TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Andrey Ermakov 184626IASM

AUTOMATION OF VEHICLES PLACEMENT IN RO-PAX SHIP LOADING

Master's thesis

Supervisor: Uljana Reinsalu
Researcher, PhD

Tallinn 2020

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Andrey Ermakov 184626IASM

SÕIDUKITE PAIGUTUSE AUTOMATISEERIMINE RO-PAX LAEVA LAADIMISEL

Magistritöö

Juhendaja: Uljana Reinsalu
Teadur, PhD

Tallinn 2020

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis and this thesis has not been presented for examination or submitted for defence anywhere else. All used materials, references to the literature and the work of others have been cited.

Author: Andrey Ermakov

04.05.2020

Abstract

The aim of this thesis is to develop a software application which can produce vehicle placement plan for Tallink Grupp AS RO-PAX (Roll-On-Roll-Off-Passenger-ship/ferry) ferry, which is typically used on the routes between Tallinn and Helsinki.

This is an effort to help Tallink Grupp AS automate the process of loading vehicles as currently used solution does not provide required automation for personnel involved. The presented solution provides acceptable placement plan based on the booking list of vehicles, i.e. quantity of different types of vehicles. Such a list is received from the ticket office for each departure. The automated placement plan can still be changed manually by the load masters, if required, in the Tallink Grupp AS internal software, thus allowing them following currently standing orders.

The automated placement in this work is implemented using three different algorithms; results are tested on real operational data and analysed; recommendation for usage is given.

This thesis is written in English and is 59 pages long, including 5 chapters, 40 figures and 16 tables.

List of abbreviations and terms

RO-RO	Roll-on/ Roll-off ship
RO-PAX	Roll-On-Roll-Off-Passenger-ship/ferry
MS	Motor Ship
IMO	International Maritime Organization
LD	Load
LCI	Loading Condition Information
ROROLOAD	RO-RO Loading
MASSLOAD	Mass Loading
RRBOOKLIST	RO-RO Booking List
GUI	Graphical User Interface
FF	First Fit
BF	Best Fit
FFD	First Fit Decreasing
BFD	Best Fit Decreasing
GRASP	Greedy Randomized Adaptive Search Procedure
STL	Standard Template Library

Table of contents

List of figures

List of tables

1 Introduction

1.1 Overview

It is only 82 kilometres between two capitals, Tallinn in Estonia, and Helsinki in Finland, but they are separated by the Gulf of Finland. “Goods have been traded between Virumaa and Southern Finland already for over 700 years (so-called seprakauppa). The increasing cross-border cooperation between Helsinki and Tallinn today is supported by the leading role both capital cities have in the economic and social development of its (Estonia) country”[1]. Number of passenger travels has increased in 2019, reaching 9 million passengers[2]. So did the cargo volumes, overall transfer was nearly 2.1 million vehicles, out of which „71% were passenger cars and 26% were trucks and trailers“[2]. Handling such high volumes of transferred vehicles require automated solutions. This thesis covers the development process of an automated solution for creating a vehicle placement plan on Tallink Grupp AS ferry.

Nowadays main option to swiftly transfer passengers, goods and vehicles by sea between countries are ferries. Tallink Grupp AS is the biggest ferry operator in Baltics[3]. The main type of ferries used by Tallink Grupp AS are RO-PAX (Roll-On-Roll-Off-Passenger-ship/ferry) type, which is able to take onboard several hundred vehicles in one trip plus passengers and other cargo. Usually, Tallinn-Helsinki and Helsinki-Tallinn trips are made three times in a day. Each trip lasts about two hours one after another with a turnaround of one hour. This tight schedule creates a pressure on the ferry crew, especially if there is additional cargo to be loaded or vehicles taken aboard with so-called “last-minute“ tickets.

If anyone has been travelling on the ferry on his or her own, carrying only personal luggage, this is easy. When you travel with your car or van, or even truck or trailer, this is a bit more complicated. Standing orders aboard ferries require check-in prior to boarding, vehicle height measurement is usually done in a harbour[3].

By construction, RO-PAX ferries usually have multiple decks to carry the cargo, vehicles and passengers. RO-PAX ships are a subclass of RO-RO (Roll-on/ Roll-off) ships. Usually classic RO-RO ships are vehicle carriers used on longer routes for transporting vehicles between countries. RO-PAX ships are usually used for moving cargo trucks along with usual passengers with cars on shorter distances. RO-PAX upper decks are for the passenger cabins. Lower decks usually carry heavy vehicles, such as trailers, lorries and buses. Middle decks usually carry passenger cars, vans, campers, shopping cars and other vehicles. Usual number of decks on such ferries is about five to six, so theoretically average ship can take up to 500 vehicles or even more, depending on its size and weight.

Our ship model is Tallink Grupp AS one of the newest vessels – MS (Motor Ship) „Megastar“. She went into service and had transferred largest passenger numbers in 2017[4]. This ship has 12 deck ship, with four cargo decks (3, 5, 6, 7) hosting the lanes where vehicles are put.

Technical data is presented in Table 1. Deck plan can be seen on Figure 1 below.

Table 1. Technical data of MS Megastar[5]

Built year	2017
Built by	Meyer Turku Oy, Finland
Passengers	2800
Decks	12
Length	212,2 meters
Breadth	30,6 meters
Speed	27 knots
Lane meters	3653 lane meters
Main engine output	40 600 kilo watts

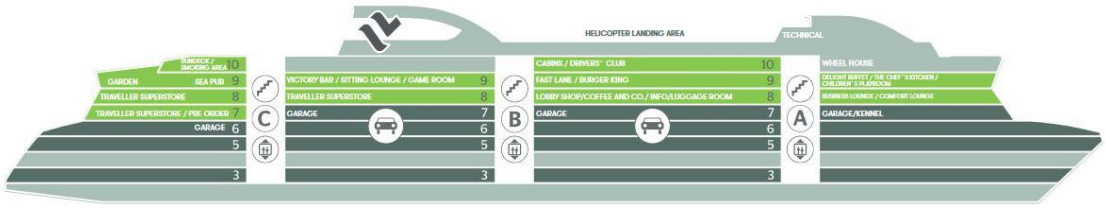

Figure 1. MS Megastar deck plan[6]

The process of boarding for ferry service consumer is relatively easy. Buy tickets, specifying to an operator the dimensions of your vehicle; arrive to a port, join the specified lane in a port for boarding, wait for the green light and follow the load master orders. The loading process for the crew is completely different.

At first, the crew, consisting of the bosun and number of deckhand seamen, receive a booking list of vehicles to be boarded. This list contains the numbers of vehicles of each type to be boarded. Bosun works with an application called NAPA. If there are 10-15 vehicles in total, he can enter vehicle types and numbers manually and put them on the placement plan, which in turn is used during the boarding. Second option is to list manually through the previously used cargo plans, find one previously used cargo plan with similar numbers of cars loaded, make manual changes to fit the current load, then place manually cars so the ship will be stable. Cargo plans are represented with LD (Load) files.

When author was checking the real LD files provided by the supervisor, the medium load was nearly 200 cars, 20-50 trucks, 40 vans and some buses or trailers. The first option is nearly impossible to accomplish, given usual loads. And second option will require time, which is always valuable for the crew.

1.2 Motivation

This thesis was inspired with my love for sea and targets helping Tallink Grupp AS bosuns or other personnel, who is involved into the vehicle loading process on a ferry.

Given just booking list, i.e. simple numbers and types of cars to be boarded, author will develop an application which automates the process of preparing the vehicle placement plan, saves time for the sailors for other tasks and can be implemented for use in real life.

1.3 Problem formulation

Cargo and vehicle placement problem itself is not only time-consuming task for deckhand crew. Because of the design of RO-RO ships, as they do not have usual ship compartments on lower decks, it may lead to a safety problem.

According to IMO (International Maritime Organization), „The problem areas of RORO ships are: the lack of internal bulkheads, cargo access doors, stability, low freeboards, cargo stowage and securing, life-saving appliances, the crew“[7]. As RO-PAX ships are a subclass of RO-RO ships, it is even more important as RO-PAX ship carry a lot of passengers. MS „Megastar“ is able to carry up to 2800 passengers in addition to cargo loads. In case of vehicle placement, Tallink bosuns and deckhand crew should not only consider the placement of vehicles themselves, but a lot of other cargo ship carries, such as liquid cargo in tanks, goods for local stores, etc. All of cargo which ship carries in a trip goes into a cargo plan. This is an important document, which is used to calculate the ship stability and exactly the same document is produced with the NAPA software, currently used by the crew. LD file is a cargo plan written in a special text format. NAPA application works as „what you see is what you get“ manner, when user just places a vehicle onto the visual deck plan, the software immediately recalculates cargo weight, ship trim and ship heel. Seaman just checks if they meet criteria for stability and can see on the screen where cars are placed on the decks.

This is a task of the bosun to place the vehicles on a plan in such a way that ship is stable. That is why currently all the responsibility lies on his shoulders. Using his knowledge and long-time experience the bosun searches for best in his mind previously used LD file and makes manual changes according to new load. Software can make final calculation, but vehicle placement itself is still a bosun’s manual task.

The manual process is pictured below on Figure 2 and planned automated process is pictured on Figure 3.

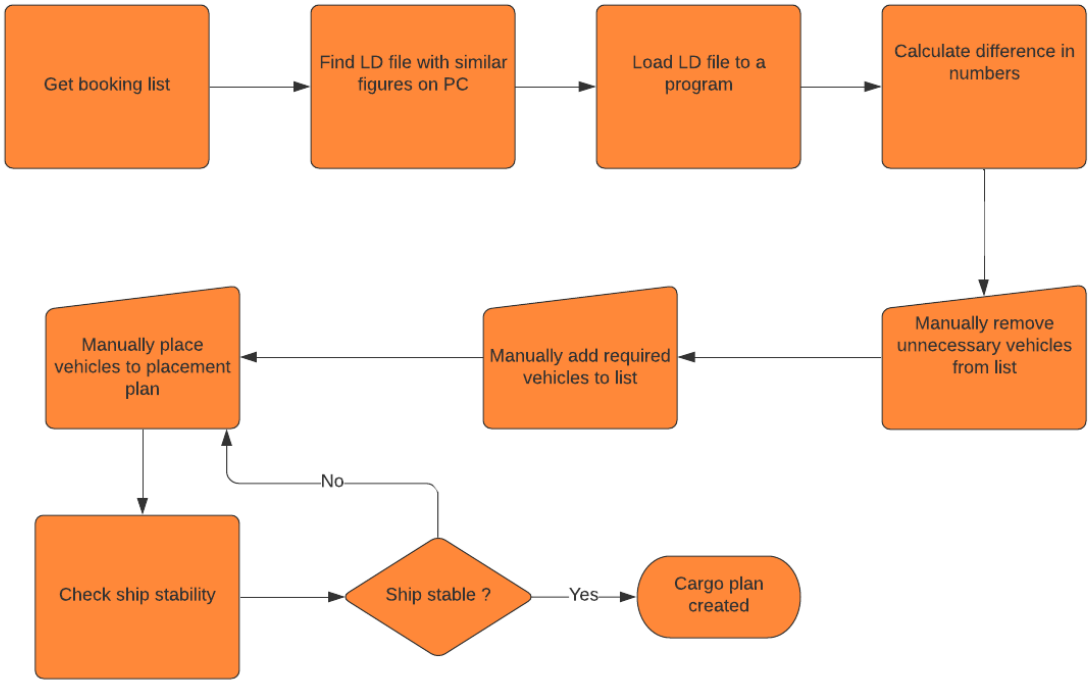

Figure 2. Manual process flow

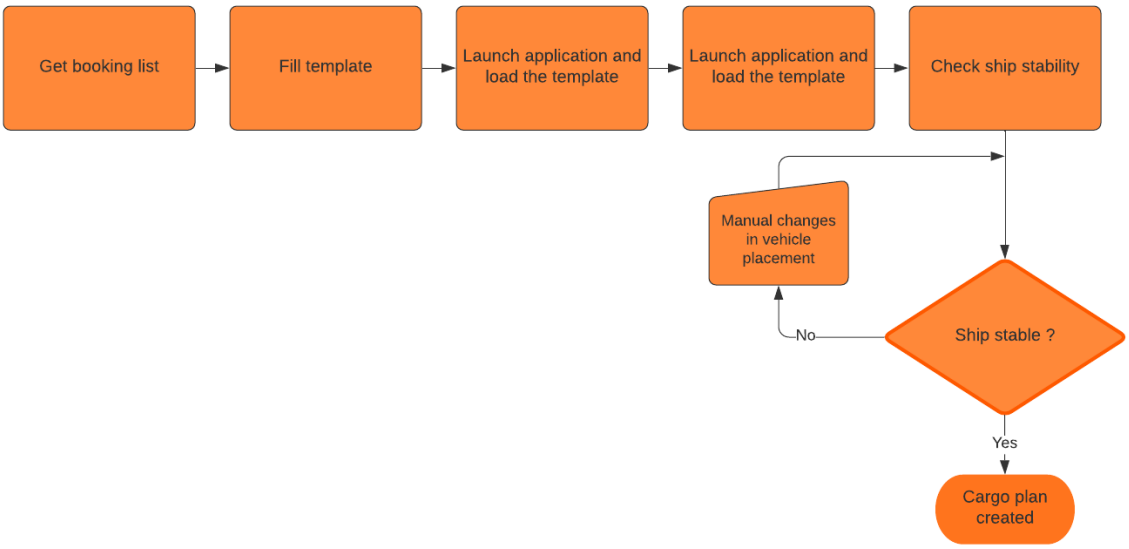

Figure 3. Automated process flow

As it can be seen on the process flow diagrams, current workflow (Figure 2) includes a lot of manual work, usually in the head of the bosun, to place the vehicles to keep ship stable. The whole operation is estimated to take not less than 15 minutes.

Current NAPA software can only ease the calculations, showing if ship is stable or not stable after new load has been placed onto the plan. Also, even 15 minutes is a time-consuming operation. Another problem which we need to address is ship stability.

1.3.1 Ship stability

As it was mentioned above, one of the problem areas of the RORO ships and thus RO-PAX ships as well is its stability. For reader, who is not familiar with ship hydrostatics, please turn to Adrian Biran[8] or Barras and Derrett[9].

In simple terms, it is required that the ship will not capsize or sink in case of unevenly distributed loads. Using Archimedes's principle, any ship even with no cargo has a geometrical centre and because its hull has a weight, thus the gravity force is applied. The vector of this force points downwards. According to same Archimedes' principle, there is also the buoyancy force, which is pointing upwards through the vertical centre of gravity. Both forces are equal by module.

Ship coordinate system will be taken as follows per Biran „X-axis runs along the ship and positive forward, Y-axis is transversal and positive to port, Z-axis is vertical and positive upwards"[8].

In the best conditions, when ship is afloat its waterline is horizontal at zero degrees, so ship bow and stern are at the very same horizontal level. If they are not at the same level, this is called a trim. Positive trim is when bow is positioned lower than the stern, and negative trim is when stern level is positioned lower than bow level. If you put some load on the ship's starboard or port side, vertical centre of gravity will also shift in the direction of added weight. According to the Archimedes's principle the vertical centre of buoyancy will also shift in the same direction and both forces will still be pointed in the opposite directions, but because new point is not located in the middle of the ship and one side has more load than other, thus ship now has a heel. The heel can be on the port side or the starboard and usually is measured in degrees.

This is a must for a captain to maintain minimum heel and trim of the ship within allowed limit for ship safety. Special attention should be given to the heel degree as ship at big angles can easily capsize.

In everyday work all calculations for ship heel and trim are calculated using the NAPA software. This task itself is hard to accomplish without special software, as parameters heavily rely on the particular ship construction and its current load.

Once all cargo is present in the cargo plan and vehicles placement has been completed, and ship heel and trim meet allowed levels, then vehicle planning task for bosun is considered completed.

1.3.2 The relevance of the problem

Considering the numbers of the ferries employed by Tallink Grupp the automation of the manual tasks might reduce costs of the ship maintenance.

Also, in addition to the high-speed RO-PAX ferries MS “Megastar”, MS “Star” and cruise ferry MS “Silja Europa”, company owns two RO-RO ships “Sea Wind” and “Regal Star”, so the automation solutions can be later extended to other types of ships. Also, Tallink ordered a new high-speed RO-PAX ferry MS “MyStar” expected to enter into service in 2022.

1.4 Initial condition for the task.

Because crew uses NAPA software for the very particular ship, all required ship technical data has already been predefined in the NAPA application. Because this program operates with cargo items, all possible vehicles types already have been predefined as well, given length, width, height and weight for each vehicle type. Information about additional cargo, for example fuel, may have been entered by other personnel, but this information was not provided to the author and it not considered in this thesis.

In addition to the types mentioned below there are priority cars, that are the same as usual cars, but they must have such places so they can exit the ferry before other cars. As current software makes no distinction between them, neither it does not know their registration number, author presumes that places that will be closer to the ramps are taken by the priority cars and the distinction is made on the spot as it is done now.

Information about possible vehicle types and their technical data is presented in Table 2.

Table 2. Vehicle types technical data

Vehicle type	Length, meters	Width, meters	Height, meters	Mass, tons	Full vehicle type name
MC	2	1	1	0.2	MOTORCYCLE
CAR	4.8	2.1	1.9	1.8	CAR
SHP_CAR	4.8	2.1	1.9	1.8	SHOPPING CAR
VAN	7	2.1	2.25	2.5	VAN
VAN_H	7	2.1	2.4	2.5	VAN HIGH
L_L	4.8	2.1	1.9	1.8	LONG & LOW
L_MH	10	2.7	4.4	5	LONG & MEDIUM/HIGH
BUS	14	2.7	4.4	12	BUS
LOR	11	2.7	4.1	12	LORRY
TT_25	17	2.7	4.1	25	TRUCK/TRAILER, 25 TONNES
TT_39	17	2.7	4.1	39	TRUCK/TRAILER, 39 TONNES
MCH	10	2.7	4.1	15	MACHINE
TRL	4.1	2.7	14	15	TRAILER
TLNK	4.1	2.7	17	25	TALLINK TRAILER

Ship decks, where vehicles are placed are divided into lanes. Each lane has a width enough to fit all types of vehicles, but there are restrictions as some types of vehicles cannot be placed to particular lanes. Different decks have different heights. Additional restrictions may include inappropriate vehicle height, desired proximity to staircases or exits, other restrictions may also apply. Also additional restrictions may be learned in the research process as author does not have opportunity to speak directly to deck personnel. Each lane also has a technical information as its absolute minimum coordinate as X_{min}, absolute maximum coordinate as X_{max}, its shift across the ship width from the ship gravity point as Y, and the vertical level of the lane on a particular deck as Z.

Also, an empty ship has its own mass known as lightship and the gravity point for a empty ship is also given with coordinates in all three dimensions, as its position in length, width, and vertical coordinate.

Database of LD files with real loading data was provided, so author is able to check the future solution on real operational data during analysis phase.

1.5 Expected outcome of the solution.

Current process is supported with only NAPA software application which require manual input. The general idea is to create a GUI (Graphical User Interface) application, which can provide required data for NAPA.

There is a previously written command line converter to create LD files. Given the type of vehicle, its assigned lane and its minimum vehicle X coordinate on lane, the converter can generate entries for LD file. Author proposes to produce an application, which generates a file with a special booking list. Each entry will consist of vehicle type code, lane name and minimal coordinate of the vehicle on the lane to pass this data to the converter.

Given a resulting file with such entries, author's application will invoke command line converter to generate a complete LD file. Opening completed LD file with NAPA application, operator can check the positioning of the vehicles on all decks, also check if heel and trim meet requirements. In case the operator is not happy with results, he can make minor movements of vehicles manually in NAPA application, without additional data input and having all vehicle locations already visualized.

