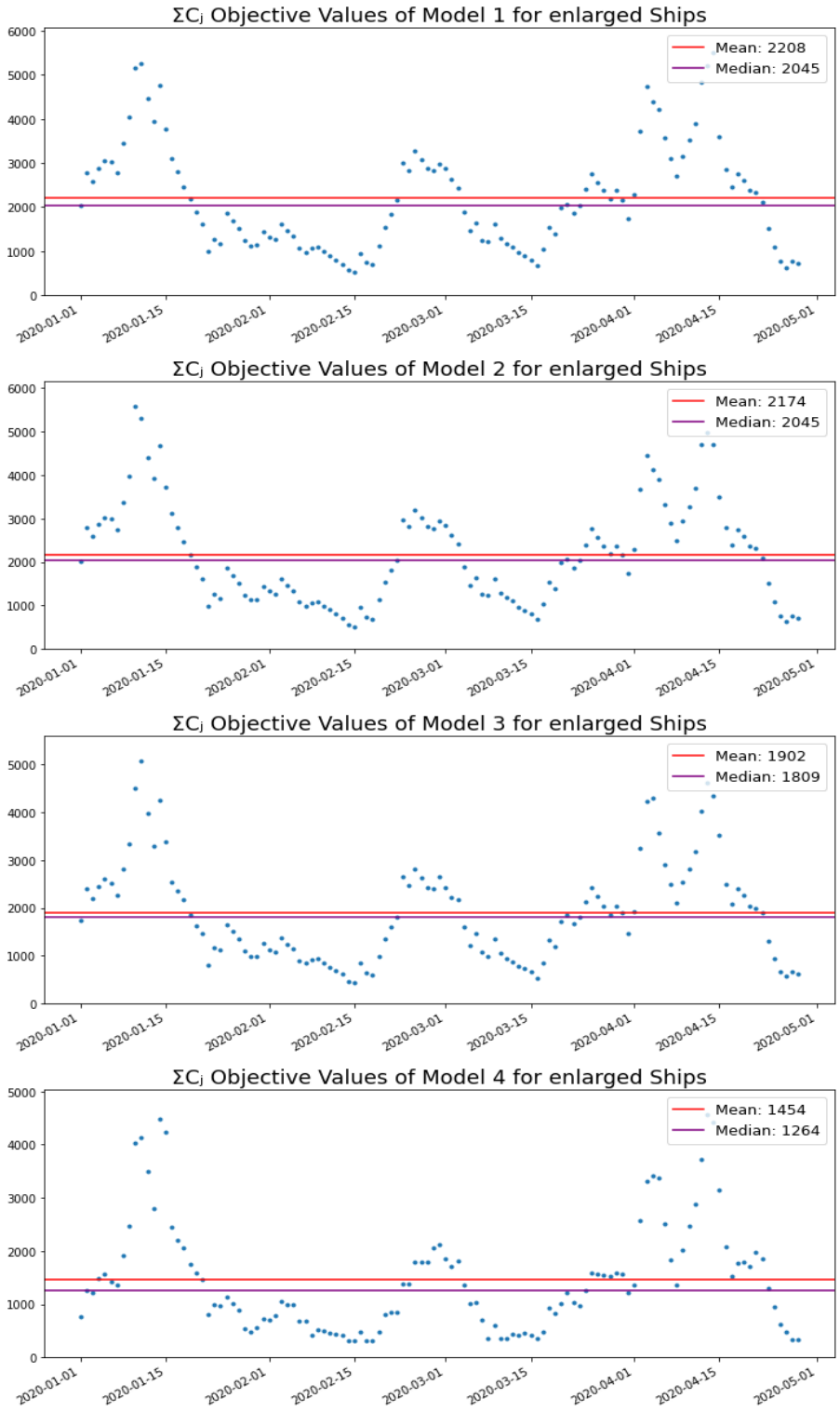


Figure 11: Objective Values of Function $\sum C_j$ of the Four Modeltypes for 50% of Ships enlarged

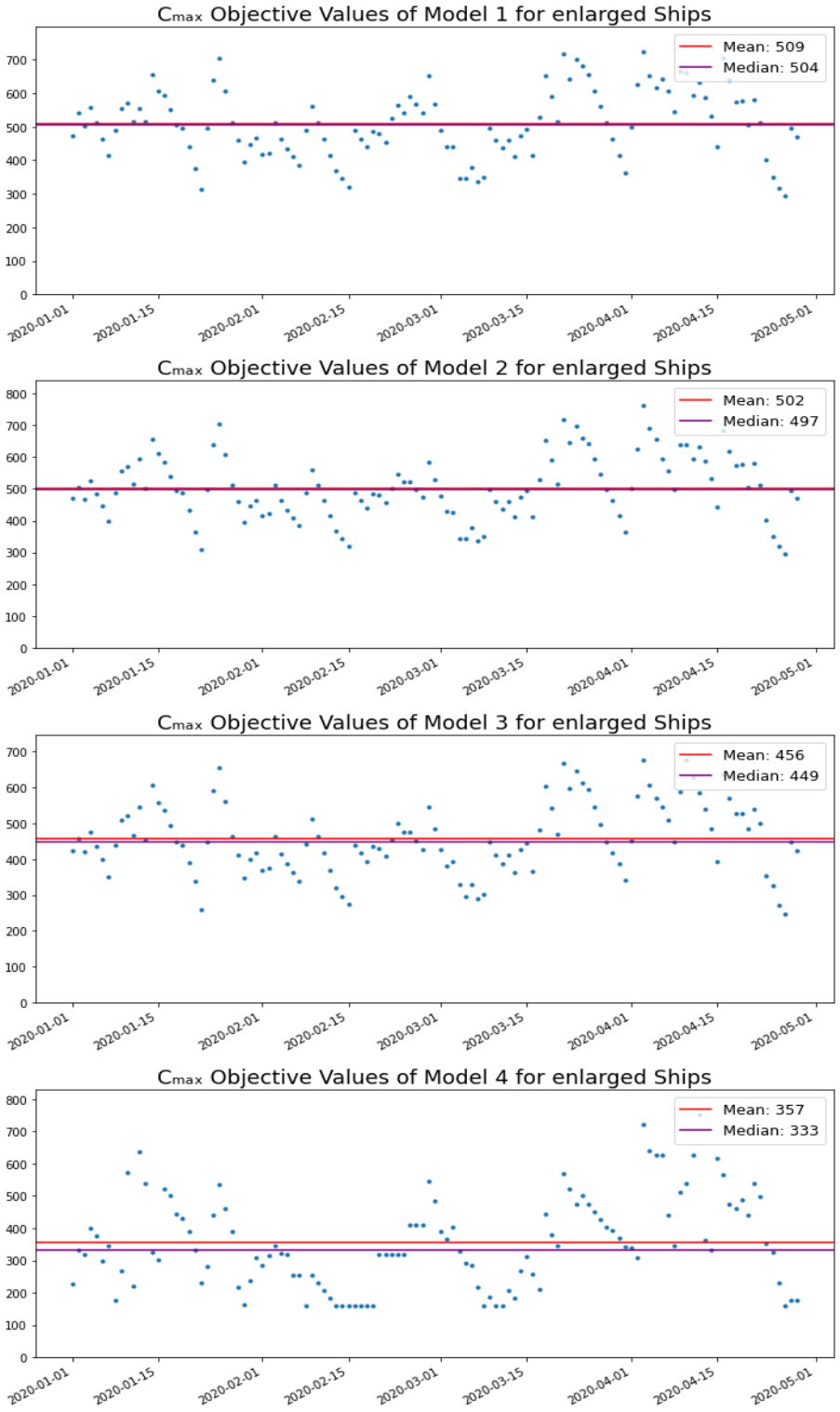


Figure 12: Objective Values of Function C_{max} of the Four Modeltypes with 50% of ships enlarged

D Outline of Linear Model

```
def solve_linear_model(df,remainders, mtype, objective_function, start):
    ''' Takes as input a dataframe with the ships that need to be scheduled and
                                                one of the following modeltypes (
                                                mtypes):

    1 : FCFS (ships can be scheduled no earlier than their ETA)
    2 : No relaxation (ships can be scheduled no earlier than their ETA)
    3 : 48-hours relaxation (ships can be scheduled up to 48hrs earlier than their
                                                    ETA)
    4 : Complete arrival time relaxation

    and one of the following objective functions as objective_function:
    1 : Minimise sum Cj
    2 : Minimise max Cj

    '''

    ships = [str(i) for i in range(len(df['MMSI']))]
    duration = list(round(df['Duration'], 0) + BUFFER)
    weight = list(round(df['Duration'], 0))
    arrival = normalize_column(df, colname='NOR Tendered', start=start)
    width = list(df['Width'])
    berths = ['AK 1W1', 'AK 1W2', 'AK 2E2', 'AK 1E1']

    ship_name_to_duration = dict(zip(ships, duration))

    prob = LpProblem("Ship_Berthing_Problem", LpMinimize)

    # Decision Variables


    s = LpVariable.dicts("service_time", ships, lowBound=0, upBound=None, cat='
                                                Integer')

    #(13)
    x = LpVariable.dicts("if_ship_in_group_berth", (str((j, i)) for j in ships for
                                                i in berths), lowBound=0, upBound=1,
                         cat='Binary')

    #(14)
    I = LpVariable.dicts('if_two_ships-berthed', ((i, (j, k)) for k in ships for j
