Project Optimization of Business Processes

Case: Containers

Final Report

Jurriaan Besenbruch 2546851
Georgios Christos Chouliaras, 2592496
Anouar Darrazi, 2599226
Derek van den Elsen, 2580100
Jan-Willem Feilzer, 2596978
Leidy Esperanza Molina, 2574864
Veronika Zhezhela, 2588505

Supervisor: Bernard Zweers

February 3, 2017

Contents

Introduction

Containerization is one of the main sources of world trade growth. The size of container ships and the number of containers transported is still growing. This growth puts a larger pressure on deep sea terminals that should accommodate sea vessels appropriately. Transportation to the hinterland also demands consideration as the thousands of containers that are usually shipped via sea vessels cannot all be transported efficiently via trucks alone.

In this paper a logistic provider that is responsible for transporting containers from several multiple deep sea terminals in Rotterdam to one inland terminal is considered. The paper presents a decision support system (DSS) that assists the logistic provider by deciding for each container on which day and via which mode of transportation it is shipped.

It is preferable to ship via barge, which takes one day, as this is cheaper than shipping via trucks. However trucks are faster, arrive at the inland terminal the same date as the departure of the deep sea terminal and are assumed to have infinite capacity in the scope of this paper, whereas barges have limited capacity in maximum TEU and maximum weight. Trucks can only carry a single container regardless of size. Containers can stay at a terminal and be subject to demurrage costs. In addition, each container has a demurrage free period, in which no demurrage costs are paid. Containers have a call date and an estimated time of arrival at the deep sea terminal that both have to be respected. Barges should also not visit too many terminals as this might lead to network congestion.

This paper firstly explains the theoretical scientific background of the solution approaches to this problem. A slow and exact solution is provided in the form of an Integer Linear Programming (ILP) formulation and a faster, but less precise, solution is derived using a Heuristic approach. Secondly, the paper will present a user manual that depicts the capabilities of the DSS in an intuitive way. Thirdly, the report will give an in depth documentation stipulating the finer details of the code underlying our decision support system. Fourthly, it will verify the results on several instances, some of which have an expected result to compare, and validate the performance of the tool. Lastly, it will summarize all the work put in to create this DSS and conclude with some points of improvement.

Task	Student
Data Analysis	George, Esperanza
ILP	Derek, George, Anouar, Jan-Willem
Heuristic	Jurriaan, Esperanza
Input Tool	Jan-Willem, Veronika
Output Tool	Esperanza, Veronika, Jan-Willem, Jurriaan
Report	Everyone
Presentation 1	Derek, Jan-Willem
Presentation 2	George, Veronika, Jan-Willem
Presentation 3	Esperanza, Anouar, Jurriaan

Table 1: Separation of Tasks

Theory

This section will elaborate on the scientific explanation behind the two main solution approaches: an Integer Linear Program and a heuristic method.

ILP approach

This method will solve the container scheduling problem in an exact manner, but it does not necessarily run very fast as ILP's are NP-hard [1]. As any ILP it has the components: decision variables, an objective function and constraints to be met.

Decision variables

The ILP decision variables x_{ijk} are defined as follows. The index i runs from 1 to I, where I represents the total amount of containers, j runs from 0 to J, where 0 stands for trucks and J represents the total number of available barges. Lastly k runs from 1 to K, where K represents the number of days. This means that if $x_{ijk} = 1$, container i uses mode of transportation j to travel on day k. Decision variables x_{ijk}, where barge j is not present on day k are not created.

In addition, y_{tjk} is included as a separate decision variable $\quad \forall j, k, t \in T$, where T equals the set of terminals. $y_{tjk} = 1$ if on day k, barge j ships any containers i belonging to terminal t, and zero otherwise. Trucks are assumed to have infinite capacity and can be excluded, hence those decision variables do not have to be created. $y_{tjk} = 1$ can be interpreted as the specific terminal t being 'open' to ship any containers via barge j on day k.

$$\text{Total Number Decision Variables} = I \cdot J \cdot K + size(T) \cdot J \cdot K$$

Objective Function

$$\text{Total Costs} = \text{Shipping Costs} + \text{Demurrage Costs} + \text{Congestion Costs}$$

$$\text{Min} \sum_{i,j,k} x_{ijk} \cdot c_{ijk} + \sum_{t \in T,j,k} y_{tjk} \cdot c_{tjk}$$

where c_{ijk} equals the cost for shipping container i using mode j on day k. c_{tjk} equals the cost for allowing any containers to be shipped via barge j from one specific terminal t on day k. c_{tjk} is modelled in the following way:

$$\text{Congestion costs } = \text{barge counter(Travel costs} + \text{Loading costs)}$$

$$c_{tjk} = \text{barges available on day k}(f \cdot \text{capacity barge j} + a \cdot \text{\#containers at terminal t})$$

where f is a constant multiplied with the TEU capacity of a barge. The intuition here is that a larger barge is slower and requires more fuel, so as a result the travel costs are higher. a is some constant that models the fact that a terminal which has more containers will likely need more time to load those containers onto the barge.

One possible rough lower bound for the objective function is the cost of transporting all containers by barge (something that is typically impossible), as transporting via barge is generally the cheaper option. This will result in the following:

$$\text{Lower Bound} = \text{Cost by barge for one TEU} \cdot \text{total TEU that needs to be shipped}$$

One rough 'upper bound' can be set by sending every container by truck as soon as it arrives. This is not a true upper bound as we could arbitrarily increase demurrage costs by having each container wait until its call date and then send them by truck, but this measure will suffice to bound the problem, as that solution is obviously not very sophisticated.

$$\text{'Upper Bound'} = \text{Cost by truck} \cdot \text{total number of containers that needs to be shipped}$$

Constraints

Each container has to be shipped once via either a barge or a truck on a feasible day. A day k is feasible if container i has its ETA + 1 before this day, as it takes one day for unloading, and its call date after this day k for barges, and after or on this day k for trucks. These 3 requirements are then fulfilled by this single constraint. The first sum demonstrates shipping via any barge and the second via truck. Note that all x_{ijk} that are not included in this constraint are by default zero, as the objective is minimized, since having any x_{ijk} be one, when it is not constrained to be, only adds to the objective.

$$\sum_{j=1}^{J} \sum_{k=\text{ETA}_i+1}^{\text{Call date}_i-1} x_{ijk} + \sum_{k=\text{ETA}_i+1}^{\text{Call date}_i} x_{i0k} = 1 \quad \forall i$$

Each barge j has certain weight and TEU capacity constraints:

$$\sum_{i=1}^{I} x_{ijk} \cdot \text{weight}_i \leq \text{weight limit}_j \quad \forall j, k$$

$$\sum_{i=1}^{I} x_{ijk} \cdot \text{TEU}_i \leq \text{TEU limit}_j \quad \forall j, k$$

Now, the x_{ijk} are bounded by the y_{tjk}. If the latter equals zero the former has to equal zero as well, effectively ensuring that any container can only leave a terminal via barge if the terminal is 'open'.

$$x_{ijk} \leq y_{tjk} \quad \forall j, k, i \text{ belonging to } t \in T$$

Integer constraints:

$$x_{ijk} \in \{0, 1\} \quad \forall i, j, k$$

$$y_{tjk} \in \{0, 1\} \quad \forall t, j, k$$

$$\text{Total number of constraints} = I + 2 \cdot J \cdot K + I \cdot J \cdot K + \text{Total Number Decision Variables}$$

Heuristic Method

As an alternative to the slow planning based on the ILP algorithm, a Heuristic method is created in order to construct a container shipment planning. The problem consists of several important factors, namely: TEU, Weight, Demurrage-free period, ETA and Call date. It is possible to select the containers based on the given factors. For example, it is preferred to ship containers with less weight and smaller size (lower TEU) by barge instead of trucks, since they do not fill up the barge as rapidly as large containers. In addition, a small container is relatively more expensive to ship by truck than a large container.

The developed Heuristic consists of two phases. The first phase is the actual developed heuristic algorithm and the second phase is the terminal optimizer. This optimizer is added to find a balance between the visited terminals and the number of containers that are delivered to one terminal. This was an additional requirement of the case. When a barge has to visit too many terminals in one day, it is not possible to deliver the containers on time.

The Heuristic starts with transforming the data. In the container information data, the demurrage free period is given for each container. As the Heuristic is heavily based on sorting the data in a certain order, it is useful to convert this information. The demurrage free period is not useful in this way, so it is converted into a deadline like the call day. As the demurrage period starts from the arrival day, the new demurrage period is defined as *arrival day + demurrage free period*. Now that it is converted, it could be used to sort on like call day.

An iterative approach is used for the Heuristic, which means that it works per day. A planning is created for each day. After all interactions, all days are combined to one schedule for the entire period. At day 1 (assuming that day 1 is the first day of the planning), all the containers that arrived before day 1 are loaded into the algorithm. On the next days (2, ..., K) only the containers that arrived at day k-1 (yesterday) are loaded and added to the containers that are already loaded, but unscheduled yet. When the container information is read, the containers are sorted in the following order: demurrage period, call day and container size (TEU). All factors are sorted in ascending order. As we want to minimize the total costs (transport and demurrage costs), it is necessary to ship the containers before their demurrage free period ends. In addition, sorting on container size is necessary, as it is better to ship small containers by barge from a cost perspective. It is relatively more expensive to ship a small container by a truck.

After the sorting process, all containers are assigned to a barge or truck. It works with a top-bottom principle. The containers that are on top of the job list are scheduled first. When there are barges available and they have capacity (TEU and Weight) left, the container is assigned to the specific barge. This process repeats till the barges are filled up completely. When the barges are full, or not available on a certain day, the heuristic only takes containers that need to be shipped on that day into consideration. In other words, only containers with the current day as call day will be shipped by truck then. The described process repeats for every day a planning should be created.

The second phase of the fast approach is the terminal optimizer. Again, this algorithm uses a heuristic approach and does not result into an optimal solution. It uses an iterative day-by-day approach. It checks everyday for the availability of two or more barges. It will take into consideration all the containers loaded on these barges and the containers scheduled to a truck are not taken into account by the algorithm. When there is just one barge available, the optimizer will not optimize anything.

The optimizer calculates the frequency of the number of times a terminal exists in the planning for that day. Then it sorts the containers on terminal frequency, TEU and Weight. All of these three parameters are sorted in descending order, as it is useful to assign the biggest containers to a barge first, and use the smallest to fill up the barges at the end. The algorithm tries to assign a whole block of containers with the same original terminal to a barge. If that is not possible (anymore), the containers are assigned one-by-one to the available barges. The final result is a balance between the visited terminals and the number of containers that should be delivered.

Based on the flow chart from figure 1, it is possible to summarize the heuristic approach in a short way. The heuristic planning is obtained as follows:

1. At day 1, take all unscheduled containers that are already arrived before day 1. At day 2 until 10, take containers that arrived one day before the planning day and add them to the job list.
2. Sort the current list with available containers on demurrage period, call date and TEU (ascending order).
3. Schedule the first containers with the shortest demurrage period left. In other words: the container that is on top of the list. When a container cannot be transported, it is placed at the end of the job list.
4. Try to assign containers to a barge:
   (a) If all available barges are full, but there are still containers left that need shipping, transport these containers by truck.
   (b) If all available barges are not full:
      i. Are there still some containers left that can be shipped today? If so, assign them to the barge until the barge is full.
      ii. Are there no more containers left? Do nothing, planning is finished for today.

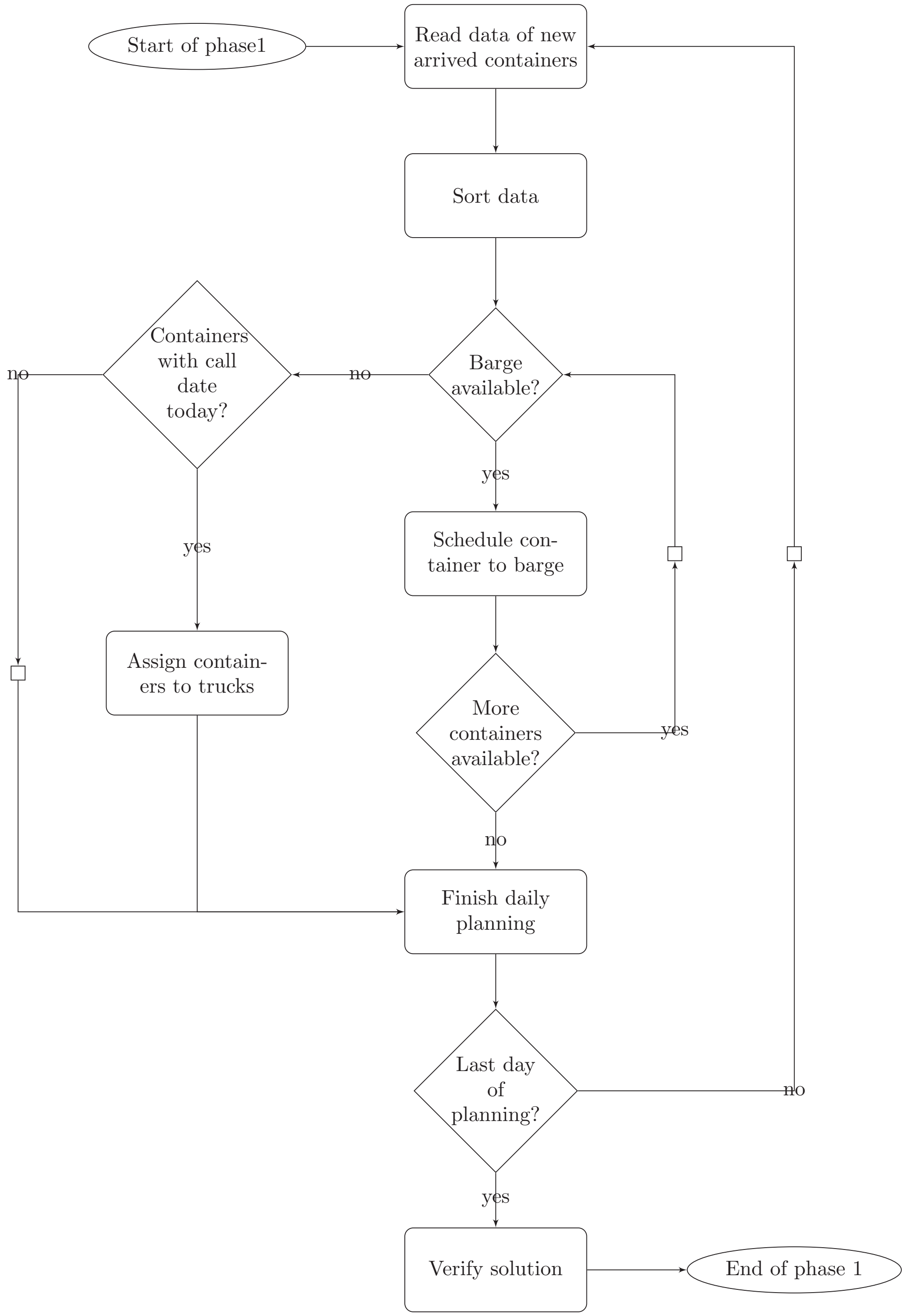


Figure 1: Flowchart of heuristic algorithm for fast planning

5. Output planning for today and remove scheduled containers. Hold unscheduled containers in job list for tomorrow.
6. Go to next day and step back to Step 1, or finish planning (on day 10).
7. Verify the solution and optimize the visited terminals by one barge.

The Heuristic as it is implemented now, gives a relatively fast solution within a 5 percent deviation of the optimal ILP solution. Besides that, there are several options to optimize the heuristic further. These optimizations are partly feature, solution and calculation time improvements.

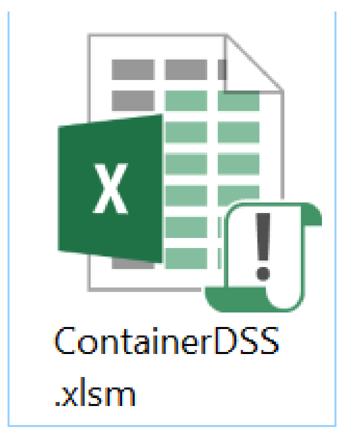
For example some functionality that is used in the slow ILP algorithm is not yet implemented in the heuristic. So the planning can be extended to work with more than ten planning days. Also terminal adjustment is not possible yet. Besides the feature improvements, the solution itself can also be improved. By calculating the transportation costs for a container when it is transported on the next day. When the call day is on the next day, the container should be transported by truck, which means a huge increase in costs for that container. To improve the solution there can be an additional variable added, where the difference between the cheapest solution of shipping tomorrow and today is calculated. When sorting this in descending order this can give some further costs reductions.

In terms of computation speed, there is a lot of overhead and unnecessary things that are calculated or read multiple times. If these parts are efficiently rewritten, there could be a enormous reduction in computation times. The heuristic algorithm itself runs in two seconds, but the overall fast approach takes up to fifteen seconds to run. Unless there are lots of improvements possible, the heuristic approach is a good quick alternative to the slower ILP algorithm. It stays close to the ILP in terms of costs and runs much faster. So it is a good alternative for last minute changes and quick cost overviews.

User Manual

This guide will walk you through the basics of using the application, including a section with downloading the additional software(OpenSolver), explanation on what files are needed and will save your time working with the user interface. As a result you will be able to create a schedule with given information, find information about any specific container in seconds, check output on effectiveness, make adjustments and generate a report. If the program displays any error, find the solution here.

Figure 2: Icon used for the DSS



First things first, check if your computer has an OpenSolver. If not, section 'How do I install and use OpenSolver?' has the information necessary concerning the installation process. Otherwise, you can already start using the tool.

To have an overview of the program, workflow of all documents can be classified as input or output of the application. Input workflow documents are files that are uploaded to the tool from your computer. Output is generated after the schedule has been created. You have an option to export the information and save it as a new file.

How do I install and use OpenSolver?

1. Go to http://opensolver.org/installing-opensolver/ and follow the instructions on the page to install OpenSolver [2].
2. Open up Excel and open VBA by pressing ALT+f11 for the Windows version. For the Mac version, go to Developer and then select Editor.
3. Select the corresponding VBA project.
4. Press tools in the header and then select references.
5. If there is an unticked box that says OpenSolver missing, click browse and select the OpenSolver.xlam file, wherever you saved it and open it.
6. If there is also a missing Solver, deselect this one and tick the Solver from the list.
7. Now the OpenSolver box should be ticked and should work appropriately and is visible under the data header in Excel.
8. For any problems please refer to this site: http://opensolver.org or contact the developers at andrew@opensolver.org, jack@opensolver.org or a.mason@auckland.ac.nz.

Windows and MacOS compatibility

The container DSS is compatible with both Windows and MacOS operating systems. The reason that there are different versions of the tool for each operating system is due to some minor issues that do not allow the Windows version to run in MacOS and vice versa. More specifically, the differences between the two versions are the following:

The VBA function `Application.GetOpenFilename()` is defined in a different way in Windows than it is in MacOS. In Windows, the function is defined as `Application.GetOpenFilename(Finfo, FilterIndex,Title)` where Finfo is a list of file filters (txt, prn, csv, etc.) and these need to be specified in order for this function to work. However, in MacOS this function needs to be specified without Finfo, FilterIndex and Title in order to run properly. This is due to the fact that MacOS reads files in a different way than Windows.

Furthermore, in the Windows version, when the container and barge input files are uploaded to the container DSS, the program closes the two workbooks after they are selected. However, MacOS cannot close or hide workbooks before the userform is closed, hence in the Mac version the worksheets need to remain opened as long as the userform is running.

It needs to be noted that the fast planning does not work on the Mac version, since there is some conflict with the `AutoFilter` function. Therefore, the user manual refers to the Windows version which is the fully functional version.

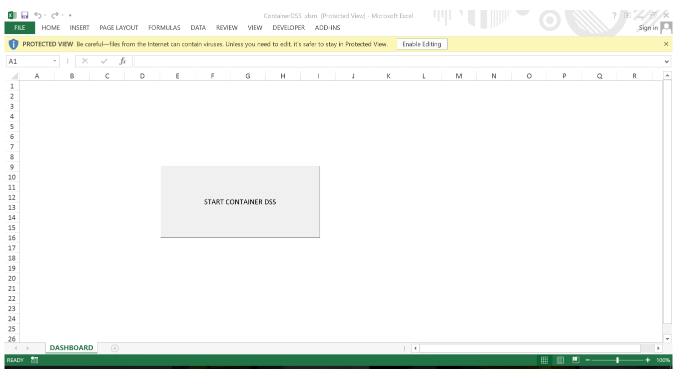
Input Screen

Welcome to the Decision Support System!

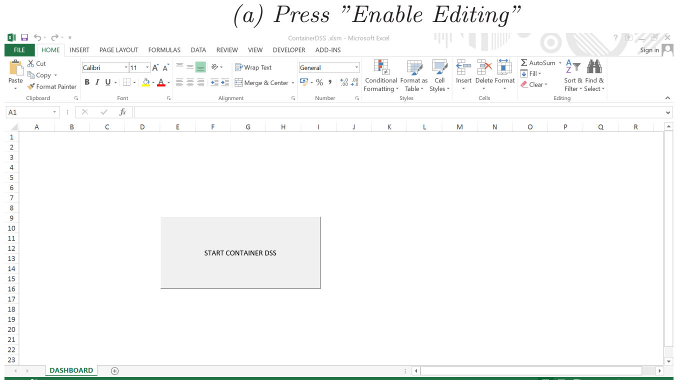
Next step to do is press "Enable Editing" to enable protected view of the content and actually start to work with the system. If something went wrong you can enable the content with the following steps:

1. Click the Microsoft Office Button.
2. Click Excel Options.
3. The Advanced category, under Editing options, select the Allow editing directly in cells check box to turn in-cell editing on.