2 Background

2.1 Related works

The ferry loading process according to Kirillova and Meleshko looks a bit old-fashioned as „at present, the analysis of the possible placement of cargo from each subsequent application and thus the formation of an operational plan for the vessel loading are carried out by the line agents using the standard features of the program Microsoft Excel spreadsheet with the entry of a large number of exogenous parameters[10]“. Øvstebø et al. in [11] defines “RO-RO stowage problem is to decide upon a deck configuration with respect to height, to decide which optional cargoes to carry, and to decide how to stow all cargoes on board the ship“.

The problem of loading cargo is mostly studied in question of loading of container ships and much less for RO-RO and RO-PAX ships. Øvstebø et al. in [11] considers stowage problem as reduction from the knapsack problem [12]. Also in [11] he concludes that it is possible to solve the problem with MIP (Multi-Integer Programming), but results may be not practical for use. Therefore, a heuristic solution can be used, which tries to load all cargo, then tries to improve cargo placement with local search[11].

Hansen et al. in [13] also uses MIP programming for stowage planning, but uses initial stowage model and shifting model as his two dimensional algorithm means also shifting cargo after partial unload. RO-RO ships usually carry a lot of vehicles during voyage and they visit several ports. In each port there might be unload and load operations as well. Also Hansen suggests using the grid system and represent each cargo as several squares in two dimensions. Obstacles, like pillars and ramps, are part of the grid also.

Puisa in [14] suggests to create an individual grids for each cargo type for each deck where it can be placed and again relies on the assumption that some cargo will be loaded and unloaded several times, before arriving to destination point. Puisa also uses MIP programming to solve it with branch and bound algorithm.

Wathe also describes that commercial software already exist, but this is mostly only a visual aid for manual placement of goods[15].

Almeida and Steiner in [16] suggest solving 1D bin packing using heuristic minimum bin slack to pack the bins efficiently.

More complicated algorithm is a novel GRASP (Greedy Randomized Adaptive Search Procedure) algorithm, where „each GRASP iteration consists of two phases: construction and local search. Construction procedure is greedy procedure to build a feasible solution and local search is to enhance the solution“[17].

All reviewed works mostly focus on the task of creating stowage plan from the mathematical point of view and it is good for stowing items in a given area. Our RO-PAX ferry is used to carry vehicles and passengers, so this is where a lot of constrains come into play. So mathematically optimal solution might not serve everyone. Therefore, author decided to review other options for the task implementation.

2.2 Data analysis

Starting the research, it is very important to perform an excellent data analysis to produce the best suitable result for the company, also making it easier to use and possibly estimate future use of the product.

Main source of the analysis were the LD files, provided by the supervisor. The following goals have been set:

- Studying data structure of the file itself
- Splitting data to essential and non-essential
- Analysis of the essential data

These steps are explained in the following chapters.

2.2.1 LD file data structure

LD file is split to the following sections:

TANKLOAD – information about liquid cargo currently present on the ship

MASSLOAD1...3 – information about passengers, crew, provision, shops, storage

MASSLOAD – summary for MASSLOAD1 … MASSLOAD3 sections

TAB_ROROLOAD – section describing each vehicle positioning on the ship

RRBOOKLIST – section containing number and types of the vehicles to be loaded

IMOLOAD – technical data of loading of lanes per IMO convention

MASS_ROROLOAD – summary for lanes loading

LCI (Loading Condition Information) – summary of the cargo list

As passengers, liquid cargo and other non-vehicle related information is a variable data and available to the crew use only, so this data is not taken into account in this thesis. Table ROROLOAD (RO-RO Loading) and sections RRBOOKLIST (RO-RO Booking List) contains essential information about the vehicles and its location on the lanes, section MASS_ROROLOAD (can be used as reference and LCI section can be used to compare results).

2.2.2 Essential data research

Table ROROLOAD has the following column, which can be useful for the task implementation:

- Car type
- Lane assigned
- Deck
- Minimum and maximum coordinates of the vehicle
- Mass of the vehicle
- Centred coordinates of the vehicle
- Colour of the vehicle for representation

Section RRBOOKLIST describes the following information:

- Type of the vehicle
- Pre-defined dimensions of the vehicle
- Maximum possible numbers of vehicles
- Actual number of vehicles

Section MASS_ROROLOAD contains information about each lane load, such as

- Total mass load
- Coordinates of centre of gravity of the lane with vehicles loaded.
- Length of the load lane

2.2.3 Analysis of the essential data

Once the source of the essential data has been defined it is very important to make decision based on the data itself. As author have researched over 500 LD files manually the following patterns have been obtained:

1. Average load not depending on the season is nearly 200 cars, 20-40 vans, 20 or more 25-ton trucks.
2. All evening cruises are usually do not exceed 150 vehicles altogether.
3. During the winter season there are motorbikes on the ferry and not more than 1 or 2 buses. A lot of long+low vehicles and long+medium/high vehicles.
4. During the summer season, number of motorbikes and buses significantly increases, the number of long+low vehicles and long+medium/high significantly decreases.
5. Heavy 39 tons trucks are present only in 22% of all cases studied.
6. Machines are transferred quite rarely, just 2 cases out of 570 studied cases, so 0.35% of all cases.

2.3 Tool selection

When author discussed the proposed topic for the thesis with the supervisor, original idea was to use the neural networks in the automation of the vehicle placement task. Given the relatively large amount of LD files, each of them containing the placement plan for each loading, neural network theoretically can be built and trained on the existing data so it can analyse requirements for new loadings and offer the placement plan. For efficient training of a neural network it is necessary to have a reliable set of data which can be recognized into the pattern. Given the LD files for three months, January (winter season), April (spring season) and June (summer season), it turned out that it is hard to predict people's behaviour as vehicles' numbers differ significantly even within one week of data. In addition, because existing software works as "what you see is that you get" program, when its operator can point to any coordinate he wants

and the vehicle can be placed at any available location on lane. Of course NAPA software will not allow vehicles overlap each other, but it does not check how much free space left on lanes. Usage of neural networks will require an additional check to avoid vehicle overlapping.

More promising option is to implement a usual desktop application, thus creating placement algorithm from the scratch. This solution allows full flexibility in implementing possible algorithms, designing graphical user interface for ease of use, and full customization of output format of data.

2.4 Selection of the evaluation criteria

The NAPA software was customized by third party developer specially for Tallink Grupp AS MS "Megastar" and generally is not available for use outside of the company. Supervisor was given an access to the computer with software installed under strict rules and conditions of usage. Thus it was agreed to perform selective tests for output data of author's future application with existing NAPA software to make sure that heel and trim of the ship stay within limits, as they are calculated automatically in NAPA software upon file load.

As it was mentioned previously, one of the available parameters is the gravity point of the empty ship. Having known coordinates of the gravity centre of empty ship and centre of loaded cargo it is possible to calculate the relative value V as follows:

$$V = \sqrt{x^2 + y^2},$$

where *x* and *y* are the length and width coordinates of these points.

The vertical coordinate Z only partially affects the stability. Shortly, we need to keep cargo vertical coordinate as low as possible comparing to vertical coordinate of ship gravity centre.

Naming as V_l the relative value of the empty ship and V_c as relative value of the loaded vehicle cargo we can calculate how close both points are located to each other. The less the difference, the closer the points, thus better stability.

2.5 Algorithm selection

Usual cargo deck on the RO-PAX ship can be represented as set of the lanes. Each lane has definitive and known length. All lanes has the equal width, but still vehicle placement restrictions apply. Last important parameter of the lane is its deck level or Z coordinate.

Decks themselves have different height, so on lower decks it is possible to place all types of the vehicles, and middle decks usually have a ceiling at height of 2.5 metres, thus restrictions apply and only particular vehicles can be parked there. Also, for example, shopping cars should be placed near shops for passenger comfort.

Looking at lanes, we can imagine virtual walls between them. Process of parking vehicles to the lane reminds stuffing the truck with the goods task. The problem of stuffing goods into the truck reminds in turn the bin packaging problem.

As defined by Garey et al., „The 1D bin packing problem: Given N items of various „sizes" and an infinite number of "bins" (each with a fixed finite capacity C), group the items into a minimal number of bins so that no bin is over-filled"[12][18]. Classical 1D bin packing problem can be solved by multiple algorithms.

In this particular case, FF (First Fit) algorithm selects first item from the desired list, in this case, a vehicle from vehicle queue, and puts it into first lane this vehicle can fit. BF (Best Fit) algorithm selects non-empty lane such vehicle can fit, so lane will be most occupied after adding the vehicle. FFD (First Fit decreasing) algorithm arranges initial vehicles in non-increasing order, sorting them by vehicle mass, and applies FF algorithm. BFD (Best Fit Decreasing) arrange initial list of vehicles in non-increasing order by vehicle mass and applies BF algorithm to the list[19].

Looking at the possible algorithm usage author have decided to use the mixed algorithm based on bin packing problem, with assumptions to select best lane for loading of next vehicle, and taking into account all placement restriction along with maintaining ship stability. This decision is based as there are too many restrictions in place and process should be controlled with each loaded vehicle.

2.6 Testing methods

As it was mentioned before, initial data for the task is original LD files from the Tallink Grupp AS. Each file contains information about the number of vehicles for each departure. After the LD file parsing, it is possible to generate input data for the new application.

Each set of test data will be loaded into the application and output X and Y coordinates of gravity point of the loaded cargo will be calculated and relative value V_c is also calculated and used as an evaluation criterion against the relative value of the empty ship.

Five hundred tests will be performed, output data obtained and analysed. Selective tests will be performed in NAPA software due its limited availability for the author to make sure heel and trim criteria are met and possible comparison with the evaluation criterion will be made. All critical cases will be reviewed as well.

3 Application development

The main goal of the application is to help the bosun to create an acceptable vehicle placement plan with resulting output as a text file, which in turn can be parsed with the converter, into a format suitable for current Tallink NAPA application.

3.1 Initial user requirements for the application

The main initial requirement for the application is simplicity of usage. NAPA software works on the usual personal computer and the target operating system is Windows. The command line LD file converter works also in Windows.

Bosun receives information provided by the ticket office before the loading. Ticket office can only provide information about vehicle types to be boarded and number of vehicles for each type. This information is used as initial data for application development and such information can be stored in text format on the computer as a template for application input.

3.2 Programming language selection

Most popular programming languages in the world are Java, C, Python and C++ [20]. It is important to select the language which suits best.

Java is an object-oriented language, developed by the Sun Microsystems. This language comes with a lot of classes, including the GUI libraries and is cross-platform.

C language is a general-purpose language, mainly intended for structured programming.

C++ is an object-oriented language, developed by Bjarne Stroustrup, includes a STL (Standard Template Library) and compilers available for multiple platforms.

Python is an interpreted programming language. It does not require a compiler and a lot of packages are available for scientific programming. GUI frameworks are available.

The comparison between these four languages are listed in Table 3 below.

Table 3. Language comparison table

Language name	Pros	Cons
Java	Has native GUI support Lot of embedded classes Easy IO functions Cross-platform	Requires Java Virtual Machine
C	Low level programming	Does not support object-oriented programming Platform-dependent No native GUI
Python	Easy to use	Hard to maintain code
C++	STL library available Supports object-oriented programming	Less user friendly than Java GUI support with external frameworks

The best choice for implementation for end-users and author skills is Java language, as Java Virtual Machine is usually installed on most workstations, or easy to obtain if absent. The other con that in this case the application can be used on a different platform if that applies in future.

3.3 Process flow comparison

Before creating new application it is important to study the process in detail to create proper data structures and design effective algorithms.

A diagram showing the real process is pictured on Figure 4.

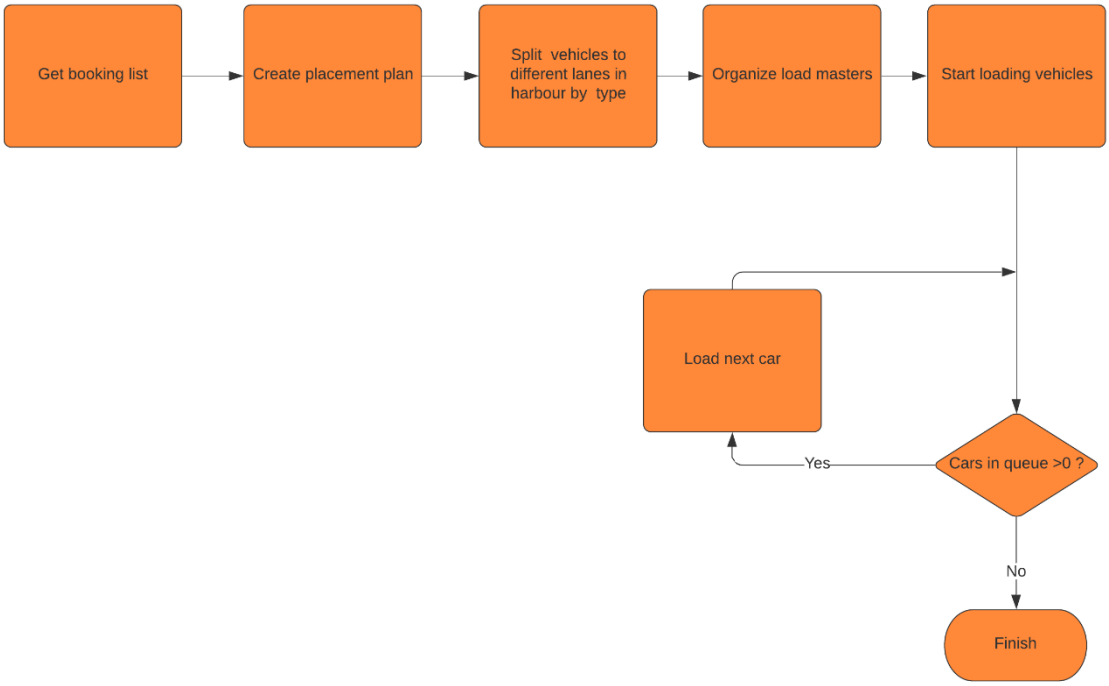

Figure 4. Real life process flow of loading vehicles

After receiving a booking list and once bosun has finished creating the cargo plan, deckhand team is ready to board vehicles. The registered vehicles are assembled in the harbour on the loading lanes. These lanes organize queues of different types of vehicles, and the crucial part that they are organized as such that it allows parallel loading and unloading of trucks and cars, minimizing boarding time. Deckhand crew splits across the ship to manage the loading process. Staying in different parts of the ship and using hand signals and other technical means crew give directions to the drivers where they should park their vehicles. Given explanation can help create an efficient algorithm, because it is essential to understand how work is organized on the spot.

The automated process, based on the explanation above, is pictured on Figure 5 below.

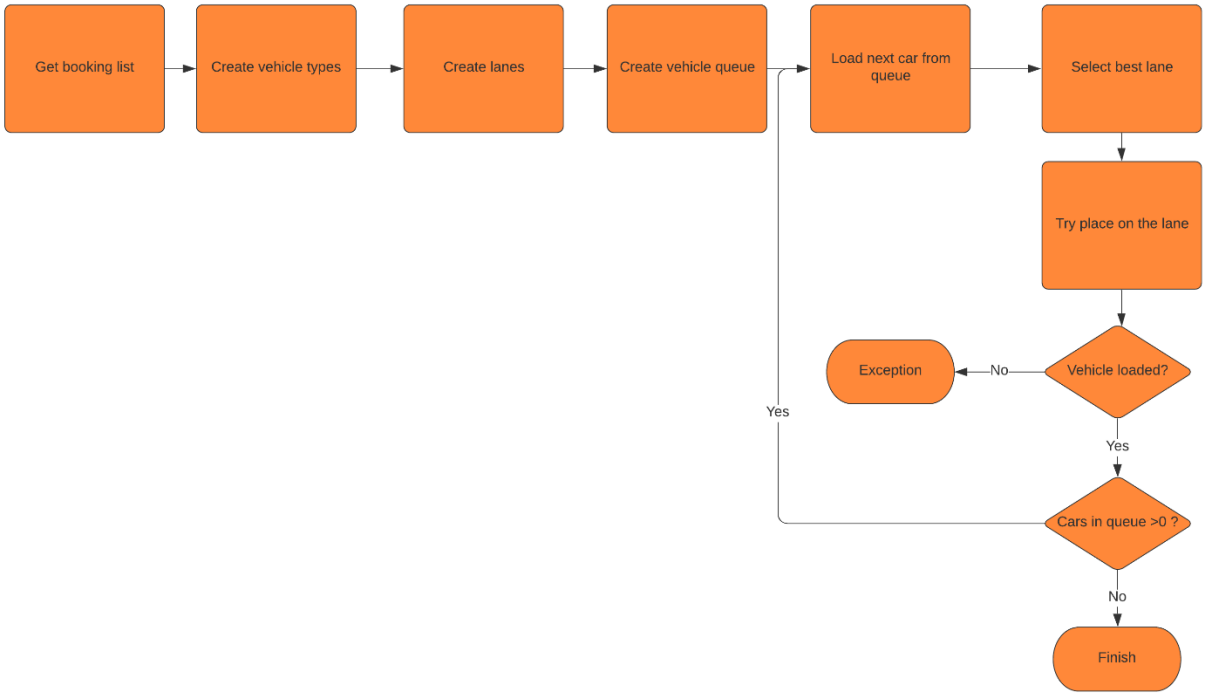

Figure 5. Proposed high-level algorithm for the solution

It is important to notice that the cargo plan already exist before the loading starts, and vehicles are loaded according to a rule that furthest places from the boarding ramp are occupied first. Key factor during boarding process is a manageable queue, that is why vehicles are organized into different lanes and usually one type of the vehicles is loaded after another.

Combining all information above, the overall task of future application development can be split to the following subtasks:

- Categorize and organize the predefined data into data structures for usage with newly created application
- Define data structures and objects
- Define interfaces so objects can exchange data
- Define input and output
- Implement an algorithm for placement calculation
- Create UI (User Interface)
- Test
- Perform analysis of the tests

3.4 Organization of the data structures

3.4.1 Data classes

Organization of data greatly affects the development of the application. If data is properly organized, the programming is simplified and there are a lot of possibilities for future modernization or adding additional features if initial conditions change. During the construction of classes the pattern of composition is used, when one class can be composed of entities of other classes[21]. Diagram of class relationship can be found on page 35. Description of essential classes is provided below.

Public enum class VehicleType is responsible for classification of vehicles. Its variables with descriptions are listed in Table 4.

Table 4. Class VehicleType

Type	Class member	Description
float	vehicleWidth	vehicle width in meters
float	vehicleHeight	vehicle height in meters
float	vehicleLength	vehicle length in meters
float	vehicleMass	mass of the vehicle in tons
String	vehicleDescription	full name of the vehicle
float	vehicleTotalType	number of vehicles to be placed
String	vehicleColour	colour of the vehicle in software

Public class StandardVehicle will encapsulate entity of VehicleType and any vehicle will be identified by the String registrationNumber. Registration numbers were not listed as an initial requirement, but such an option can be useful in the future. Class variables with descriptions are represented in Table 5.

Table 5. Class StandardVehicle

Type	Class member	Description
VehicleType	itsType	vehicle type
String	registrationNumber	vehicle registration number

In StandardVehicle class author uses the composition of two classes as this will allow storing fewer overhead data and it is easier to maintain the code.

Class LoadedVehicle is used for storing information of the vehicles, which have been placed on the lane. This class also uses the principle of class composition. Its variables with descriptions are listed in Table 6.

Table 6. Class LoadedVehicle

Type	Class member	Description
StandardVehicle	loadedVehicle	vehicle, which has been placed on the lane
String	laneNum	lane number, compatibility value
String	laneDeck	deck number, compatibility value
float	xMin	minimum X coordinate of vehicle on lane
float	xMax	maximum X coordinate of vehicle on lane

Class VehicleQueue contains information about vehicles to be boarded so there are objects of StandardVehicle class.

The lanes configuration is listed in StandardLane class. Class variables with descriptions are listed in Table 7.