                                                in ships for i in berths),
                         lowBound=0, upBound=1, cat='Binary')

    #  (6)
    if mtype == 3:
        relax = LpVariable.dicts('arrival_hours_earlier', ships, lowBound=0,
                                                    upBound=48, cat='Integer')

    # (1) Objective Function
    if objective_function == 1:
        prob += lpSum(s[ships[j]] + duration[j] for j in range(len(ships)))

    #(2)
    elif objective_function == 2:
        CMax = LpVariable('max_completion_time', upBound=None, cat='Integer')
```

```
        prob += CMax

        #(7)
        for j in range(len(ships)):  # Find maximum completion time
            prob += CMax >= s[ships[j]] + duration[j], "Max Completion Time: {}".
                                                    format(ships[j])

    # CONSTRAINTS

    # (5) FCFS
    if mtype == 1:
        for j in range(len(ships)):
            for j_prime in range(len(ships)):
                if arrival[j] <= arrival[j_prime]:
                    prob += s[ships[j]] <= s[ships[j_prime]]

    # (8) Occupation in the harbour
    for i in range(len(berths)):
        for j in range(len(ships)):
            prob += s[ships[j]] >= (remainders[i] + BUFFER) * x["('{}', '{}')".
                                                    format(ships[j], berths[i])]

    # (3) One berth per ship
    for j in range(len(ships)):
        prob += lpSum([x["('{}', '{}')".format(ships[j], i)] for i in berths]) ==
                                                1

    # (4) Start service after arrival
    if mtype == 1 or mtype == 2:
        for i in range(len(ships)):
            prob += s[ships[i]] >= arrival[i], "Service > Arrival for: {}".format(
                                                    ships[i])
    elif mtype == 3:
        for j in range(len(ships)):
            prob += s[ships[j]] >= arrival[j] - relax[ships[j]]

    # (9)