Please, check if your screen matches Fig 3(b).



(a) Press "Enable Editing"



(b) Upload files

Figure 3: Enable Macro

The last step to begin operating is pressing the "Start Container DSS" button. Once you clicked you are redirected to the input screen Fig 6. The program runs automatically and the flash screen will be displayed as seen in Figure 5.

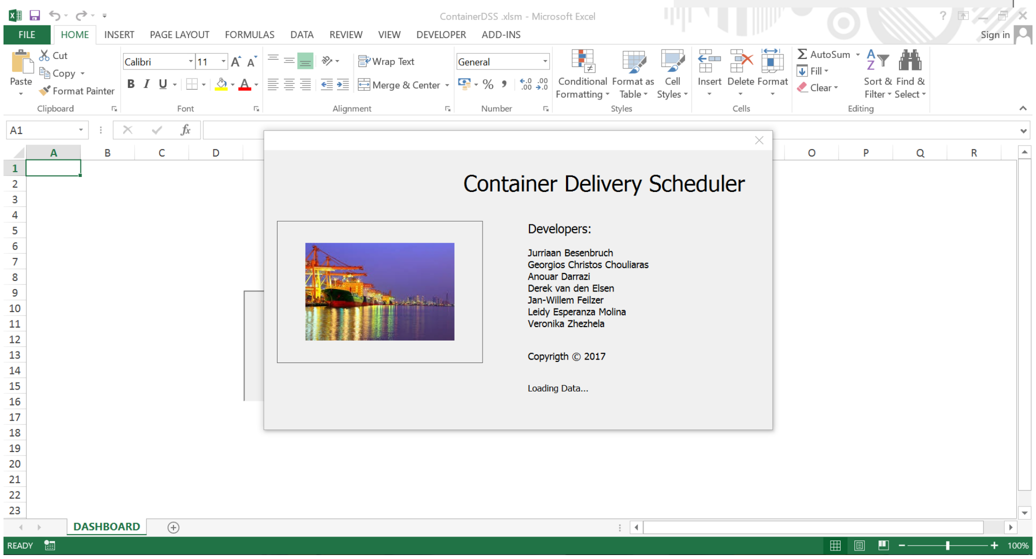


Figure 5: Flash Screen

The screen will be switched to the Input Screen that has the following features:

- ability to choose files on your computer.
- make adjustments to the scheduling and terminal visits.
- check the correctness of the files based on set rules.
- select the preferable type of planning.

The process of selecting and validating the files is shown in Fig 7.

How to start the planning:

1. Upload the file with Container information in the first empty field:
   (a) Click on Browse button.
   (b) Select the file as it is shown on the Fig 7(a).
   (c) Name of the chosen file should appear on the screen.
2. Repeat step one for adding the correct file with barge data under the "Barge Information" label.
3. If the error was shown Fig 7(b) verify the file and choose an appropriate one.

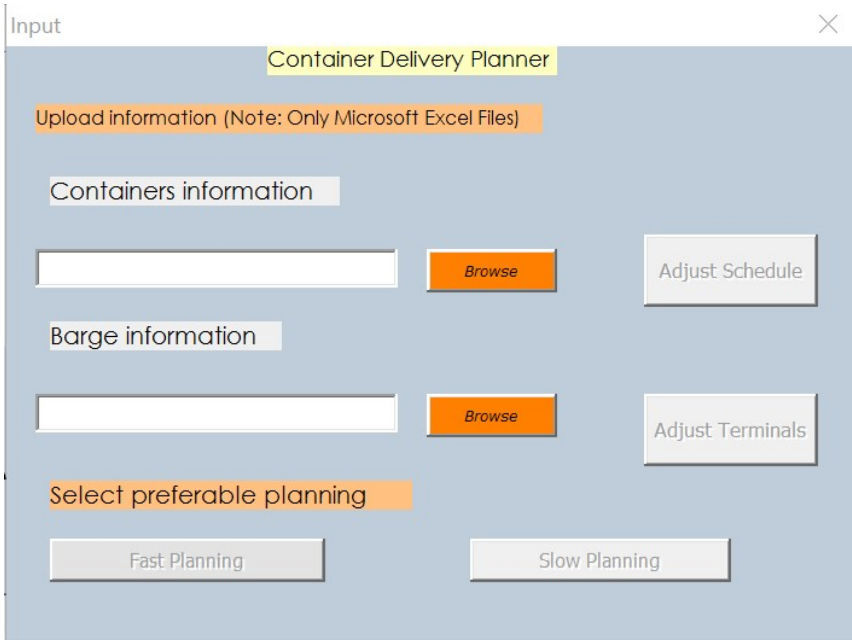
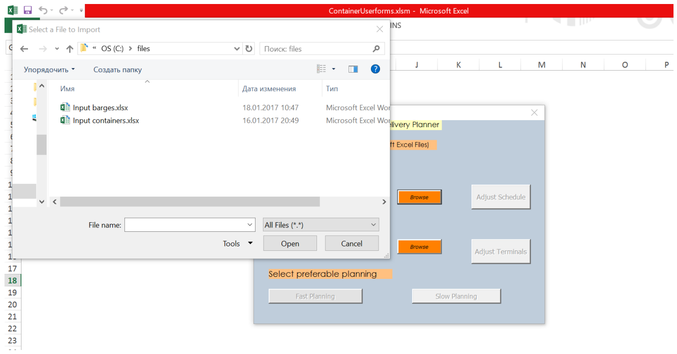


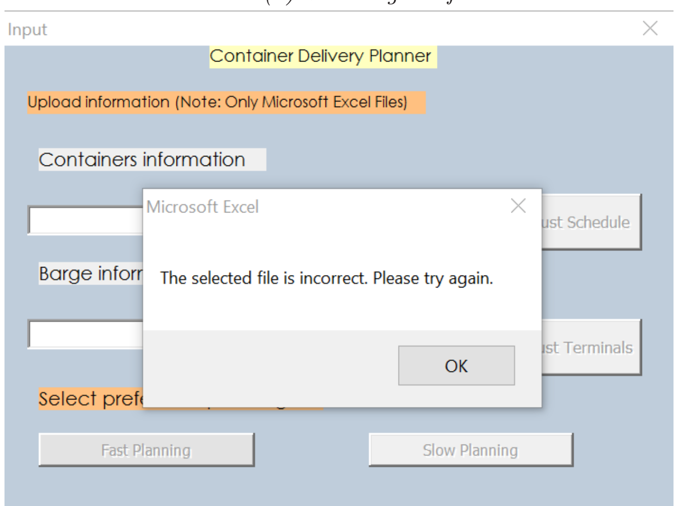
Figure 6: Input Screen

4. If you get an error twice see below **NOTE** for additional information about files.
5. buttons for the FAST and SLOW planning being enabled indicates correct input has been contributed. Fig 7(c).

NOTE File-selection option can display an error while you try to add specific file. Only one file is allowed to be added per field. Make sure that the file has extension "Excel Files (*.xls), *.xls,", "Excel Files 2007 (*.xlsx), (*.xlsx"). You are not able to upload (".csv") file. Examples of Containers Information and Barges Information are shown on Fig 8. It has to be mentioned that one of the possible causes for an error is wrong column order or name. So, please, make sure that your file is similar to the one attached in the example.



(a) Selecting the files



(b) Wrong Selection

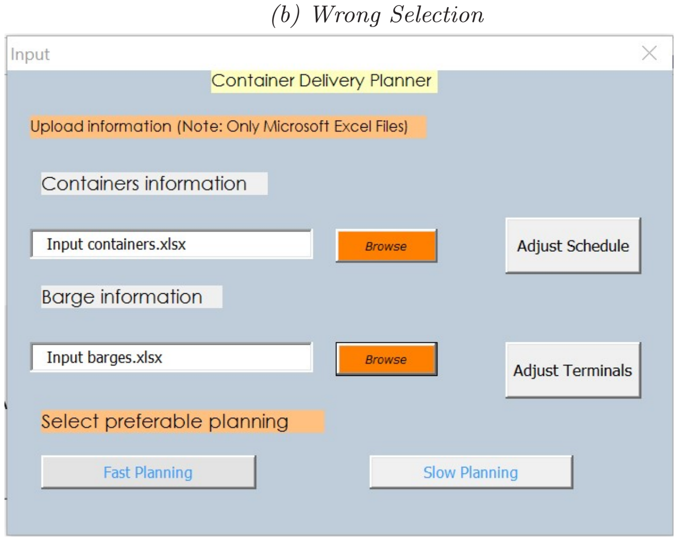
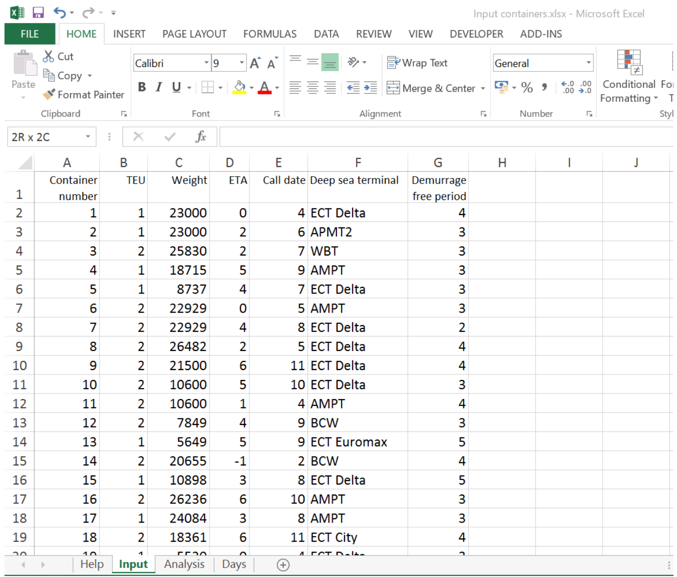
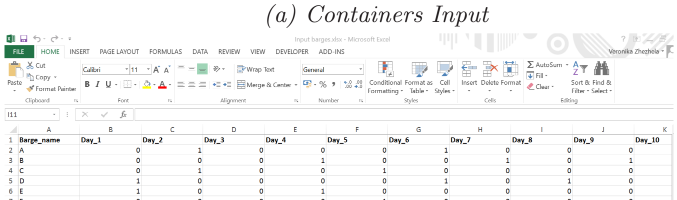


Figure 7: Upload Files



Container number	TEU	Weight	ETA	Call date	Deep sea terminal	Demurrage free period
1	1	23000	0	4	ECT Delta	4
2	1	23000	2	6	APMT2	3
3	2	25830	2	7	WBT	3
4	1	18715	5	9	AMPT	3
5	1	8737	4	7	ECT Delta	3
6	2	22929	0	5	AMPT	3
7	2	22929	4	8	ECT Delta	2
8	2	26482	2	5	ECT Delta	4
9	2	21500	6	11	ECT Delta	4
10	2	10600	5	10	ECT Delta	3
11	2	10600	1	4	AMPT	4
12	2	7849	4	9	BCW	3
13	1	5649	5	9	ECT Euromax	5
14	2	20655	-1	2	BCW	4
15	1	10898	3	8	ECT Delta	5
16	2	26236	6	10	AMPT	3
17	1	24084	3	8	AMPT	3
18	2	18361	6	11	ECT City	4

(a) *Containers Input*



Barge_name	Day_1	Day_2	Day_3	Day_4	Day_5	Day_6	Day_7	Day_8	Day_9	Day_10
A	0	1	0	0	0	1	0	0	0	
B	0	0	0	1	0	0	1	0	1	
C	0	1	0	0	1	0	0	0	0	
D	1	0	0	0	0	1	0	1	0	
E	1	0	0	1	0	0	0	0	0	
F	0	0	0	0	1	0	0	0	0	

(b) *Barges Input*

Figure 8: File Examples

How do I select type of planning?

One more click! Time to decide which type is right for you.

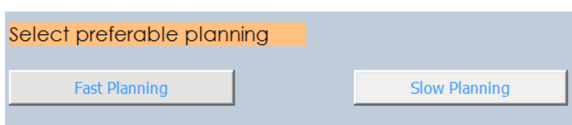


Figure 9: Type of planning

In the bottom of the screen Fig 9 you have two options to select depending on your desired output:

- **Fast planning**: will take you less than 20 seconds and provide a solution that is less accurate but very close to the opti-

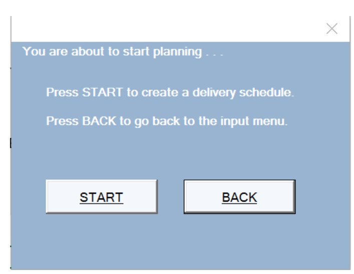


Figure 10: Start Planning

mal. It is based on a Heuristic method. An additional feature is "Adjust Schedule".

- **Slow planning**: will take you approximately 20 minutes (depending on computer speed) and will provide the best possible solution, among all alternative solutions, and is based on an ILP approach. You can make both available manual adjustments for this type of planning.

To choose the preferable type just click on one of the buttons. Then press START to proceed and BACK to go to the Input Screen as displayed in Fig 10.

Manual Adjustments

Manual Adjustments are two additional features to adapt the schedule according to your needs and preferences. Also, to make the tool more flexible from the changes you make to the output. If you do not want to set a new constraint to the system you can create the schedule from now on and move to the Output Screen section.

After successfully uploading the documents you can make manual adjustments to the schedule and terminal visits(at the right side of the Input screen). However, not both of them are available for both types of planning. If you are not sure what type of planning you want to use, read the previous section.

If you want to a container being delivered on a specific day using a specific mode of transportation → go the the Manual Schedule Adjustments.

If you want a specific barge to go or not go to a specific deep sea terminal → go to the Manual Terminal Adjustment.

How do I modify the schedule?

To modify the delivery schedule press the button "Adjust Schedule" on the input Screen and start filling in the form.

Guidelines:

1. Enter the container ID that should be transported, for instance "10".
2. Enter the day that the container should leave the inland terminal, for instance "6".
3. Scroll to choose the Barge or Truck, for instance "D" that should deliver the container "10" on day "6".
4. Select ADD to confirm the adjustment.
5. Select BACK to go back to the Input Screen.
6. Select RESET to clean the entries.

NOTE: All of the boxes should be filled in. If you have got an error message after selecting "Slow Planning" and attempting to add a constraint for a container, for which a constraint has already been added, this message informs you about it and you can replace the constraint or cancel the action, as it is shown in Fig 28.

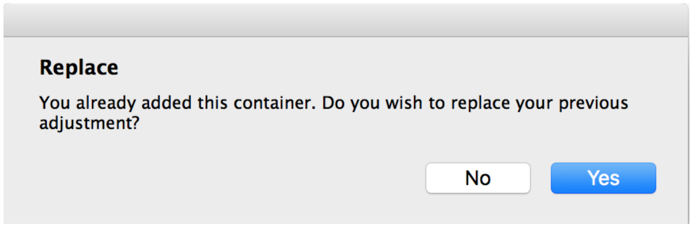


Figure 11: Possible Error.

You also should be aware of conflicting constraints. For example, if you select for a container from one of the terminals to be shipped by barge, and then also add a constraint for the same terminal to be closed on the same day, then the two constraints are conflicting and OpenSolver cannot find a feasible solution. In this case OpenSolver returns the error message which can be seen in Fig 12.

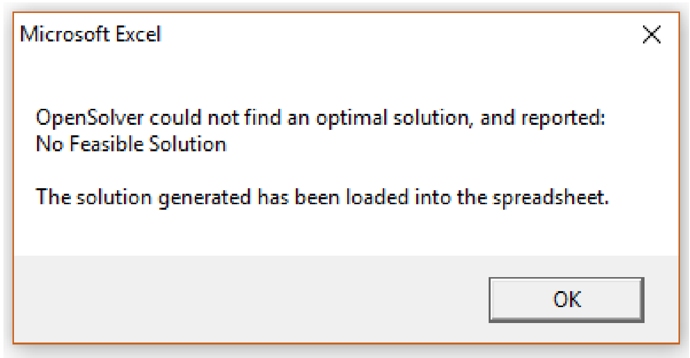


Figure 12: Infeasibility error when there are conflicting constraints.

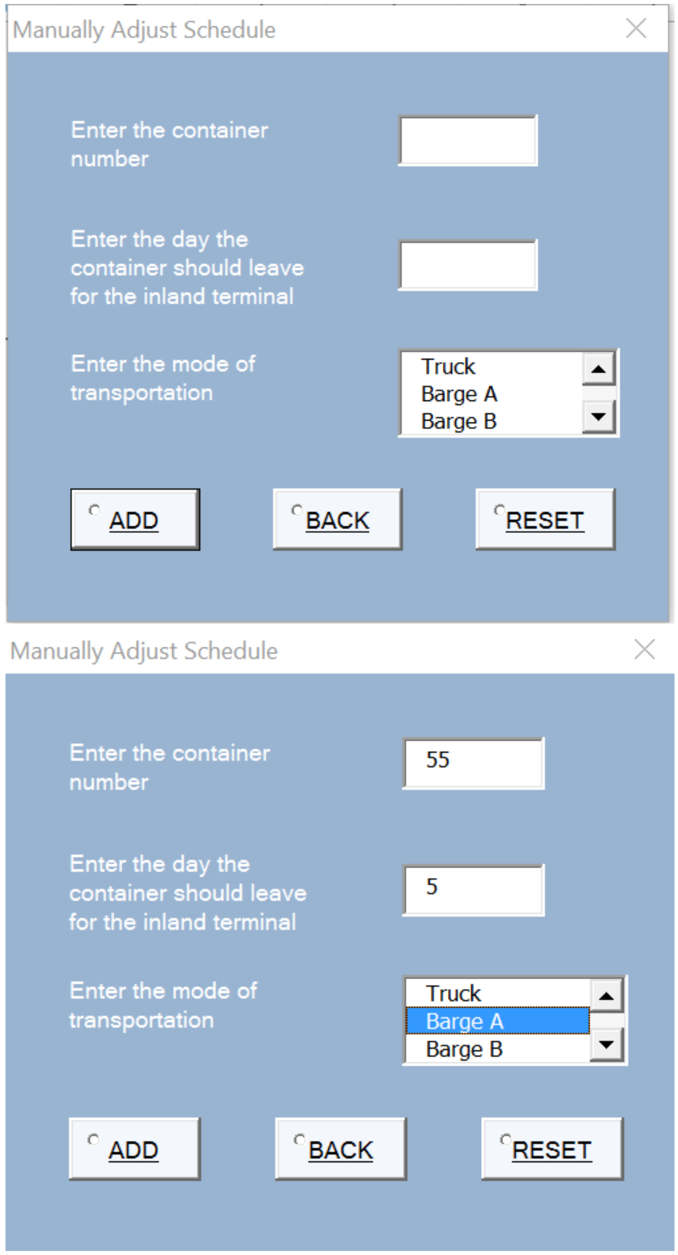
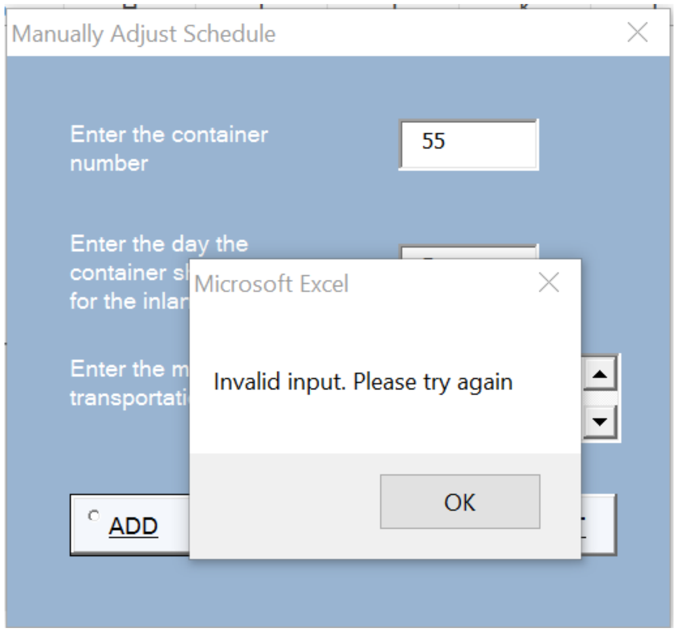
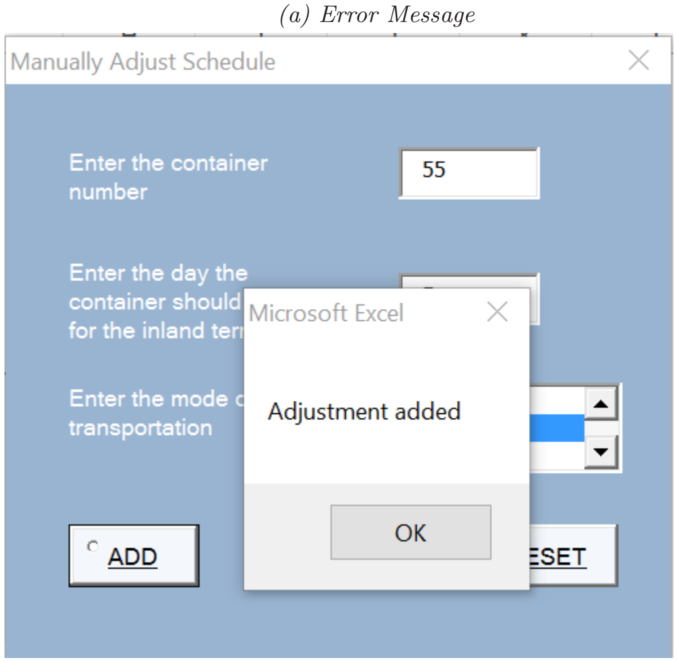