Table 7. StandardLane class

Type	Class member	Description
ArrayList <Vehicle>	laneList	list of vehicles loaded onto this lane
String	laneName	name of deck
String	laneDeck	name of deck where lane is located
int	deckNum	integer deck number for compatibility
double	lanePriority	priority of lane
int	defaultLanePriority	default priority of lane

double	laneCoefficient	lane coefficient, depends upon lane location for forward loading algorithm
double	laneCoefficientReversed	lane coefficient, depends upon lane location for reverse loading algorithm
float	laneMass	total lane load
float	centerX	X coordinate of lane, located in the middle of its length
float	centerY	Y coordinate of lane, indicates lane centre across the ship
float	centerZ	Z coordinate of lane, indicates deck level
float	minX	minimal X coordinate of lane
float	maxX	maximum X coordinate of lane
float	setPoint	point for next vehicle placement
float	availableLength	available space on the lane
List<LaneLocation>	laneLocation	lane location on the ship, reserved
List<VehicleType>	allowedVehicleList	list of allowed vehicles on lane
List<VehicleType>	notAllowedVehicleList	list of not allowed vehicles on lane

Class StandardLane comprises much more information about lane itself than originally was learnt from LD files. This will allow effective usage of lane information with all algorithms explained later. Also, previously defined class VehicleType is used as a classifier.

Class LaneCoefficient represents lane coefficient. This is an enumeration class, used in the calculation of the lane priority during the selection of best suitable lane for a vehicle. Lane coefficient is a lane value, which allows to define a sequence in which order lanes are considered for vehicle placement. Lane coefficient is assigned on the lane initialization and depends on its location and lane own metrics. Same value of lane coefficient can be applied to a single lane or to a group of lanes. If same value is applied to a group of lanes, it means that each lane in the group has equal chance to accept a

vehicle. To resolve a problem which lane in this case will take vehicle first, the lane priority will be used in the selection of the best suitable lane and priority calculation logic is introduced. This is explained in sections 3.5.5, 3.5.6 and 3.5.7.

Example of the class definitions as follows:

```
public enum LaneCoefficient {
    P71 (1), P72 (Math.pow(10,2)),
    P73 (Math.pow(10,4)), P74 (Math.pow(10,6)),
    P61 (Math.pow(10,8)), P62 (Math.pow(10,10)),
    P63 (Math.pow(10,12)), P64 (Math.pow(10,14)),
    P51 (Math.pow(10,16)), P52 (Math.pow(10,18)),
    P53 (Math.pow(10,20)), P54 (Math.pow(10,22)),
    P55 (Math.pow(10,24)), P31 (Math.pow(10,26)),
    P32 (Math.pow(10,28)), P33 (Math.pow(10,30)),
    P34 (Math.pow(10,32)), P35 (Math.pow(10,34));

      private double laneCoefficient;

      LaneCoefficient(double laneCoefficient) {
              this.laneCoefficient=laneCoefficient;}
      double getLaneCoefficient () {
              return laneCoefficient; }
      }
```

Class Ferry represents a ship. Class variables and its descriptions are listed in Table 7.

Table 8. Class Ferry

Type	Class member	Description
List<Lane>	lanes	list of lanes, present on the ship
List<Record>	outputTable	List of values for cargo plan based on loaded vehicles on lanes
List<LoadRecord>	outputLoadTable	List of values for cargo plan compatibility in book list.
String	exportTableHeader	header for cargo plan ROROLOAD section. Not used
String	exportLoadTableHeader	header for cargo plan RRBOOKLOAD section. Not used.

As in life lanes are located on the decks on the ship, so they belong to this class. Other variables were created to represent the sections in the real cargo plan in NAPA format.

Class InputData is responsible for reading input data from the text file.

Class LaneLocation indicates positioning of the lane on the deck (is it bow lane, stern lane, etc.) is listed for compatibility and extended functionality, but is not currently used.

Class MainFrame contains GUI and is implemented with Java Swing library.

The class structure is pictured on Figure 6 below.

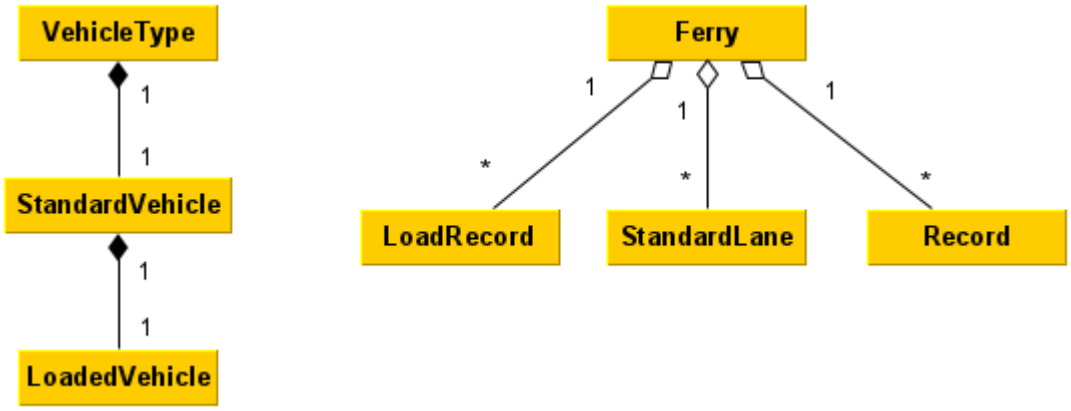

Figure 6. Class relationship

Such organization of data structures allows to solve the task using minimum of different classes and make it easier to maintain the code.

3.4.2 Interfaces

The good style of programming in Java is to implement interfaces to ease the communication between the classes. Logically these are sets of abstract methods, but the great advantage of interface is that any class which supports an interface will be implementing its own version of the method.

Interface Vehicle is used to handle vehicle data, such as return its vehicle length, width, height, registration number, mass, or the whole vehicle type.

Interface Lane offers all required methods to work with lane, including adding vehicles to lane, calculation available space, new set point calculation, lane priority calculation, getting and calculating all coordinates.

Interface Ship works not with particular lane, but with collection of lanes, as lanes are organized in the list. This interface allows to get the full list of lanes, invoke priority calculations for the list, exporting the output tables, selecting best lane etc.

Interface Table works with output tables only and receives a list of records.

3.4.3 Organization of data structures summary

Data organization is crucial before the algorithm implementation. Using the composition of classes which communicate via interfaces makes vehicle placement task easier to accomplish and maintain the code during the application development.

3.5 Algorithm implementation

3.5.1 Background for algorithm implementation

The task of creating effective cargo plan includes the proper vehicle location selection, including proper calculation of its coordinates, and in the meantime maintaining ship stability. As it was stated above, the one-dimensional bin packing algorithm is used as a base algorithm. It targets to fill highly prioritized lanes first in order to maintain ship stability. It should be noted that in order to maintain ship stability, it is crucial to keep the Y coordinate of centre of gravity of cargo as close as possible to the Y coordinate of centre of the empty ship to avoid the ship heel. Also, to avoid ship trim, either positive or negative, is also crucial to maintain X coordinate of centre of gravity of vehicle cargo as close as possible to X coordinate of centre of gravity of the empty ship.

Given the formulae below for calculating the cargo gravity point, where vehicles as cargo items C_1, C_2 … C_N have X coordinates as of CX_1, CX_2 … CX_N , Y coordinates as CY_1, CY_2 … CY_N with masses CM_1, CM_2,…CM_N, we can calculate the coordinates of the centre of the mass of all cargo items as follows:

$$X_{cargo} = \frac{\sum_{i=1}^{N}(CX_i * CM_i)}{\sum_{i=1}^{N} CM_i}$$

$$Y_{cargo} = \frac{\sum_{i=1}^{N}(CY_i * CM_i)}{\sum_{i=1}^{N} CM_i}$$

Given such an equation, and taking into account the stability rules, the vehicle placement task can be formulated as follows: to avoid heel and trim, this is required to keep the centre of the cargo mass for all vehicles closer to the ship gravity point in X and Y dimension. This can be accomplished if heaviest vehicles will be put closer to the centre of gravity or, alternatively, they can be put to the ship sides in equal distances from centre of gravity of the empty ship, thus making sure the load is evenly balanced to both ports to avoid ship heel.

3.5.2 Input from the user

Input is made with simple algorithm to collect initial input data. Process diagram is listed on Figure 7.

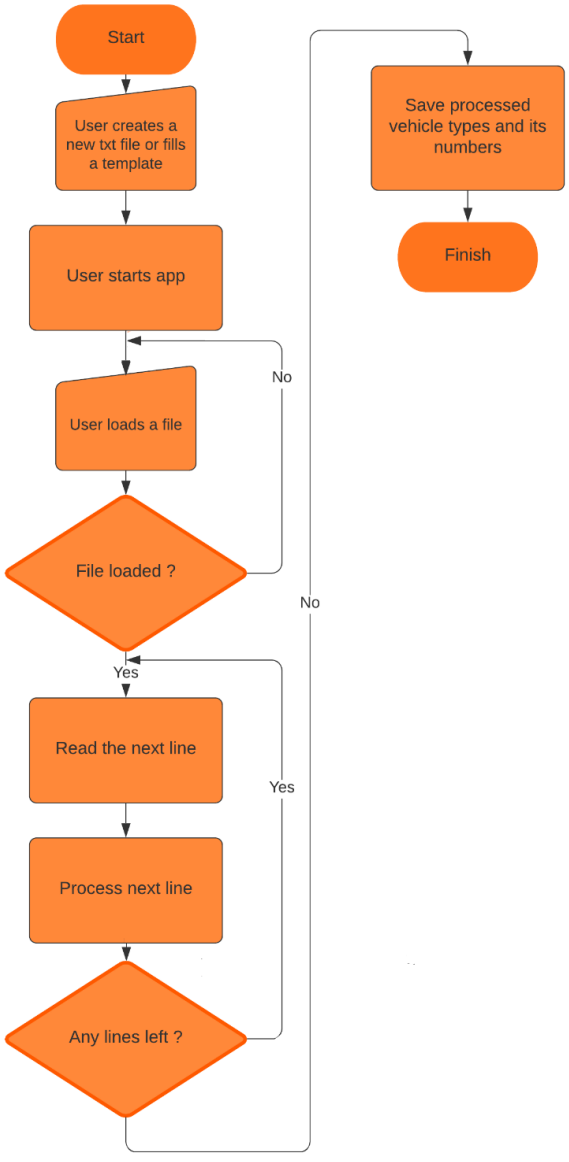

Figure 7. Input process diagram

Format of the file: TYPE_OF_VEHICLE AMOUNT_OF_VEHICLES

Example of the input file:

CAR 200
VAN 15
TT_25 20

3.5.3 Creating of vehicles queue

Based on the data gathered from the user, class VehicleQueue creates the list of vehicles. Each vehicle is automatically assigned a generated registration number. Registration numbers are presented in a format which have all vehicles registered in Estonia. In future development, it can be used for real registration numbers of any country, given the number contains Latin letters and Arabic numerals.

To keep the balance of the ship and avoid heel the heaviest vehicles come first into boarding queue, except for buses, which always will be first in the list. The idea is to put heaviest vehicles to lanes with the highest priority, because such lanes have minimum influence on ship heel. Lane priority is explained in detail in section 3.5.5. Buses should be positioned in the middle of the ship, as ship staircases are located closer to middle lanes. The algorithm, except registration number assignment part, is pictured on Figure 8 below.

The vehicles are put into the list of the StandardVehicle objects. Once all types of vehicles have been put into the queue, the registration numbers are assigned. Generator iterates through the list by randomizing 3 numbers and 3 Latin letters to create standard Estonian registration number.

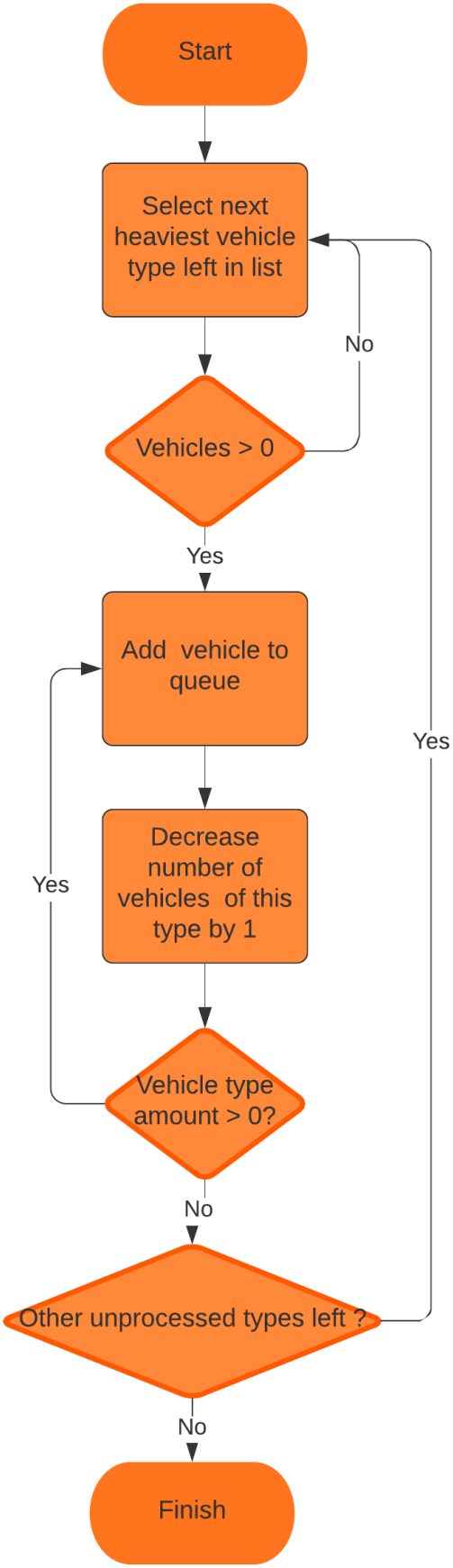

Figure 8. Queue creation algorithm

3.5.4 General loading algorithm.

Loading algorithm by design loads vehicles onto lanes. It picks next vehicle from the queue, finds best suitable lane and tries to add this vehicle to this lane. If this operation successes, it picks up the next vehicle from the queue. If there's not enough place on the ferry, it reports the loading operation was not successful. All loading operations are executed until queue is empty.

As ferry have two departure ports as Tallinn and Helsinki, vehicles embark it from stern ramp in Helsinki and from bow ramp in Tallinn. So, two loading directions exist: forward and reverse. Vehicles normally disembark from the opposite side they entered, thus entering the ferry in Helsinki from stern they can be placed closer to bow and ones entering ferry in Tallinn are placed closer to stern. Forward loading is set to trips from Helsinki to Tallinn, and reverse loading is set to trips from Tallinn to Helsinki.

Trip direction will also affect the calculation of the set point. Set point for the forward algorithm uses maximum X lane coordinate as initial set point for next vehicle. Set point for reverse algorithm uses minimal X lane coordinate as initial set point for next vehicle. After each placement operation set point will be recalculated in all implementations.

All algorithm implementations take into account free space available on the lane, thus the goal is to use all possible lane length until no more vehicles can be placed.

General loading algorithm is applied to all possible implementations of the application. The difference lies in two things: how the lane priority is calculated and how the vehicle coordinates are calculated depending on the loading direction.

Each lane has a list of allowed types of vehicles and a list of not allowed type of vehicles. In case vehicle is allowed, it can be placed on lane if free space is available. In case vehicle type is not allowed to be placed on the lane, this lane priority is calculated with negative value and until next vehicle load such lane is not considered suitable for current vehicle.

The logic of the general loading algorithm is pictured on Figure 9.

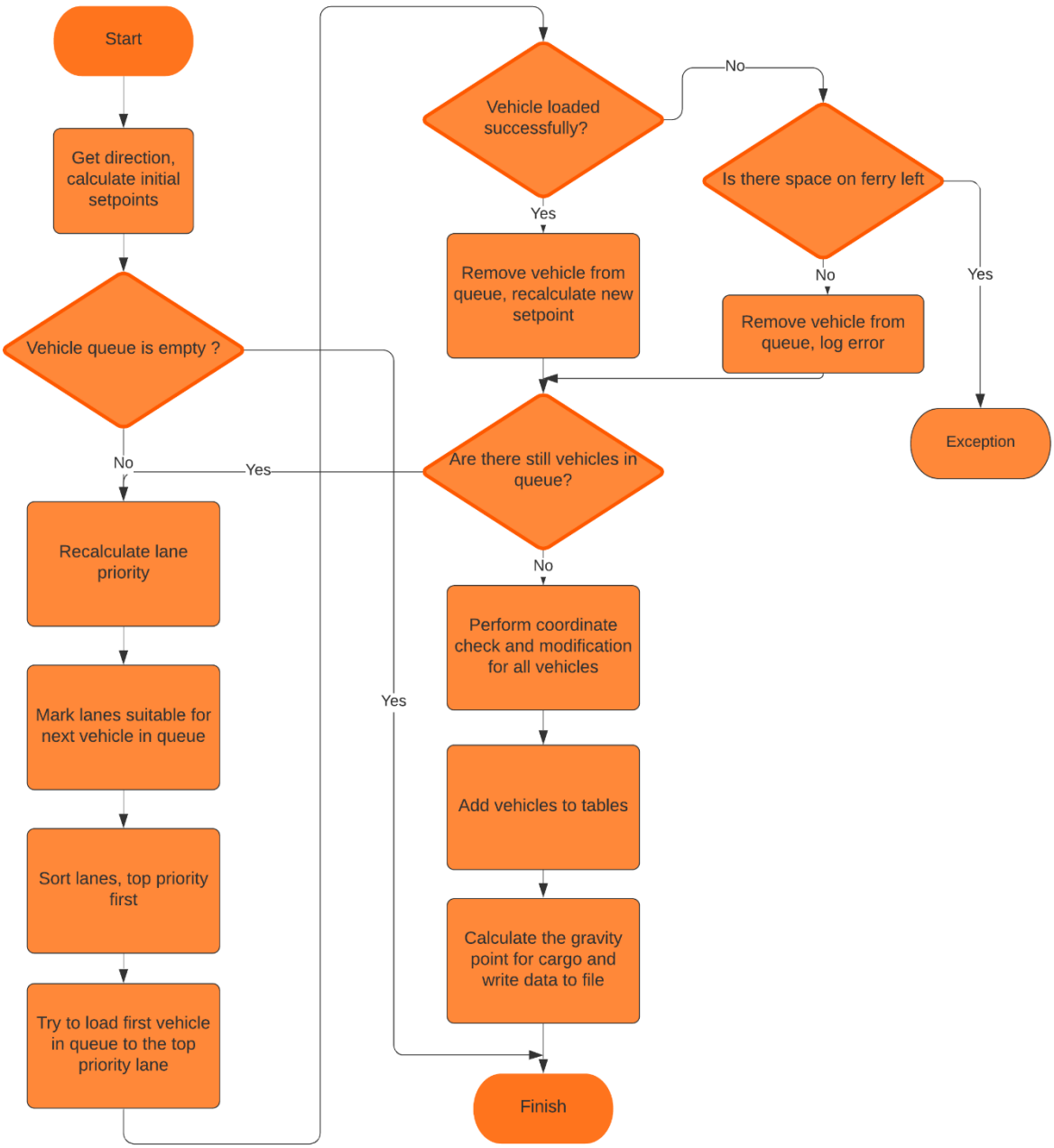

Figure 9. General loading algorithm

Author of this thesis develops and evaluates three algorithms. Algorithm 1 uses lane priority calculation “outer lanes first” and centred lane alignment mechanism. Algorithms 2 and 3 use lane priority calculation “central lane first” logic with vehicle centred lane alignment and “central lane first” with full shift alignment mechanism respectively. Lane priority calculation algorithms are explained in sections 3.5.6 and 3.5.7, vehicle alignment mechanisms are explained in sections 3.5.9 and 3.5.10.

3.5.5 Lane priority calculation algorithms background

Lane priority calculating algorithm basically is the crucial algorithm in vehicle placement task. As vehicles are pushed onto the best suitable lane one after another, there is only one factor that can stop putting vehicles to the currently selected lane and start putting them on another one. This is a lane priority.

