# Final project assignment: Simulation of a base scenario

Jef Jacobs
Toon Eeraerts
Wout Deleu

Semester 2

## Contents

**1 Introduction** **3**

**2 Probability distributions** **4**

2.1 Inter arrival time . . . . . 4

2.2 Container group size . . . . . 5

2.3 Service time . . . . . 6

2.4 Container type . . . . . 6

2.5 Arrival and departure points . . . . . 6

2.6 Flow type . . . . . 7

**3 Algorithm** **7**

3.1 Amount of simulations . . . . . 8

3.2 Flow . . . . . 8

3.3 Core . . . . . 9

3.4 Visualization . . . . . 10

**4 What-if scenarios** **13**

4.1 Base Scenario . . . . . 13

4.2 Smallest Remaining capacity . . . . . 14

4.3 Possible split ups . . . . . 15

4.4 Not allowing mixing container types in a single yardblock . . . . 16

**5 Performance** **17**

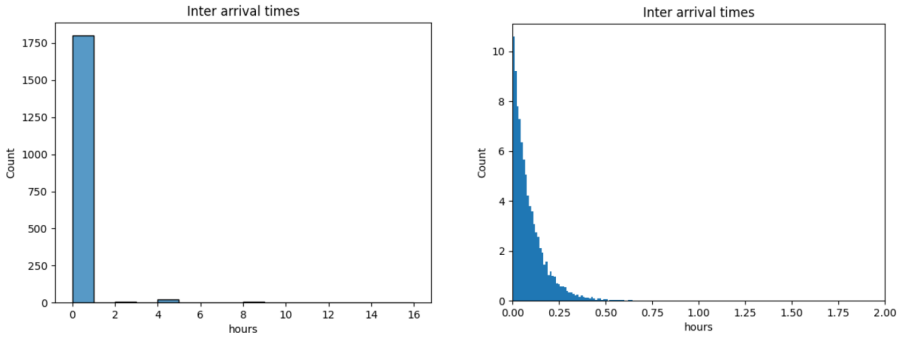
**6 Conclusion** **19**

# 1 Introduction

During this report, we will discuss the final assignment of the 'Yard Storage Assignment Problem'. Over the past labs, a significant amount of work and effort has gone into designing a simulation that can visualize a given port complex. The simulation focuses on the storage situation in the yard. In this report, we will discuss the basic design choices and the results of the simulation. Python is the primary technology used for the simulation. The code is written to be as dynamic as possible, utilizing global variables and booleans to vary parameters and control the overall flow of the simulation. It is based on a discrete event simulation, with the option for online simulation. To provide a brief recap, the goal is to simulate a yard where container groups arrive and depart. Containers arrive and leave in groups, and they are typically not considered individually. This means that they cannot be split up most of the time and are stored together in the same location. In the report, we aim to simulate the storage in the yard as a result of the inflow and outflow of containers.

To summarize, the focus of the first lab was on analyzing the existing data samples. The intention was to be able to sample them throughout the project to approximate reality. The objective of the second assignment was to set up a base scenario for simulation and obtain initial results. The samples obtained from previous sessions were converted into usable distributions and used to generate parameters for the container arrival and departure schedule. With a basic simulation scenario in place, the initial results could be examined. It is important to note that some minor errors occurred during the development and design of the sample functions. These errors stemmed from a misinterpretation of the graphs from the initial reports, resulting in a significant underestimation of the number of generated containers. This issue was also mentioned in the second report. However, while designing the final simulation, the mistake resurfaced but was promptly identified and resolved, no longer posing a problem.

The objective of this final assignment is to integrate all the components and make the simulation fully functional and usable. Three additional scenarios have been implemented to provide new insights into how the yard operates. Additionally, along with addressing the previous mistake, a visualization feature has been implemented. This visualization was one of the primary goals for this semester and serves as the finishing touch for this course. It brings the entire project together, making all the realizations visible, and provides genuine insights into the movements of the various elements within the yard.