Figure 13: Manual Adjustments of the Schedule



(a) Error Message



(b) Correct Adjustment

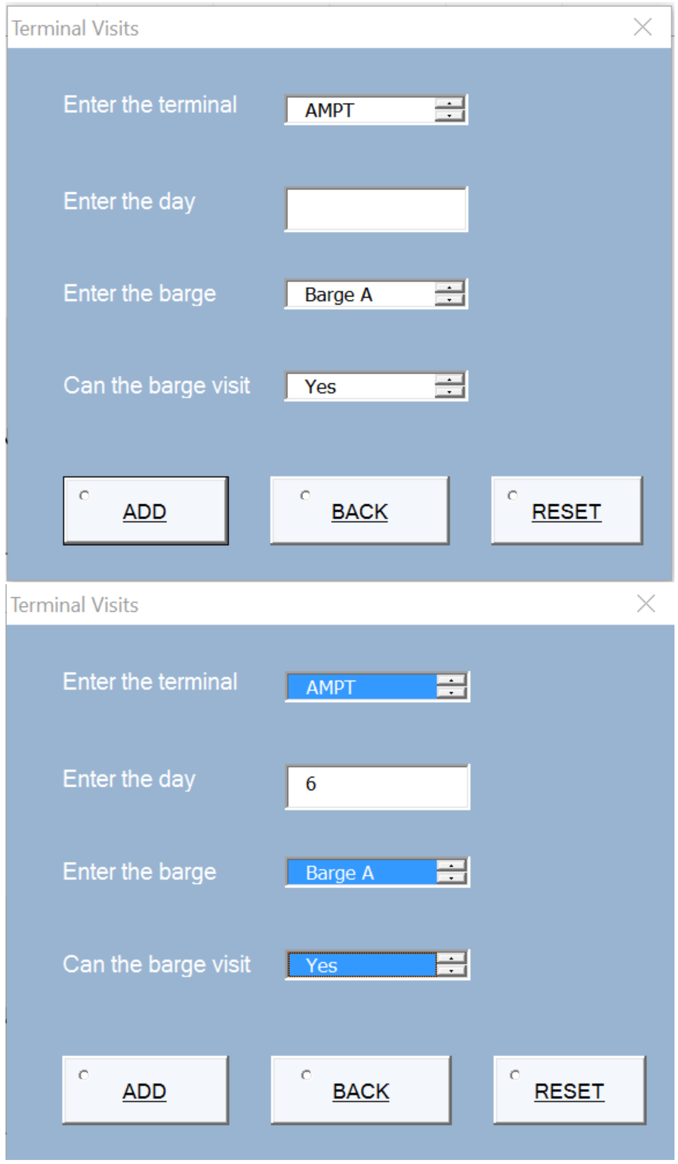
Figure 14: Add Adjustments to the Schedule

How do I arrange terminal visits?

If you want to update information regarding terminals, use the following method after pressing "Adjust Terminals" button on the main screen:

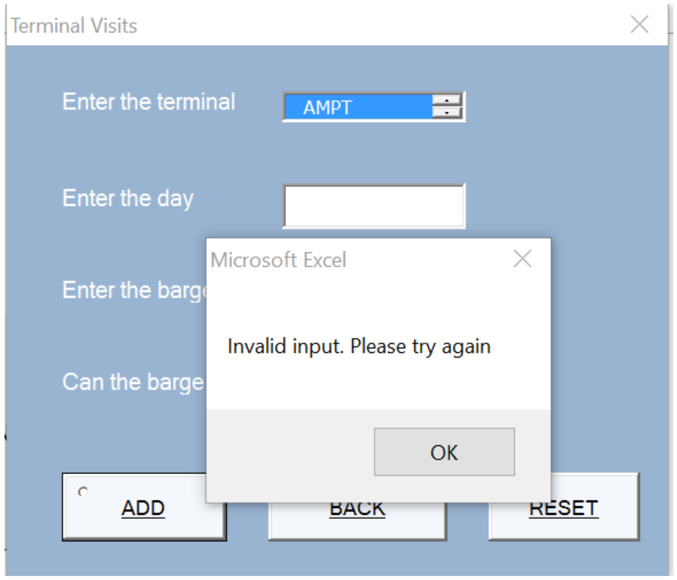
1. Insert container ID in the text box "Enter the container number".
2. Fill in the next text box with the day that you want to adjust.
3. Next, choose the transportation mode.
4. Press Yes if you want this barge to visit the terminal, otherwise press on the scroll bar and then select No.
5. Ensure that all of the fields are filled up. Check if they are colored in blue as in Fig 15 (a). Make sure to press on the field to make all boxes blue.
6. Select <u>ADD</u> to confirm the adjustment.
7. Select <u>BACK</u> to go back to the Input Screen.
8. Select <u>RESET</u> and confirm by clicking on "Yes" to start the procedure again.
9. Your changes are submitted once the "Terminal adjustment added" window is displayed.
10. To go to the input screen press <u>BACK</u>.

NOTE: In case of an error message like in Fig 16 (a) one of the following possible mistakes were made: non-numerical input, not all of the boxes are filled with information, invalid Container ID or wrong day of scheduling.

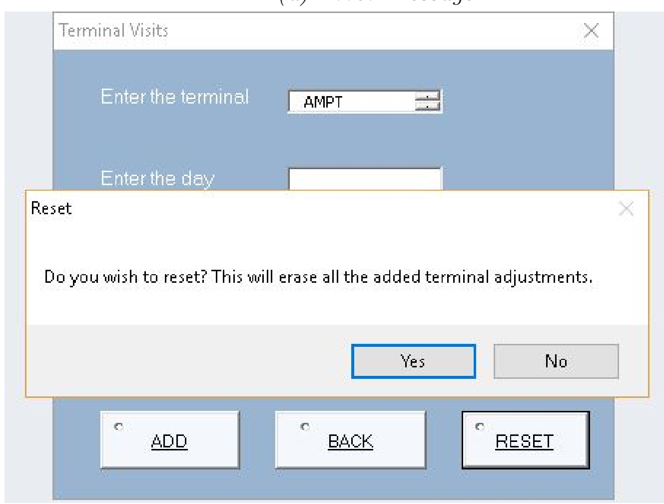


(a) Correct Adjustment

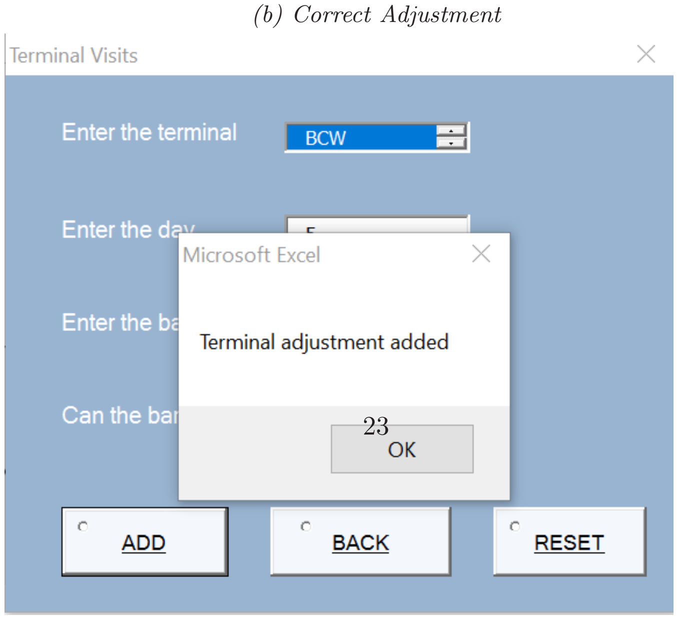
Figure 15: Add Adjustments to the Schedule



(a) *Error Message*



(b) *Correct Adjustment*



(c) *Error Message*

Figure 16: Add Adjustments to the Schedule

Output

Dialog window in Fig 17 indicates the completion of generating the desired results. As soon as its done, the output screen is placed on your laptop screen as in Fig 18.

The output will give you all the possible information about the schedule. It includes the day, mode and cost of the container and additionally provides the total overall cost,total transportation costs and demurrage costs for all containers, as an extension summaries are created to get an understanding of the schedule's effectiveness. All of the aforementioned, will be covered in this section.

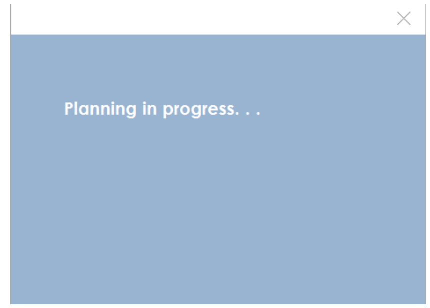


Figure 17: Planning in progress

How do I find information about a container?

The advanced search dialog helps you to find data about containers. You can type in the container's ID number and information about this specific container will be shown. In Figure 18 (b) the information of the container number 44 is provided as an example.

If the ID is not correct, you can clear up the textbox via the "Clear" button or the message of the "Incorrect ID" will appear as shown in Figure 19 (b). Another option to find the information about the container of interest is to filter the given dataset in Search Tab of the output as in Fig 19 (b) . You can also filter on each column. The procedure is the following:

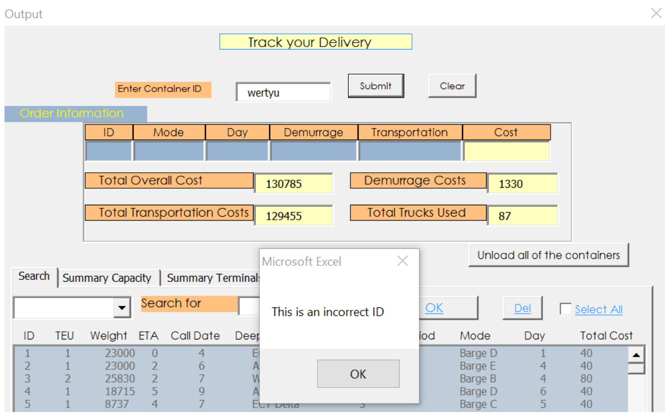
1. Choose the filter in the box.
2. Type the number or the name of the Terminal (with capital letter).
3. Press "OK".
4. Press "Del" to clean up the Listbox.
5. Press "Upload the containers" in order to see all the information again.

Furthermore, you can find information concerning each button and tab of the Output Screen.

Summaries of the datasets are placed in the tabs next to the search page.

(a) Advanced Search

Figure 18: Output Screen



(a) Invalid enter

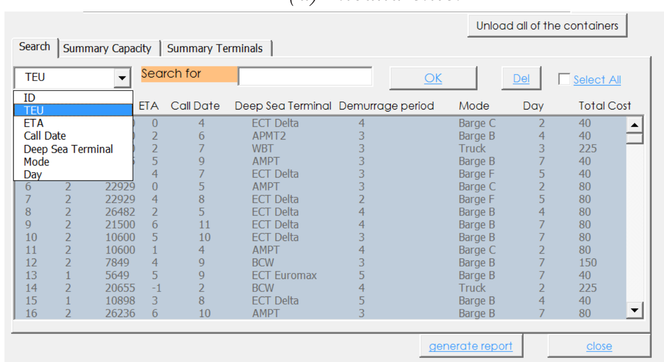


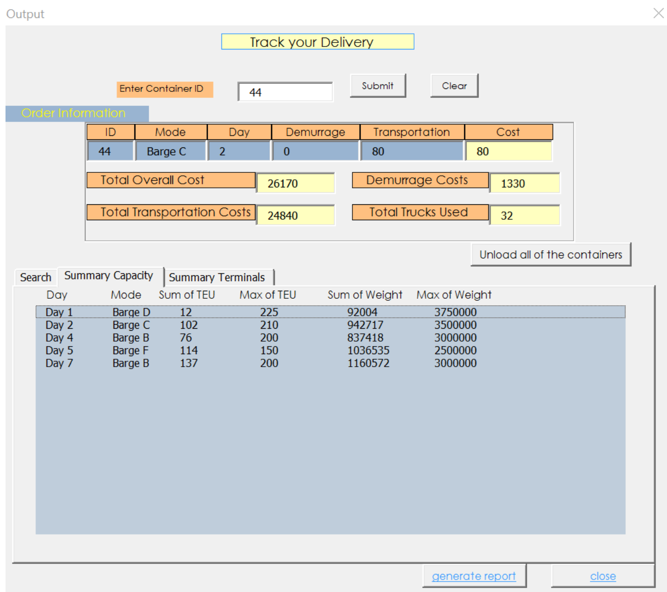
Figure 19: Filter Search

Summary Capacity

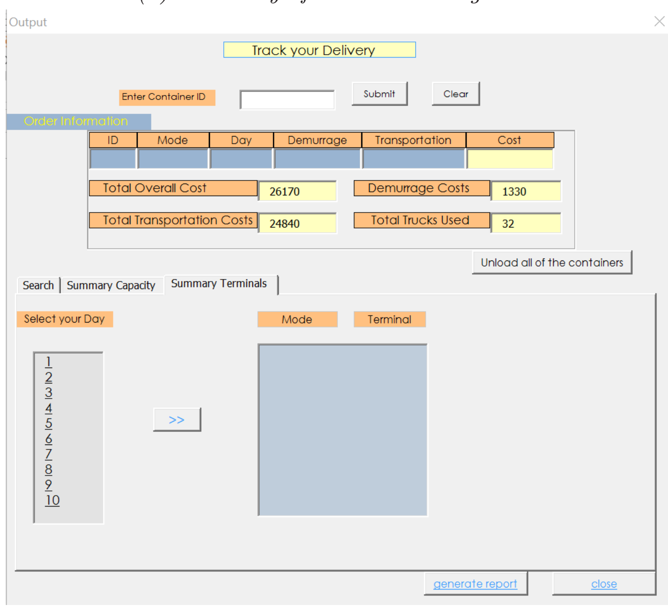
The summary on the TEU and Weight gives you information such as if the capacity meets possible shipment and how each mode is loaded comparing to the maximum values. To go to the section click on "Summary Capacity" label as in Fig 20(a).

Summary terminals

Last page of the data summary contains information regarding the terminal visits that are planned for each day for available barge on that day Fig 20 (b). After selecting the day of interest and pressing " >> " button the summary will appear as it is shown in Fig 24.



(a) Summary of TEU and Weight Measures.



(b) Summary of Terminals.

Figure 20: Summaries

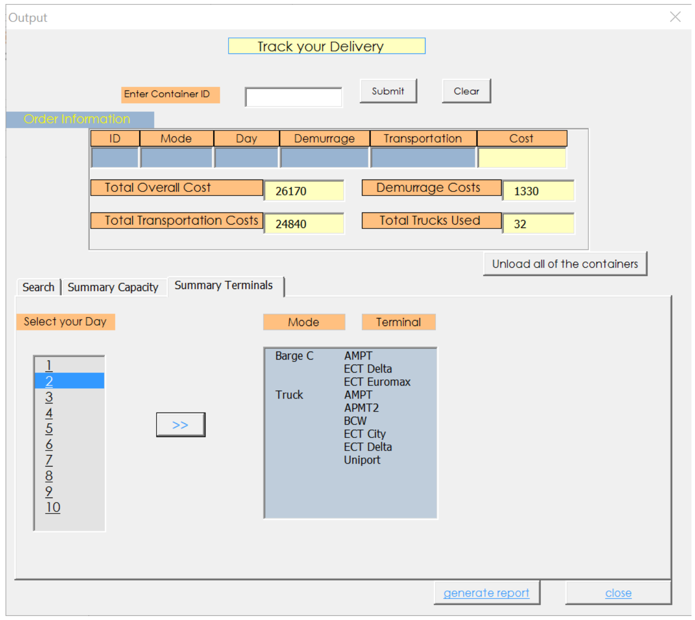
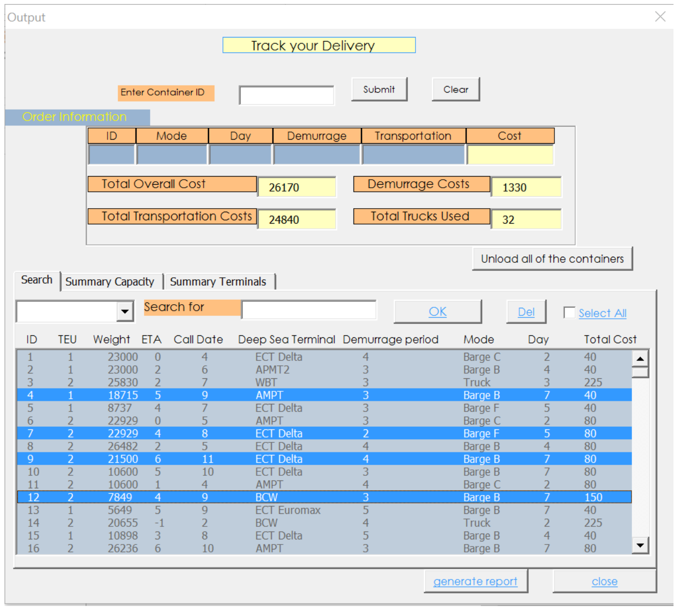


Figure 21: Summary for Day 2 on Terminals

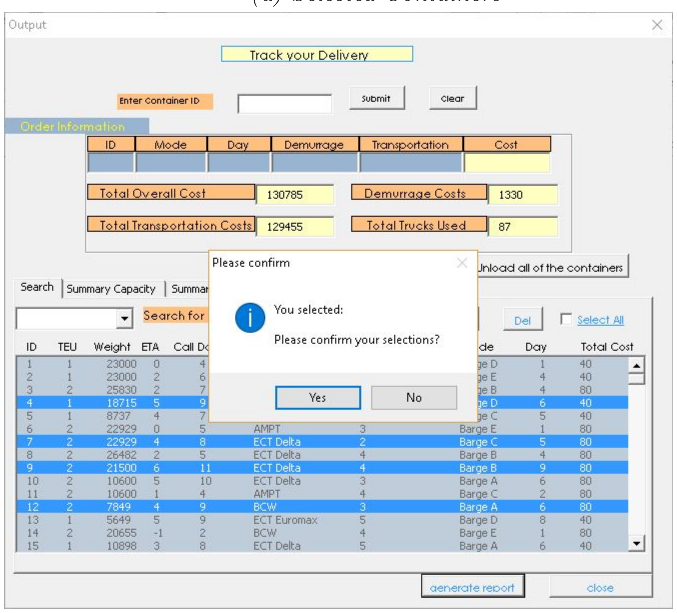
How do I generate a report?

In case that you need to save schedule for one, some of or all the containers on your computer, do the following:

1. Go to the main Output menu in Fig 18.
2. Select containers or click on the checkbox on the right "Select all".
3. Check if the right containers are highlighted with blue. Fig 22(a).
4. Confirm your selection-click on "Yes", "No" to go back one step Fig 22(b).
5. Click on "generate report" in the bottom of the form.
6. In "File name" enter the name for the report and press save Fig 23(a).
7. The report will be open automatically and the program will be closed. You can modify the report and save the changes Fig 23(b).

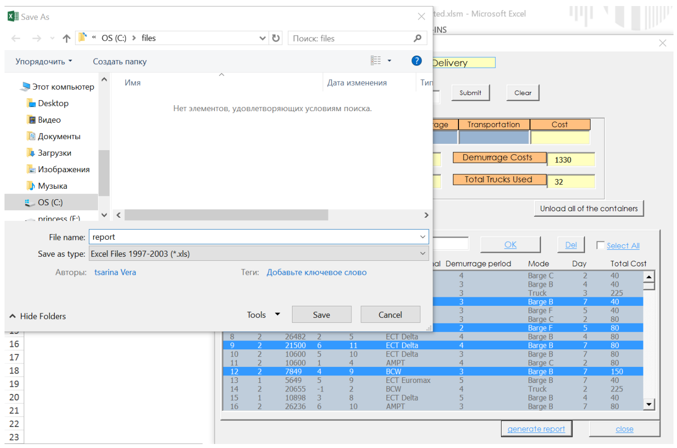


(a) Selected Containers

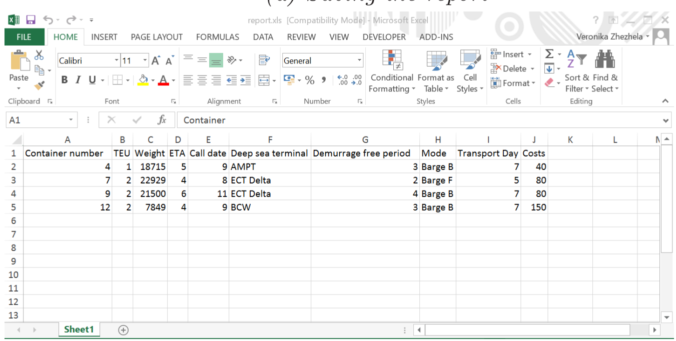


(b) Verify Selection

Figure 22: First steps to generate report



(a) Saving the report



(b) report

Figure 23: Generate Report

FAQ

The process can be canceled at any time by clicking Esc or close button in the right corner. After reopening the file the sheets of a previous session could have been stored. In that case do the following to reset:

1. Go to the Developer Menu in Excel.
2. Click on Visual Basic.
3. Choose the project "ContainerDss".
4. Press on the "+" to view all of the Modules.
5. Doubleclick on "hDump". Fig 24
6. Click on area "Sub DeleteSheets()" , press F5.
7. Click on area "Sub DeleteSheets2()" , press F5.