    for i in berths:
        for j in range(len(ships)):
            for j_prime in range(len(ships)):
                if j != j_prime:
                    prob += s[ships[j_prime]] >= s[ships[j]] + duration[j] -
                        M * (1 - I[i, (ships[j], ships[j_prime])]),
                        "{}-{}-{}".format(i, ships[j], ships[j_prime])

    # (10)
    for i in berths:
        for j in range(len(ships)):
            for j_prime in range(len(ships)):
                if j < j_prime:
                    prob += I[i, (ships[j], ships[j_prime])] +
                    I[i, (ships[j_prime], ships[j])] <= (1 / 2)
                    * (x["('{}', '{}')".format(ships[j], i)]
                    + x["('{}', '{}')".format(ships[j_prime], i)])

    # (11)
    for i in berths:
        for j in range(len(ships)):
            for j_prime in range(len(ships)):
```

```
                if j < j_prime:
                    prob += I[i, (ships[j], ships[j_prime])] + I[i, (ships[j_prime
                                                                ], ships[j])] >= x[
                        "('{}', '{}')".format(ships[j], i)] + x["('{}', '{}')".
                                                                format(ships[
                                                                j_prime], i)] - 1

    # (12)
    for j in range(len(ships)):
        for j_prime in range(len(ships)):
            for j_prime_prime in range(len(ships)):
                prob += width[j] * x["('{}', '{}')".format(ships[j], berths[0])]
                + width[j_prime] * x["('{}', '{}')".
                format(ships[j_prime], berths[1])] + width[j_prime_prime]
                * x["('{}', '{}')".format(ships[j_prime_prime], berths[2])] <= 106

    print('I will now try to solve the problem, please wait a moment...')
    prob.solve(pulp.PULP_CBC_CMD(maxSeconds=120))
    print("Status:", LpStatus[prob.status], 'with value', value(prob.objective))
```