All implemented algorithms use the following logic: lane priority is calculated as multiplication of the lane available length (free space) and lane coefficient. Lane coefficient was introduced previously in section 3.4.1. Because author uses forward and reverse loading of the ferry, lane coefficient may also vary for each lane, depending where the ferry is loaded.

Lane coefficient assigns a value to each lane to create order of lanes in which they should be filled. But what happens when top priority lane has been almost full and next vehicle cannot be put, but lane itself theoretically can still accept smaller vehicles in the queue? That is where lane free or available length should be taken into account. If it is physically possible to fit a vehicle into the lane with highest lane coefficient, such lane must always have higher priority over any other lane which has less value of lane coefficient. If vehicle does not fit into lane free space, lane automatically is not considered suitable for the vehicle and its lane priority is not taken into account.

Calculation of the lane available space is just calculation of available space from the set point of next vehicle placement point till end of the lane. Initial set point location has been introduced in the previous section.

In selection of criterion how the lane priority will be calculated following key factors are considered:

- Proximity of the lane to the Y coordinate of empty ship
- Lane location on deck, for example stern, bow, full hull-length lane
- Available space on the lane
- Deck where lane is located

For better understanding of lane positioning, please refer to Table 16 in Appendix 2. This table shows for each lane centre X coordinate (or XCG), lane Y coordinate (or YCG). Lane middle point XCG calculated as equidistant point between XMIN and

XMAX, being minimal and maximal X lane coordinates respectively. Lane length has been calculated using these coordinates and is listed below in the length column. Columns X_DIFF and Y_DIFF show the difference between lane XCG and X coordinate of the centre of the gravity of the empty ship, and lane YCG and Y coordinate of the centre of the gravity of the empty ship. All distances are given in meters.

Given the following data, it is possible to establish, where each lane is located on the ship and understand which lanes lie in symmetry to ship X axis.

The sketch of lanes locations (based on lanes data, presented in Appendix 2) on decks are listed below. Figure 10 pictures lane locations on deck 3. Deck 5 lane layout is pictured on Figure 11. Lanes of deck 6 and deck 7 are pictured on Figures 12 and 13 respectively.

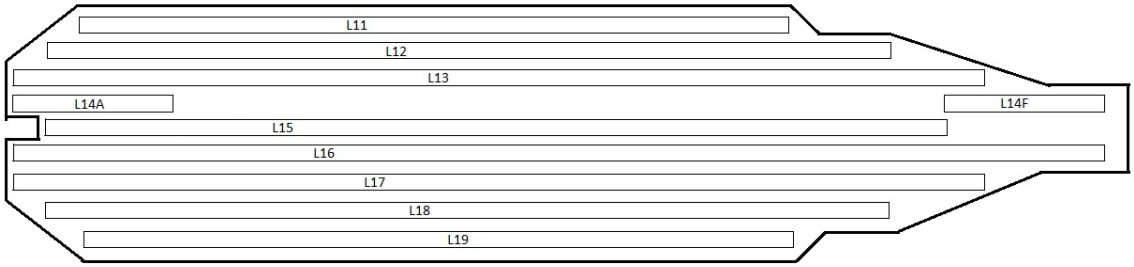

Figure 10. Deck 3 lane layout

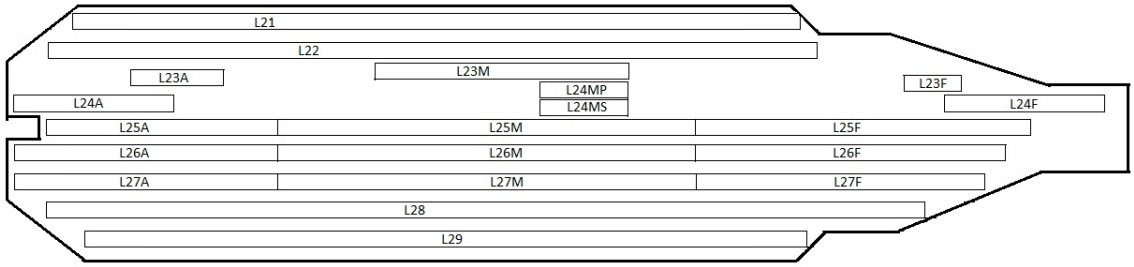

Figure 11. Deck 5 lane layout

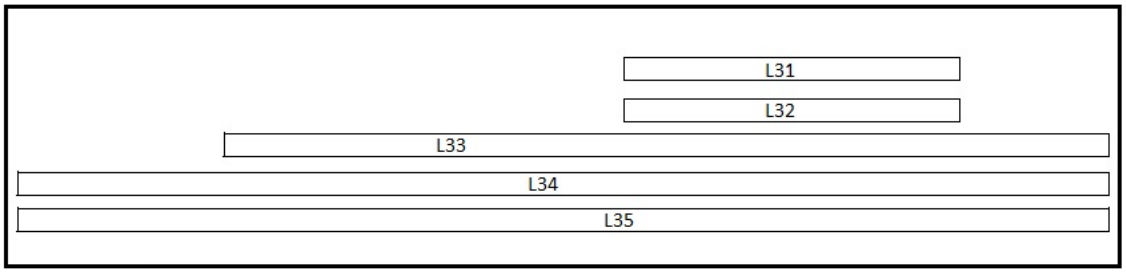

Figure 12. Deck 6 lane layout

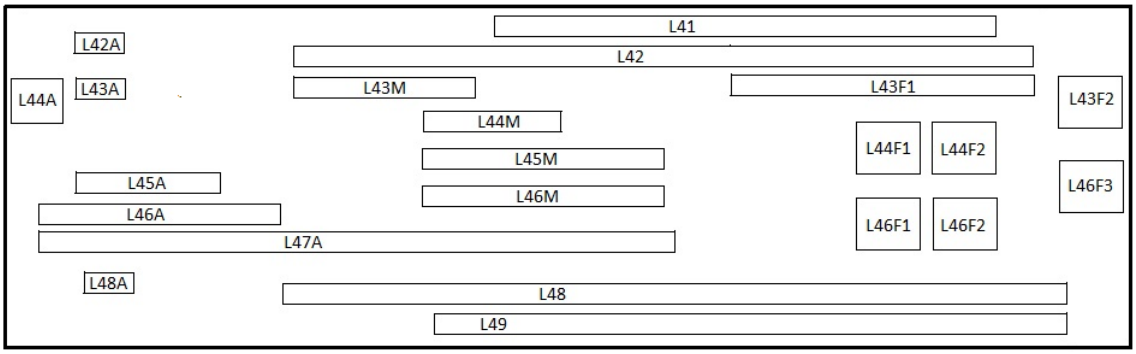

Figure 13. Deck 7 lane layout

3.5.6 Lane priority calculation “outer lanes first” logic

The logic of lane priority calculation “outer lanes first” was explained in previous section. Lane coefficient in this case is deck independent.

According to ship stability principles, this is crucial to load both ports equally. Bosuns tend to load outer lanes first. Thus, the following logic is selected: lane coefficient has highest values closer to the port side or starboard and lowest coefficients are for lanes in the centre of the ship. Once loading of vehicles is finished the vehicle centred alignment mechanism is applied. This mechanism is explained in section 3.5.9.

Each deck is using the same criterion of assigning lane coefficients, so the decks are populated purely depending the lane coefficient and lane available length irrelevant to the deck number.

Example of the lane coefficients is presented for deck 3 in Table 9. Outer lanes have highest values.

Table 9. Example of lane coefficient for deck 3

Lane(s)	Forward loading coefficient	Reverse loading priority coefficient
L11, L19	100 000	100 000
L12, L18	10 00	10 00
L13, L17	10	10
L14A	1	0.1
L14F	0.1	1
L15	0.1	0.1
L16	1	1

The selection of the coefficient presented in Table 9 is explained here. The difference between neighbour coefficients in 100 times dictates that even smallest vehicle should be placed to a lane with higher lane coefficient value. Given that the maximum length of any lane is no more than 188 meters and smallest vehicle length is just 2 metres, a lane with 2 spare metres with coefficient value of 1000 should get higher priority than completely free lane with spare 188 metres and with lane coefficient of 10. Small lanes L14A and L14F have also non-standard coefficient of 1, because they are small, and they are close to central lane and its load does not significantly affect the heel.

3.5.7 Lane priority calculation "central lane first" logic

Second algorithm of calculating lane priority will be using same idea that lane coefficients will be using the criterion of each next lane coefficient is 100 times greater than the previous one, but in this case the coefficients will be increased or decreased depending on the position of lane on the deck plus depending on which deck lane itself is located. In this scenario, the lane coefficient is deck dependent: the higher the deck, the lower the coefficient.

Figure 14 shows virtual X-axis of the ship and lane symmetry is seen alongside it for deck 5.

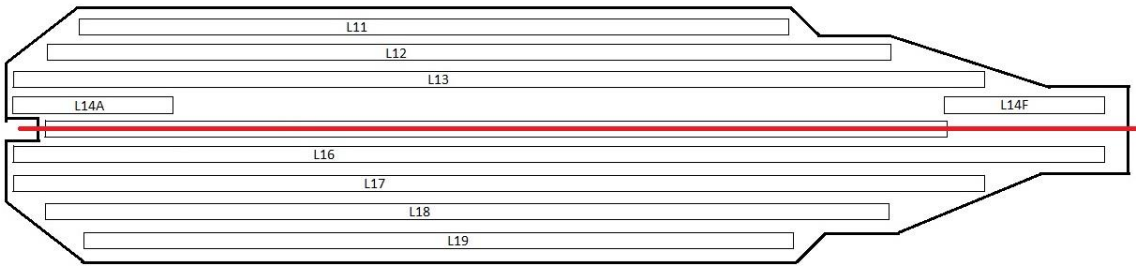

Figure 14. Lane symmetry example for deck 5

The X-axis of the ship almost overlaps the lane L15 on deck 3. According to the ship stability rules, it is important to place vehicles evenly on each side of that line, so vehicles' weight is distributed evenly on both ports. It is even better to place vehicles to the lower deck for the crew. In this algorithm decks are filled starting from lower decks and going to upper decks, only after lower decks have been fully loaded, except cases, where vehicles should be placed on higher decks because of standing restrictions.

In vehicle placement algorithms 2 and 3, which use this priority calculation algorithm, the heaviest vehicles will be placed closer to the centre of the ship, so central lanes will have the highest priority and lower priority will be gradually assigned to the outer lanes. Lanes with lowest priority are located directly on the port side and starboard. But, according to the logic of filling lower decks first, the rule that even 2 meters on lane with lowest priority on lower deck will outweigh by priority fully free central lane on the next deck up.

The example coefficients are represented in Tables 10 and 11. Lane locations are taken into account as seen from Table 16 located in Appendix 2.

Looking at coefficients in Tables 10 and 11 it is easy to see that under normal circumstances, the lanes on deck 3 will be loaded prior to loading of deck 5.

Table 10. Lane coefficients for deck 3 in second algorithm

Lane(s)	**Forward loading coefficient**	**Reverse loading priority coefficient**
L11, L19	10^{26}	10^{26}
L12, L18	10^{28}	10^{28}
L13, L17	10^{30}	10^{30}
L14A	10^{32}	10^{32}
L14F	10^{32}	10^{32}
L15	10^{34}	10^{34}
L16	10^{32}	10^{32}

Table 11. Lane coefficients for deck 5 in second algorithm

Lane(s)	**Forward loading coefficient**	**Reverse loading priority coefficient**
L21, L29	10^{16}	10^{16}
L22, L28	10^{18}	10^{18}
L23A	10^{22}	10^{16}
L23M, L26M	10^{22}	10^{22}
L23F	10^{22}	10^{24}
L24A	10^{22}	10^{24}

L24F	10^{22}	10^{16}
L24MP, L24MS	10^{24}	10^{24}
L25M. L25F	10^{24}	10^{24}
L25A	10^{24}	10^{16}
L26A	10^{16}	10^{16}
L26F	10^{18}	10^{24}
L27M	10^{20}	10^{20}
L27A	10^{16}	10^{16}
L27F	10^{16}	10^{24}

3.5.8 Vehicle alignment mechanism background

In order to keep the ship stable within the limits for heel and trim, author introduces the mechanism of vehicle alignment. This mechanism allows adjusting of the cargo placement within one particular lane.

The alignment of cargo takes place after all vehicles have been loaded onto all lanes and is performed in all implementations of vehicle placement algorithms.

3.5.9 Vehicle centred lane alignment mechanism

This mechanism places vehicles in such a way that the centre of the mass of the lane cargo is always placed to the centre coordinate of the lane. The algorithm is pictured on Figure 15.

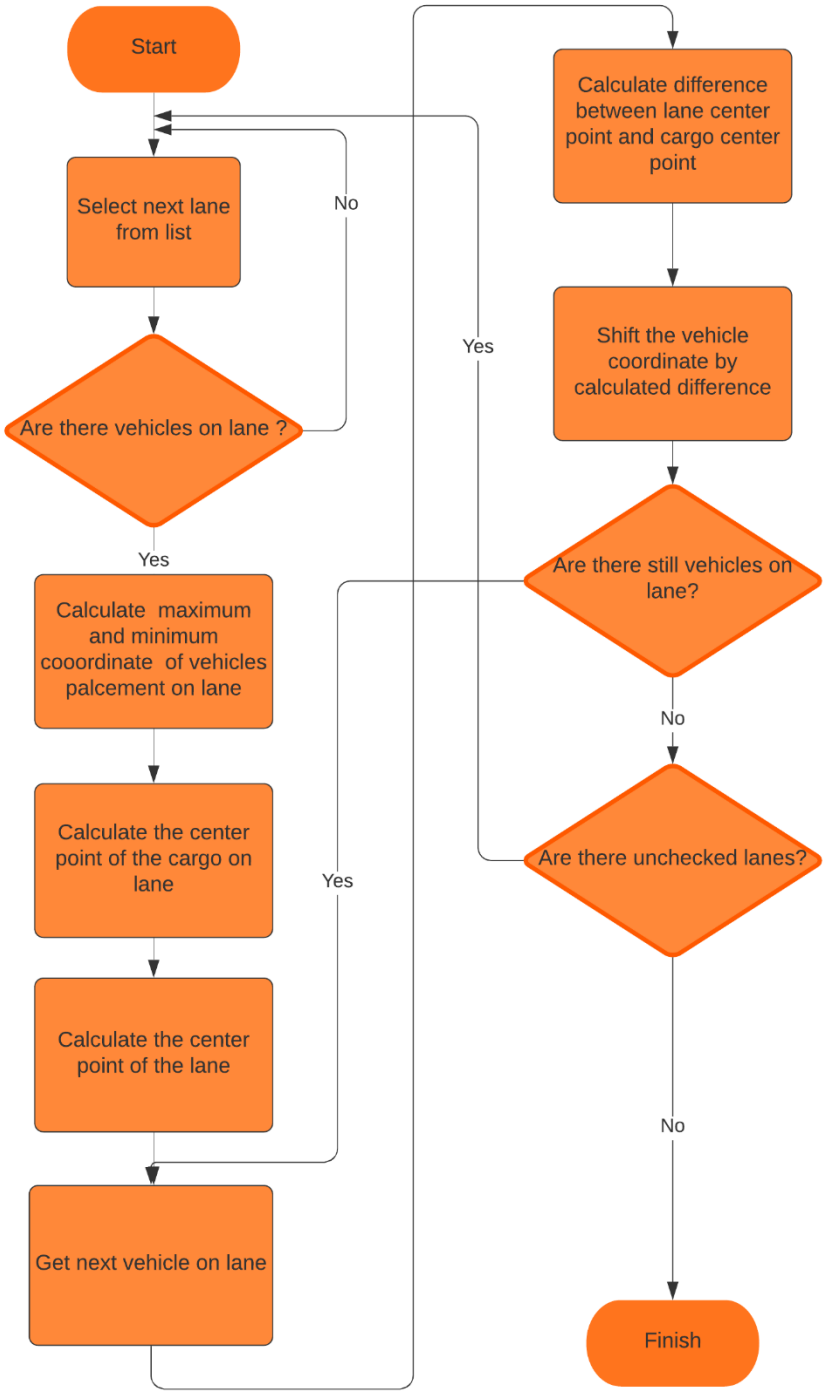

Figure 15. Vehicle centred lane alignment mechanism

The example of mechanism is pictured below on Figures 16 and 17 on next page.

Length L defines the length on X axis from the ship gravity centre to cargo centre of mass. Length L is decreased with such mechanism applied.

Figure 16 shows vehicle placement location before the alignment and Figure 17 shows the cargo position after alignment. The shift is calculated using lane minimum and

maximum X coordinates and cargo minimum and maximum coordinates, as such that free space after alignment on both sides of the lane are equal and cargo is centred on lane.

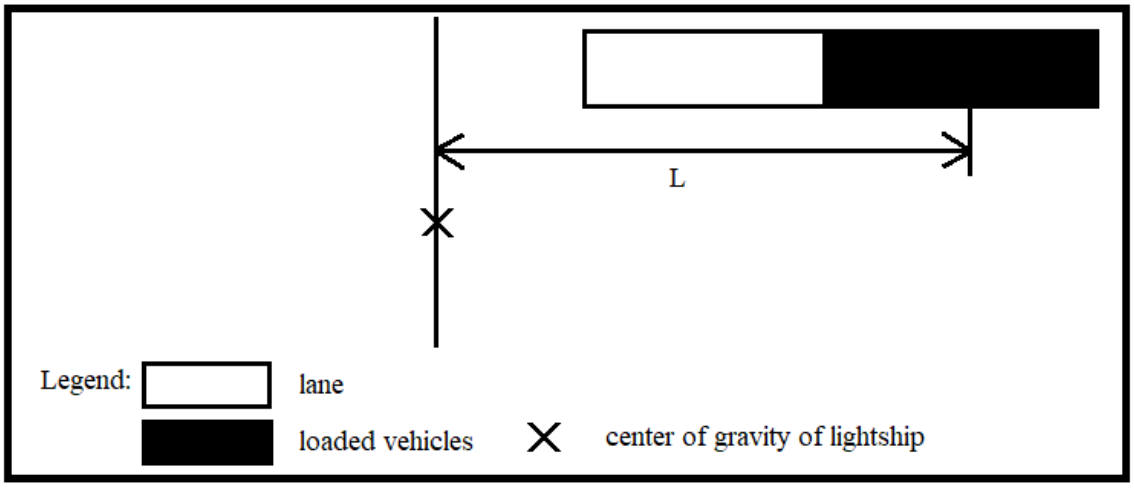

Figure 16. Cargo placement without the centred lane mechanism applied

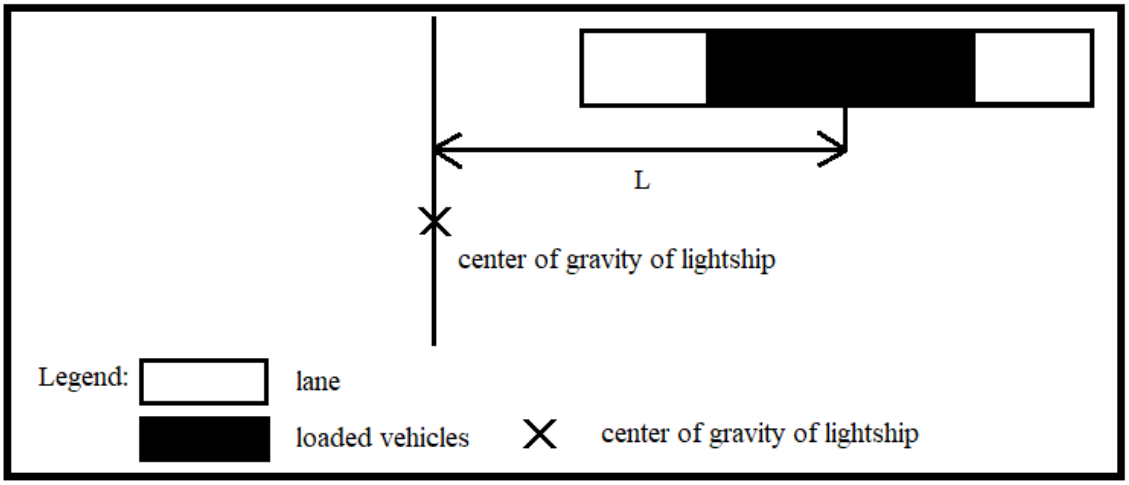

Figure 17. Cargo placement with the centred lane mechanism applied

Once vehicles are aligned, it allows to keep cargo closer to the empty ship gravity point and improves the ship trim.