8. Click on Save icon in the left corner.

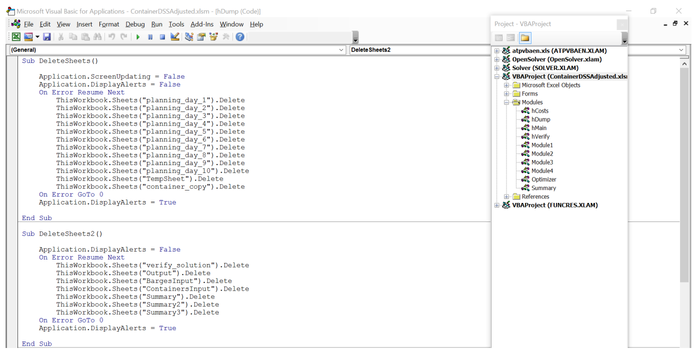


Figure 24: Visual Basic

Documentation

This section can be viewed as a user manual for a developer. Firstly, it will discuss per function its code. It will in turn explain general functions, followed by functions solely dealing with an ILP and functions relating to the Heuristic approach. Secondly, the ILP and Heuristic have some limitations that are addressed here. Thirdly, some of these limitations can be remedied by appropriately updating some parameters in the code, which is discussed here. Lastly an Excel sheet is created behind the scenes for the ILP while running the user interface and its components are elaborated upon here.

General functions

Userform Input_form

SUB BTN_ADDCONSTRAINTS_CLICK(): Opens the userform, ManAdjustSchedule to adjust the schedule prior to the schedule being created. It counts the number of days, barges and containers in order to test if the user inputs valid adjustments. This button will only be enabled if both the container and barge file are provided.
Code:

1. Defines several arrays that store and preserve the constraints as defined by the user.
2. Counts the `nrContainers`, `nrBarges` and `nrDays` to validate the users input.
3. If `ListboxBool` = True (this Boolean is added to make sure that constraints are not added twice).
   (a) Loops through the `nrBarges` + `nrTrucks`.
      i. If the `nrBarges` = 0 (truck) calls MANADJUSTSCHEDULE and adds the constraint.

ii. Else it is a barge and adds the constraint.

4. Sets `ListboxBool` = False.

5. Calls HideInputshowManAdjust() (hides the input sheet and shows the ManAdjust sheet).

Sub Btn_Adjust_Terminals_Click(): Opens the TerminalVisits userform to adjust the schedule prior to the schedule being created. It counts the number of days and barges in order to test if the user inputs valid adjustments.
Code:

1. Defines several arrays that store and preserves the constraints as defined by the user

2. Counts the `nrBarges` and `nrDays` to validate the users input

3. If `ListBargeBool` = True (this Boolean is added to make sure that constraints are not added twice)

    (a) Loops through the `nrBarges`.

    (a) Calls the TerminalVisits and adds the constraint.

4. Sets `ListBargeBool` = False.

5. If `ListTerminalBool` = True,

    (a) Adds all the terminals.

    (b) Sets `ListTerminalBool` = False.

6. Sets `ListTerminalBool` = False.

    (a) Adds the constraint.

    (b) Sets `ListYesNoBool` = False.

7. Calls HideInputshowManAdjust() (hides the input sheet and shows the TerminalVisits sheet).

Sub Btn_Browse_Barges_Click(): Allows the user to input the desired barge file. The function checks if the provided input is feasible by calling the ValidateData(). If this is the case it will copy the input worksheet of the barge file to the macro workbook by calling CopyInputSheetToMacroWB(). If not, a corresponding error message will be displayed. Code:

1. Sets screen updating to false and defines several string variables.

2. Filters on the type of files.

3. Sets the dialog box caption to select the files.

4. Handles the information as specified by the user.

5. If `BargeFileName` = True (the file is one of the correct types of files),

    (a) Sets variables to open the given barge file.

   (b) Validates the file by calling ValidateData and setting `bool` equal to the return.
   (c) If `bool` = True
      i. `BargeInputBoolean` = True.
      ii. Prints the name of the file that is chosen.
      iii. Calls CopyInputSheetToMacroWB() to copy all the information into the sheet called `Bargesinput`
6. Else the selected file is incorrect.
7. If the user uploaded any input sets `Checkforinput` = True.
8. If `Checkforinput` = True,
   (a) Shows the buttons in the input sheet.

Sub Btn_Browse_containers_Click(): Allows the user to input the desired container file. The function checks if the provided input is feasible by calling the ValidateData(). If this is the case it will copy the input worksheet of the container file to the macro workbook by calling CopyInputSheetToMacroWB(). If not, a corresponding error message will be displayed.
Code:

1. Sets screen updating to false and defines several string variables.
2. Filters on the type of files.
3. Sets the dialog box caption to select the files.
4. Handles the information as specified by the user.
5. If `ContainerFileName` = True (the file is one of the correct types of files),
   (a) Sets variables to open the given container file.
   (b) Validates the file by calling ValidateData and sets `bool` equal to the return.
   (c) If `bool` = True,
      i. `ContainerInputBoolean` = True.
      ii. Prints the name of the file that is chosen.
      iii. Calls CopyInputSheetToMacroWB() to copy all the information into the sheet called `Containerinput`.
6. Else the selected file is incorrect.
7. If the user uploaded any input sets `Checkforinput` = True.
8. If `Checkforinput` = True,
   (a) Shows the buttons in the input sheet.

SUB BTN_FAST_PLANNING_CLICK(): Starts the fast planning procedure. At the same time, it resets the Booleans that enable or disable the fast and slow planning buttons.
Code:

1. Sets screen updating false.
2. Calls `ResetInputBooleans()`.
3. Hides the input form.
4. Shows the progress of the fast planning.

SUB BTN_SLOW_PLANNING_CLICK(): Opens the userform OPENSOLVERPROGRESS, which first asks the user if he wants to continue. If so, the slow planning procedure will start and the Booleans that enables or disables the fast and slow planning buttons will be reset.
Code:

1. Sets screen updating false.
2. Calls `ResetInputBooleans()`.
3. Hides the input form.
4. Shows the progress of the OpenSolver.

Userform ManAdjustSchedule

SUB BTN_ADD_CLICK(): Checks if the users input is valid. If so, a message will be displayed saying that the adjustment is added. In addition, the three pieces of data will be added to three corresponding arrays. These will later be used to add the constraints to the OpenSolver.
Code:

1. Checks if the user input is valid.
    (a) If not, show message box Invalid input.
2. Loops through size of the array.
    (a) Checks if the user input is not defined before.
        i. Shows message box and if answer is no, empties the particular values.
    (b) Else stores the values as defined by the user.
3. Sets the different variables equal to the input as given by the user.
4. Increments the `NrofUserConstraints` (to add to OpenSolver).
5. Adjusts the values of the different arrays.
6. Shows message box that all the adjustment are added.
7. Resets the values of the input dialog.

SUB BTN_BACK_CLICK(): Resets all the values of the textboxes and returns to the INPUTFORM Userform.
Code:

1. If the user clicks on the back button empties the previous values.
2. Calls HIDEMANADJUSTSHOWOUTPUT().

SUB BTN_RESET_CLICK(): Asks if the user wants to reset all of the added constraints. If so, all the added constraints will be deleted.
Code:

1. If the user clicks reset, gives a message box to confirm action.
2. If answer = no,
   (a) Exits Sub.
3. Else, resets all the values.
   (a) Adjusts the arrays
   (b) Enables the fast planning button (since no constraints are added in MANADJUSTSCHEDULE)
   (c) Calls HIDEMANADJUSTSHOWOUTPUT().

Userform OpenSolverProgress

SUB BTN_BACK_CLICK(): Closes the userform OPENSOLVERPROGRESS and displays the INPUT FORM Userform.

SUB BTN_START_CLICK(): Starts the Slow Planning process and displays the message that the planning is in progress.
Code:

1. Shows the different buttons.
2. Shows the planning in progress message.
3. Resets all the previous input given in the dialog boxes.
4. Calls BUILDOS
5. Calls HIDEOSPSHOWUF2

Userform SplashUserForm

Sub UserForm_Activate(): Displays the opening screen.
Code:

1. Displays the loading data message for a second.
2. Displays the creating forms message for a second.
3. Displays the opening message for a second.
4. Waits for another second and gets rid of the SplashUserForm.

Userform TerminalVisits

Sub Btn_ADD_Click(): Checks if the users input for Terminal Visits are valid. If so, a message will be displayed saying that the adjustment is added. In addition, the three pieces of data will be added to three corresponding arrays. These will later be used to add the constraints to the OpenSolver.
Code:

1. Checks if the user input is valid.
   (a) If not, shows message box Invalid input.
2. Loops through size of the array.
   (a) Checks if the terminal adjustment is not defined before.
      i. Shows message box and if answer is no, empties the particular values.
      ii. Else stores the values as defined by the user.
   (b) Checks if the inputs for the barge, terminal and day are not the same.
      i. Shows message box and if answer is no, empties the particular values.
      ii. Else stores the values as defined by the user.
3. Sets the different variables equal to the input as given by the user.
4. Increments the `NrofUserConstraints` (to add to OpenSolver).
5. Adjusts the values of the different arrays.
6. Shows message box that all the adjustment are added.
7. Resets the values of the input dialog.

Sub Btn_TerminalBACK_Click(): Resets all the values of the textboxes and returns and calls HideTerminalShowInput().
Code:

1. If the user clicks on the back button, empties the previous values.
2. Calls HideTerminalShowInput().

Sub Btn_TerminalRESET_Click(): Resets all the values of the textboxes and returns and calls HideTerminalShowInput().
Code:

1. If the user clicks reset, gives message box to confirm action.
2. If answer = no.
   (a) Exits Sub.
3. Else, resets all the values.
   (a) Adjusts the arrays.
   (b) Enables the fast planning button (since no constraints are added in ManAdjustSchedule())
   (c) Call HideTerminalShowInput().

Userform FastProgress

Sub Btn_Back_Click(): This sub makes sure that if the user clicks on the back button the FastProgress screen is hidden and the input form is shown.

Sub Btn_Start_Click(): Starts the Fast Planning process and displays the message that the planning is in progress.
Code:

1. Shows the different buttons.
2. Shows the planning in progress message.
3. Resets all the previous input given in the dialog boxes.
4. Calls hMain.FastSchedule().
5. Calls hMain.ShowProgress().

Userform UserForm2(OutputForm)

Private Sub cmdSubmit_Click(): Submits button for displaying the output. Checks if the input of the Container ID is valid or the cell is not empty.
Code:

1. If the input is not numeric or the field is empty it gives an error.

2. Else looks for the input in the column with Container ID.
3. Finds the value using VLookup() function and in each of textboxes for the Day, Mode etc. gives the corresponding column value.

Private Sub cmbc_clear_Click(): Clears the values for the output boxes.

Private Sub cmbUload_Click(): Displays the Excel Sheet with output information.

Private Sub OK_Click():
Code:

1. Checks if the search field is not empty.
2. If is empty, returns the message "Please enter a value".
3. Check the combobox selection, if it is not selected, message "Choose a filter field".
4. Clears the listbox after a valid input.
5. Distinguishes for each of the filters in the combobox using the Select Case property.
6. After case is selected finds the corresponding row first, second and etc. the symbol that coincides with input to the search field.
7. Adds Item to the listbox with corresponding indexed column.

Private Sub Del_Click(): Clears the listbox.

Private Sub CheckBox1_Click():
Code:

1. Checks if the checkbox `Value = True`.
2. Selects all values in Listbox.

Private Sub cmd_close_Click(): closes Userform.
Code:

1. Calls the function hDump.DeleteSheets2().
2. Unloads the Userform if the FastProgress is enabled.

Private Sub CommandButton3_Click():generates report
Code:

1. Sets selection of container `contselect = False`.
2. If the row was selected `contselect = True` .
   (a) Check = MsgBox with confirmation of selection.
   (b) Adds the Worbook hides it.
   (c) Loops through selected rows and Adds values of "A" column to the "Sheet 1" in hidden Workbook.
   (d) Adds values of B column and each next column one by one with selected row.

3. Calls `NewFile = Application.GetSaveAsFilename` for user to save and name new hidden Workbook.
4. Else, messages "Please select at least one container."
5. If `Check` = "Yes" go add the Workbook.
6. If `Check` ="No" unselect rows.
7. Open added Workbook.

Private Sub UserForm_Initialize(): For output values as Total Transportation Costs, demurrage Costs displays sum of corresponding columns till last non-empty cells. For "Total truck" that was used counts the number of "Truck" cells.

For summary listboxes shows values from Sheet "Summary2" .

To the "Summary Terminal" lisbox with "Day" counts the number of day and displays it.

Summary

Sub GenerateSummary(): Generates the three summaries necessary for the output userform. Code:

1. As the output of the ILP and Heuristic is not the same, it should be converted into one general form. First the subroutine starts checking if the sheet "Output" exists. If this does not exists, it is created and checks for the sheet "TESTOutput".

   The sheet "Output" is the heuristic solution, and already converted in the general form. Sheet "TESTOutput" is from the ILP and needs some little tweaks before it is in the general form.

   (a) Copy TESTOutput information into the Output sheet.

   (b) Loop through the containers of .

      i. Copy the value for the Transport in each container.

      ii. Calculate the Transport Cost and Demurrage costs.

2. If the "Output" sheet exists.

   (a) Loop through all the containers.

      i. Calculate the Transport Cost and demurrage costs.

3. Generate the summaries automatically from output file. There are 3 different summaries that needs to be generated. Summary 1 gives the general overview. Here is visible to what terminal each truck/barge goes and how many they transport. Summary 2 focuses on the barges and they give a quick overview of the load. The loaded and maximum capacity (TEU and Weight) of each barge is here visible. So this makes it possible to check if the barges are definitely not overloaded. Also they could obtain the amount of space left on a certain barge. The last summary (summary 3) gives an overview of which terminal is visited by which barge at a certain day.

(a) Creates the sheets Summary, Summary2 and Summary3 with its respective headers.

(b) Loop through the days of planning.

i. Generate a report for each day and verify solution.

ii. Extract unique transport methods for this day.

iii. Look at the terminals that are visited that day for those barges.

ILP functions

Module 1

Function sheetExists(WB As Workbook, sheetToFind As String) As Boolean: This function searches through the name of available sheets and returns true if the selected sheet exists, else false.
Code:

1. Sets `SheetsExists` = False.
2. Activates the corresponding worksheet.
   (a) Loops through all the available worksheets.
      i. If the name of the sheet is found sets `SheetsExists` = True.
3. Next sheet.

Function GetFilenameFromPath(ByVal strPath As String) As String: If the user opens any file, this function makes sure that the file name is selected without the additional directory name.
Code:

1. If there is no backslashes available and the length of the string is > 0,
   (a) Returns the rightmost characters without the backslash.

Function CheckForInput() As Boolean: Before enabling the planning buttons in the `inputform` Userform this function checks if the user defined any input. If true, planning buttons are enabled, if false they are not.
Code:

1. Sets `CheckForInput` = False.
   (a) If the user selected a containers inputsheet and barges inputsheet,
      i. Sets `CheckForInput` = True, this enables the planning buttons for the user.

Function CreateOutputWorksheet(WB As Workbook, SheetName As String) As Worksheet: When the user imports any schedule this function makes sure a new worksheet is

created where this information can be stored.
Code:

1. Sets the WB and WS variables to the corresponding workbook and worksheet variables.
2. Adds the new sheet with `Sheetname` to the workbook.
3. Creates the new output worksheet.

Function CountNrRowsinWS(WS As Worksheet) As Long: For any given worksheet this function counts the number of rows.
Code:

1. Activates the corresponding worksheet.
2. Counts number of rows, starting from the bottom, ending at the top.

Function CountNrColumnsinWS(WS As Worksheet) As Long: For any given worksheet this function counts the number of columns.
Code:

1. Activates the corresponding worksheet.
2. Counts number of columns, starting from the last cell at the right, ending at the left.

Sub ClearSheet(WS As Worksheet: Every time the model is initialized this function makes sure all the sheets are cleared.

Sub CopyInputSheetToMacroWB(WSName As String): Takes a worksheet name as input and copies either the barge or container input file to the corresponding worksheet in the macro workbook.
Code:

1. Checks if the provided input string corresponds to a container or barge input file and sets the corresponding worksheet and workbook variables.
2. Checks if the destination sheet already exists. If so, clear that sheet. If not, creates a new sheet.
3. Copies the entire input sheet.
4. Pastes it to the set worksheet in the macro workbook.

Sub ResetInputBooleans(): Resets all the input Booleans in the tool to check for new input.
Code:

1. Resets the `ContainerInputBoolean` and `BargeInputBoolean` back to false.

Sub DeleteWorksheet(WS As Worksheet): Deletes a selected worksheet.

Function ValidateData(WB As Workbook, DocumentName As String) As Boolean: Every time the user uploads the container and barges files, this functions validates that all the attributes are defined and placed in the right columns.
Code:

1. Defines several variables to compare the text in the different files
2. If the selected file is the container data,
   (a) Checks if the `TEU` is placed in column 2, `Weight` in column 3, `ETA` in column 4, `CallDate` in column 5, `Deep sea terminal` in column 6.
      i. If the sum of all the string comparisons is equal to zero,
      ii. Then sets `Correctdata` = True.
3. If the selected file is the barge input data file,
   (a) Checks in the first column if the names of the barges are defined, if so sets `Correctdata` = True.
4. Sets `ValidateData` = `Correctdata`.

Sub BuildOS(): Initializes the model by calling all the different functions and then runs OpenSolver.
Code:

1. Calls `SetAndClearSheetsforOSAlgorithm`.
2. Resets the OpenSolver.
3. Calls `CopyInputToOSWS`.
4. Calls `CountDaysBargesContainer`.
5. Sorts the container input alphabetically on Deep Sea Terminal.
6. Calls `AllocateSpaceForDecVar`.
7. Calls `AssignCostsConstraints`.
8. Calls `AssignCapacityConstraints`.
9. Creates the array `InfeasibleConstraints` to store all the infeasible adjustments added by the user.
10. Loops through all the adjustments added by the user. If there are none, this part of the code can be ignored.
    (a) Calls `AddUserConstraint` for each added adjustment.
11. Calls `ETACallConstraint`.
12. Calls `TEUWeightConstraint`.

13. Calls `SetBinaryAndObjectiveCell`.

Module 2

Sub DecVarSum(): Creates a column of length `nrContainers` filled with ones. The ones are used to make sure that all the values of the `ETACallConstraints` are equal to one.
Code:

1. Allocates space for the Transport Constraint header.
2. Loops through the number of containers.
    (a) Pastes ones under the Transport Constraint header.

Sub ETACallConstraint(): Create `nrContainers` amount of constraints. These constraints ensure that each container is shipped one day after its ETA and arrives at its Call date at the latest. In order to achieve this, the sum of the decision variables between ETA + 1 and the Call date have to equal one.
Code:

1. Calls DecVarSum().
2. Sets the `TransportRange` equal to all the ones produced in DecVarSum().
3. Creates `ETAbool` and `Callbool` to ensure that the start and end of the range can only be set once. These Booleans are set to True, once the start and end of the containers `ETACallDate` range are set.
4. Sets `CurrentDay` = 0.
5. Loops through the number of containers.
    (a) Finds the ETA and Call date value for the corresponding container.
    (b) Loops through the amount of available trucks and barges per day.
        i. Updates `CurrentDay` accordingly.
        ii. Sets `startOfRange` equal to the current column, if `CurrentDay` = ETA + 1.
        iii. Sets `endOfRange` equal to the current column, if `CurrentDay` = CallDate.
        iv. If at the end of the loop the `endOfRange` has not been found yet, sets `endOfRange` equal to the last column.
    (c) Sets `tempRange` equal to the current container row, starting from `startOfRange` and ending at `endOfRange`.
    (d) In order to add this constraint to OpenSolver, pastes (and keeps the reference) the sum of `tempRange` next to the corresponding value of the `TransportRange`.
    (e) Resets `CurrentDay`, `ETAbool` and `Callbool`.

(f) Sets `finalRange` equal to the sum of all the `tempRanges`.

(g) Adds a constraint to the OpenSolver, by ensuring that all the values in the `finalRange` are equal to the corresponding value in the `TransportRange`.

Sub SetAndClearSheetsforOSAlgorithm(): In order to reference the corresponding worksheets in the workbook, this sub sets the values of the variables equal to these sheets. At the same time it calls the ClearSheet() function.
Code:

1. Sets the WB and WS variables to the corresponding workbook and worksheet variables.
2. Clears the corresponding sheets.
3. Activates the OpenSolver worksheet.