(a) Real distribution (b) Sample distribution

Figure 1: Inter-arrival times

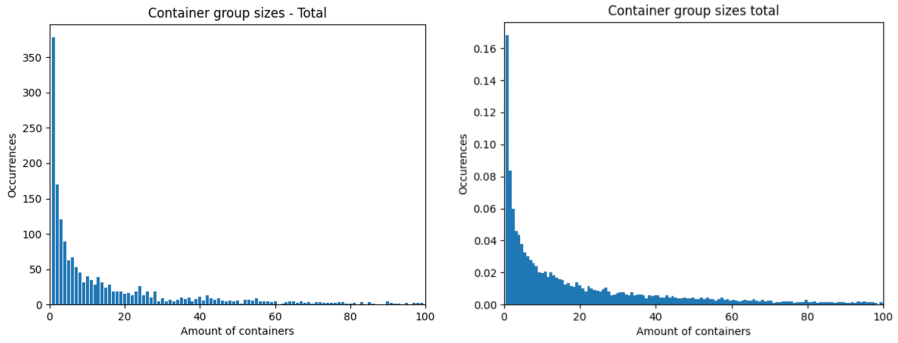
# 2 Probability distributions

The simulation makes use of probability distributions derived from real yard data provided by Ir. E. Thanos. Each distribution is carefully designed to yield the same average results as the input data. The simulation involves generating samples of container groups, each having distinct characteristics determined by these distributions.

## 2.1 Inter arrival time

The sampling frequency of container groups is a critical parameter in the simulation. In the original input data, container groups frequently arrived at the same time on the same vessel, leading to a high occurrence of zero inter-arrival times, shown in Figure 1a. Consequently, the average inter-arrival time has a very low value of 0.092 hours or 5 minutes and 30 seconds.

To ensure a representative distribution, this average value had to be present in our sample distribution. To represent the steepness of the graph, an exponential function was used, resulting in the distribution shown in Figure 1b.




(a) Real distribution (b) Sample distribution

Figure 2: Container group sizes

## 2.2 Container group size

The container group sizes in the input data indicate significant variance, ranging from single-container groups to groups with over 2000 containers. This strong variance caused a difficult decision in determining an appropriate distribution. In the end, a Weibull distribution was chosen to achieve a steep descent at the lower end while simultaneously allowing for the possibility of larger group sizes. The average size of container groups in the input data, as shown in Figure 2a, is 32 containers. The Weibull distribution used for this sample generation is shown in Figure 2b.

Based on the previous average outcome, we can ascertain the weekly generation of containers. This number corresponds to the quantity of containers being transported each week. The result is nearly identical to the number of containers from the input data, thereby validating the average values of the preceding distributions.

$$Generated\_containers = \frac{Total\_time}{inter\_arrival\_time} \cdot Average\_container\_group\_size \tag{1}$$

$$= \frac{168}{0.091} \cdot 32 \tag{2}$$

$$\approx 59000 \tag{3}$$

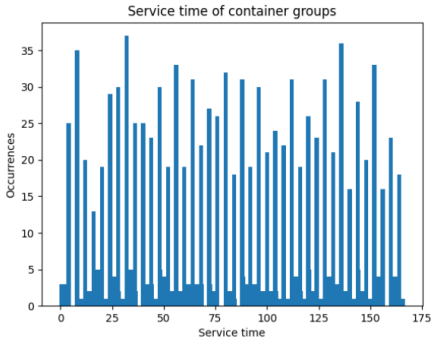



Figure 3: Service time sample distribution

## 2.3 Service time

The service time of each container group represents how long they stay in the yard before departing. The input data revealed a uniform distribution ranging from zero to 166 hours, shown in Figure 3. However, it is important to note that this distribution solely represents the container groups falling under transshipment. The other kind of container groups (export and import) always have a service time of 48 hours.

While it would be interesting to consider generating consistent container groups with a service time of 48 hours, we chose to simplify the distribution due to the larger number of transshipment container groups compared to the other types. The resulting distribution is a unfiorm distribution between zero and 166 hours.

## 2.4 Container type

Each container group can be of normal or reefer type. This distribution was based upon the input analysis of Nick De Bruyckere, Enrique Miron and Dries Van de Velde represented in Figure 4. Normal containers and reefer container occur respectively 69% and 31% of the times.