3.5.10 Vehicle full shift alignment mechanism

This mechanism is almost the same as mechanism explained in previous section, but with the additional shift for all cargo on particular lanes to minimal or maximum coordinate, which lies closer to the lightship gravity point. This additional shift applies only to the bow or stern lanes, which maximal or minimal coordinates do not cross the midships. For bow lanes this is will be shift to minimal coordinate, for stern lanes to the maximum coordinate. For all other lanes centred lane alignment mechanism is apllied.

For example, the bow lane cargo is shifted on Figure 18.

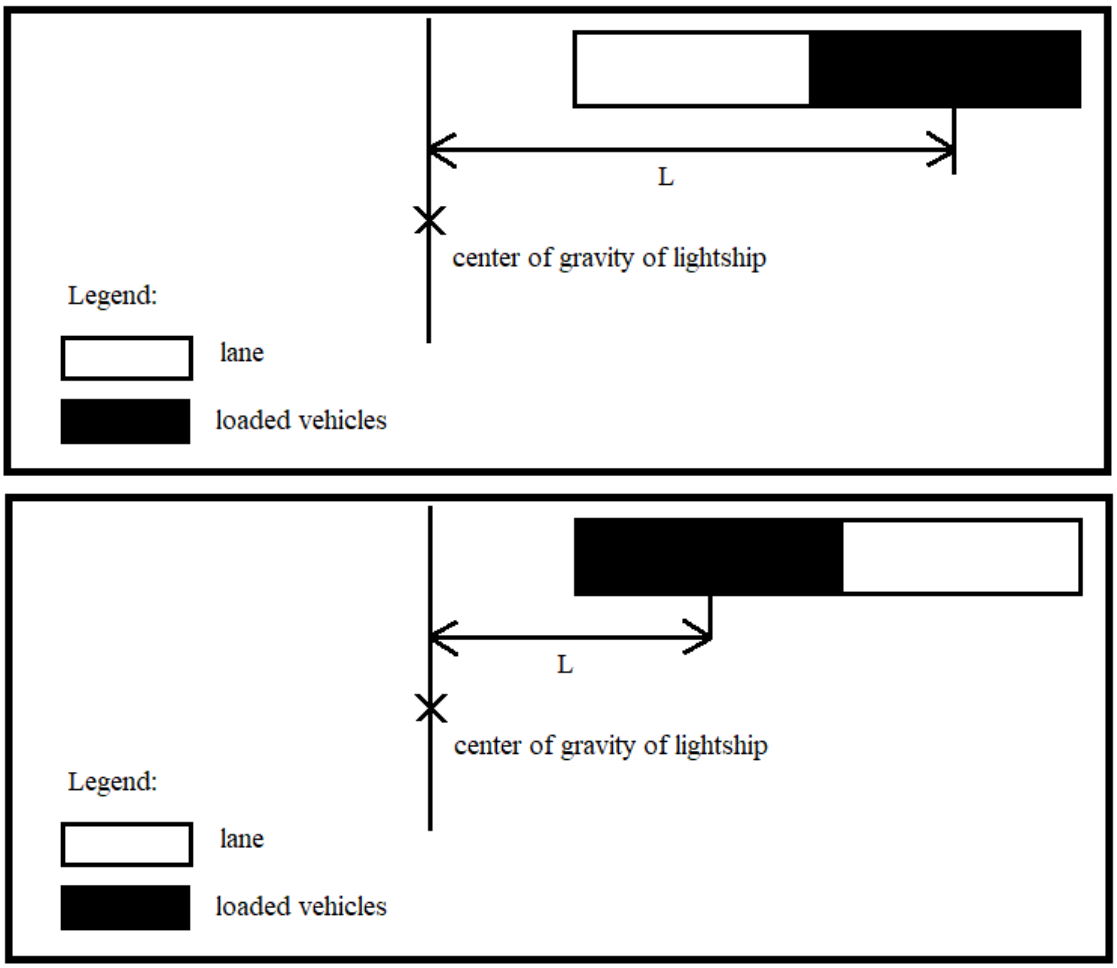

Figure 18. Vehicle full shift alignment mechanism

Such shift can decrease the ship trim even more than the previous algorithm.

3.6 Graphical user interface

Graphical interface is made especially for simple use. Start of the program is pictured on Figure 19 below. After program starts, user is required to load a filled text template, based on the data from the ticket office. This is pictured on Figure 20 below.

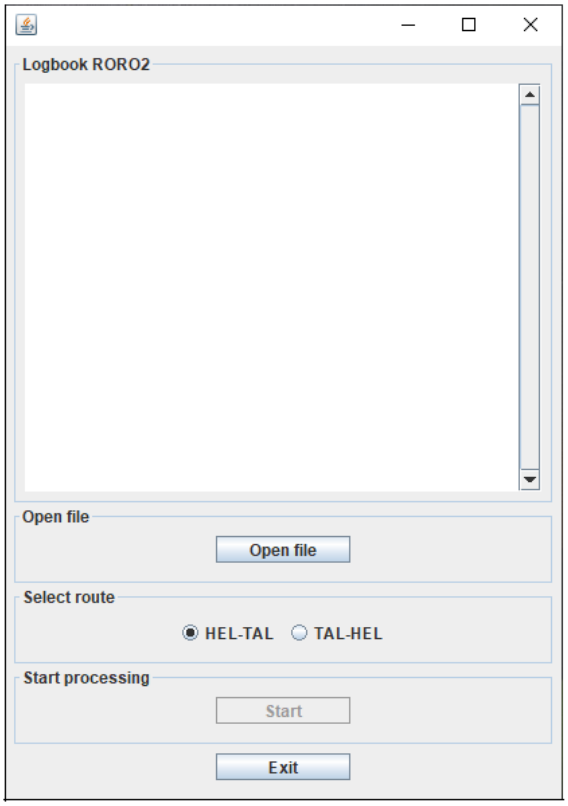

Figure 19. Start of the program screenshot

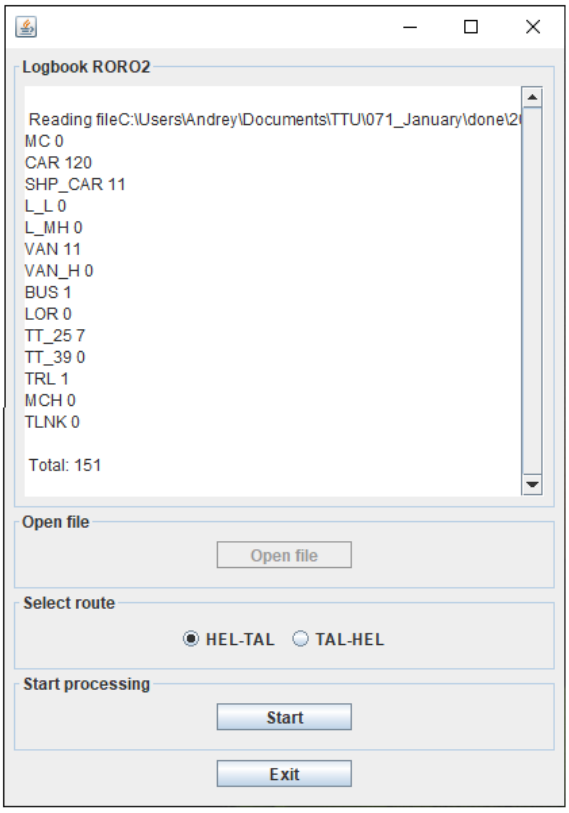

Figure 20. User input screenshot

Once initial data has been read from file, user is required to select trip direction and click Start button to get resulting file in text format and technical information is printed in an application text field. The output of the program is pictured on the Figure 21.

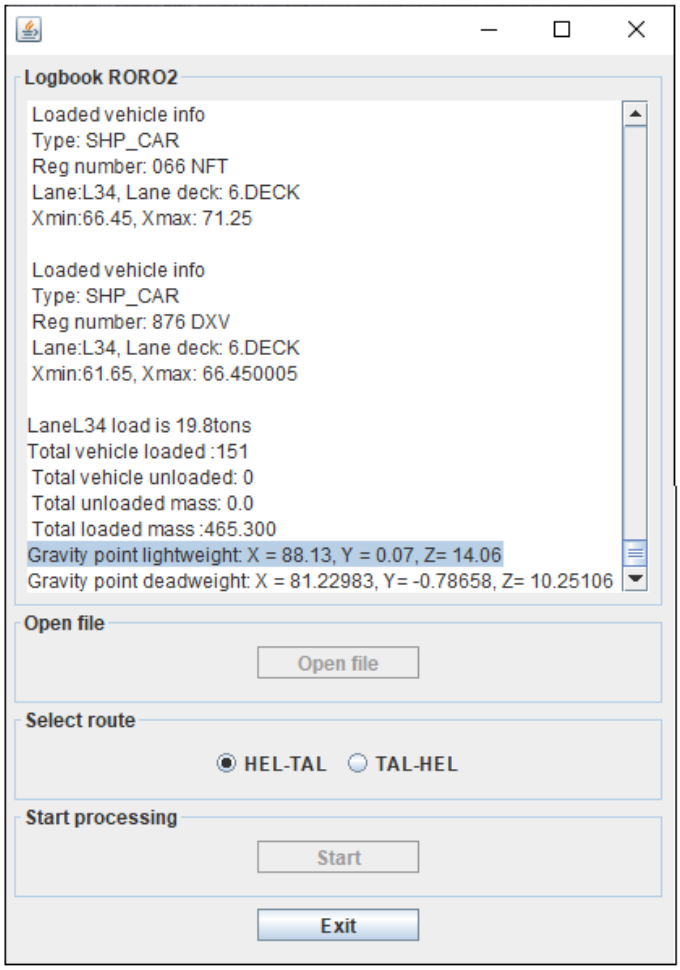

Figure 21. Output of the program screenshot

User is provided with the information of the vehicles loaded and centre of gravity coordinates. In case some vehicles have not been loaded, the information is printed in text area. Resulting text file is pipelined to the command line converter and resulting LD file is generated automatically.

3.7 Future development

The program can be improved in the future. The vehicles were supplied with the registration numbers, so it can be used with real registration numbers if needed.

The structure of the program allows adding more lanes and changing lane restrictions.

4 Results analysis

4.1 Testing background

For analysis of the results, 500 loading conditions have been selected. Each of three algorithms was tested with same 500 loading conditions of real operational data. The empty ship centre of gravity coordinate is X=88.13, Y=0.07 and Z=14.06. All coordinates are given in metres. In addition for analysis a selective testing on the real Tallink software is made to confirm the proposal suggested by the evaluation criterion.

4.2 Analysis of the data

Output data (cargo gravity point X, Y and Z coordinates) produced after 1500 runs of the application (3 algorithms were tested each with 500 loading conditions) was organized into groups to estimate how algorithms work with different cargo load. The loads are split into groups: from 0 to 500 tons, 500 tons to 1000 tonnes. 1000 to 2000 tons, 2000 to 3000 tons, 3000 to 4000 tons and above 4000 tons.

Evaluation criterion explained in section 2.4 will be used during the testing without the NAPA software. During the selective testing with the NAPA software heel and ship information will be used for algorithm perfomance evaluation.

4.2.1 Analysis using evaluation criterion and selective tests

In this section, the author considers vehicle placement plan results evaluated by the coordinates of cargo centre of gravity point and then analyses data with NAPA software from the selective tests for each cargo group.

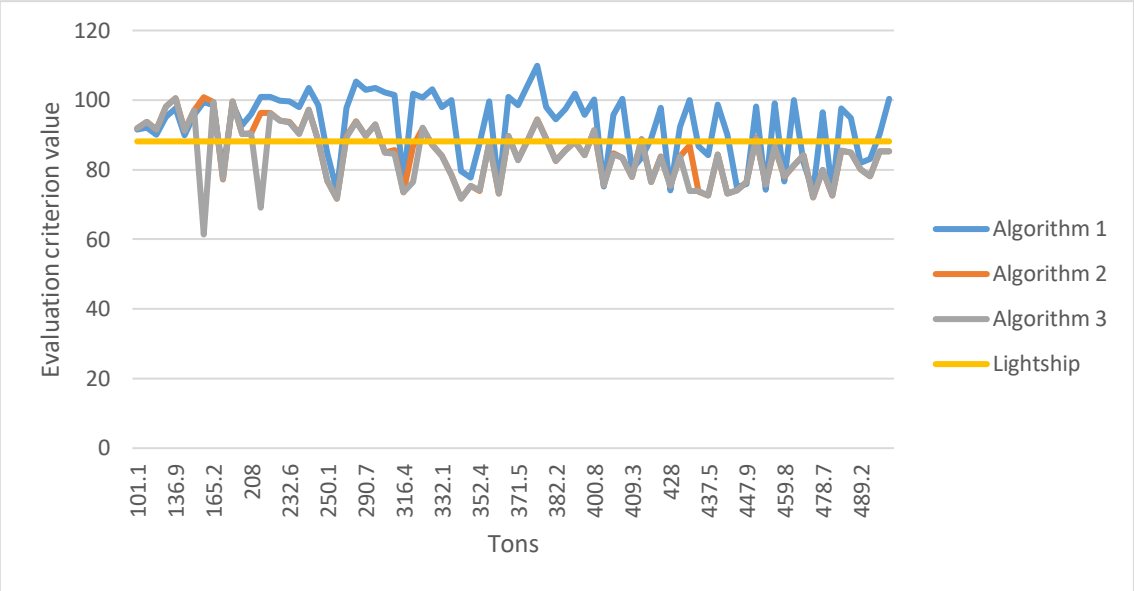

Figure 22. Algorithm comparison with loads up to 500 tons

Figure 22 shows the evaluation criterion values for loads up to 500 tons. Total number of all cases reviewed is 80 or 16% of all cases. As can be seen from the figure, algorithms 2 and 3 overlap each other and lie closer to the empty ship (lightship mass) value. The chart shows that algorithms 2 and 3 should give better results than algorithm 1.

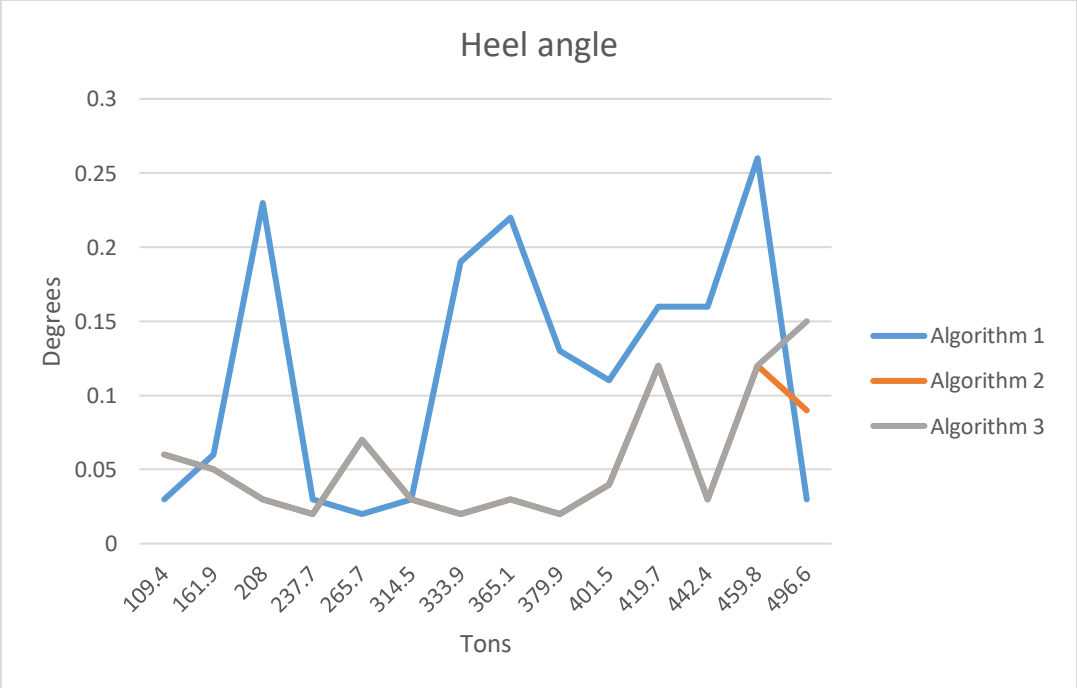

Figure 23. Heel comparison for loads up to 500 tons

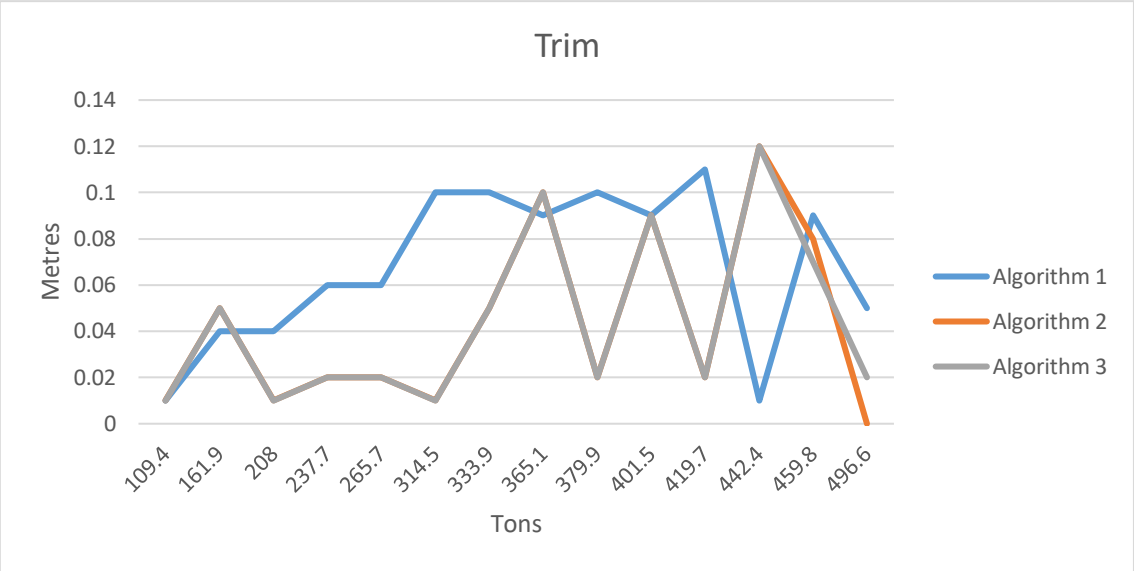

Figure 24. Trim comparison for loads up to 500 tons

Figures 23 and 24 show results of selective tests in this cargo group in NAPA software.

Chart indicate that algorithms 2 and 3 are more effective in minimizing heeling, because vehicles are placed closer to the empty ship gravity point and cargo mass is not too heavy to cause ship heeling.

Algorithms 1 is less stable with increased heel angles as vehicles are placed further away from ship gravity point and even a single heavy vehicle put on the port side or starboard can break the ship balance.

Trim for all three algorithms is usually less than 0.12 metre and this can be considered as excellent result.