Sub CopyInputToOSWS(): Copies the container and barges input information to the OpenSolver worksheet and pastes them into the desired position.
Code:

1. Counts the number of rows and amount of columns in the container and barge worksheets.
2. Copies the barge and container input to the OpenSolver worksheet.

Sub CountDaysBargesContainers(): Calculates the number of days, barges and containers that are provided by the user.
Code:

1. Sets `nrBarges` = `BargeNrRows - 1` (Subtract one due to the header) .
2. Sets `nrDays` = `BargeNrColumns - 3` (Subtract 3 for the max Weight, max TEU and Barge Names headers).
3. Sets `nrContainers` = `ContainerNrRows - 1` (Subtract one due to the header).

Sub AllocateSpaceForDecVar(): Allocates space for the decision variable header by counting the number of days and counting the available barges and trucks per day. Once this is done, it will allocate the same amount of space for the corresponding costs header.
Code:

1. Creates a one-dimensional array, `NrAvailableBargesPerDay` and a two-dimensional array `BargeArray`.
2. Creates variables `currentColumn` and `currentColumnCost` which stand for the current column in the decision variable space to be allocated and the cost space to be allocated.

3. Sets `totalBarges` = 0, `counter` = 0.
4. Loops through number of days
   (a) Loops through all the barges.
      i. Stores the values of the barge input worksheet into `BargeArray`.
      ii. If `BargeArray(value)` = 1, increments counter by 1.
   (b) Stores counter into the corresponding `NrAvailableBargesPerDay` position.
   (c) Increments `totalBarges` by counter.
   (d) Resets counter.
5. Loops through the number of days
   (a) Pastes the corresponding day into the right cell of the OpenSolver worksheet using `currentColumn` and `currentColumnCost`.
   (b) Under this header, allocates space for the available barges that day + a truck.
   (c) If the current column does not stand for a truck, loops through the available barges on the current day.
      i. Increments the variables `currentColumn` and `currentColumnCost` by 1.
   (d) Again, increments the variables `currentColumn` and `currentColumnCost` by 1.
6. Creates a temporary array of size `totalBarges`
7. Loops through the number of days (i)
   (a) Loops through the number of barges (j)
      i. If `BargeArray(i,j)` = 1 stores that barge into `temparray`.
8. Loops through the number of columns of the decision variable and cost space, using `currentcolumn`.
   (a) Under the day header row, checks if the cell does not correspond to a truck. If so, pastes a truck header value of zero. If not, pastes the corresponding barge header.

   Copies the header range of the decision variables and pastes this header next to itself to create a header for the costs.

Sub AssignCostsConstraints(): This function calculates the total costs corresponding to each decision variable and pastes them into the costs range. The total costs consist of the demurrage costs and the transportation costs.
Code:

1. Creates an array `day_to_pay` of size `nrContainers`.
2. Loops through the number of containers.

(a) For each container, add its ETA + demurrage free period and store this value into the corresponding `day_to_pay` position.

3. Creates a two-dimensional array `demurrage_costs` of size `nrContainers` and `nrDays`.
4. Loops through the number of containers (i).
   (a) Loops through the number of available barges + trucks (j).
      i. Updates `CurrentDay` accordingly.
      ii. If we are dealing with a truck, `total_costs(i,j)` = 225 + the corresponding `demurrage_costs`. Stores this value and its reference to it into its position in the cost range.
      iii. If we are dealing with a barge, `total_costs(i,j)` = 40 · TEU + the corresponding `demurrage_costs`. Stores this value and its reference to it into its position in the cost range.

Sub AssignCapacityConstraints(): This function uses a barges maximum weight and maximum TEU to create constraints for the OpenSolver.
Code:

1. Creates two arrays `max_teu` and `max_weight` of size `nrBarges`.
2. Loops through the number of Barges.
   (a) For each barge stores its maximum weight and maximum TEU into the corresponding arrays.
3. Defines variables `WeightRange`, `TEURange`, `current_barge` and `auxiliary`.
4. Loops through the available barges and trucks.
   (a) Only takes columns into account which do not represent a truck.
   (b) Updates `current_barge` to the current column.
   (c) Creates one temporary range `tRange` and sets this equal to the range under the current header. It has a length of `nrContainers`. Calculates the sumproduct of `tRange` · all the TEU values. Likewise, calculates the sumproduct of `tRange` · all the weight values.
   (d) Adds two constraints saying that both previously specified ranges need to be less than or equal to the corresponding `max_teu` and `max_weight` values.
5. Adds headers to the cells in which these constraints will be placed.

Sub TEUWeightConstraint(): After calculating the TEU and Weight constraints this functions adds these constraints to OpenSolver.
Code:

1. Creates two ranges (`usedRange` and `limitRange`) for both the TEU and the Weight.

2. Sets these ranges equal to the corresponding cells in the sheet.
3. Adds constraints to open solver by setting the `usedRange` equal or smaller than the `limitRange`.

SetBinaryAndObjectiveCell(): This sub combines all the transportation costs with the congestions costs into one objective cell.
Code:

1. Sets `tempRange_2` equal to the container total costs range.
2. Sets `tempRange_3` equal to the congestion costs range.
3. Sums the values of both ranges and pastes the reference in the corresponding cell.

Sub HideOSPShowUF2(): This function hides the OpenSolverProgress userform and shows the output userform.

Function AddUserConstraint(ConNr As Long, Day As Integer, Mode As Integer) As Boolean: This function adds the manually added user constraints to the OpenSolver if these adjustments are feasible. If not, a corresponding error message will be added to the `InfeasibleConstraints` array. These messages will later be displayed.
Code:

1. Initializes `ErrorMessage`, set `CurrentDay` = 0 and set `tempBool` = False.
2. Loops through all the container ID's.
   (a) Finds the container corresponding to the input `ConNr`.
   (b) If the added adjustment is infeasible due to a chosen shipping date prior to the container's ETA + 1, due to a chosen shipping date post the container's call date or due to barges not being available on the chosen day, store the corresponding error message in `ErrorMessage`.
   (c) If the constraint is feasible, add the constraint to the OpenSolver and set the corresponding boolean to true.
3. If the chosen adjustment was infeasible, resize and preserve the `InfeasibleConstraint` array and add the `ErrorMessage` to the array.

Sub HideManAdjustShowInput(): Hides the ManAdjustSchedule userform and shows the Input_Form userform.

Sub HideInputShowManAdjust(): Hides the Input_Form and shows the ManAdjustSchedule userform.

Sub AddTerminalConstraints(): Adds the user's added terminal constraints to the OpenSolver if these constraints are feasible. If not, a corresponding error message will be stored and later be displayed.

1. Allocates space for `NrTerminals`, `ErrorMessage` `TerminalName` and `FeasibleBool`.

2. Loops through all the available terminals per day.
    (a) If the specified input data corresponds to one of the cells, adds a constraint specifying what value this cell needs to have to the OpenSolver.
    (b) Sets `FeasibleBool` to true.
3. If `FeasibleBool = False`, adds the corresponding error message to the `InFeasibleTerminalConstraints` array.

Module 3

Sub CongestionConstraint(): This function creates constraints for the OpenSolver relating to minimizing congestion and adds the decision variables to the OpenSolver.
Code:

1. Creates several variables used in this function to hold specific information.
2. Loops through the days.
    (a) Loops through the barges.
        i. If there is a barge present on this day in the barge input file,
        ii. Adds 1 to the `barge_counter` vector on this day.
3. Sets parameters `f_multiplication_teu_constant` and `a_multiplication_container_constant` the higher the f, the higher the travel costs, the higher the a the higher the loading costs.
4. Counts the number of containers at each specific terminal and save this is in a vector `container_counts`.
5. Declares several more variables to aid in making the constraints.
6. Loops through the total available modes of transportation per day (header).
    (a) If we have reached truck as another mode of transportation iterates `current_day_2` as we have reached the next day,
    (b) If not we must still be at a barge so do the following, loops through all the terminals.
        i. Sets `current_terminal` to the appropriate value, they are ordered alphabetically.
        ii. Saves the current barge and finds its corresponding capacity limits and stores them in `Teu_to_check` and `weight_to_check`.
        iii. Calculates loading, travel and congestion costs according to the formula mentioned at the theory section.
        iv. Calculates total loading, travel and congestion costs by summing the above for all feasible day, barge, terminal combinations.
        v. Sets `current_range` to be a pointer towards the range containing the decision variables x_{ijk}, where only the i coming from `curent_terminal` t are selected.

vi. Sets `current_range_2` to be a pointer towards the corresponding y_{tjk}.

vii. Uses `current_range` and `current_range_2` to add all congestion constraints for this day, barge and terminal combination.

(c) Resets some of the count variables for the next iteration.

7. Selects the decision variable range corresponding to the y_{tjk} in `decision_range_2` and unionizes this with the x_{ijk} stored in `DecVarRange` to then add them to the solver as the decision variable cells.
8. Selects the range of costs in `cost_range` to then reference a sum to this to output the total congestion costs in the sheet to later construct the objective.

Sub TheOutput(): This sub creates the output file that translates all the decision variables into the correct information. It gives exactly the amount of containers, TEU and Weight with the corresponding barge and day. In addition it also outputs the amount of terminals each barge visits per day.
Code:

1. Creates 3 different columns where the information of the Mode, Transport Day and Total Costs per container can be stored.
2. Declares several arrays and variables.
3. Sorts the file on the Deep Sea Terminals alphabetically.
4. Sets Boolean `AlreadySet` = False.
5. Loops through the containers.
   (a) Loops through the available barges and trucks.
      i. If the corresponding header is a truck update `current_day`.
         A. If the value is not zero and `AlreadySet` = False,
         B. Updates `nrTrucks`, `teuTrucks`, `weightTrucks`.
         C. Outputs the Mode, Day and Total Costs per container.
      ii. Else it is a barge.
         A. If the value is not zero and `AlreadySet` = False,
         B. Updates `Barges`, `teuBarges`, `weightBarges`.
         C. Outputs the Mode, Day and Total Costs per container.
6. To create output for the number of terminals visited by each barge per day declares several arrays and variables.
7. Counts the number of containers at each terminal and stores the information in `container_counts`.
8. Sets counters to zero, create another day counter `current_day_2`.
9. Loops through the available barges and trucks.

(a) If the corresponding header is a truck update `current_day_2`,

10. Loops through the `nrTerminals`.
    (a) For each specific terminal take exactly the start cell of the range and the end cell of the range.
        i. If the sum of values in that specific range is > 0,
        ii. Updates Terminals array.
11. Next step is to output all the information stored in the different arrays and variables
12. Copies the titles to specific cells in the sheet.
    (a) Loops through the number of available trucks.
    (b) Outputs the `Day`, `nrTrucks`, `teuTrucks`, `weightTrucks`.
        i. Loops through the number of barges.
        ii. Outputs `BargeName`, `Terminals`, `teuBarges`, `weightBarges`.
13. Unsort everything.

Heuristic functions

hMain

SUB FASTSCHEDULE(): It calls all the necessary functions in the correct order for the fast planing to start.

SUB SHOWPROGRESS(): Hides the INPUT form and the progress of the fast planning is shown.

SUB SHOWOUTPUT(): Closes the progress of the fast planning, and the OUTPUT window is shown.

SUB START() : This is the Main function of the Heuristic method, all the procedures for the heuristic method are executed or called from this function.
Code:

1. A temporary worksheet `wsTemp` is created to save records for processing, the first row keeps the same header as the `ContainersInput` file, if the worksheet already exists, it is deleted and is created again.
2. `ContainersInput` and `BargesInput` are defined as worksheets, `ws0` and `bs` respectively. The information from the `ContainerInput` is copied into a new Sheet `container_copy` defined as `ws`, if the worksheet already exists, it is deleted and is created again.
3. Data modification, the values for the demurrage free period are rewritten as `demurrage free period + ETA`.

4. Loop through the possible days of transportation `today`.
    (a) Creation of a planning sheet per day `planning(today)` worksheet for all transportation days `today`.
5. If the user decided to make any modification.
    (a) Loops through the number of desired changes `UserConNrArr`.
    (b) Selects the `planning(today)` worksheet of the change.
    (c) Calls the function HandChangeData().
    (d) Loops through the infeasible constraints and shows an error message for each of the errors.
6. Loops through the days `today` (days of transportation).
    (a) Defines last row `lrow` from `input_container` sheet. Defines range `rng` based on the last row and the last column (`G`).
    (b) Reads barge information into an array named `barges`. If there are available barges for the day then,
        i. Determines the number of available barges for this day and loops through this value in the barges information.
        ii. From this loop, the name, Maximum TEU and Maximum weight will be obtained and stored in `barges`.
    (c) In the containers information `ws`, ETA filters are applied.
        i. If it is the first day of planning, all the containers with ETA smaller than the planning day are selected, `ETA<today`.
        ii. If today is between the second day and the day before the last pick up, all the containers with ETA smaller than today minus one are selected, `today` $>$ 1 and `today` $<$ last day.
        iii. Otherwise everything is selected.
        iv. The selected information is pasted in the temporary work sheet `wsTemp`.
    (d) Sorts the temporary worksheet.
        i. Sorts by ascending demurrage period.
        ii. Sorts by ascending call date.
        iii. Sorts by ascending TEU.
    (e) Verifies if the user did any change by hand. If there is any value already included in the `planning(today)` sheet.
        i. Loops through the number of changes that have been done in this day `RangeHand`, select the information of the containers.
        ii. Loops through the number of available barges for today `numBargesToday`.

iii. Selects the TEU and the Weight of this container.

iv. Updates TEU and weight left in the `barges` array.

(f) Copies all the information from the temporary sheet to the `planning (today)`, loops as long as there are values in the temporary sheet `wsTemp`.

i. Defines the minimum weight `miw` and the Minimum TEU/size `mit` that are in the `wsTemp`.

ii. Checks the availability of the barges, using the function `BargesFree`.

iii. If there is an available barge, tries to schedule the first in the row.

A. Selects the size/TEU, weight and the call date from the container on the top, defined as `cSize`, `cWght`,`cCall` respectively.

B. `pSship` indicates which is the barge that is available today and has enough space left for the selected container.

C. If there is any available barge, indicates the number of the barge in the column `H`, then cut the information from `wsTemp` and places it in the next row available in the `planning(today)` sheet.

D. When the first one in the row is not possible to schedule and there is enough time left, places them at the last line of the temporary file. First option to plan this container again, is tomorrow.

E. When Call date is equal to `today`, then ship by truck, it is indicated in the column `I`, then cuts the information from `wsTemp` and places it in the next row available in the `planning(today)` sheet.

iv. If there are no barges available, then just ships the containers that needs to be available at the inland terminal today.

A. The mode of transportation is indicated in the column `I`, then cuts the information from `wsTemp` and places it in the next row available in the `planning(today)` sheet.

7. Verifies that the solution fits all the basic constraints.

Sub PrepareOutput(): It creates a new work sheet with the necessary format for the output form.
Code:

1. A new worksheet output is created, the header is defined with the same names as the containers input file, but now also the Mode, Day and Total cost are added.

2. Loops through the days of planning.

(a) The full name of the mode of transportation is set, either the name of the barge or the word Truck is written in the column Mode.

(b) Reverses demurrage free period modifications on input.

(c) Copies the full information from the planning(i) sheet in the available space in the output sheet.

Function LastRow(sh As Worksheet): The function gives the row or column number of the last cell with data in one row or one column. The input for this function is a worksheet.
Code:

1. If there is any error it will return 0.
2. Looks for the last value in the selected Sheet.

Function TotalWeight(barge As Worksheet, bargeID As String) As Long: Calculates total weight in barge, based on loaded containers. The input is a Worksheet with all the final information after applying the heuristic method `barge`, and the barge that is selected `bargeID`.
Code:

1. Calculates the range of the `barge`.
2. Applies a filter in the column where mode in the barge worksheet = `bargeID`.
3. Sums the total Weight for this barge.

Function TotalTEU(barge As Worksheet, bargeID As String) As Integer: Calculate total TEU in barge based on loaded containers.The input is a Worksheet with all the final information after applying the heuristic method `barge`, and the barge that is selected `bargeID`.
Code:

1. Calculates the range of the `barge`.
2. Applies a filter in the column where mode in the barge worksheet = `bargeID`.
3. Sums the total TEU for this barge.

Function TotalTrucks(plan As Worksheet) As Integer: Calculate total number of trucks needed in planning for this day. The input is a Worksheet with the final information after applying the heuristic method per day `plan`.
Code:

1. Calculates the range of the `plan`.
2. Applies a filter to select the containers that are shipped via truck.
3. Sums the total of Trucks used for this day.

Function WhichBarge(brg As Variant, nob As Integer, cw As Long, ct As Integer) As Integer: Assigns the available barge to the container. The input is the `barge` array, `nob` the number of barges available today, `cw` the container weight and `ct` the container TEU.
Code:

1. Loops through the number of available barges today.
   (a) If there is enough space left for the weight and the TEU of the container in the barge, then the container is assigned to that barge.
2. Otherwise any other barge is assigned.

Function BargeFree(brg As Variant, nob As Integer, mw As Long, mt As Integer) As Integer: Determines whether there is or not any available barge for the container that is going to be scheduled. The input is the `barge` array, `nob` the number of barges available today, `mw` the minimum weight necessarily today and `mt` the minimum TEU/size necessarily today.
Code:

1. If there is any available barges today. Then, loops through the number of available barges.
   (a) Checks if there is no space left on the barge for TEU, or for the minimum weight `mw`.
2. Else there are no available barges today.
   (a) Checks if there is space at least for a TEU=2, and for the minimum weight `mw`.
      i. Assigns the container to the barge and `BargeFree` = 1.
3. Otherwise there is no free barge `BargeFree` = 0.

Function HandChangeData(ID As Long, newDay As Integer, newMode As Integer, planning As Worksheet, ws As Worksheet, bs As Worksheet) As Boolean: Verifies and executes the adjustments done by the user. The input is the container that the user wants to reschedule `ID`, the day `newDay`, the new mode of transportation `newMode`, the worksheet `planning` corresponding to `newDay`, the containers information `ws` and the barges information `bs`
Code:

1. Sets `HandChangeData` as False, creates an `ErrorMessage` and defines the last row included in the containers information `ws`.
2. Defines the range to look up for any value into the containers information. The `VLookup` function is used to look for the container `ID`, in the defined range `Lookup`, in the column indicated (2 TEU, 3 Weight, 4 ETA , 5 Call date, 6 Deep sea terminal, 7 Demurrage free period).
   (a) If the ETA is larger or equal than the `newDay` and the `newMode` is not Truck, then shows the error message depending on the constraints that are not met, and sets `HandChangeData = False`.

(b) If the ETA is larger or equal than the `newDay` and the `newMode` is Truck, then shows the error message depending on the constraints that are not met, and sets `HandChangeData = False`.

(c) If the Call date is smaller or equal than the `newDay` and the `newMode` is not Truck, then shows the error message depending on the constraints that are not met, and sets `HandChangeData = False`.

(d) If the Call date is smaller or equal than the `newDay` and the `newMode` is Truck, then shows the error message depending on the constraints that are not met, and sets `HandChangeData = False`.

(e) If `HandChangeData = False`

i. Counts how many Infeasible constrains the adjustment has.

ii. Indicates which is the `ErrorMessage` that needs to be shown.

3. If all the constraints are satisfied then it does the changes the user wants.

(a) Verifies if there is another value in the planning day sheet, in order to define the position of the new information of the container in this sheet.

(b) Adds all the information from the container to the planning sheet, and the desired mode of transportation. This is done using the function Vlookup.

(c) Deletes the information of the container from the `ws` sheet.

FUNCTION N2L(BYVAL RNG AS INTEGER): Changes the numbers to the letters from the alphabet.

hCosts

FUNCTION CALCTOTALCOSTS(TRANSPORTDAY AS INTEGER, CALLDAY AS INTEGER, FREEPERIOD AS INTEGER, ARRIVALDAY AS INTEGER, CONTSIZE AS INTEGER, OPTIONAL MODE AS STRING) AS LONG: This function calculates the cost for the shipping of each container in the `planning` Sheet.
Code:

1. The demurrage costs are calculated.

2. If there is no input for mode.

(a) uses default shipping rules, If `transportDay < callDay`,

(b) Calculates costs for shipping by barge.

(c) Else if `transportDay = callDay`, calculates costs for shipping by truck.

(d) Else there is an error, not possible to ship anymore.

3. Calculates costs for preference mode.

Function TotalPlanningCosts() As Long: It calculates the cost for the final schedule, looping through all the days.

Function TotalCosts(plan As Worksheet, day As Integer) As Long: Calculates the total cost per container.
Code:

1. Defines the range of the `plan` worksheet, which is the plan per day.
2. Loops through all the containers in the worksheet.
   (a) If the 8th column does not indicate the name of the barge, then the container is shipped by Truck. Else by the barge.
   (b) Calculates the total costs using the CalcTotalCosts function.
3. The costs values are added and defined as `TotalCosts`.

Function TransportCosts(Mode As Boolean, contSize As Integer) As Long: Calculates the costs of transportation, depending on the selected mode.

