SAPIENZA
UNIVERSITÀ DI ROMA

Analysis of Intermodal Terminal Operations through Simulation of Resource Utilization and Truck-Train Synchronization Effects

Faculty of Civil and Industrial Engineering
Corso di Laurea Magistrale in Transport Systems Engineering

Mohammadreza Keramati
ID number 1937929

Advisor
Prof. Gaetano Fusco

Co-Advisor
Prof. Chiara Colombaroni

Academic Year 2023/2024

Thesis not yet defended

Analysis of Intermodal Terminal Operations through Simulation of Resource Utilization and Truck-Train Synchronization Effects
Master's Thesis. Sapienza University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: keramati.1937929@studenti.uniroma1.it

Contents

0.1 Abstract

This thesis investigates the operational efficiency of intermodal terminals through advanced discrete event simulation (DES) techniques, focusing on resource utilization and truck-train synchronization. As global supply chains increasingly rely on intermodal transportation, optimizing terminal operations becomes crucial for cost reduction and environmental sustainability. The study employs a DES model developed in Python using SimPy and Salabim libraries to simulate complex terminal interactions, including truck arrivals, train schedules, crane operations, and storage yard management.

Six scenarios were designed to test various synchronization levels, arrival distributions, and demand patterns, applied to a case study of the Piedimonte San Germano Terminal in Italy. The research objectives include evaluating the impact of truck-train synchronization on efficiency, assessing the effects of different resource configurations on key performance indicators, and providing actionable recommendations for terminal optimization.

Key findings reveal that tighter synchronization between truck arrivals and train schedules significantly enhances terminal performance. Poisson-distributed truck arrivals, when tightly synchronized with train schedules, improve operational efficiency compared to Gaussian distributions. While increasing the number of cranes improves workload distribution, it may lead to underutilization in low-demand scenarios. High-demand scenarios demonstrate the terminal's capacity to handle increased volumes but may reduce direct transshipments. Additionally, increasing the number of road gates substantially reduces average truck waiting times, identifying them as critical bottlenecks.

The study concludes that prioritizing synchronization strategies, implementing dynamic resource allocation, and investing in additional road gates are crucial for improving intermodal terminal efficiency. This research contributes to the field by offering a comprehensive, adaptable simulation model and providing valuable insights for terminal operators, logistics planners, and policymakers aiming to enhance the efficiency and sustainability of global supply chains.

Chapter 1

Introduction

Intermodal transportation is a critical component of modern logistics, aimed at enhancing efficiency and sustainability by integrating various modes of transport, such as road, rail, and maritime. This approach reduces environmental impact and transportation costs while ensuring a seamless transfer of goods. Intermodal terminals are key nodes where different modes converge to facilitate the efficient movement of intermodal transport units (ITUs) between transportation modes.

1.1 Background

This study focuses on intermodal terminals and aims to improve their operational efficiency using simulation-based analysis. Intermodal terminals handle a variety of operations, including loading, unloading, transferring, and storing ITUs, while also managing synchronization between trucks and trains. The efficient use of resources such as cranes, road gates, and storage yards is crucial for terminal performance. Effective management can reduce costs, minimize delays, and increase competitiveness.

Intermodal transport offers significant benefits, such as reducing road congestion and lowering carbon emissions by shifting long-distance transport from road to rail. However, challenges include delays, sub-optimal resource utilization, and the need for precise synchronization between truck and train schedules. Addressing these challenges is essential for optimizing terminal performance.

1.2 Concepts and Definitions

An **Intermodal Terminal** is a facility where ITUs are transferred between different modes of transport, such as trucks and trains. This study employs a **Discrete Event Simulation (DES)** approach to model the dynamic interactions between components of an intermodal terminal. DES allows for capturing the stochastic nature of terminal operations by simulating distinct events that change the system state over time.

Intermodal Transport Units (ITUs) are standardized cargo units, such as containers or swap bodies, designed for transportation across multiple modes without handling the contents. **Synchronization** between truck arrivals and train schedules

is vital for minimizing delays and improving throughput. **Resource Utilization** is measured by evaluating the effective use of cranes, road gates, and storage yards within the terminal.

1.3 Scope of the Study

This research explores the performance of an intermodal terminal under different operational scenarios using a simulation model. Six scenarios with varying synchronization levels and resource configurations are analyzed to understand the impact on key performance indicators (KPIs), such as truck waiting time, crane utilization rate, and the percentage of direct transshipments.

The simulation model is developed in Python, using the **SimPy** library for DES and **Salabim** for visualizing certain processes, such as truck arrivals and road gate operations. The methodology aims to provide insights into how synchronization and resource configurations influence operational efficiency, providing recommendations for optimizing terminal operations.

1.4 Problem Statement

Operations at intermodal terminals are complex, influenced by fluctuating demand, resource availability, and timing constraints. Key challenges include achieving optimal **synchronization between truck arrivals and train schedules**, ensuring efficient **resource utilization**, and minimizing delays to enhance throughput.

This study addresses the following questions: How do different levels of truck-train synchronization impact terminal operations? What effects do varying resource configurations, such as the number of cranes and road gates, have on terminal performance? To answer these questions, a DES model is developed to simulate terminal operations under several scenarios.

1.5 Figures and Visualization

Figures are used to enhance understanding of the concepts:

- Figure 1.1 illustrates the layout of an intermodal terminal, highlighting sections like road gates, cranes, and storage areas. - Figure 1.4 depicts the sequential processes involved in handling ITUs, demonstrating potential bottlenecks.

These visuals help convey the complexity of terminal operations and the factors affecting efficiency.