## 2.5 Arrival and departure points

The arrival and departure positions of vessels and trucks are chosen randomly from the available locations. The selection of either a vessel location or a truck

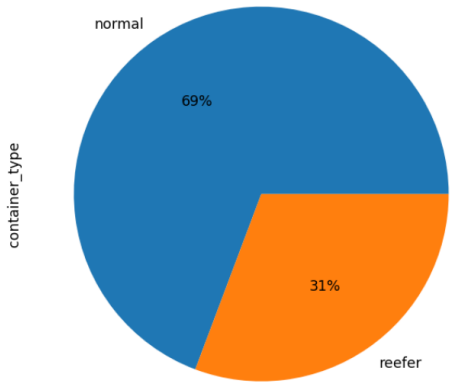



Figure 4: Container type analysis

location depends on the flow type, which will be discussed later.

### 2.6 Flow type

Each container group can be categorized as either export or import. In the input data, all groups that were categorized as transshipment are now categorized as export. The distribution of these flow types is based on the proportion of containers in the input data belonging to each category, resulting in 81% export and 19% import.

For the container groups classified as export, further division was necessary to determine their berthing locations and potential truck locations. This is because transshipment's would go from vessel to vessel, while true exports from truck to vessel. This distribution resulted in 69% transshipment's and 31% true export.

## 3 Algorithm

In this section, we will elaborate and discuss the flow, architecture, and design of the code. Since this code was developed in multiple steps, each with its own objectives, the program exhibits a highly modular structure. This modularity proves to be particularly advantageous for potential code reuse, both presently and in the future.

## 3.1 Amount of simulations

The first and maybe most primary parameter to know is the amount of simulations that needs to be run to get significant results. This can be calculated by the following formula:

$$S = \sqrt{\frac{\sum_{i=1}^{n}(X_i - \overline{X})^2}{n-1}}$$

$$\frac{S}{\sqrt{k}} < d$$

The key parameter we wanted to consider in the simulation is the travel distance, denoted as X. To assess the reliability of our results, we use the sample variance, denoted as S. The number of simulations performed is represented by k. Additionally, we define an accepted standard deviation, denoted as d. This threshold can be chosen based on the desired level of accuracy or confidence in the results.

In our case, X represents either the average or total travel distance obtained from running the simulation. The calculation process involves repeatedly running the simulation and collecting X values. As the number of simulations increases, the sample variance based on these k values is continuously evaluated. Once the sample variance falls within an acceptable range determined by the chosen standard deviation d, the calculation process is stopped, and the value of k is selected as the number of simulations required to achieve the desired level of accuracy.

For the total distance the accepted deviation is 1000. If we look at the magnitude of the total travel distance, we see it surpasses a million. A deviation of a thousands seems in that case reasonable. The total distance travelled is a much larger value than the average, because of this the accepted deviation is larger. From this information we can conclude that at least 120 simulations must be done to get consistent results.

## 3.2 Flow

Due to the requirement of testing multiple scenarios in the simulation, designing and structuring the program to allow for flexible flow manipulation without significant internal changes posed a challenge. To address this, the program utilizes booleans to map out different paths within the simulation. These booleans serve as input parameters and can be configured to define predefined scenarios.

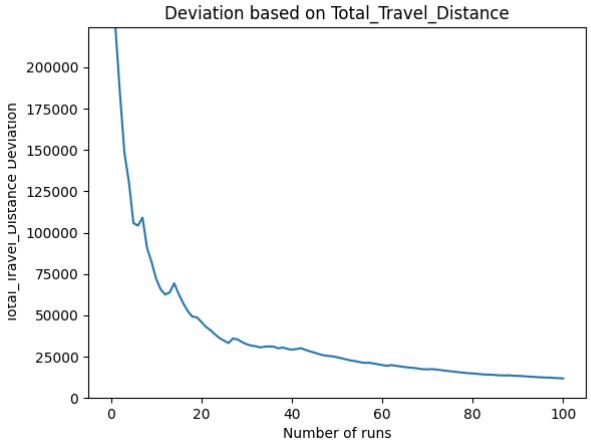



Figure 5: Deviation based on total travel distance

To ensure consistency and prevent unintended configurations, checks are implemented to throw exceptions if the booleans are set in an invalid combination. In the current implementation, four booleans were utilized:

1. ARRIVAL BASED
2. DEPARTURE BASED
3. CLOSEST
4. LOWEST OCCUPANCY
5. MIXED RULE
6. SPLIT UP

## 3.3 Core

The program operates on an event-based model, where the simulation timer progresses from one event to another. To achieve this, two lists are maintained to hold the events: one for container arrivals at the yard and another for container departures from the yard. Whenever a container is generated, a new arrival event is scheduled based on the interarrival time sampling function. Upon arrival, the container is checked to see if there is available space in the yard for storage. If there is, a yard block is selected based on the defined scenario to

store the container. However, if there is no available space, the container is rejected.