DemurrageCosts(CurrentDay As Integer, arrivalDay As Integer, freePeriod As Integer, contSize As Integer) As Long: Calculates demurrage costs. These costs are based on the container size (TEU) and demurrage free period.
Code:

1. Calculates the demurrage free period `dfp = freePeriod - arrivalDay`.
2. Defines the maximum value between `(current Day - arrivalDay)` - demurrage free period and 0.
3. Calculates the demurrage costs, the maximum value calculated above times the demurrage costs depending on the size of the container.

Function DemurrageContCost(size): Calculates the demurrage costs per container based on its size, 50 for 1 TEU containers and 70 for 2 and 3 TEU containers.

hVerify

Sub VerifySolution() : Verifies that the heuristic solution met all the constraints.
Code:

1. A verification sheet is created, this will be used in the final report.
2. A container verification temporary sheet is also created.

3. The total costs is calculated and included in the verification sheet.
4. Loops through the schedule days.
   (a) Sets planning worksheet for each day.
   (b) Reads barge information into array. If there is any available barge today then saves the TEU and weight in `barges`.
   (c) If there is any available barges then loops through the barge range.
      i. For the selected day and the correspondent barge, includes in the verification sheet the information in regards to:
         A. Total weight in the barge Vs maximum weight in the Barge.
         B. Total TEU in the barge Vs maximum TEU in the Barge.
         C. When barges are not overloaded (TEU and Weight), makes cell green.
   (d) Else an error message is displayed "No barges today".
   (e) For the selected day, calculates the same values above but regarding Trucks.
5. Loops through the schedule days.
   (a) Creates a list of errors warning messages at the end of the report.
   (b) Verifies that basic constraints are met.
      i. Checks if all containers are transported after arrival date.
         A. Loops through the containers that are scheduled for the day.
         B. If the constraint is not satisfied, then prints the error and stores it in the list of errors.
      ii. Checks if all containers are transported before or at call date.
         A. Loops through the containers that are scheduled for the day.
         B. If the container is transported by truck and transport day <= call date, then prints the error and stores it in the list of errors.
         C. Else, transported by barge and transport day <call day, then prints the error and stores it in the list of errors.
         D. Copies all container numbers in container verification temporary sheet.
      iii. Checks if all containers are transported exactly once.
         A. Loops through the containers.
         B. Counts how many times is the container included in the schedule.
         C. If the containers is included more than once, then prints the error and stores it in the list of errors.

6. If the range of the error list is larger than 1 then prints the message "There are some error's. Perhaps solution is not correct.".

hDump

Sub DeleteSheets(): Deletes all the temporary sheets that are used through the development of the heuristic method.

Sub DeleteSheets2(): Deletes the input and output information that is used when the planning start over again.

Sub DeactivateScreenUpdating(): Deactivates the screen updating.

Sub ActivateScreenUpdating(): Activates the screen updating.

Limitations ILP

The tool is fairly general but there are some limitations to the ILP. Namely the input file size is dictated via 4 factors, `nrContainers`, `nrBarges`, `nrDays` and `nrTerminals`. `nrContainers` and `nrDays` are completely general and can be made arbitrarily large, as long as they are positive integers. `nrBarges` is constrained to be a positive integer up to 15 at most. This is due to the fact that this factor has not been taken into account when writing the code and trying to alter this would require a complete overhaul of the code to dynamically update more appropriately. `nrTerminals` is fixed at 9, but the following subsection explains how to add one more terminal. We trust that the reader can extend this to adding 2 or more terminals.

How to add a terminal with name = TERMINALNAME

In AddTerminalConstraints()

change

```
NrTerminals = 9
```

to

```
NrTerminals = 10
```

and add

```
TerminalName = "TERMINALNAME"
```

after line

```
TerminalName = "WBT"
```

In Btn_Adjust_Terminals_Click() add

```
TerminalVisits.ListBox_Terminal.AddItem "TERMINALNAME"
```

after line

```
TerminalVisits.ListBox_Terminal.AddItem "WBT"
```

In CongestionConstraint() change

```
NrTerminals = 9          'currently we are fixed at 9 terminals
```

to

```
NrTerminals = 10          'currently we are fixed at 10 terminals
```

and add

```
containercounts(9) = Application.WorksheetFunction.CountIf(Range("F19:F" & r), "TERMINALNAME")
```

after

```
containercounts(8) = Application.WorksheetFunction.CountIf(Range("F19:F" & r), "WBT")
```

and add

```
current_terminal = "TERMINALNAME"
```

after

```
current_terminal = "WBT"
```

In TheOutput() change

```
NrTerminals = 9          'currently we are fixed at 9 terminals
```

to

```
NrTerminals = 10          'currently we are fixed at 10 terminals
```

and add

```
containercounts(10) = Application.WorksheetFunction.CountIf(Range("F19:F" & r), "TERMINALNAME"
    )
```

after

```
containercounts(9) = Application.WorksheetFunction.CountIf(Range("F19:F" & r), "WBT")
```

Limitations Heuristic

The heuristic has at this time a few limitations:

1. The heuristic is limited to a planning horizon of 10 days. Besides that, the algorithm is by design appropriate to work without a limited planning horizon. This functionality is not implemented yet and for now the algorithm makes a planning for 10 days. Also it requires a barge information sheet with exactly 10 barge days mentioned as in the initial dataset.
2. Terminal adjustments are not implemented. It is not possible to forbid a barge to visit a terminal at a certain day. It requires some modification in the heuristic to filter these containers from the barge planning.
3. The heuristic does not take costs for the next day into account. So it cannot predict if it is much more expensive to ship tomorrow and it will only base its solution on demurrage period, call day and container size.

There are no restrictions on number of barges per day, terminals or containers. The limitation for these parameters is based on the limitation of an integer in our programming language VBA. The integer will overflow when there are more than 32,767 containers, terminals or barges. As the weight of a container is stored as long integer, this will overflow at 2,147,483,647. Both integer values are signed.

Terminal optimizer

The optimizer is not limited to two barges, it will work with any amount of barges available. When there are at least two barges available on a certain day, the optimizer can work. The optimizer will not find any optimal solution, but it tries to find a balance in a heuristic way. So from this it also follows that it is not possible to save some barges for another day.

Changing Parameters ILP

Truck Cost

The current truck costs are a standard 225 for a container of any size or weight. To change this number to for example 150, change the following line of code:

In AssignCostsConstraints()

```
total_costs(i, j - (totalBarges + nrDays + 8)) = (225 + demurrage_costs(i, Day - 1)) '*
    WSOpenSolver.Cells(i + 19, j - (totalBarges + nrDays))
```

To the following line of code:

```
total_costs(i, j - (totalBarges + nrDays + 8)) = (150 + demurrage_costs(i, Day - 1)) '*
    WSOpenSolver.Cells(i + 19, j - (totalBarges + nrDays))
```

Anything more elaborate like making the costs dependent on the TEU or weight is recommended to be done within the loop that this line of code is in.

Barge Transportation Cost

The current barge costs for a container are a set 40 per its TEU. To change this number to for example 60, change the following line of code:

In AssignCostsConstraints()

```
total_costs(i, j - (totalBarges + nrDays + 8)) = (40 * WSOpenSolver.Cells(i + 19, 2) +
     demurrage_costs(i, Day - 1)) '* WSOpenSolver.Cells(i + 19, j - (totalBarges + nrDays))
```

To the following line of code:

```
total_costs(i, j - (totalBarges + nrDays + 8)) = (60 * WSOpenSolver.Cells(i + 19, 2) +
     demurrage_costs(i, Day - 1)) '* WSOpenSolver.Cells(i + 19, j - (totalBarges + nrDays))
```

Anything more elaborate like making these costs depend non linearly on the TEU or the weight is recommended to be done within the loop that this line of code is in.

Demurrage Cost

The current demurrage costs are a set vector of (50,70,70) for respectively 1, 2 and 3 TEU to change this to for example (60, 80, 90) do the following:
Change this block of code:

In AssignCostsConstraints()

```
If j > day_to_pay(i) - 1 Then 'if number of days that exceed demurrage >0
   If WSOpenSolver.Cells(i + 19, 2) = 1 Then
       demurrage_costs(i, j - 1) = (j - (day_to_pay(i) - 1)) * 50
    Else
      demurrage_costs(i, j - 1) = (j - (day_to_pay(i) - 1)) * 70
    End If
Else
    demurrage_costs(i, j - 1) = 0
End If
```

To the following block of code:

```
If j > day_to_pay(i) - 1 Then 'if number of days that exceed demurrage >0
   If WSOpenSolver.Cells(i + 19, 2) = 1 Then
      demurrage_costs(i, j - 1) = (j - (day_to_pay(i) - 1)) * 60
   Else
      If WSOpenSolver.Cells(i + 19, 2) = 2 Then
         demurrage_costs(i, j - 1) = (j - (day_to_pay(i) - 1)) * 80
      Else
         demurrage_costs(i, j - 1) = (j - (day_to_pay(i) - 1)) * 90
      End if
   End If
Else
   demurrage_costs(i, j - 1) = 0
End If
```

Where the 50 and 70 have been replaced by 60 and 80 and another if else statement was added. This was done because the original code only took into account that the containers with 1 TEU had 50 demurrage costs and anything else 70. A more extensive generalization like aking it dependent on the number of days it has been since the demurrage free period requires more significant rewriting of this particular code block, but it is recommended to be done at this place in the code.

Travel Cost

By default the travel costs of a corresponding barge are its TEU limit times a constant 2. To change this to for example 5 change the following line of code:

In CongestionConstraint()

```
f_multiplication_teu_constant = 2 'set parameters
```

to the following line of code:

```
f_multiplication_teu_constant = 5 'set parameters
```

Loading Cost

By default the loading costs of a corresponding terminal are its number of containers leaving from that terminal times a constant 1.5. To change this to for example 3 change the following line of code:

In CongestionConstraint()

```
f_multiplication_teu_constant = 1.5 'set parameters
```

to the following line of code:

```
a_multiplication_container_constant = 3
```

Congestion Cost

For a more particular direct change to the Congestion costs, it is recommended to change the following line of code to for example make the congestion costs a constant 3400

In CongestionConstraint()

```
WSOpenSolver.Cells(9, nrDays + 8 + auxi_2 + j) = barge_counter(current_day_2 - 1) * (
    f_multiplication_teu_constant * Teu_to_check + a_multiplication_container_constant *
    containercounts(j)) 'calculates total cost
```

to the following line of code:

```
WSOpenSolver.Cells(9, nrDays + 8 + auxi_2 + j) = 3400 'calculates total cost
```

Changing Parameters Heuristic

Transportation Cost

In the current code the truck transportation costs are fixed at 225 and for the barge, they are set to 40 per TEU. To change these values it is necessary to rewrite two lines of code. As example the truck costs are changed to 400 and the barge costs to 60 per TEU.

In TransportCosts(Mode, ContainerSize)

```
If Mode = 0 Then
   TransportCosts = 40 * contSize
Else
   TransportCosts = 225
End If
```

To the following line of code:

```
If Mode = 0 Then
   TransportCosts = 60 * contSize
Else
   TransportCosts = 400
End If
```

When the barge transport costs are also fixed and do not rely on the container size anymore, it is possible to change it into a fixed number too (like the truck costs).

Demurrage Cost

The current demurrage costs are set as 50, 70, 70 fro respectively 1, 2, and 3 TEU. To adjust the demurrage costs, it is possible to change these values in the code. Change the following lines:

In DemurrageContCost(size)

```
   If size < 2 Then
      DemurrageContCost = 50
   Else
      DemurrageContCost = 70
   End If
```

To the following block of code:

```
If size = 1 Then
   DemurrageContCost = 10
ElseIf size = 2 Then
   DemurrageContCost = 20
ElseIf size = 3 Then
   DemurrageContCost = 30
End If
```

When the demurrage costs should be 10, 20 and 30 respectively for 1, 2, and 3 TEU.

Explanation Excel Sheets ILP

Behind the user interface a sheet is created with helpful output information that gives a deeper overview of how the code works. This section will delve into the topography of the sheet and explain each region of it in turn. The creation of this sheet was done by the initial instance, so if the input file will change, most of the regions will expand and update dynamically.

In sheet `OSalgorithm`:

A1:M7, barge input file
A18:G1780, container input file
H17:AD1780, decision variables, x_{ijk}
AE17:BA1780, cells holding transportation and demurrage costs combined
BC18:BD31, cells relating to TEU capacity constraints of barges
BF18:BG31, cells relating to weight capacity constraints of barges
BM17:BN1780, cells relating to shipping constraints of containers
Q1:ED5, decision variables, y_{tjk}
Q6:ED8, actual costs made by opening terminals
Q9:ED14, extra information about terminal costs
P1, objective function cell
O2:P5, extra information about total costs per type

In sheet `TESTOutput`:

A1:G1763, container input file
H1:I1763, mode of transportation and day that the container in this row leaves
J1:J1763, cost related to transporting the container in this row
L1:S11, the number of containers that is being shipped on a specific barge or via trucks per day
L14S24, the number of TEU that is being shipped on a specific barge or via trucks per day
L27:S37, the number of weight that is being shipped on a specific barge or via trucks per day
U1:AA11, the number of terminals that are visited by a specific barge per day

Verification ILP

In order to verify the functionality of the shipping tool, several instances were tested to check whether all the requirements are met or not. Firstly, some findings from data analysis, which aid the explanation of some of the results that are provided. The constraints are then tested in order to evaluate the performance of the model.

Initial Data Set

The provided data set is composed of 1762 containers, distributed throughout 9 deep sea terminals. In table 2 the total number of containers per ETA value per terminal are listed, as well as the total amount of TEU per ETA value. This shows a clear difference in the number of arrivals at the different terminals, with a maximum number of arrivals at terminal ECT Delta and a minimum number of arrivals at terminal WBT.

Terminal \ ETA	-1	0	1	2	3	4	5	6	Total # Containers
APMT	57	51	62	72	73	79	68	65	527
APMT2	17	24	12	18	29	25	33	16	174
BCW	4	3	7	14	5	7	9	5	54
ECT City	3	4	2	8	5	6	10	5	43
ECT Delta	92	64	82	105	115	86	104	93	741
ECT Euromax	11	13	10	11	8	18	17	12	100
RWG	11	14	8	11	9	10	7	15	85
Uniport	0	2	3	4	5	6	0	1	21
WBT	0	1	2	3	3	3	5	0	17
Total # Containers	195	174	191	246	252	240	253	212	1762
Total TEU	308	284	307	414	412	403	415	348	2891

Table 2: Number of arriving containers per terminal and ETA and total TEU per ETA.

Transport	Barges	Max TEU	Max Weight	Call	Total TEU	Total Weight
Day 1	D,E	475	7500000	2	111	1294583
Day 2	A,C	460	7500000	3	195	2295840
Day 3		0	0	4	326	3446186
Day 4	B,E	450	6750000	5	315	3264205
Day 5	C,F	360	6000000	6	359	3866595
Day 6	A,D	475	7750000	7	391	3960879
Day 7	B	200	3000000	8	407	4309959
Day 8	D	225	3750000	9	431	4516530
Day 9	B	200	3000000	10	259	2734121
Day 10		0	0	11	97	1031933
Total		2845	45250000		**2891**	30720831

Table 3: Available Barges for each possible day of transportation along with its max TEU and Weight. Call dates with their total TEU and Weight for the initial data set.

Table 3 depicts the available barges per day, along with the total amount of available TEU and weight per day. The table also includes the amount of TEU and weight that need to be transported for each call date. It can be observed that for the days 7, 8, 9 only one barge is available. However the last day that new containers arrive is day 6 (max ETA is 6). Hence, these containers have to be distributed onto the barges that arrive at days 7, 8 and 9. It can be seen that for days 7, 8 and 9 there is only one available barge (B or D). This is likely due to the fact that from day 7 onward there are no new arriving containers.

The instances where the total TEU and weight for transportation exceed the maximum capacity are shown in bold. Note that the grand total of TEU for transportation exceeds the maximum capacity only by 46 TEU, while the total weight is less than the maximum capacity of the barges. This means that with optimal planning, the vast majority of the containers can be transported by barges. An interesting observation is that the sum of the TEU of containers with ETA value -1, 0 or 1 equals 899 (as can be seen in table 2). This amount is less than the available TEU of the available barges for day 1 and 2 (935 available TEU). Having this in mind, it is to be expected that in the optimal solution for days 1 and 2, the available barges will not be full, since there will not be more available containers to add.

In order to have a sharper lower bound than the one mentioned in the Theory section for the objective function, we can calculate the costs of transporting the available maximum TEU in barges and the rest of TEU by truck, using the minimum number of trucks. From table 3 it can be seen that the total amount of TEU need to be transported equals 2891, while the maximum TEU in the barges equals 2845. Hence, if 2845 TEU are transported by barge, there are 46 TEU left, that need to be transported by truck. In order to find the minimum number of trucks that would be needed in this case, we can assume that the 6 containers of size 3 TEU will be transported by truck (18 TEU in total). Thus, 28 TEU remain to be transported by truck. In this case, 14 trucks can be used to transport 2 TEU containers. Thus, we conclude that the minimum possible costs are:

$$
\begin{aligned}
\text{Lower bound} &= \text{Cost by barge} \cdot \text{Available TEU by barge} + \text{Cost by Truck} \cdot \text{minimum Trucks} \\
&= 40 \cdot 2845 + 225 \cdot (18 + 14) = 121000
\end{aligned}
$$

Hence, the optimal costs cannot possibly be lower than this lower bound, since it is impossible to have less costs than transporting everything possible through barge and the remaining via truck, since that is the cheapest option.

The first instance to check, contains the given barge and containers data sets. The results of the model are now used to check if all the constraints are fulfilled and also to evaluate the quality of the findings.

Weight and TEU

In order to check if the weight and TEU constraints are met, total used TEU and weight for each barge per day are shown in tables 4 and 5 along with the corresponding maximum capacities. The times where a barge has reached its maximum capacity are shown in bold. It can be noticed that the barges have reached their maximum TEU capacities several times, however none of the barges has reached its maximum weight capacity. This agrees with the findings in table 2 where it is shown that the total weight to be transported is less than the maximum available capacity, something that does not hold for the TEU. Furthermore, in table 3 it is observed that the amount of TEU to be transported for days 1 and 2 is less than the maximum available capacity of the barges and hence, the barges in the first two days are not full, as it is shown in table 4.

	Barge A	Barge B	Barge C	Barge D	Barge E	Barge F
Day 1	0	0	0	172	248	0
Day 2	234	0	**210**	0	0	0
Day 3	0	0	0	0	0	0
Day 4	0	**200**	0	0	**250**	0
Day 5	0	0	**210**	0	0	136
Day 6	**250**	0	0	**225**	0	0
Day 7	0	**200**	0	0	0	0
Day 8	0	0	0	**225**	0	0
Day 9	0	184	0	0	0	0
Day 10	0	0	0	0	0	0
Max TEU	250	200	210	225	250	150

Table 4: Amount of TEU used in each barge per day. Full barges in bold.

	Barge A	Barge B	Barge C	Barge D	Barge E	Barge F
Day 1	0	0	0	1970510	2734291	0
Day 2	2455478	0	2338803	0	0	0
Day 3	0	0	0	0	0	0
Day 4	0	2236113	0	0	2766936	0
Day 5	0	0	2077071	0	0	1311281
Day 6	2807939	0	0	2466486	0	0
Day 7	0	1893440	0	0	0	0
Day 8	0	0	0	2539233	0	0
Day 9	0	1838896	0	0	0	0
Day 10	0	0	0	0	0	0
Max Weight	4000000	3000000	3500000	3750000	3750000	2500000

Table 5: Total weight used in each barge per day.

The weight constraint so far has not been binding for the given instance. This is not a proper check in this way so we modify the original instance and triple the weight for every container. The results are as expected and shown in table 6 and table 7. Now the weight constraint is binding most of the times and the TEU constraint is not. Note that tripling the weight was quite an excessive change and a lot of containers went by truck as a result.

	Barge A	Barge B	Barge C	Barge D	Barge E	Barge F
Day 1	0	0	0	73	147	0
Day 2	140	0	130	0	0	0
Day 3	0	0	0	0	0	0
Day 4	0	73	0	0	155	0
Day 5	0	0	184	0	0	97
Day 6	183	0	0	114	0	0
Day 7	0	150	0	0	0	0
Day 8	0	0	0	117	0	0
Day 9	0	87	0	0	0	0
Day 10	0	0	0	0	0	0
Max TEU	250	200	210	225	250	150

Table 6: Amount of TEU used in each barge per day.

	Barge A	Barge B	Barge C	Barge D	Barge E	Barge F
Day 1	0	0	0	2073795	**3724908**	0
Day 2	**3941991**	0	**3448971**	0	0	0
Day 3	0	0	0	0	0	0
Day 4	0	2021211	0	0	**3700689**	0
Day 5	0	0	**3496965**	0	0	2262792
Day 6	**3972471**	0	0	**3702861**	0	0
Day 7	0	**2994837**	0	0	0	0
Day 8	0	0	0	**3624669**	0	0
Day 9	0	**2969739**	0	0	0	0
Day 10	0	0	0	0	0	0
Max Weight	4000000	3000000	3500000	3750000	3750000	2500000

Table 7: Total weight used in each barge per day.