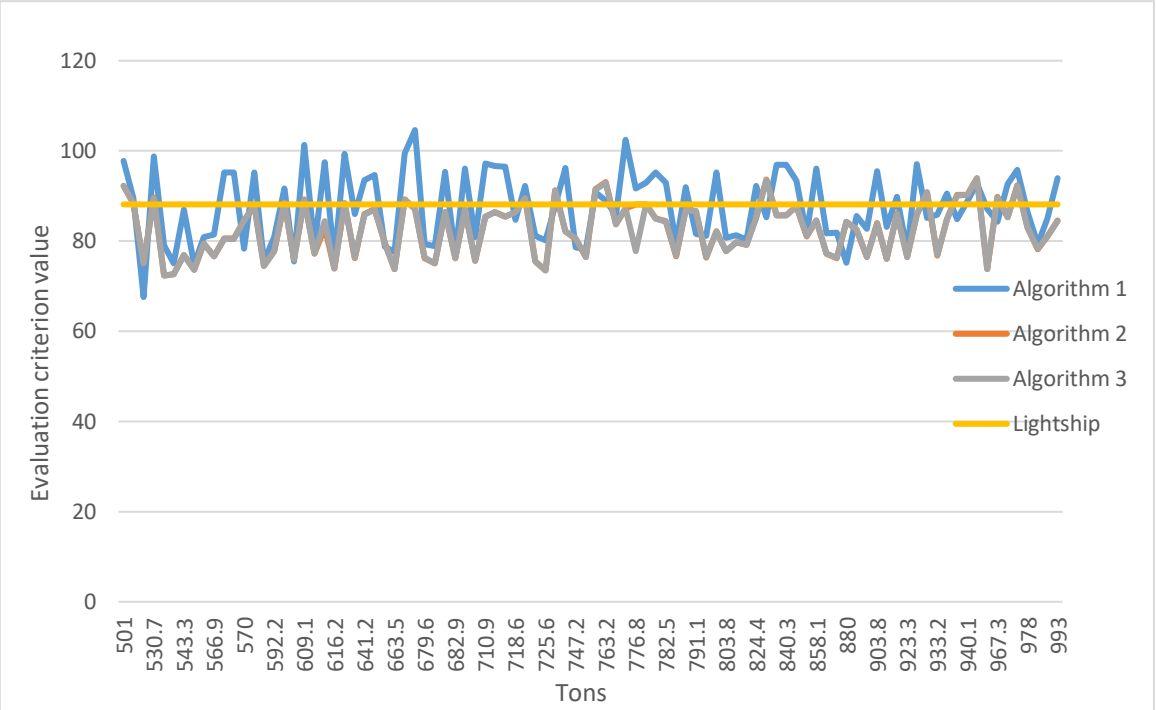

Figure 25. Algorithm comparison with loads between 500 and 1000 tons

Figure 25 represents loads from 500 to 1000 tons and comprises 94 cases out of 500 and this is 18.8% of all cases.

Chart shows that evaluation criterion value of algorithm 1 is located above empty ship criterion value and it indicates that X coordinate is shifted to bow. Algorithm 2 and 3 in turn put the cargo closer to aft and overlap each other on the chart.

The average X coordinate for algorithm 1 is 87.45 and average Y coordinate is -0.53, and for algorithms 2 and 3 X coordinates are 82.73 and 82.08 respectively and average Y coordinate value is -1,01.

Heel and trim for loads from 500 to 1000 tons are pictured on Figures 26 and 27 respectively.

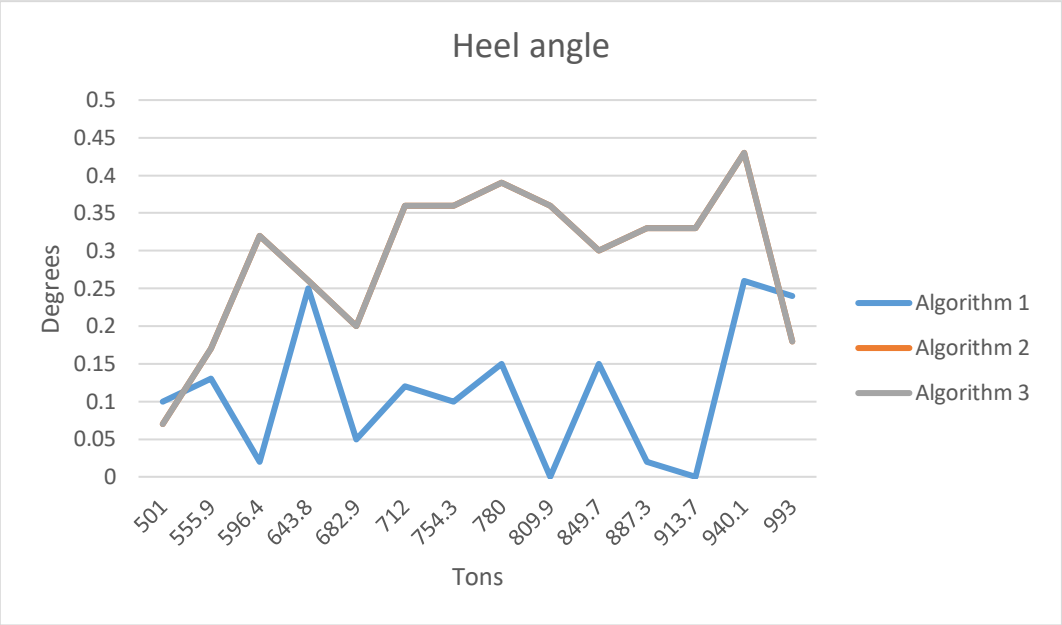

Figure 26. Heel comparison for loads from 500 to 1000 tons

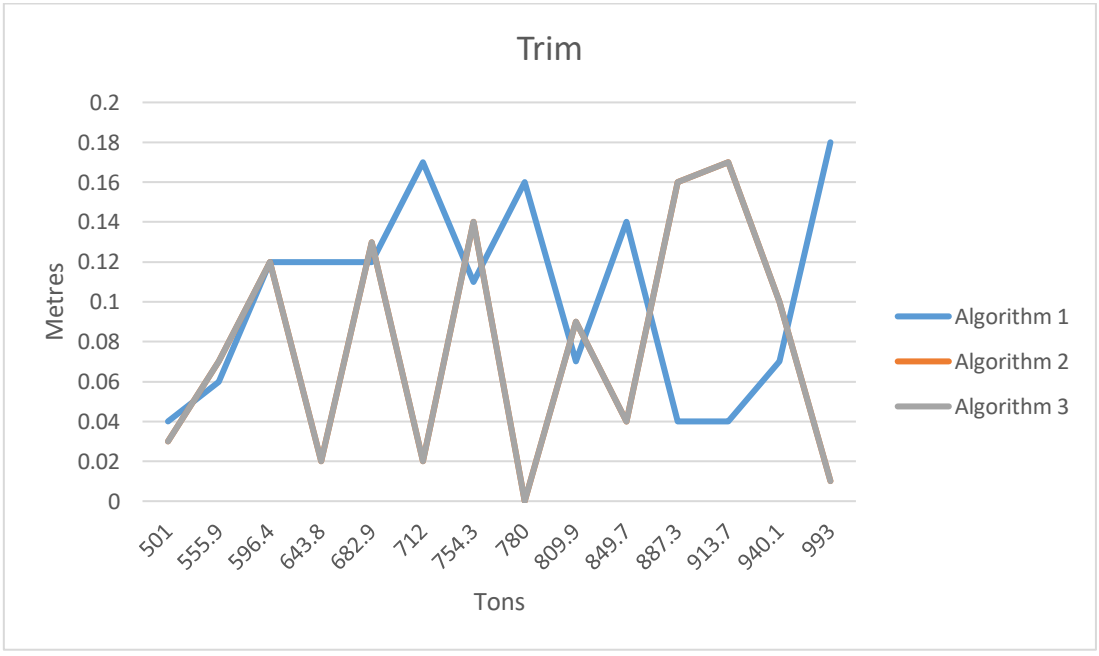

Figure 27. Trim comparison for loads from 500 to 1000 tons

Figures 26 and 27 confirm, that on loads up to 1000 tons algorithm 1 has a better heel, than algorithms 2 and 3 have. They overlap each other on chart. The load is heavier than in previous group and now a single truck does not affect the ship balance evidently. Trim is also good with average value about 0.1 metre for all algorithms.

Overall, this two groups of cargo loads represents 34,8% of all cases reviewed.

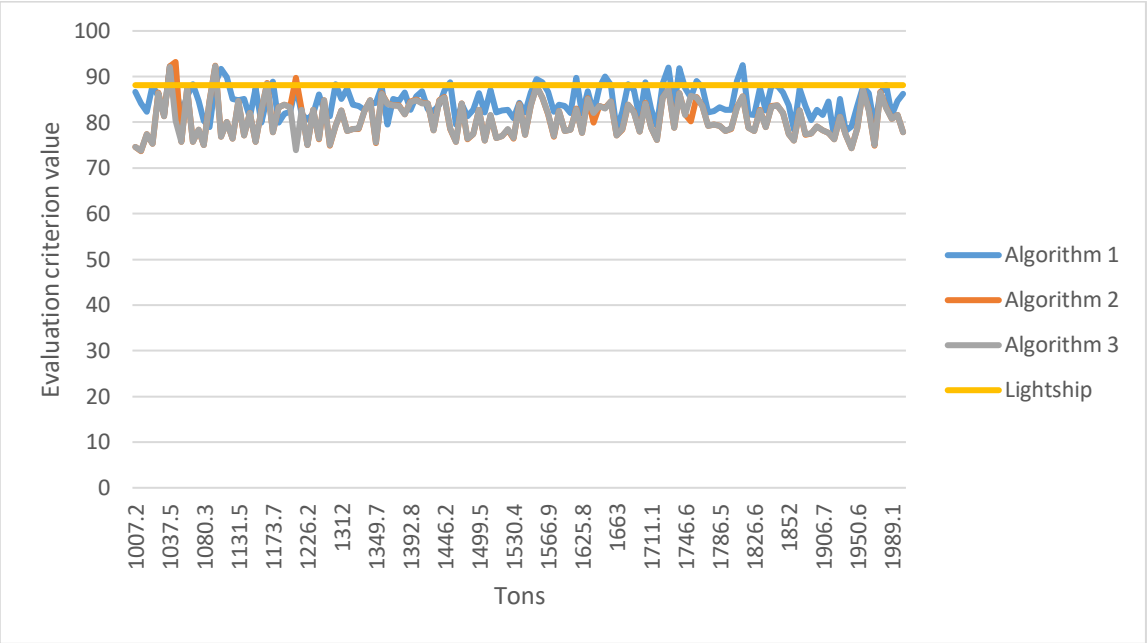

Figure 28. Algorithm comparison with loads between 1000 and 2000 tons

The loads between 1000 and 2000 tons represent 136 cases or 27,2 % of all cases. The relative values for corresponding loads are pictured on Figure 28.

There is a strong indication that algorithm 1 places cargo closer to the empty ship gravity point than algorithms 2 and 3, which almost overlap each other on chart. This is can indicate that either heel, or trim, or even both values after processing data with algorithm 1 should be better, than heel and trim produced with algorithms 2 and 3.

Average X coordinate value for algorithm 1 is 84.62 versus algorithm 2 and 3, coordinate X is 80.96 and 80.76 respectively. Average Y coordinate is still better at value -0.63 for algorithm 1 and average Y coordinates for algorithms 2 and 3 respectively -0.75 and -0.748.

With the increase of tonnage vehicle alignment mechanism can slightly decrease the heel.

Heel and trim comparison for loads between 1000 and 2000 tons are pictured on Figures 29 and 30 respectively.

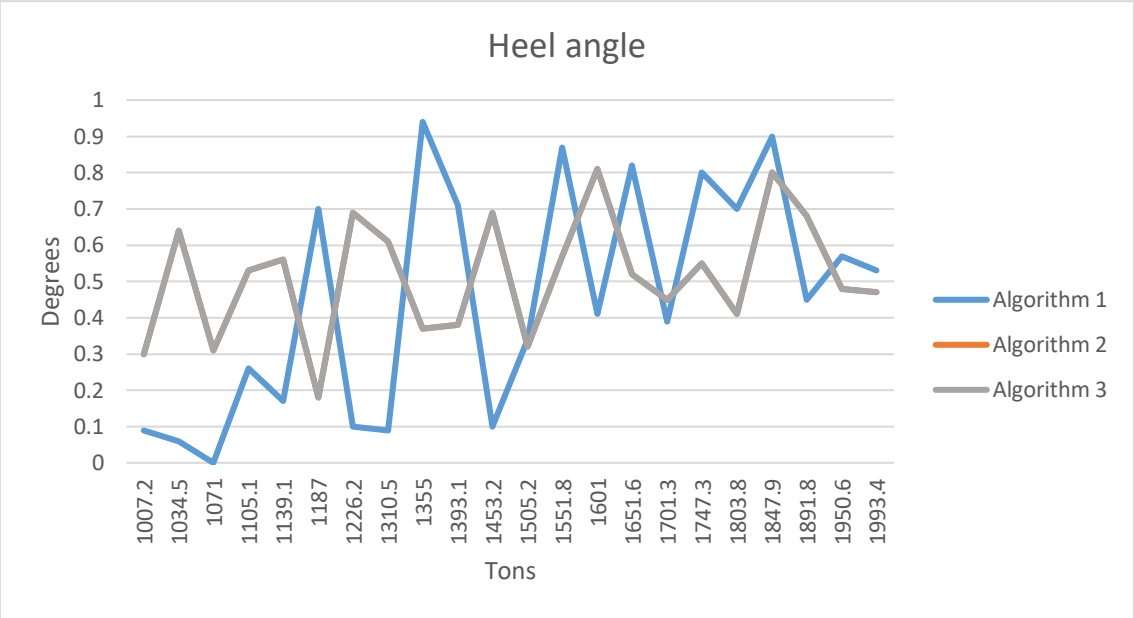

Figure 29. Heel comparison for loads between 1000 and 2000 tons

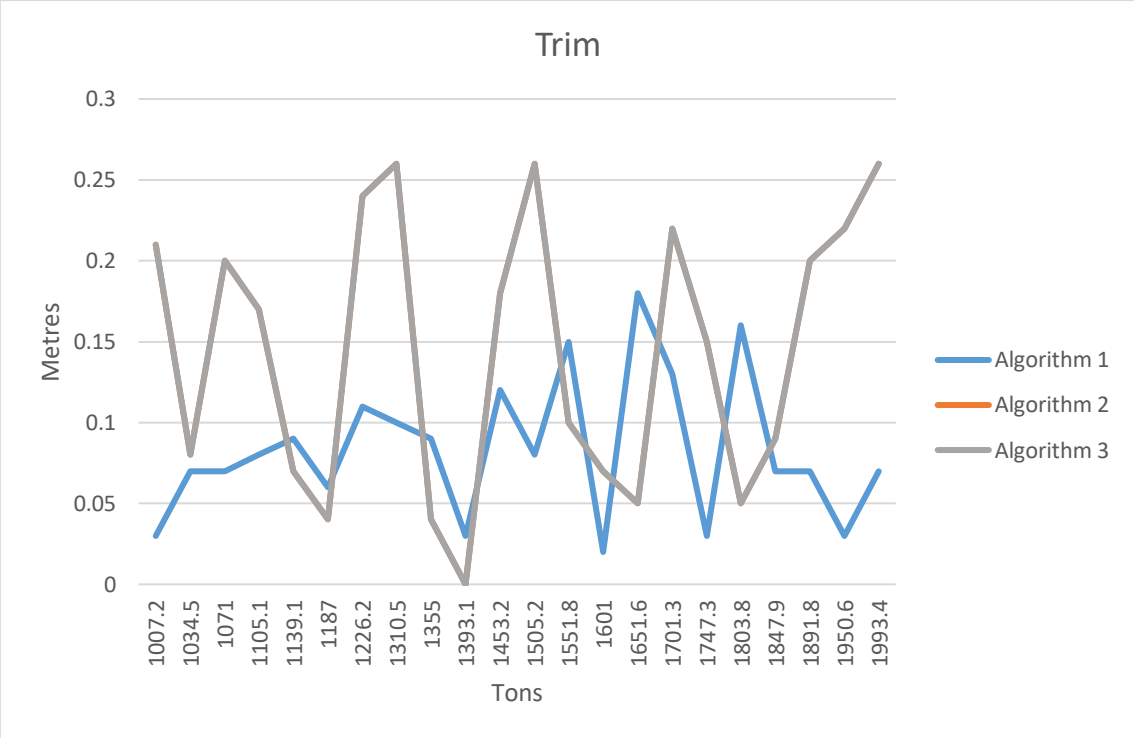

Figure 30. Trim comparison for loads between 1000 and 2000 tons

Figures 29 and 30 confirm the suggestion, that for loads between 1000 and 2000 tons algorithm 1 is preferred over algorithms 2 and 3, which overlap on chart.

Algorithm 1 is heavily dependent on the types of vehicles and is most efficient when number of trucks is near 30 in total. Algorithms 2 and 3 are less sensitive to the type of vehicles in the load.

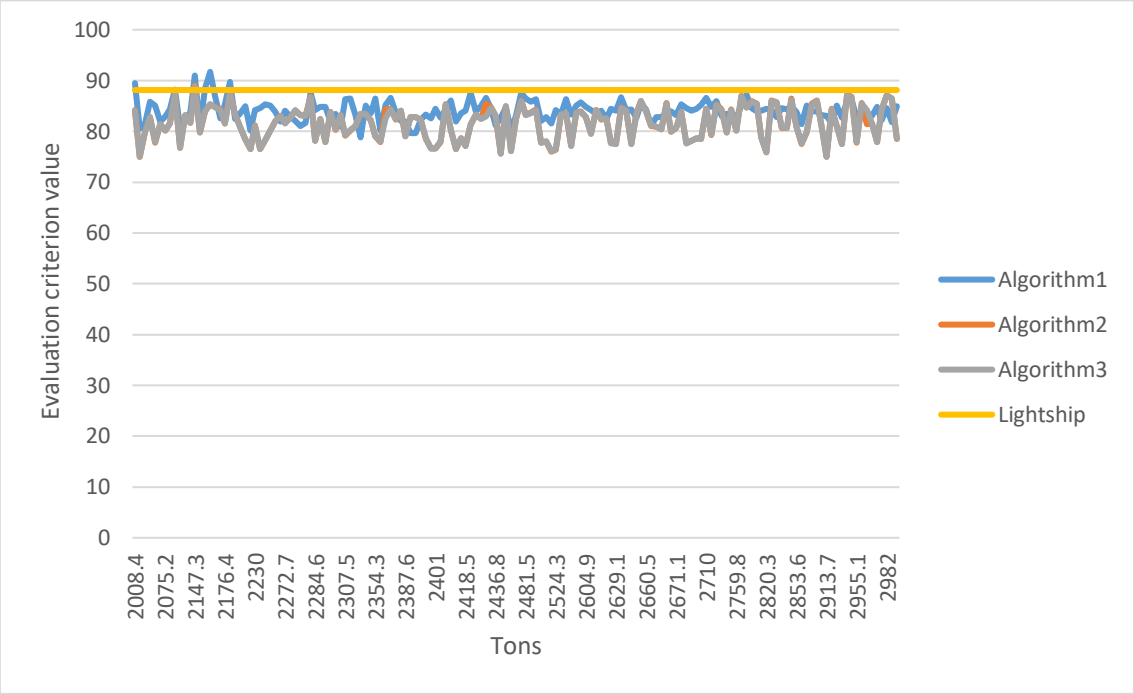

Figure 31. Algorithm comparison with loads between 2000 and 3000 tons

Figure 31 shows the evaluation criterion values for loads between 2000 and 3000 tons. They represent 153 cases or 30.6 % of all cases.

The chart indicates that algorithm 1 should perform better that algorithms 2 and 3. Average X coordinate value for algorithm 1 is 84.02 against is 81.67 and 81.60 respectively. Average Y coordinate is now worse for algorithm 1 with average value -0.71 for algorithm 1 and average Y coordinates for algorithms 2 and 3 respectively -0.69.

The numbers suggest, that difference between all algorithms is not very significant in this cargo group.