In events where containers depart, the respective container is removed from the yard. For each event that occurs, all relevant statistics are updated. This process continues until the simulation time is reached, completing the simulation.

## 3.4 Visualization

The visualization component of the simulation is implemented using Tkinter, a Python library widely used for creating graphical user interfaces (GUIs). Tkinter is based on the Tk GUI toolkit, hence its name "Tkinter." One of the key advantages of Tkinter is its inclusion in the standard Python distribution, eliminating the need for additional library installations. This accessibility makes it beginner-friendly and convenient for developing cross-platform applications.

In summary, Tkinter provides a robust framework for developing graphical applications in Python, allowing users to create intuitive and interactive user interfaces for their programs.

Some key features of the simulation is that the possibility to setup a flow using the intro screen, seen in Figure 6. Here you can setup all the necessary parameters to run a simulation, as well as the duration. All the possible scenarios can be chosen in this place, prior to start.

The visualization is multithreaded, which allows it to run smooth. It is possible to run both the visualization and gatter results formatted in a LaTex table (which has to be run multiple times to get correct results). But a strong computer is needed to pull this off. Because of the multithreaded visualization was it not usable, but in case of stronger pc's, it should work fluently. More on performance later in Section 5.

During the execution of the graphical representation, the boats are depicted on the water at the top of the screen. They appear and disappear as required by the simulation. The yard is represented by blocks of varying sizes, each indicating its respective capacity. These blocks are predominantly displayed in a vibrant green color. It is worth noting that the reefer blocks, which are used for refrigerated containers, are comparatively smaller in size. The degree to which a block is filled is represented by the color red. The more red there is in a block, the more it is filled in terms of percentage.