Call Date and ETA

10 randomly selected containers alongside the 6 containers of size 3 TEU are visualized in table 8. For all the containers it can be observed that the transportation day is always between ETA + 1 and the call date. The container with ID 14 has call date 2 which means that it can only go by barge on day 1, however this container is shipped by truck. This is due to the fact that this container comes from the small BCW terminal (table 2) which is ”closed” on day 1. Furthermore, it can be seen that 5 out of 6 containers with size 3 TEU are shipped by truck. This makes sense since it is preferable to send containers with small TEU sizes by barge, as in this way the barge can ship more containers. However, the container with ID 1315 is shipped by barge E on day 1, since it comes from ECT Delta which is ”open” on this day and also barge E is not full on that day.

ID	TEU	Weight	ETA	Call date	Terminal	Mode	Transport Day
14	2	20655	-1	2	BCW	Truck	Day 2
44	2	22950	0	4	ECT Delta	Barge E	Day 1
147	2	14600	5	9	ECT Delta	Barge A	Day 6
211	1	23440	4	8	ECT Euromax	Barge F	Day 5
232	2	12686	-1	4	RWG	Barge C	Day 2
265	2	17431	0	4	AMPT	Barge D	Day 1
271	2	11700	4	8	ECT Delta	Barge A	Day 6
285	2	7588	4	9	ECT Delta	Barge A	Day 6
1173	2	24200	5	10	APMT2	Barge D	Day 8
1312	3	29027	2	5	RWG	Truck	Day 4
1313	3	29027	3	6	APMT2	Truck	Day 4
1314	3	29027	4	7	ECT Delta	Truck	Day 6
1315	3	29027	0	3	ECT Delta	Barge E	Day 1
1316	3	29027	5	8	AMPT	Truck	Day 7
1317	3	29027	2	7	AMPT	Truck	Day 4
1442	1	2300	5	8	ECT Euromax	Barge B	Day 7

Table 8: Schedule of random containers to verify the ETA and call date constraints.

Terminal Visits

In order to have a good trade-off between the terminal visits and the total costs, certain congestion costs are imposed at each terminal for each day. The desired result is to prevent a barge to visit too many terminals in one day since then there might be a great delay in the terminals which might result to a delay in the planning. On the other hand, the number of visited terminals should not be too low, since then the transportation and the demurrage costs are higher due to more trucks. The congestion costs function that is imposed at terminal t for barge j in day k is the following:

$$c_{tjk} = \text{barges available on day } k(f \cdot \text{capacity barge } j + a \cdot \# \text{ containers at terminal } t)$$

where f and a are some constants. The intuition behind this cost function is that a barge with larger capacity should cost more to travel and also a terminal with a larger amount of containers should have more loading costs. The reason that the available barges per day are also included is to allow for a relaxation in the costs when there is only one barge available, meaning that the costs will be lower when there is one barge available, and higher when there are more available barges, so as to have a better balance in the terminal visits. In order to determine the constants f and a several variations were tested and the constants that resulted to the most balanced amount of terminal visits were selected. The amount of terminals visited by each barge per day for $f = 2$ and $a = 1.5$ can be seen in the table below.

	Barge A	Barge B	Barge C	Barge D	Barge E	Barge F	Total
Day 1	0	0	0	1	1	0	2
Day 2	2	0	5	0	0	0	7
Day 3	0	0	0	0	0	0	0
Day 4	0	3	0	0	1	0	4
Day 5	0	0	1	0	0	4	5
Day 6	1	0	0	2	0	0	3
Day 7	0	6	0	0	0	0	6
Day 8	0	0	0	4	0	0	4
Day 9	0	2	0	0	0	0	2
Day 10	0	0	0	0	0	0	0

Table 9: Number of terminals visited by each barge per day case 1

The same table is also presented for the same constants but without taking into account the number of available barges per day in order to conduct a comparison between the two cost functions. The amount of terminals visited by each barge per day with cost function $c_{tjk} = 2 \cdot \text{capacity barge } j + 1.5 \cdot \#\text{ containers at terminal } t$ is presented below. For convenience, we name the case with cost function that takes into account available barges per day as *case 1* and the case with cost function that does not include available barges per day as *case 2*.

	Barge A	Barge B	Barge C	Barge D	Barge E	Barge F	Total
Day 1	0	0	0	3	2	0	5
Day 2	1	0	8	0	0	0	9
Day 3	0	0	0	0	0	0	0
Day 4	0	4	0	0	1	0	5
Day 5	0	0	1	0	0	5	6
Day 6	1	0	0	3	0	0	4
Day 7	0	4	0	0	0	0	4
Day 8	0	0	0	3	0	0	3
Day 9	0	2	0	0	0	0	2
Day 10	0	0	0	0	0	0	0

Table 10: Number of terminals visited by each barge per day case 2

From a comparison between tables 9 and 10 it can be seen that in case 2, for the days when there are two barges available the total amount of terminal visits is higher than in case 1. More specifically, in table 10 on day 2 barge C visits 8 terminals and barge A only one while in table 9 barge A visits 2 terminals and barge C visits 5. Allowing barge C to visit 8 terminals might result to a great delay to the schedule due to congestion in the terminals. It is be possible to lower the number of terminal visits in case 2 by having larger constants f and a, however it will result to a very low number of terminal visits for the days when one barge is available. As it can be seen, in table 10 for days 7 to 9 when there is one barge available, the amount of terminal visits is lower than in table 9, something that results to more trucks for these days. For these two cases, the amount of trucks used per day along with the total costs are provided in table 11.

	# Trucks in case 2	# Trucks in case 1
Day 1	0	11
Day 2	1	5
Day 3	13	23
Day 4	8	19
Day 5	15	8
Day 6	8	10
Day 7	8	1
Day 8	0	2
Day 9	0	0
Day 10	0	0
Total Trucks	53	79
Total Costs	128275	133285

Table 11: Amount of trucks and total costs for case 1 and 2

In table 11 it is shown that the total number of trucks used in the planning in case 1 is higher than case 2 and due to this, the total costs are also higher. However, for the days 7 to 9 when there is one barge available, case 1 uses less trucks since there are more visited terminals for these days. Due to the aforementioned, we can argue that the cost function of case 1 (the one that takes into account the available barges per day) provides a better balance between the amount of terminal visits and the total costs, since in this case the maximum number of visited terminals by a barge is 5 when there are two barges available and 6 when there is one barge available, while for case 2 these numbers are 8 and 4 respectively. Conclusively, since a barge visiting 8 different terminals might result to a delay in the schedule, it is preferable to have a bit higher costs than delayed deliveries. In order to have a more detailed look in the functionality of the congestion costs, table 12 depicts which specific terminals are visited by each barge each day.

	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9
APMT	D	A		B	C	D		D	B
APMT2		C		B		D	B	D	
BCW		C		B			B		
ECT City		C			F		B		
ECT Delta	E	A		E		A	B	D	B
ECT Euromax		C			F		B		
RWG		C			F			D	
Uniport					F				
WBT							B		

Table 12: Terminals visited by each barge for each day. The letters denote the corresponding barge.

It can be seen that APMT and ECT Delta which are the terminals with the largest amount of containers according to table 2, are the most frequently visited. On the other hand, Uniport and WBT which are the less busy terminals, are only visited once. Furthermore, on day 2 barge A visits the two largest terminals APMT and ECT Delta, while barge C visits 5 smaller terminals. Also, on day 7 when one barge is only available, this barge visits ECT Delta and 5 smaller terminals, while on day 8 barge D visits both big terminals, but only two smaller. Both big terminals are also visited on day 9 with barge B, however this does not visit any other terminals than these. This is due to the fact that on day 9 barge B after filling up with containers from APMT and ECT Delta, has only 16 TEU left before it is full, hence it is cheaper to send the remaining containers by truck, than pay the congestion costs for visiting one more terminal.

Extreme Congestion Costs

In order to show the impact of the terminal visits to the total costs, two extreme scenarios were tested: one with zero congestion costs (Scenario 1) and one with extremely high congestion costs (Scenario 2). More specifically, for the first scenario constants f and a were set to zero, while for the second scenario they were both set to 10000. Table 13 summarizes the amount of trucks and the total costs for each scenario, while table 14 depicts the used capacity per barge and per day when there is no limit to terminal visits.

	# Trucks Scenario 1	# Trucks Scenario 2
Day 1	0	201
Day 2	0	205
Day 3	15	204
Day 4	11	251
Day 5	4	240
Day 6	2	272
Day 7	6	220
Day 8	1	119
Day 9	0	47
Day 10	0	3
Total Trucks	39	1762
Total Costs	123665	397020

Table 13: Number of trucks and total costs when congestion costs are zero (scenario 1) and maximum (scenario 2).

In table 13 it can be seen that in the scenario without congestion costs the minimum amount of trucks is used (39), since the vast majority of the containers is shipped by barges, something that is visible in table 14 where it is shown that almost all the barges are full. Also all the terminals are "open" in this scenario. On the other hand, if the congestion costs are too high, then all the terminals are "closed", meaning that it is preferable to send every container by truck instead of using barges. This is the reason that in scenario 2, all 1762 containers were shipped by truck.

	Barge A	Barge B	Barge C	Barge D	Barge E	Barge F
Day 1	0	0	0	207	241	0
Day 2	244	0	207	0	0	0
Day 3	0	0	0	0	0	0
Day 4	0	**200**	0	0	**250**	0
Day 5	0	0	**210**	0	0	**150**
Day 6	**250**	0	0	**225**	0	0
Day 7	0	**200**	0	0	0	0
Day 8	0	0	0	**225**	0	0
Day 9	0	199	0	0	0	0
Day 10	0	0	0	0	0	0
Max TEU	250	200	210	225	250	150

Table 14: Amount of used TEU per barge and per day for the scenario with zero congestion costs.

In table 13 it can also be seen that when the barges are able to visit any terminal the costs are 123365, while the theoretical lower bound is 121000. The fact that the costs in scenario with zero congestion costs are only 2365 more than the lower bound verifies the efficiency of the model. On the other hand, the total costs in scenario 2 are 397020 implying some 'upper bound' for total costs of this specific data set. It should be noted here that these costs are a bit higher than the 'upper bound' which was mentioned in the Theory section, which is the following:

$$\begin{aligned} \text{'Upper Bound'} &= \text{Cost by truck} \cdot \text{total number of containers that needs to be shipped} \\ &= 1762 \cdot 225 = 396450 \end{aligned}$$

The reason that the costs in scenario 2 are higher than the aforementioned Upper Bound, is due to the containers depicted in table 15. These containers arrive at day -1 and have demurrage free period of 2 days, which means that they have to pay demurrage costs since day 1. It can be seen that the 1 TEU containers cost 275, which is 50 for demurrage and 225 for truck transport, while 2 TEU containers cost 70 for each day that exceeds demurrage free period.

ID	TEU	ETA	Call date	Demurrage free	Transport Day	Costs
225	2	-1	2	2	Day 1	295
333	2	-1	2	2	Day 1	295
549	1	-1	2	2	Day 1	275
575	2	-1	4	2	Day 1	295
874	1	-1	3	2	Day 1	275
1394	2	-1	4	2	Day 1	295
1612	1	-1	3	2	Day 1	275
1636	2	-1	4	2	Day 1	295
1736	2	-1	3	2	Day 1	295

Table 15: Containers which in day 1 have exceeded their demurrage free period.

$$\text{'Upper Bound'} = \text{Theoretical 'Upper Bound'} + \text{Demurrage Costs}$$

$$\text{'Upper Bound'} = 396450 + 70 \cdot 6 + 50 \cdot 3 = 397020$$

Finally, it should be noted that instances with less available terminals were also tested. In these instances the ILP worked properly as only the available terminals were used.

Large Data set

In order to verify that the model can also work properly with a different data set, a larger data set was created with the following specifications:

- 2147 containers
- 12 possible days of transportation
- Call dates from 1 to 13
- ETA from -1 to 11
- 8 barges: A, B, C, D, E, F, G, H
- On day 11 there are 3 available barges

Table 16 contains information concerning the larger data set. More specifically, it shows the available barges per transportation day along with the available TEU and weight in these barges. Furthermore, it depicts the amount of TEU and Weight for each call date. Also, it can be seen that the total TEU for transportation are a bit more than the available TEU while the total weight for transportation is less than the maximum available, similarly to the initial data set. The cost function used in this formulation is:

$$c_{tjk} = \text{barges available on day } k(2 \cdot \text{capacity barge } j + 1.5 \cdot \# \text{ containers at terminal } t)$$

Moreover, the order of the terminals when it comes to amount of containers in them is the same as in the initial data set.

	Barges	Available TEU	Available Weight	Call	Total TEU	Total Weight
Day 1	D, E	475	7500000	2	111	1294583
Day 2	A, C	460	7500000	3	195	2295840
Day 3		0	0	4	326	3446186
Day 4	B,E	450	6750000	5	315	3264205
Day 5	C,F	360	6000000	6	359	3866595
Day 6	A, D	475	7750000	7	391	3960879
Day 7	B	200	3000000	8	409	4318659
Day 8	D	225	3750000	9	459	4783086
Day 9	B	200	3000000	10	310	2949221
Day 10	C	210	3500000	11	294	2819176
Day 11	F,G,H	370	6700000	12	145	1614674
Day 12		0	0	13	215	2265392
Total		3425	55450000		**3529**	36878496

Table 16: Available Barges for each possible day of transportation along with its max TEU and Weight. Call dates with their total TEU and Weight for the large data set.

Table 17 contains the amount of the TEU capacity used in each barge for each day, along with the corresponding maximum capacities. The full barges are shown in bold. It can be observed that the majority of the barges reached its full capacity, while the rest of the barges have a few TEU left. The only barge that has a lot of TEU left is barge D on day 1, due to the fact that it only visited the APMT terminal, which is the second largest, while barge E visited ECT Delta, which is the terminal with the most arrivals. It is interesting to note here, that the reason that Barge E visited the largest terminal is due to the fact that it has larger capacity than barge D. In addition, it should be noted that every time that ECT Delta was visited it was by the largest available barge for this day.

	Barge A	Barge B	Barge C	Barge D	Barge E	Barge F	Barge G	Barge H
Day 1	0	0	0	161	**250**	0	0	0
Day 2	244	0	209	0	0	0	0	0
Day 3	0	0	0	0	0	0	0	0
Day 4	0	**200**	0	0	**250**	0	0	0
Day 5	0	0	**210**	0	0	136	0	0
Day 6	**250**	0	0	**225**	0	0	0	0
Day 7	0	**200**	0	0	0	0	0	0
Day 8	0	0	0	**225**	0	0	0	0
Day 9	0	**200**	0	0	0	0	0	0
Day 10	0	0	**210**	0	0	0	0	0
Day 11	0	0	0	0	0	0	98	98
Day 12	0	0	0	0	0	0	0	0
Max TEU	250	200	210	225	250	150	100	120

Table 17: Amount of TEU used in each barge per day for the large data set.

	Barge A	Barge B	Barge C	Barge D	Barge E	Barge F	Barge G	Barge H
Day 1	0	0	0	1	1	0	0	0
Day 2	2	0	5	0	0	0	0	0
Day 3	0	0	0	0	0	0	0	0
Day 4	0	3	0	0	1	0	0	0
Day 5	0	0	2	0	0	4	0	0
Day 6	1	0	0	1	0	0	0	0
Day 7	0	5	0	0	0	0	0	0
Day 8	0	0	0	3	0	0	0	0
Day 9	0	4	0	0	0	0	0	0
Day 10	0	0	6	0	0	0	0	0
Day 11	0	0	0	0	0	0	2	1
Day 12	0	0	0	0	0	0	0	0

Table 18: Number of terminals visited by each barge for the large data set.

Table 18 includes the number of terminals that were visited by each barge per day. The total number of visited terminals per day varies from 2 to 7 when there are two available barges and from 3 to 6 when there is one available. It needs to be noted here that, on day 11 there are 3 available barges, F,G and H, however only G and H are used, since all the terminals for barge F on that day are "closed". Note here that barge F is the one with the largest capacity between these three barges, something that means its congestion costs are more than the other two barges. The reason that barge F was not used on this day, is due to the fact that there were only 22 containers which could be transported by barge, however they were from different terminals as it can be seen in table 19. Hence, it is cheaper to send these containers by truck, than pay the congestion costs for a barge to visit another terminal to collect only a few containers. It should also be noted that the on day 11 barge G visited APMT and ECT Euromax, while barge H visited ECT Delta. Note that these terminals are not listed in table 19 since there were not any containers left in them.

ID	TEU	ETA	Call date	Terminal	Demurrage free	Mode	Transport Day
1857	1	10	13	ECT City	5	Truck	Day 11
1861	1	10	13	APMT2	5	Truck	Day 11
1870	1	10	12	BCW	4	Truck	Day 12
1882	1	10	12	RWG	5	Truck	Day 11
1884	1	10	12	APMT2	4	Truck	Day 11
1889	1	10	12	BCW	3	Truck	Day 11
2084	1	10	13	RWG	5	Truck	Day 11
2085	2	10	13	APMT2	4	Truck	Day 11
2087	2	10	13	Uniport	4	Truck	Day 11
2095	2	10	13	APMT2	4	Truck	Day 12
2101	2	10	13	APMT2	3	Truck	Day 11
2103	2	10	13	APMT2	3	Truck	Day 12
2109	2	10	13	RWG	3	Truck	Day 11
2110	1	10	13	APMT2	3	Truck	Day 12
2112	1	10	13	WBT	3	Truck	Day 12
2113	2	10	13	ECT City	4	Truck	Day 11
2115	2	10	13	ECT City	4	Truck	Day 11
2116	2	10	13	RWG	3	Truck	Day 12
2124	1	10	13	RWG	5	Truck	Day 12
2135	3	10	13	APMT2	3	Truck	Day 11
2136	3	10	13	AMPT	4	Truck	Day 11
2137	2	10	13	WBT	4	Truck	Day 11

Table 19: Containers with call dates 12 and 13 that transported by truck.

Table 20 includes the number of trucks used for each day along with the total costs (transportation and demurrage costs). The days with the most trucks are days 3 and 12 for which there is no barge available. It should be added here that for day 12, 46 out of 53 containers that were moved by truck have ETA 11 and hence, they could not be transported by a barge on day 11.

	# Trucks
Day 1	8
Day 2	10
Day 3	31
Day 4	8
Day 5	18
Day 6	17
Day 7	20
Day 8	8
Day 9	8
Day 10	9
Day 11	19
Day 12	53
Total Trucks	209
Total Costs	177965

Table 20: Number of trucks used per day along with total costs.

Manual Adjustments

In order to verify the functionality of the manual adjustments, a smaller data set that includes the first 299 containers from the initial data set was created. Furthermore, the congestion costs were set to 0 in order for all the terminals to be "open", so as to verify that when the user selects that a terminal should no be visited, then it should be the only closed terminal. Furthermore, since the barge schedule is the initial one, and there are only 299 containers in this data set, none of the containers is transported by truck as the total capacity of the barges is more than the total TEU and weight need to be transported. The optimal planning without any manual adjustment for the first 8 containers is shown in table 21, in order to compare this with the planning after the manual adjustments.

ID	TEU	Weight	ETA	Call date	Terminal	Mode	Transport Day
1	1	23000	0	4	ECT Delta	Barge E	Day 1
2	1	23000	2	6	APMT2	Barge B	Day 4
3	2	25830	2	7	WBT	Barge B	Day 4
4	1	18715	5	9	AMPT	Barge A	Day 6
5	1	8737	4	7	ECT Delta	Barge C	Day 5
6	2	22929	0	5	AMPT	Barge E	Day 1
7	2	22929	4	8	ECT Delta	Barge C	Day 5
8	2	26482	2	5	ECT Delta	Barge E	Day 4

Table 21: Container shipping information before the manual adjustments.

In table 22 can be seen several possible manual adjustments for some of the containers, along with the expected possible error according to the criteria of the adjustments. For example, the user can choose container 1 to be shipped by barge C on day 5, however, as it can be seen in table 21, the call date for the container 1 is day 4. Thus, it is not feasible to add this constraint and an error message should warn the user that this manual adjustment is not possible. On the other hand, the user can also select for the container 3 to be shipped by barge E on day 4. Hence, after the planning this container should be shipped by barge E, instead of barge B that is initially used as it can be seen in table 21. In this case no error message will be displayed.

Container ID	Day	Mode	Expected Error
1	5	Barge C	After Call Date
2	1	Truck	Before ETA
3	4	Barge E	N/A
4	9	Barge B	On Call date
5	6	Barge C	Barge not available
7	7	Barge B	N/A
8	4	Truck	N/A

Table 22: Manual adjustments for the containers.

Table 23 contains two possible user adjustments concerning the terminals. More specifically, barge D on day 1 should not visit ECT Delta. This should force the model to set the decision variable for this terminal on this day to be 0, by adding an extra constraint for that. Note that since there are no congestion costs in this instance, all the terminals should be open, except of ECT Delta on day 1. A second adjustment is for barge F to visit ECT Euromax on day 3, however since there are no available barges on that day, the program should return an infeasible error.

Barge	Day	Action	Terminal	Expected Error
D	1	not visit	ECT Delta	N/A
F	3	visit	ECT Euromax	Barge not available

Table 23: Manual adjustments for the terminals.