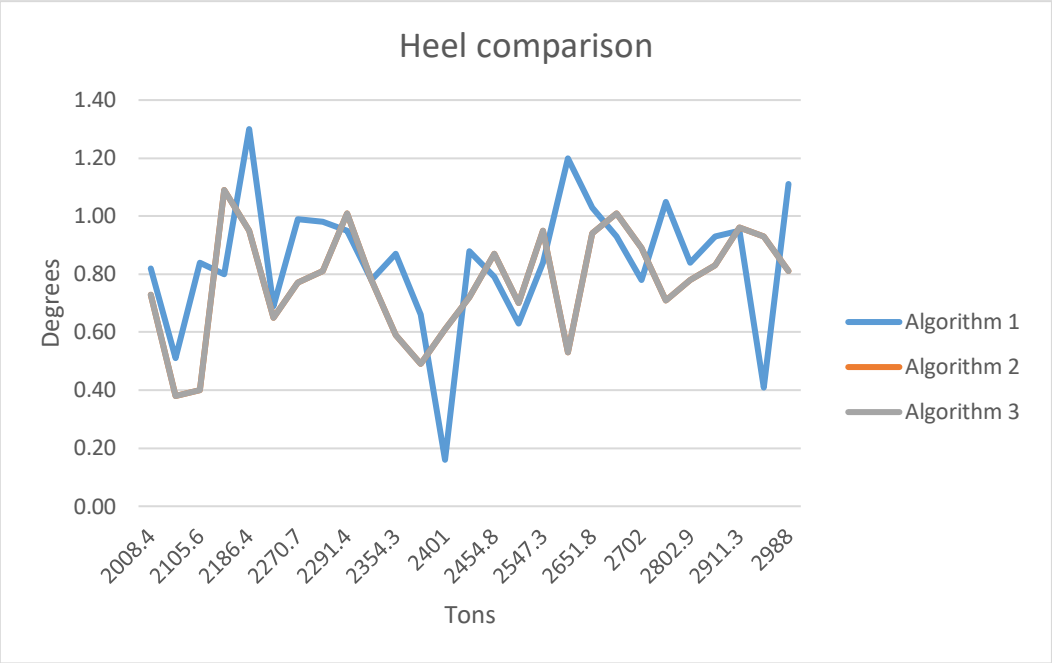

Figure 32. Heel comparison for loads between 2000 and 3000 tons

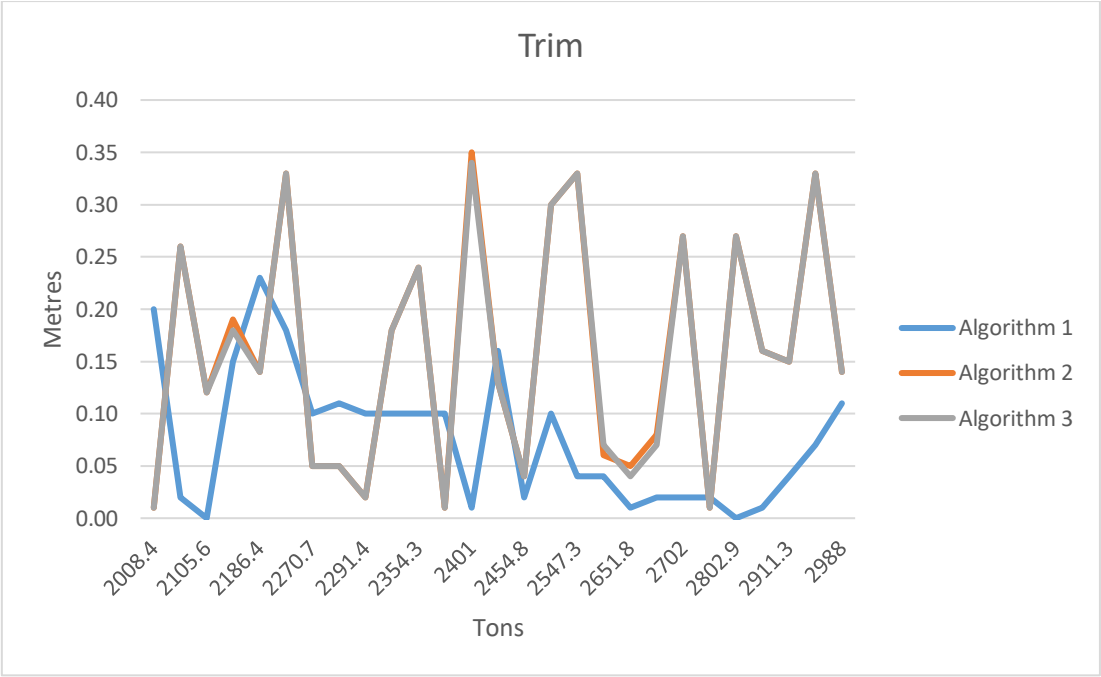

Figure 33. Trim comparison for loads between 2000 and 3000 tons

Figures 32 and 33 contain the heel and trim comparison data for loads between 2000 and 3000 tons and confirm the proposal that the output results will not differ significantly.

Average heel for algorithm 1 is 0.84 degree against average heel 0.77 degree for algorithm 2 and 3.

Generally, algorithm 1 produces better results for trim again, because of its strategy of placing vehicles on all decks. Algorithms 2 and 3 use the schema of loading decks after deck going up, that is why location of vehicles can differ, but as trim does not exceed level of 0.35 meter this is a good result.

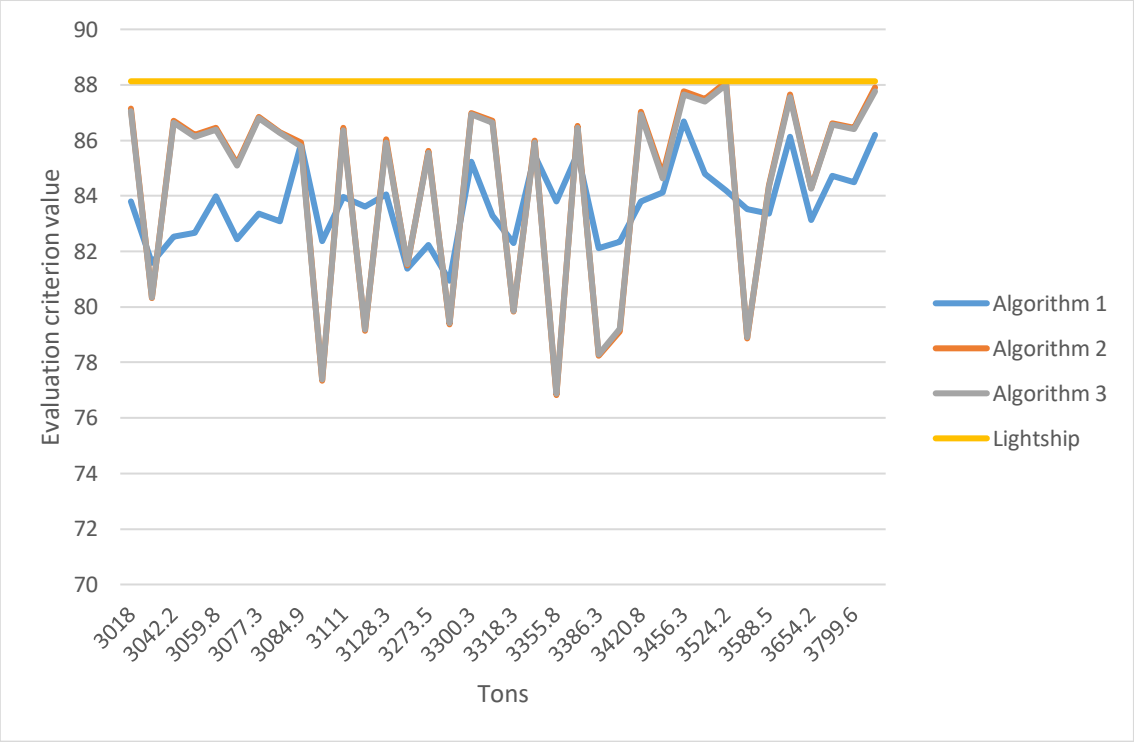

Figure 34. Algorithm comparison with loads between 3000 and 4000 tons

Figure 34 shows the comparison of evaluation criterion for loads between 3000 and 4000 tons. They represent 36 cases or 7,2 % of all cases.

Chart suggests, that algorithm 1 is more stable at least with one of the parameters, and algorithms 2 and 3 results indicate that most likely two parameters will vary.

Average X value coordinate for this tonnage for algorithm 1 is 83.69 versus algorithm 2 and 3, coordinate X is 84.38 and 84.34 respectively. Average Y coordinate is still better at -0.73 for algorithm 1 and average Y coordinates for algorithms 2 and 3 respectively -0.63 and -0.678.

Heel and trim comparison for loads between 3000 and 4000 tons are pictured on Figures 35 and 36 respectively.

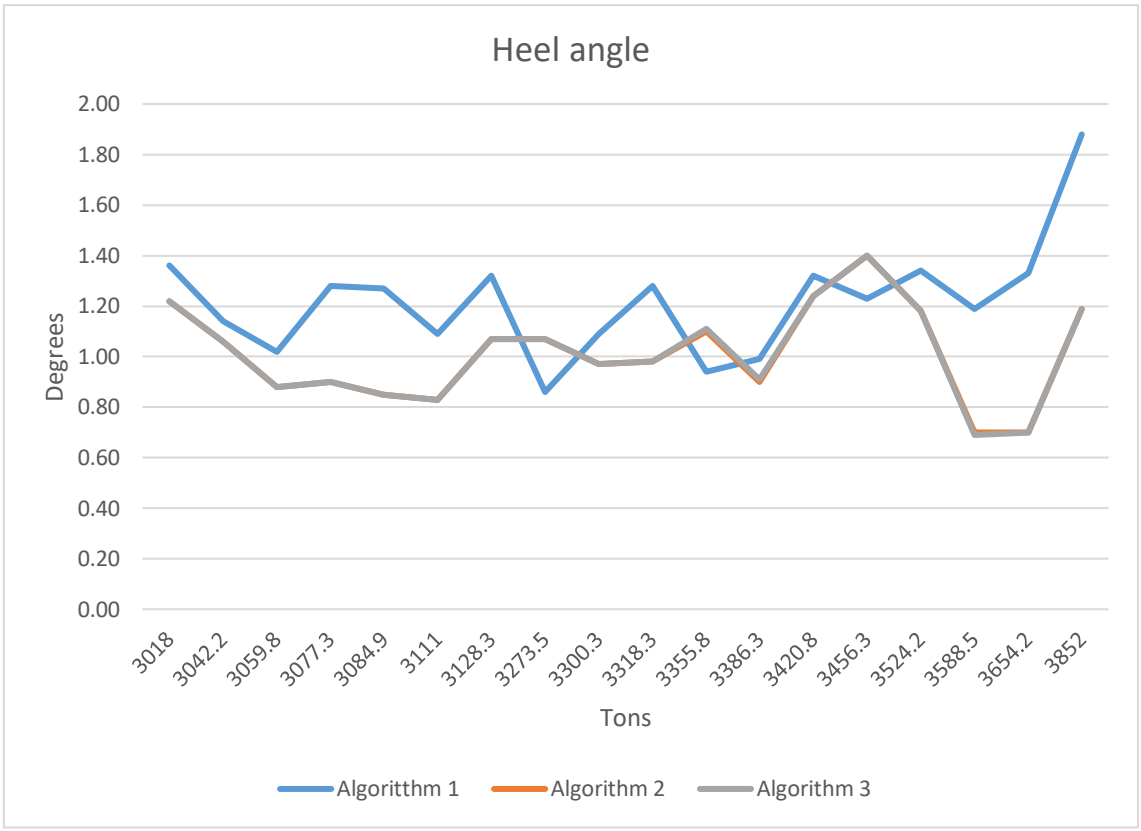

Figure 35. Heel comparison for loads between 3000 and 4000 tons

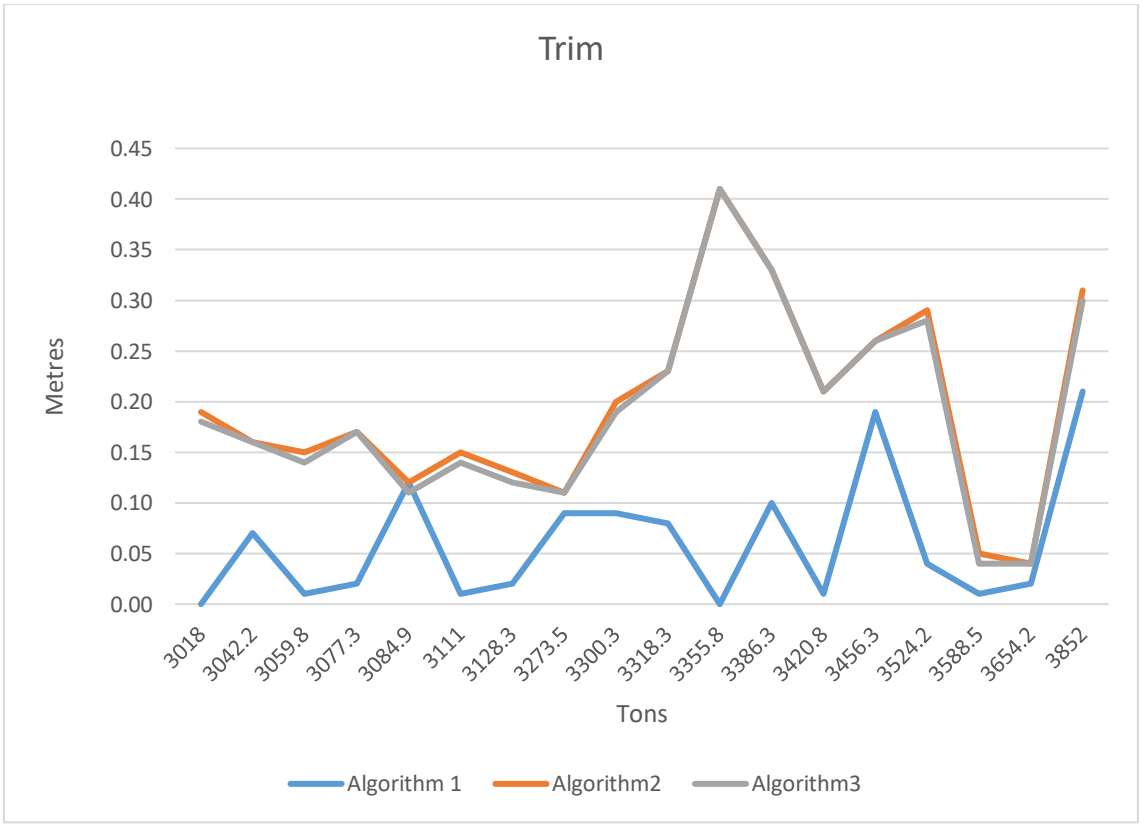

Figure 36. Trim comparison for loads between 3000 and 4000 tons

Figures 35 and 36 indicate that the proposal about algorithm 1 was correct. Trim is just within limit of 0.22 metres, but heel is even higher than algorithm 2 and 3 shows. This is because with such heavy loads all decks are loaded and the strategy to keep heavy weight closer to the centre of the ship wins over strategy to balance weight on ship sides. Also, algorithm 3 finally makes a small difference against algorithm 2.

Peak trim of 0.41 metre happened because some light Tallink trailers were put to the bow, and their weight was too small to decrease the trim.

There's only load for cargo over 4000 tons and it is exactly 4076 tons and algorithms 2 and 3 outperform algorithm 1 with X coordinate 85.78 and 85.72 versus 84.36 and Y coordinates for algorithms 2 and 3 -0.59 comparing -0.81 with algorithm 1 used. Algorithm 1 heel is 1.65 with trim 0.08, algorithm 2 with heel 1.13 and trim as 0.18 and algorithm 3 with same heel as 1.13 and trim 0.17. Again algorithm 3 makes a small difference against algorithm 2 with slightly better trim.

In overall it indicates that algorithms 2 and 3 are recommended for use starting from 0 to 500 tons and for loads of 3000 tons and more. For loads till from 500 to 3000 tons, user can select between algorithms 1, 2 or 3 whichever suggest better results for a user.

All algorithms provide ship trim and heel algorithms within allowed limits.

4.2.2 Analysis of truck cargo

In this section author analyses the dependability of percentage of the heavy 25- and 39-ton trucks in the loads on the heel and trim of the ferry. In case of loads over 3000 tons, the biggest part of the load are heavy trucks. Figures 37 and 38 show how heavy cargo influences the ship heel and trim.

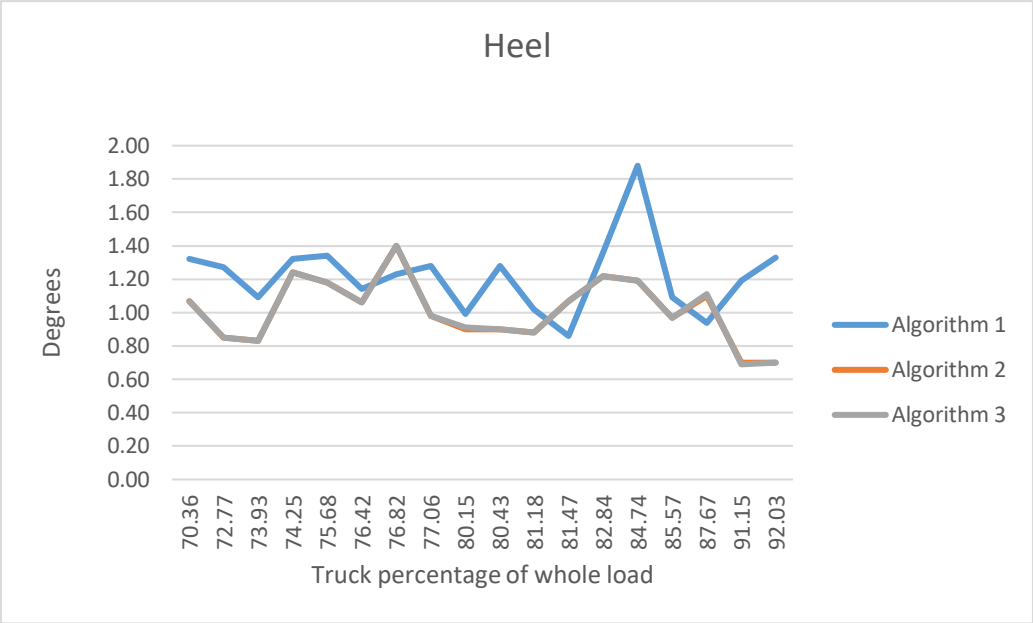

Figure 37. Heel comparison to truck percentage

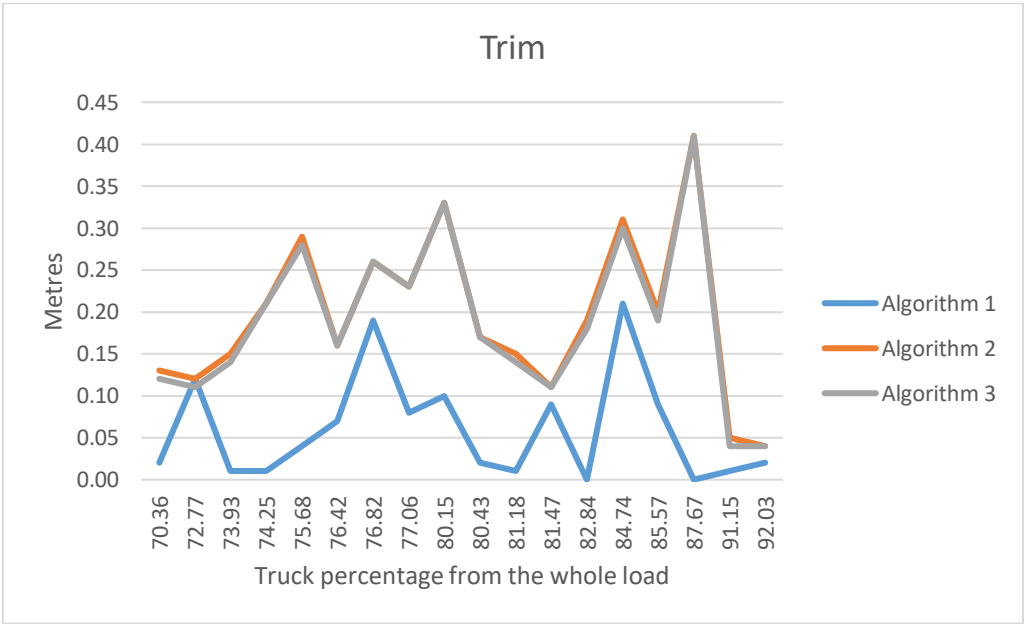

Figure 38. Trim comparison for truck percentage

Algorithms 2 and 3 suggest that the heavy load is relatively evenly distributed around the value of 1 degree of the ship heel and this heel is usually less than ship have if algorithm 1 is used. Again algorithms 2 and 3 almost overlap each other.