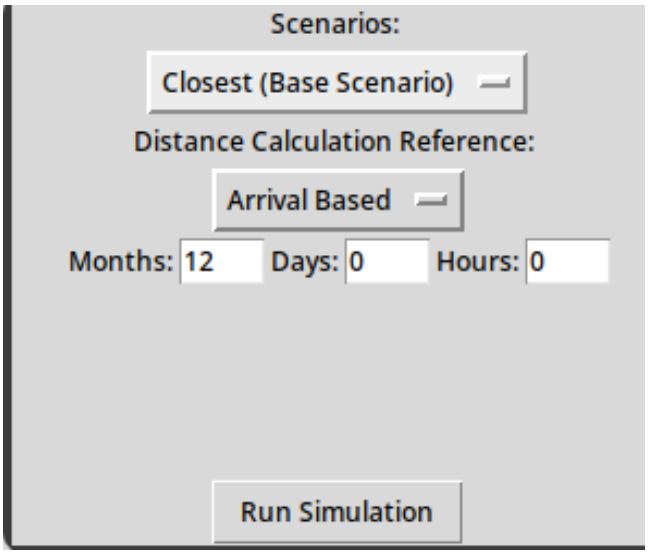



Figure 6: Intro screen


Figure 7: Progress bar of the simulations being run in order to get results

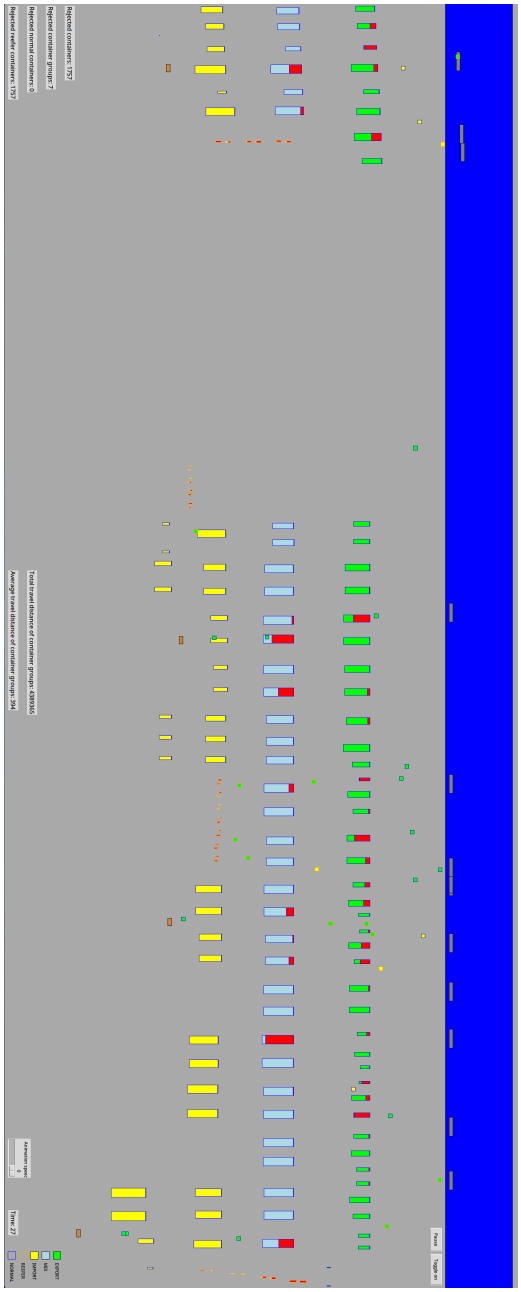



Figure 8: Visualization full layout

At the bottom of the screen, various statistics related to the simulation are displayed, such as the number of rejected containers and the total travel distance. The legend, indicating the meaning of different colors used in the visualization, is also located at the bottom for reference.

An interesting feature of the interface is the ability to modify the simulation's running state on the fly. This means that the simulation can be paused, allowing for temporary suspension of its progress. Additionally, the speed of the simulation can be adjusted, allowing users to control the pace of the simulation. Another option is to toggle the animations on or off. When animations are turned off, the simulation runs significantly faster, which can be useful when running the simulation for an extended duration.

# 4 What-if scenarios

A significant aspect of the assignment involved implementing various scenarios to explore the most "efficient" scenarios or to examine the impact of different scenarios. This provided an opportunity to test the visualization and gain insights into the workings of the yard.

As mentioned in previous reports, two different approaches to block assignments were implemented. These approaches remain applicable to all the scenarios described in this section. Therefore, for most scenarios, both approaches to block assignment are possible. The block assignment rule influences the selection of a yard block to store a container group. The two situations studied here are arrival-based and departure-based assignments.

In the case of arrival-based assignment, the yard block chosen is the one closest to the arrival point. Conversely, in departure-based assignment, the yard block selected is the one closest to the departure point. In both cases, the selection is based on minimizing the distance between two points.

## 4.1 Base Scenario

The base scenario remains unchanged from the previous report, where the decision rule is that a container group is accepted if there is available space in the yard. This scenario is referred to as FIFO (First In First Out). The FIFO rule applies to the arrival of containers and determines whether they are stored in the yard. According to FIFO, the first container group that arrives is given

priority over subsequent arrivals. If there is no space available for an arriving container group, it will be rejected. The block assignment rule determines which yard block is assigned to store the respective containers.

We see the base scenario as a reference point for the other scenarios. It is a scenario where most statistics point to a solid model. AANVULLEN

| | |
|---|---|
| Containers Rejected | 548306.083 |
| CG Rejected | 4034.933 |
| Normal Rejected | 9816.133 |
| Reefer Rejected | 538489.95 |
| Total Travel Distance | 5314784151.897 |
| AVG Travel Distance Containers | 1754.928 |
| Portion of YB close to full (at some point) | 0.698 |
| Portion of YB never used | 0.195 |
| Portion of YB close to full (average) | 0.698 |
| AVG daily total Occupancy | 0.745 |

Table 1: Base scenario, distance reference: arrival based

| | |
|---|---|
| Containers Rejected | 549260.158 |
| CG Rejected | 4029.158 |
| Normal Rejected | 9932.708 |
| Reefer Rejected | 539327.45 |
| Total Travel Distance | 5823736051.32 |
| AVG Travel Distance Containers | 1923.124 |
| Portion of YB close to full (at some point) | 0.591 |
| Portion of YB never used | 0.258 |
| Portion of YB close to full (average) | 0.591 |
| AVG daily total Occupancy | 0.673 |