After setting the aforementioned adjustments, the solution of the model for the first 8 containers is provided in table 24. The changes in the planning compared to table 21 are shown in bold. As it can be seen, container 3 was shipped by barge E instead of barge B, container 7 was shipped by barge B on day 7, instead of barge C on day 5 and finally container 8 was shipped by truck instead of barge E. The aforementioned verify the functionality of the manual adjustments concerning the containers.

ID	TEU	Weight	ETA	Call date	Deep sea terminal	Mode	Transport Day
1	1	23000	0	4	ECT Delta	Barge E	Day 1
2	1	23000	2	6	APMT2	Barge B	Day 4
3	2	25830	2	7	WBT	**Barge E**	Day 4
4	1	18715	5	9	AMPT	Barge A	Day 6
5	1	8737	4	7	ECT Delta	Barge C	Day 5
6	2	22929	0	5	AMPT	Barge E	Day 1
7	2	22929	4	8	ECT Delta	**Barge B**	**Day 7**
8	2	26482	2	5	ECT Delta	**Truck**	Day 4

Table 24: Container shipping information after the manual adjustments.

After the planning has finished, the program displays the infeasible error messages for the constraints that could not be met, as it can be seen in figures 25, 26 and 27. The order in which the messages appear is the same as the order of the corresponding manual adjustments. Figure 25 shows the errors for containers 1 and 2. More specifically, since the call date for container 1 is day 4, the manual adjustment is not able to be implemented and an error message informs the user about that. Similarly, container 2 arrives in Rotterdam on day 2, hence it is not able to be shipped on day 1 and the corresponding error message appears. Moreover, a comparison between the tables 21 and 24 shows that the planning for these two containers after the adjustments is the same as before the manual adjustments.

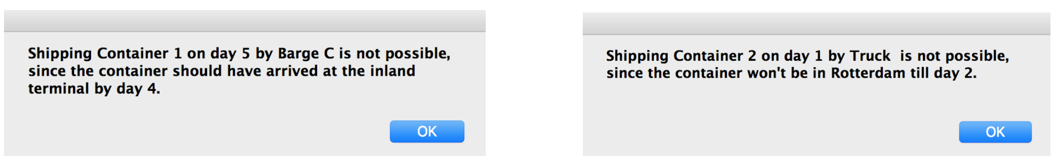


Figure 25: Error Messages for containers 2 (left) and 4 (right)

The error messages for containers 4 and 5 are shown in figure 26. Container 9 has call date 9, hence it is not possible to be shipped by barge on that day, as the barge needs one day to arrive at the inland terminal. Similarly, on day 6 only barges A and D are available and thus, shipping container 5 by barge C on that day is not possible, since this barge is not available on that day.

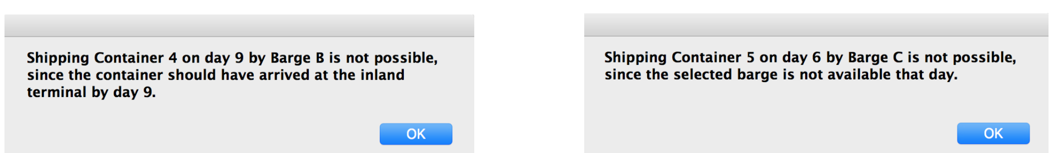


Figure 26: Error Messages for containers 4 (left) and 5 (right).

When it comes to the terminal manual adjustments, barge F is not possible to visit on day 3 ECT Euromax, since on this day the barge is not available, and the corresponding error message can be seen in figure 27.

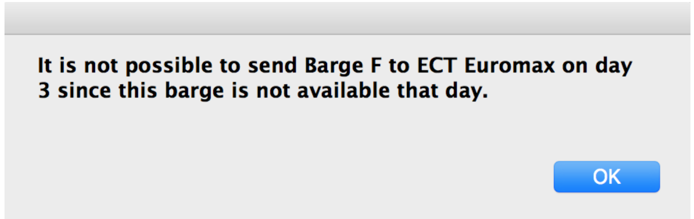


Figure 27: Error message for barge F.

Furthermore, the last adjustment which says that barge D should not visit ECT Delta on day 1 was implemented successfully. This can be verified by the snapshot of the ILP model in figure 28, where it can be seen that ECT Delta is the only terminal for that day for which the decision variable is 0. It is noted here, that the decision variables for the rest of the terminals are 1, as there were no congestion costs imposed in this instance.

	Congestion Costs Decision Variables							
Day	1	1	1	1	1	1	1	1
Barge	4	4	4	4	4	4	4	4
Terminal	AMPT	APMT2	BCW	ECT City	ECT Delta	ECT Eurom:	RWG	Uniport
Decision Variables	≤ 1	≤ 1	≤ 1	≤ 1	≤ 0	≤ 1	≤ 1	≤ 1
Costs incurred	0	0	0	0	0	0	0	0
Loading Costs incurred	0	0	0	0	0	0	0	0
Travel Costs incurred	0	0	0	0	0	0	0	0
Total Costs	0	0	0	0	0	0	0	0
Loading Costs	0	0	0	0	0	0	0	0
Travel Costs	0	0	0	0	0	0	0	0

Figure 28: Snapshot from the ILP model's Excel sheet, where it is shown that ECT Delta is the only "closed" terminal.

Verification Heuristic

Similarly to the verification of the ILP, several instances were tested in order to check whether all the requirements are met or not.

When a planning for each day is created, the solution will be verified. To check if the given solution is feasible, there is a built-in verifier that runs a couple tests automatically (SUB VERIFY()). For a feasible solution, there are several conditions that need to be satisfied. These conditions are:

1. Containers should be transported after the arrival day.
2. Containers should be transported before call date when loaded on a barge.
3. Containers should be transported at most at call day or before when delivered by truck.
4. Barges are not overloaded: checks maximum TEU and maximum weight.
5. All containers should be transported exactly once.

If one of these constraints is not satisfied, the solution is not feasible anymore. Hence, it is an important part of the Heuristic to verify these constraints.

Initial Data set

Using the given data set, with the containers information and the availability of the barges. The constraints are verified, the verification is done day by day.

Call date and ETA

In order to verify some of the conditions, the shipment day for each container is compared with the arrival date (ETA) plus one day. Likewise, a similar procedure is executed for the call date minus one day. These procedures are executed no matter what the mode of transportation is.
Some randomly picked containers are displayed in Table 25. It is noticeable that all of them met the three first conditions described abovefor the day of transportation.

ID	TEU	Weight	ETA	Call date	Terminal	Mode	Transport Day
117	2	10305	1	6	ECT Delta	Barge A	Day 2
176	1	18890	0	4	APMT2	Barge E	Day 1
212	1	23440	5	9	ECT Euromax	Barge D	Day 6
216	1	23440	3	6	ECT Delta	Barge C	Day 5
232	2	12686	-1	4	RWG	Barge E	Day 1
375	1	23600	6	9	AMPT	Barge D	Day 8
419	2	4200	5	9	ECT Delta	Barge A	Day 6
526	1	14980	3	7	ECT Delta	Barge A	Day 6
844	1	23084	6	11	ECT Delta	Barge D	Day 8
893	2	30156	-1	4	AMPT	Barge E	Day 1
1115	2	24200	3	8	ECT Delta	Barge C	Day 5
1241	1	22300	-1	3	ECT Delta	Barge D	Day 1
1360	1	19664	3	6	APMT2	Truck	Day 6
1465	1	19740	4	9	ECT Delta	Barge A	Day 6
1606	2	7666	5	8	AMPT	Barge B	Day 7
1726	1	4664	2	5	ECT Delta	Barge B	Day 4

Table 25: Schedule of random containers to verify the ETA and call date constraints

Weight and TEU

The total scheduled TEU and weight are calculated per day and per barge and compared with the maximum capacities per barge in order to verify that the barges are not overloaded. Table 26 depicts a summary of the final solution. More specifically, it shows the Total Weight, TEU and cost of transportation per day and mode of transportation.

As it is shown in Table 26 none of the barges will be overloaded in any of the days, because in all of the cases the scheduled weight and scheduled TEU do not exceed the corresponding maximum capacity. Additionally, the barges that have met their maximum TEU capacities are shown in bold. Lastly, it should be noted that the maximum weight capacity has not been met in any of the days.

Report	Total Costs:	130785			
Day	Mode	Scheduled weight	Max Weight	Scheduled TEU	Max TEU
Day 1	Barge D	2730848	3750000	**225**	225
	Barge E	2631096	3750000	248	250
	Trucks	0			
	Costs	19490			
Day 2	Barge A	2536477	4000000	**250**	250
	Barge C	1876189	3500000	172	210
	Trucks	4			
	Costs	17780			
Day 3	Trucks	0			
	Costs	0			
Day 4	Barge B	2129017	3000000	**200**	200
	Barge E	2557690	3750000	249	250
	Trucks	0			
	Costs	18720			
Day 5	Barge C	2349684	3500000	**210**	210
	Barge F	1579572	2500000	149	150
	Trucks	7			
	Costs	15935			
Day 6	Barge A	2484072	4000000	**250**	250
	Barge D	2003691	3750000	223	225
	Trucks	17			
	Costs	22745			
Day 7	Barge B	2110081	3000000	199	200
	Trucks	13			
	Costs	10885			
Day 8	Barge D	2592572	3750000	224	225
	Trucks	11			
	Costs	11435			
Day 9	Barge B	1492214	3000000	148	200
	Trucks	35			
	Costs	13795			
Day 10	Trucks	0			
	Costs	0			

Table 26: Verify Weight and TEU constraints for the Heuristic method

Terminal Visits

In order to avoid congestion in the terminals, it is important that the days there are more than one barge available, each terminal be visited just by one barge. Table 27 includes the information of the days with more than one barge available, the terminal visited by each barge each day. There, it is noticeable that none of the terminals are visited by two barges in the same day.

In addition, it can be seen than in every day ECT delta and AMPT, which are the terminals with the largest amount of containers, are in different barges and the space left is filled up with the containers from other terminals.

Day 1		**Day 2**		**Day 4**		**Day 5**		**Day 6**	
Mode	Terminal	Mode	Terminal	Mode	Terminal	Mode	Terminal	Mode	Terminal
D	ECT Delta ECT Euro.	A	APMT2 ECT City ECT Delta RWG	B	ECT Delta Uniport WBT	C	APMT2 BCW ECT Delta Uniport	A	APMT2 BCW ECT Delta
E	AMPT APMT2 BCW ECT City RWG Uniport WBT	C	AMPT BCW ECT Euro. Uniport WBT	E	AMPT APMT2 BCW ECT City ECT Euro. RWG	F	AMPT ECT City ECT Euro. RWG WBT	D	AMPT ECT City ECT Euro. RWG Uniport WBT

Table 27: Terminal visited by each barche, the days there are more than two barges available

On the other hand, to verify that a container is shipped just one time we count the number of times the container ID is included in the final output and it should be only one, which it was for all test instances

Comparison ILP and Heuristic

In order to compare the ILP and the Heuristic approach, the total costs, the terminal visits as well as the running speed time need to be taken into account. However, a straightforward comparison between the two methods is a tricky task, due to the fact that each approach has a different way to balance the terminal visits and the total costs. ILP adds penalty congestion costs to the objective function in order to prevent barges from visiting too many terminals, while the Heuristic approach swaps containers between the barges after the planning of each day is done. In this way, the Heuristic method does not lower the total amount of visited terminals per day, however succeeds in keeping the number of terminals visited by each barge at relatively low levels.

A way to make the comparison reliable is by comparing the Heuristic approach with two different instances of the ILP: an instance where the congestion costs are unaccounted for, disabling the corresponding constraints and decision variables, meaning there is no limit to the terminal visits (instance 1) and another one where the congestion costs are given by the cost function that is mentioned earlier (instance 2):

$$c_{tjk} = \text{barges available on day } k(2 \cdot \text{capacity barge } j + 1.5 \cdot \# \text{ containers at terminal } t)$$

For each instance, the transportation and the demurrage costs will be compared along with the number of visited terminals per day. It needs to be noted that these instances refer to the initial data set.

Costs and Running Time

Firstly, the costs as well as the running time of each method are compared. The results can be seen in table 28.

	Heuristic	ILP instance 1	ILP instance 2
Transportation Costs	129455	121095	127535
Demurrage Costs	1330	2570	5800
Total Costs	130785	123665	133285
Running Time	<30 s	≈ 10 min*	≈ 20 min*

Table 28: Comparison of costs and running time between Heuristic and ILP. Running time of the ILP is an average.

In this table it can be seen that when there are no congestion costs imposed in the ILP, the total costs are lower than the Heuristic by 7120. On the other hand, in the case that the ILP includes congestion costs, the total costs are higher than the Heuristic's costs by 2500. This difference in the results is due to the fact that in instance 1 barges visit more terminals than in the Heuristic method, while in instance 2 there are less terminal visits than the Heuristic. Another interesting observation in table 28 is that the demurrage costs for the Heuristic are the lowest between the 3 cases and the reason for this is that the demurrage free period is the first sorting criterion in the heuristic is the demurrage free period, then the call date and lastly the TEU. Hence, as the Heuristic prioritizes the demurrage free period it is reasonable for this method to provide lower demurrage costs than the ILP. At this point it is also worth mentioning that the transportation costs for the ILP in instance 1 are only 95 higher than the 121000 lower bound found in the Verification ILP section. This very small difference between the ILP and the lowest possible transportation costs verifies the performance of the method. Note that the heuristic's total costs seems to be within 5% of the optimal ILP solution's total costs on all the instances that we have tested.

When it comes to running times, the Heuristic is way faster than the ILP, since it runs in a few seconds, while the full ILP needs approximately 20 minutes. Without congestion the ILP runs in approximately 10 minutes, since the problem is less complex. It needs to be noted here, that the running time of the ILP depends on the computer speed to a great extent and hence the 20 minutes that are displayed in table 28 is an average of the running times in several different computers.

Trucks and Barges

Another interesting comparison between the two methods is the amount of trucks used in each occasion. The summary of the trucks used per day and per method can be seen in table 29.

# Trucks	Heuristic	ILP instance 1	ILP instance 2
Day 1	0	0	11
Day 2	4	0	5
Day 3	0	15	23
Day 4	0	11	19
Day 5	7	4	8
Day 6	17	2	10
Day 7	13	6	1
Day 8	11	1	2
Day 9	35	0	0
Day 10	0	0	0
Total Trucks	87	39	79

Table 29: Number of trucks used in the Heuristic and in the ILP for instances 1 and 2.

It can be noticed that the Heuristic method uses the most trucks compared to the two instances of the ILP. Furthermore, by comparing tables 28 and 29 it can be seen that there is a correspondence between the number of trucks used and the transportation costs for each method. This correspondence makes sense, as the transportation costs are dependent on the amount of trucks used in each occasion. It should be noted that while the ILP for instance 2 uses a lower amount of trucks than the Heuristic, its total costs are higher than the Heuristic. This happens because the ILP in instance 2 ships more containers by barge than the Heuristic, however some of the containers need to wait longer in the terminals, something that results to higher demurrage costs.

Table 30 provides a summary of the total TEU used in each barge per day for the Heuristic method. It is noticed that the barges use their full capacity only two times, however in the majority of the cases the barges are almost full. The corresponding table for instance 1 is table 14 while table 4 corresponds to instance 2. By comparing these three tables it can be seen that the Heuristic uses the least amount of the TEU capacities in the barges and this is the reason why this method uses the largest amount of trucks compared to the other two methods.

Terminal Visits

A key factor that affects directly the costs for each method is the amount of terminals visited by each available barge per day. Table 31 contains information about the terminals visited using the ILP for instance 1. The corresponding table for instance 2 is table 9.

Table 32 summarizes the amount of terminals visited by each barge for the Heuristic method. By comparing tables 31 and 32 it can be seen that for the days when two barges are available, the Heuristic provides a better classification of the containers in the barges as the total amount of terminals visited per day is much lower. This happens because when the ILP does not contain

	Barge A	Barge B	Barge C	Barge D	Barge E	Barge F
Day 1	0	0	0	224	249	0
Day 2	249	0	173	0	0	0
Day 3	0	0	0	0	0	0
Day 4	0	**200**	0	0	249	0
Day 5	0	0	209	0	0	**150**
Day 6	249	0	0	224	0	0
Day 7	0	199	0	0	0	0
Day 8	0	0	0	224	0	0
Day 9	0	148	0	0	0	0
Day 10	0	0	0	0	0	0
Max TEU	250	200	210	225	250	150

Table 30: Amount of TEU used in each barge per day for the Heuristic method.

	Barge A	Barge B	Barge C	Barge D	Barge E	Barge F	Total
Day 1	0	0	0	8	8	0	16
Day 2	8	0	9	0	0	0	17
Day 3	0	0	0	0	0	0	0
Day 4	0	8	0	0	9	0	17
Day 5	0	0	9	0	0	8	17
Day 6	9	0	0	9	0	0	18
Day 7	0	8	0	0	0	0	8
Day 8	0	0	0	8	0	0	8
Day 9	0	7	0	0	0	0	7
Day 10	0	0	0	0	0	0	0

Table 31: Amount of terminals visited each day by the available barges for ILP instance 1.

congestion costs, two barges can visit the same terminal in the same day, something that increases significantly the number of visited terminals per day. When it comes to the days with one barge available, it can be seen that the amount of terminals visited by the available barge per day is identical for the two methods.

	Barge A	Barge B	Barge C	Barge D	Barge E	Barge F	Total
Day 1	0	0	0	2	7	0	9
Day 2	4	0	5	0	0	0	9
Day 3	0	0	0	0	0	0	0
Day 4	0	3	0	0	6	0	9
Day 5	0	0	4	0	0	5	9
Day 6	3	0	0	6	0	0	6
Day 7	0	8	0	0	0	0	8
Day 8	0	0	0	8	0	0	8
Day 9	0	7	0	0	0	0	7
Day 10	0	0	0	0	0	0	0

Table 32: Amount of terminals visited each day by the available barges for the Heuristic.

By comparing tables 32 and 9 it can be noticed that the ILP which includes the congestion costs has lower numbers in terminals visited per day by the available barges. This is reflected by the higher demurrage costs for the ILP in instance 2 compared to the Heuristic. It is noticed that the Heuristic can keep the number of terminals visited by each barge at relatively low levels when there are two barges available. However, when only one barge is available, it visits all the terminals which have containers that need to be shipped, something that results to the relatively high number of visited terminals for days 7 to 9. On the other hand, in table 9 it can be seen that the corresponding numbers for the ILP in instance 2 are lower.

This inability of the Heuristic method to lower the total number of visited terminals per day is one of the disadvantages compared to the ILP, which is able to change the balance between the visited terminals and the total costs by changing the congestion costs function.

Generalization

Another aspect that can be compared between the two methods is at which extent each method can be generalized. In other words, for what changes in the input files each method allows. Table 33 summarizes the generalizations as well as the manual adjustments than can or cannot be made in each method.

	Heuristic	ILP
Number of Days	exactly 10	> 0
Number of Barges in one Day	≥ 0	≥ 0
Number of Total Barges	> 0	< 16
Add Terminal	Yes	No
Remove Terminal	Yes	Yes
Manual Adjustment Container	Yes	Yes
Manual Adjustment Terminal	No	Yes

Table 33: Comparison between the allowable generalizations in the two methods.

Conclusion

The goal of this project was to send a set of containers from a set of deep sea terminals to an inland terminal via some mode of transportation, respecting all types of constraints, namely capacity, shipping necessity and congestion possibilities. Ultimately, the objective was to drive down the total costs and number of terminal visits by barges and find this solution in an appropriate time frame. Two solutions were created, the ILP that solves the problem to optimality in a still relativy short time, and the heuristic that gives a close to optimal solution nearly instantly. Both solution methods are hardly sensitive to changes in input parameters. The decision support system has an enormously user friendly GUI, that flexibly allows for manual adjustments to the schedule and shows the output in an elegant way, alongside with sorting, searching, summarizing and reporting functionalities. The user manual provided in this report makes the use of this tool even simpler. Accompanied with the extensive documentation in this report, it is also trivial to adapt any other important parameters in the VBA code, a language that is widely used across business industries.

However powerful the tool is, there is always room for improvement. The ILP can be extended to allow for an arbitrary number of barges and more than 9 terminals should be able to be read in dynamically. The heuristic can be changed to not be sensitive towards having a different number of days. The way the terminal visits are handled by the heuristics can also be improved when it comes to 1 barge for a day. Manually added constraints that ensure a infeasible solution should display an error message before running the program instead of after. Terminal constraints for the heuristic are not an option as of now. Conflicting manually adjusted constraints are unaccounted for by the error messages currently. The change of objective function value could be displayed after the user tries to input a constraint manually. Any parameters that can be altered following the changing parameters sections in the documentation could be altered automatically in the GUI. An explicit version for the MacOs could be released and the user manual could support multiple languages.

References

[1] Richard M. Karp (1972) *Reducibility Among Combinatorial Problems* Retrieved from: https://people.eecs.berkeley.edu/ luca/cs172/karp.pdf (2017, Feb)

[2] Mason, A.J., OpenSolver An Open Source Add-in to Solve Linear and Integer Progammes in Excel, Operations Research Proceedings 2011, eds. Klatte, Diethard, Lthi, Hans-Jakob, Schmedders, Karl, Springer Berlin Heidelberg pp 401-406, 2012, http://dx.doi.org/10.1007/978-3-642-29210-1_64, http://opensolver.org (2017, Feb)