The low trim value, as it has been learnt previously, is great advantage of algorithm 1 and it confirms that it is still kept low even with higher loads thanks to proper placement

of load along X axis. During the testing there was no indication that trim will significantly increase with the load increase in algorithm 1 implementation.

4.2.3 Critical cases study

In this section author analyses the maximal difference of output coordinates of the cargo gravity point and the coordinates of gravity point of the empty ship. This is done to make sure that even big difference of coordinate X from the empty ship centre of gravity point still give the result for trim and heel values to stay inside allowed boundaries.

Tables 12 and 13 represent maximal and minimal X coordinates in relation to the ship heel and trim for algorithm 1. Respectively tables 14 and 15 represent maximal and minimal X coordinates in relation to ship heel and trim for algorithms 2 and 3.

Table 12. Minimal X coordinates for algorithm 1

Minimal X coordinate value[m]	**Weight [10^3 kg]**	**Heel [°]**	**Trim [m]**
67.57	514.7	0.00	0.19
73.32	479.2	0.05	0.12
73.75	473.9	0.00	0.12

Table 13. Maximal X coordinates for algorithm 1

Maximal X coordinate value[m]	**Weight [10^3 kg]**	**Heel [°]**	**Trim [m]**
109.87	377.4	0.09	0.19
105.39	289.3	0.1	0.12
104.62	671.9	0.05	0.27

Table 14. Minimal X coordinates for algorithms 2 and 3

Minimal X coordinate value[m]	**Weight [10^3 kg]**	**Heel [°]**	**Trim [m]**
71.70	339.9	0,04	0.10
71.74	252.1	0.02	0.08
72.1	473.9	0.18	0.13

Table 15. Maximal X coordinates for algorithms 2 and 3

Maximal X coordinate value[m]	Weight [10^3 kg]	Heel[°]	Trim [m]
100.82	161.9	0.05	0.05
100.49	136.9	0.06	0.04
99.67	169.9	0.00	0.04

All cases indicate that even X coordinate of the cargo does not lie close to the X coordinate of the empty ship, so the ship is stable.

All output Y coordinates might not be taken in consideration as in all tests heel never reached critical levels of 2 degrees.

4.2.4 Graphical representation in Tallink software

The results of the program are put into the LD file, which in turn should be loaded into Tallink software. This is an example how end users see the vehicle placement plan.

Figure 39 represent the vehicle placement plan for decks 3 and 5.

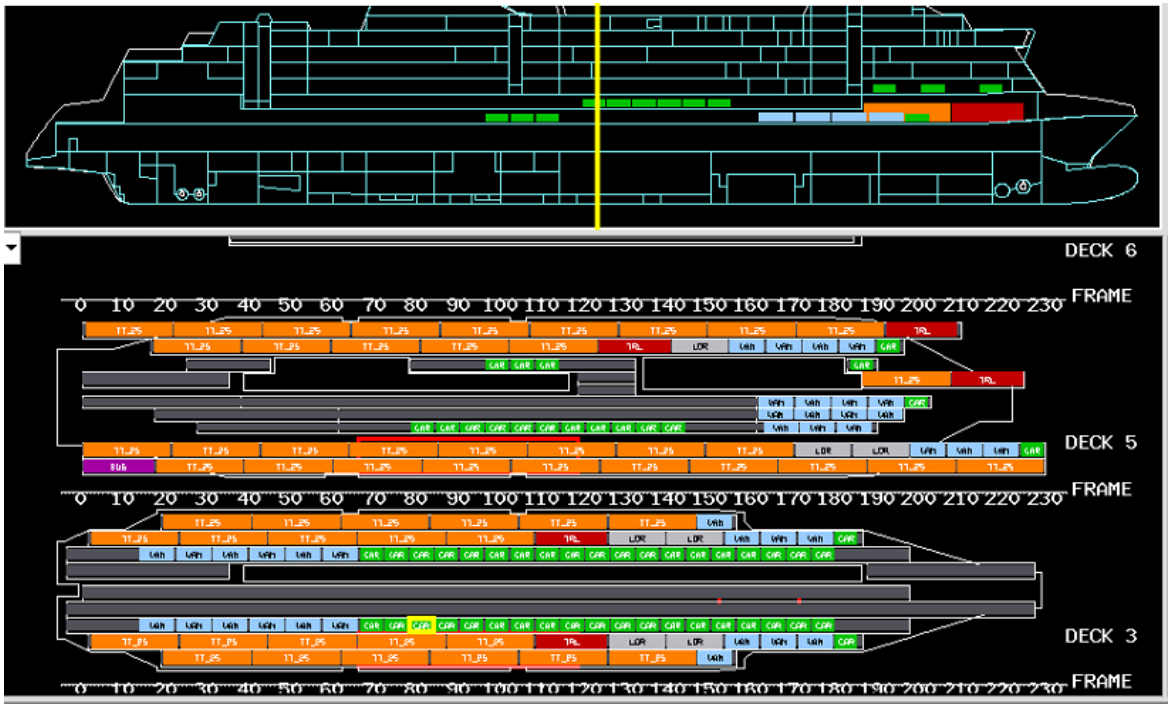

Figure 39. Visual representation of 3 and 5 deck vehicles placement

Figure 40 represent a vehicle placement plan for decks 6 and 7 along with ship heel and trim information.

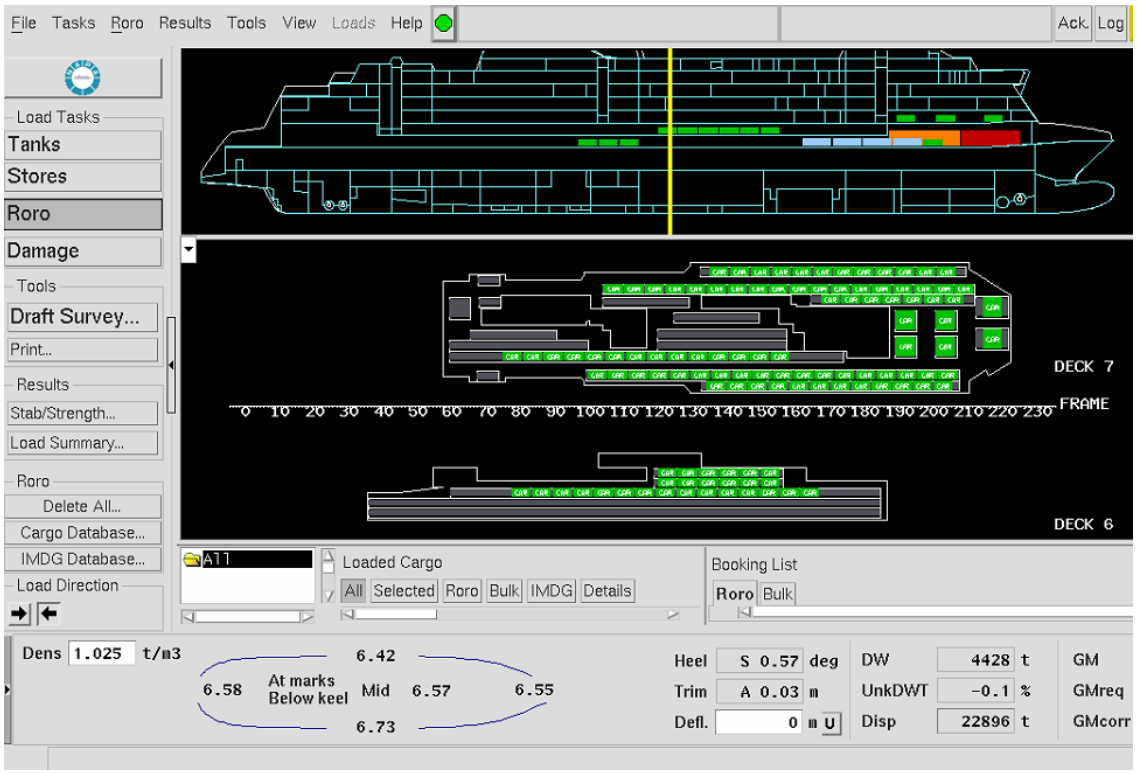

Figure 40. Visual representation of 6 and 7 deck vehicles placement and ship stability information

5 Summary

The goal of the thesis was to create a tool for the Tallink bosuns of MS “Megastar” for automation of vehicle placement onboard the ship on Tallinn - Helsinki line.

Author reviewed multiple options for a vehicle placement plans and as result modified 1D bin packing algorithms has been implemented. Three algorithms were implemented, tested and analysed with real operational data provided by the Tallink.

Solution was developed using the Java language and the outcome is a Java application with simple user interface, which can run on multiple platforms. Application automatically invokes a special converter, so application results are used for creation of cargo plans compatible with Tallink existing software.

For all tested cases Tallink software confirmed that generated cargo plans meet ship stability requirements, thus the objectives have been achieved and application can be used for real ship loading.

References

[1] V. T. (City of H. Jaakola Ari, Oksanen-Sarela Katja, Berglund Petri and P. K. (City of T. Kuulpak Peeter, *Helsinki - Tallinn, Facts and figures 2014*. 2014.

[2] "Passenger and goods transport through ports increased last year - Statistics Estonia," 2020. https://www.stat.ee/news-release-2020-036 (accessed Apr. 01, 2020).

[3] "Corporate factsheet 2017." https://www.tallink.com/documents/12397/79137327/Tallink-Grupp-Investors-Company-Factsheet-2017-03.pdf (accessed Apr. 27, 2020).

[4] "Megastar generates largest passenger numbers on Baltic Sea in 6 months | Economy | ERR." https://news.err.ee/610186/megastar-generates-largest-passenger-numbers-on-baltic-sea-in-6-months (accessed Apr. 27, 2020).

[5] "Tallink Shuttle Megastar - Tallink & Silja Line." https://www.tallinksilja.com/tallink-shuttle-megastar-helsinki-tallinn-helsinki#tabs-content-9 (accessed Apr. 01, 2020).

[6] "MS Megastar deck plan," Accessed: Apr. 01, 2020. [Online]. Available: https://www.tallinksilja.com/documents/10192/122897942/Megastar+deck+plan.pdf/e06bef49-552a-a5c5-738a-2ad7c3fb80bd.

[7] "Focus on IMO IMO and ro-ro safety," 1997.

[8] A. Biran, *Ship Hydrostatics and Stability*, vol. 363, no. 9. 2003.

[9] B. Barrass and C. R. Derrett, *Ship Stability for Masters and Mates*. 2006.

[10] Y. Kirillova and Y. Meleshenko, "Development of an economic and mathematical model of loading a freight and passenger ferry," *Eastern-European J. Enterp. Technol.*, vol. 3, no. 4–81, pp. 28–37, 2016, doi: 10.15587/1729-4061.2016.71215.

[11] B. O. Øvstebø, L. M. Hvattum, and K. Fagerholt, “Optimization of stowage plans for RoRo ships,” *Comput. Oper. Res.*, vol. 38, no. 10, pp. 1425–1434, 2011, doi: 10.1016/j.cor.2011.01.004.

[12] J. D. Garey MR, *Computers and Intractability - A Guide to the Theory of NP-Completeness*. San Francisco: WH Freeman and Company, 1979.

[13] J. R. Hansen, I. Hukkelberg, K. Fagerholt, M. Stålhane, and J. G. Rakke, “2D-packing with an application to stowage in Roll-on Roll-off liner shipping,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9855 LNCS, pp. 35–49, 2016, doi: 10.1007/978-3-319-44896-1_3.

[14] R. Puisa, “Optimal stowage on Ro-Ro decks for efficiency and safety,” *J. Mar. Eng. Technol.*, pp. 1–17, Sep. 2018, doi: 10.1080/20464177.2018.1516942.

[15] E. Wathne and J. Rakke, “Cargo Stowage Planning in RoRo Shipping Optimisation Based Naval Architecture Marine Technology (2 year).”

[16] R. De Almeida and M. T. A. Steiner, “Resolution of 1-D Bin Packing Problem using Augmented Neural Networks and Minimum Bin Slack,” *2015 Latin-America Congr. Comput. Intell. LA-CCI 2015*, pp. 1–6, 2016, doi: 10.1109/LA-CCI.2015.7435943.

[17] A. Layeb and S. Chenche, “A Novel GRASP Algorithm for Solving the Bin Packing Problem,” *Int. J. Inf. Eng. Electron. Bus.*, vol. 4, no. 2, pp. 8–14, 2012, doi: 10.5815/ijieeb.2012.02.02.

[18] Falkenauer E., *Genetic Algorithms and grouping problems*. New York: Wiley, 1998.

[19] D. S. Johnson, A. Demers, J. D. Ullman, M. R. Garey, and R. L. Graham, “Worst-Case Performance Bounds for Simple One-Dimensional Packing Algorithms,” *SIAM J. Comput.*, vol. 3, no. 4, pp. 299–325, 1974, doi: 10.1137/0203025.

[20] “TIOBE - The Software Quality Company.” https://www.tiobe.com/tiobe-index/

(accessed Apr. 07, 2020).

[21] K. S. and E. R. E. Freeman, B. Bates, *Head First Design Patterns*. O'Reilly Media, 2014.

Appendix 1 – Code location

The code of the project has been published at the following locations with public access:

Algorithm 1 - https://gitlab.com/anerma_ttu/roro.git

Algorithm 2 - https://gitlab.com/anerma_ttu/roro2.git

Algorithm 3 - https://gitlab.com/anerma_ttu/roro3.git

Binary - https://drive.google.com/open?id=11yq1FzJnB-JfHt1_wc15WySpuSAOKZx6

Appendix 2 – Lanes data

Table 16. Lanes data

DECK	LANE	XMIN	XMAX	XCG	YCG	Length	X_DIFF	Y_DIFF
3. DECK	L11	15	125	70.00	12.95	110	18.13	-12.88
3. DECK	L12	1.00	149.00	75.00	9.85	148	13.13	-9.78
3. DECK	L13	-3.00	158.00	77.50	6.75	161	10.63	-6.68
3. DECK	L14A	-3.00	28.00	12.50	3.65	31	75.63	-3.58
3. DECK	L14F	150	182	166.00	3.65	32	-77.87	-3.58
3. DECK	L15	0.00	158.00	79.00	-0.55	158	9.13	0.62
3. DECK	L16	-3.00	182.00	89.50	-3.65	185	-1.37	3.72
3. DECK	L17	-3.00	158.00	77.50	-6.75	161	10.63	6.82
3. DECK	L18	1	149	75.00	-9.85	148	13.13	9.92
3. DECK	L19	15.00	125.00	70.00	-12.95	110	18.13	13.02
5. DECK	L21	0.00	168.00	84.00	12.85	168	4.13	-12.78
5. DECK	L22	13.00	157.00	85.00	9.75	144	3.13	-9.68
5. DECK	L23A	20	36	28.00	6.20	16	60.13	-6.13
5. DECK	L23M	62.60	105.60	84.10	6.20	43	4.03	-6.13
5. DECK	L23F	146.00	152.00	149.00	6.20	6	-60.87	-6.13
5. DECK	L24A	0.00	28.00	14.00	3.45	28	74.13	-3.38
5. DECK	L24F	149	180	164.50	3.45	31	-76.37	-3.38
5. DECK	L24MP	94.60	105.60	100.10	3.80	11	-11.97	-3.73
5. DECK	L24MS	94.60	105.60	100.10	1.40	11	-11.97	-1.33
5. DECK	L25A	0.00	32.50	16.25	-0.90	32.5	71.88	0.97
5. DECK	L25F	129	162	145.50	-0.90	33	-57.37	0.97
5. DECK	L25M	32.50	129.00	80.75	-0.90	96.5	7.38	0.97
5. DECK	L26A	14.00	49.00	31.50	-3.30	35	56.63	3.37
5. DECK	L26F	129.00	157.00	143.00	-3.30	28	-54.87	3.37
5. DECK	L26M	49	129	89.00	-3.30	80	-0.87	3.37
5. DECK	L27A	22.00	49.00	35.50	-5.70	27	52.63	5.77
5. DECK	L27F	129.00	152.00	140.50	-5.70	23	-52.37	5.77
5. DECK	L27M	49.00	129.00	89.00	-5.70	80	-0.87	5.77

5. DECK	L28	0	184	92.00	-10.05	184	-3.87	10.12
5. DECK	L29	0.00	184.00	92.00	-13.15	184	-3.87	13.22
6. DECK	L31	94.80	124.80	109.80	3.25	30	-21.67	-3.18
6. DECK	L32	94.8	124.8	109.80	0.95	30	-21.67	-0.88
6. DECK	L33	47.40	147.40	97.40	-1.35	100	-9.27	1.42
6. DECK	L34	28.70	147.40	88.05	-3.65	118.7	0.08	3.72
6. DECK	L35	28.70	147.40	88.05	-5.95	118.7	0.08	6.02
7.DECK	L41	105.30	167.30	136.30	13.20	62	-48.17	-13.13
7.DECK	L42	82.80	169.30	126.05	9.15	86.5	-37.92	-9.08
7.DECK	L42A	53.70	58.60	56.15	11.20	4.9	31.98	-11.13
7.DECK	L43A	54.2	59	56.60	6.10	4.8	31.53	-6.03
7.DECK	L43M	83.00	102.70	92.85	6.10	19.7	-4.72	-6.03
7.DECK	L43F1	131.00	169.00	150.00	6.65	38	-61.87	-6.58
7.DECK	L43F2	170.00	177.00	173.50	4.90	7	-85.37	-4.83
7.DECK	L44A	47.2	52	49.60	4.70	4.8	38.53	-4.63
7.DECK	L44M	99.00	119.20	109.10	2.50	20.2	-20.97	-2.43
7.DECK	L44F1	150.70	155.70	153.20	1.80	5	-65.07	-1.73
7.DECK	L44F2	160.00	165.00	162.50	1.80	5	-74.37	-1.73
7.DECK	L45A	52	72	62.00	-1.55	20	26.13	1.62
7.DECK	L45M	95.40	125.40	110.40	-1.45	30	-22.27	1.52
7.DECK	L46A	47.00	79.50	63.25	-4.05	32.5	24.88	4.12
7.DECK	L46M	95.40	125.40	110.40	-3.95	30	-22.27	4.02
7.DECK	L46F1	150.7	155.7	153.20	-4.20	5	-65.07	4.27
7.DECK	L46F2	160.00	165.00	162.50	-4.20	5	-74.37	4.27
7.DECK	L46F3	170.00	177.00	173.50	-2.50	7	-85.37	2.57
7.DECK	L47A	47.00	138.80	92.90	-6.55	91.8	-4.77	6.62
7.DECK	L48A	53.7	58.6	56.15	-11.05	4.9	31.98	11.12
7.DECK	L48	79.00	165.60	122.30	-10.75	86.6	-34.17	10.82
7.DECK	L49	105.60	165.60	135.60	-13.25	60	-47.47	13.32