Table 2: Base scenario, distance: arrival & departure based

## 4.2 Smallest Remaining capacity

The first new scenario involves storing all containers of the same container group in the same yard block. The containers are placed in the yard block with the largest remaining capacity. This approach ensures that the yard gradually fills up in a balanced manner across all available storage blocks. However, there were two possible interpretations for the concept of "remaining capacity" without explicit emphasis. In this implementation, the choice was made to consider the

| | |
|---|---|
| Containers Rejected | 551553.892 |
| CG Rejected | 4053.15 |
| Normal Rejected | 12869.05 |
| Reefer Rejected | 538684.842 |
| Total Travel Distance | 6622102730.796 |
| AVG Travel Distance Containers | 2184.101 |
| Portion of YB close to full (at some point) | 0.648 |
| Portion of YB never used | 0.05 |
| Portion of YB close to full (average) | 0.648 |
| AVG daily total Occupancy | 0.842 |

Table 3: Base scenario, distance reference: departure based

absolute number of available container slots in each yard block. An alternative interpretation could have been to use normalized remaining capacity, which represents the percentage of capacity that is still available in each block. In cases where two yard blocks have exactly the same remaining capacity, the closest one is chosen based on the block assignment rule.

BESCHRIJVEN RESULTATEN...

| | |
|---|---|
| Containers Rejected | 650443.533 |
| CG Rejected | 5013.842 |
| Normal Rejected | 48756.217 |
| Reefer Rejected | 601687.317 |
| Total Travel Distance | 8270153316.425 |
| AVG Travel Distance Containers | 2728.917 |
| Portion of YB close to full (at some point) | 0.453 |
| Portion of YB never used | 0.245 |
| Portion of YB close to full (average) | 0.453 |
| AVG daily total Occupancy | 0.645 |

Table 4: Lowest remaining capacity, distance reference: arrival & departure based

### 4.3 Possible split ups

In this scenario, container groups have the possibility of being split up into individual containers, which can be treated separately. This means that a yard block does not necessarily have to accommodate a full container group. Consequently, larger container groups should be handled more easily, and smaller

| | |
|---|---|
| Containers Rejected | 651509.433 |
| CG Rejected | 5014.367 |
| Normal Rejected | 49211.8 |
| Reefer Rejected | 602297.633 |
| Total Travel Distance | 8268808654.143 |
| AVG Travel Distance Containers | 2725.312 |
| Portion of YB close to full (at some point) | 0.447 |
| Portion of YB never used | 0.245 |
| Portion of YB close to full (average) | 0.447 |
| AVG daily total Occupancy | 0.646 |

Table 5: Lowest remaining capacity, distance reference: departure based

| | |
|---|---|
| Containers Rejected | 650775.533 |
| CG Rejected | 5008.1 |
| Normal Rejected | 48856.217 |
| Reefer Rejected | 601919.317 |
| Total Travel Distance | 8263128846.329 |
| AVG Travel Distance Containers | 2725.798 |
| Portion of YB close to full (at some point) | 0.465 |
| Portion of YB never used | 0.245 |
| Portion of YB close to full (average) | 0.465 |
| AVG daily total Occupancy | 0.647 |

Table 6: Lowest remaining capacity, distance reference: arrival-based

yard blocks can be utilized more effectively within the overall system.

The yard blocks selected to store containers are determined based on their proximity to the arrival or departure point (depending on the decision rule) and their available space to accommodate one or more containers. The primary focus is on identifying the closest block that can hold one or more containers.

BESCHRIJVEN RESULTATEN...

## 4.4 Not allowing mixing container types in a single yard-block

The final scenario introduces a rule where MIX yard blocks can only accommodate containers of a single flow type. This means that once a container arrives and occupies a MIX yard block, only containers of the same flow type can be

| | |
|---|---|
| Containers Rejected | 472353.375 |
| CG Rejected | 5127.158 |
| Normal Rejected | 0.0 |
| Reefer Rejected | 472353.375 |
| Total Travel Distance | 5611085409.514 |
| AVG Travel Distance Containers | 1851.933 |
| Portion of YB close to full (at some point) | 0.623 |
| Portion of YB never used | 0.22 |
| Portion of YB close to full (average) | 0.623 |
| AVG daily total Occupancy | 0.699 |
| Average amount of split ups | 53848.1 |

Table 7: Possible split up, distance: arrival-based

| | |
|---|---|
| Containers Rejected | 473158.492 |
| CG Rejected | 5131.692 |
| Normal Rejected | 0.0 |
| Reefer Rejected | 473158.492 |
| Total Travel Distance | 6959654819.095 |
| AVG Travel Distance Containers | 2297.05 |
| Portion of YB close to full (at some point) | 0.629 |
| Portion of YB never used | 0.101 |
| Portion of YB close to full (average) | 0.629 |
| AVG daily total Occupancy | 0.788 |
| Average amount of split ups | 65446.74 |

Table 8: Possible split up, distance reference: departure based

added to that particular yard block. In other words, containers with different flow types cannot be mixed within the same yard block. This rule ensures segregation and grouping of containers based on their flow type in the yard.

BESCHRIJVEN RESULTATEN...

# 5 Performance

The entire simulation was implemented in Python, which is an interpreted programming language. Unlike compiled languages like C or C++, Python is executed by an interpreter rather than being compiled into machine code. As a result, Python can be perceived as slower in terms of performance compared to compiled languages.

| | |
|---|---|
| Containers Rejected | 472326.425 |
| CG Rejected | 5144.042 |
| Normal Rejected | 0.0 |
| Reefer Rejected | 472326.425 |
| Total Travel Distance | 6077806085.211 |
| AVG Travel Distance Containers | 2006.154 |
| Portion of YB close to full (at some point) | 0.547 |
| Portion of YB never used | 0.321 |
| Portion of YB close to full (average) | 0.547 |
| AVG daily total Occupancy | 0.613 |
| Average amount of split ups | 74397.59 |

Table 9: Possible split up, distance reference: arrival & departure based

| | |
|---|---|
| Containers Rejected | 562005.908 |
| CG Rejected | 4285.208 |
| Normal Rejected | 10305.508 |
| Reefer Rejected | 551700.4 |
| Total Travel Distance | 5329012083.215 |
| AVG Travel Distance Containers | 1757.727 |
| Portion of YB close to full (at some point) | 0.679 |
| Portion of YB never used | 0.176 |
| Portion of YB close to full (average) | 0.679 |
| AVG daily total Occupancy | 0.748 |

Table 10: Mixed rule, distance reference: arrival-based

The simulations were conducted over a period of one year, which involved extensive computations. When running the simulations without animations, the average time per simulation was 127.24 seconds. While this may not seem excessively high, considering the number of simulations required to obtain meaningful results, it becomes a significant factor. For instance, running 100 simulations would take approximately 3 hours and 32 minutes.

The graphical aspect of the simulation is fully multithreaded, with each moving container assigned its own thread. This enables smooth animations and enhances the visual experience. However, this also poses challenges when attempting to run the simulation multiple times while simultaneously displaying the visualization, particularly on less powerful computers. Performing such tasks on separate threads becomes complex due to the already resource-intensive nature of the visualization process.

| | |
|---|---|
| Containers Rejected | 562459.5 |
| CG Rejected | 4311.817 |
| Normal Rejected | 12785.225 |
| Reefer Rejected | 549674.275 |
| Total Travel Distance | 6606445719.566 |
| AVG Travel Distance Containers | 2180.617 |
| Portion of YB close to full (at some point) | 0.648 |
| Portion of YB never used | 0.044 |
| Portion of YB close to full (average) | 0.648 |
| AVG daily total Occupancy | 0.842 |

Table 11: Mixed rule, distance reference: departure based

| | |
|---|---|
| Containers Rejected | 557756.75 |
| CG Rejected | 4226.058 |
| Normal Rejected | 9384.333 |
| Reefer Rejected | 548372.417 |
| Total Travel Distance | 5823224027.74 |
| AVG Travel Distance Containers | 1922.426 |
| Portion of YB close to full (at some point) | 0.61 |
| Portion of YB never used | 0.245 |
| Portion of YB close to full (average) | 0.61 |
| AVG daily total Occupancy | 0.678 |

Table 12: Mixed rule, distance reference: arrival & departure based

# 6 Conclusion