1.6 Objectives of the Study

The main objective of this study is to analyze the impact of varying levels of synchronization between truck arrivals and train schedules, along with different resource allocations, on intermodal terminal performance. Specific objectives include:

1. Evaluating the **impact of synchronization** on operational efficiency.

2. Assessing **resource utilization** by varying the number of cranes, road gates, and storage yard configurations. Providing recommendations for improving the

3. **efficiency and capacity utilization** of intermodal terminals.

1.7 Structure of the Thesis

The remainder of this thesis is organized as follows: - Chapter 2 presents a **Literature Review** of intermodal terminal operations and optimization strategies. - Chapter 3 describes the **Methodology**, detailing the DES model and its underlying assumptions. - Chapter 4 contains the **Results and Discussion**, comparing different KPIs across scenarios and highlighting key findings. - Chapter 5 concludes with a summary of the **findings** and provides **recommendations** for future research and practical applications.

1.8 Contribution of the Study

This research contributes to the **development and analysis of a simulation-based approach** for evaluating intermodal terminal operations, focusing on the effects of synchronization and resource allocation. By introducing multiple scenarios that reflect realistic variability in truck-train synchronization and terminal resource configurations, the study builds on previous research and aims to optimize terminal efficiency. The use of **Salabim** for visualizing processes offers additional insights into bottlenecks, enhancing the understanding of complex terminal dynamics.

1.9 Background on Intermodal Transport

Intermodal transport involves the movement of goods using multiple modes of transportation without handling the goods themselves when changing modes (UNECE, 2010). This concept focuses on the efficient use of standardized Intermodal Transport Units (ITUs) such as containers, swap bodies, or semi-trailers, which are adapted to be easily transferred between modes. This method is beneficial for reducing transit costs, enhancing environmental sustainability, and facilitating seamless movement across diverse regions.

The efficiency of intermodal transport depends on several factors, including the type of product being transported and the geographical positioning of intermodal terminals. Typically, the intermodal rail-road transport chain involves three major phases: pre-haulage, main haulage, and post-haulage. In pre-haulage, ITUs are collected from various origins by road transport and brought to an intermodal terminal. During the main haulage phase, ITUs are transported over long distances by rail. Finally, in the post-haulage phase, ITUs are delivered to their final destination using road transport.

1.10 Intermodal Freight Terminals

Intermodal freight terminals are crucial links in the intermodal transport chain. These facilities serve as transfer points where ITUs are transferred from road vehicles to trains and vice versa. These terminals play a significant role in the efficiency of the overall intermodal transportation system and are also responsible for a considerable portion of the total logistics cost. Depending on the scale of operations, intermodal terminals can be categorized into three types:

- **Small terminals**: Capable of transshipping up to 70-80 ITUs per day.
- **Medium terminals**: Capable of transshipping up to 140-150 ITUs per day.
- **Large terminals**: Capable of transshipping up to 250 ITUs per day.

Intermodal terminals are characterized by specific features such as layout, handling equipment, storage capacity, and operating policies. Inland rail-road terminals are usually smaller than maritime terminals, and the residence time of ITUs at such facilities is shorter. The terminal must be carefully managed to ensure optimal utilization of its storage area, transshipment equipment, and gate facilities.

The terminal layout typically includes designated areas for road gates, cranes, and buffer storage, with a direct link to the railway platform for train loading/unloading. The efficient functioning of these terminals depends on the synchronization between truck and train arrivals and the effective allocation of resources like cranes and road gates to handle incoming/outgoing ITUs.

1.11 Rail-road Intermodal Terminal Description

Rail-road intermodal terminals serve as critical nodes within the logistics network, facilitating the movement of goods by connecting rail and road transportation modes. These terminals perform the transshipment of standardized Intermodal Transport Units (ITUs), such as containers or swap bodies, and ensure smooth transitions between transport types to optimize efficiency and reduce environmental impact.

General Considerations

Intermodal rail-road terminals are designed to maximize the efficient movement of ITUs while minimizing handling time and costs. The terminal's design, handling equipment, layout, and operating policies must be optimized to achieve high throughput while maintaining safety and reducing operational complexity. The terminal's location is also a critical factor, ideally situated near major production and consumption areas or within efficient access to main rail lines and road networks.

The efficiency of such terminals depends on several factors, including the timely coordination between road and rail operations, the availability of appropriate handling equipment, and the correct use of storage areas. The key challenge lies in the terminal's capacity to handle fluctuations in demand, while ensuring resources are optimally utilized.

Vehicles and Intermodal Transport Units

Intermodal Transport Units (ITUs) are central to the functioning of intermodal terminals, designed to be transported efficiently across different transport modes without handling the contents directly. The ITUs may be containers, swap bodies, or semi-trailers, with dimensions standardized according to international regulations, allowing for seamless handling by various types of vehicles.

Road vehicles at the terminal are primarily trucks equipped with trailers that can load and unload ITUs. Rail vehicles consist of specialized flat wagons that can carry ITUs across long distances. The use of ITUs simplifies transshipment between trucks and trains, reduces handling costs, and minimizes the risk of damage to the goods [10].

Terminal Layout

The layout of an intermodal terminal plays a critical role in its operational efficiency. The terminal is typically divided into several key components, each designated for specific functions related to the transfer and storage of ITUs.

Terminal Components

The major components of a rail-road intermodal terminal include:

- **Rail Side**: The rail side of the terminal consists of multiple rail tracks used for loading and unloading ITUs. These tracks should be easily accessible by cranes and should allow sufficient space for the handling and shunting of wagons.

- **Road Side**: The road side comprises road gates, truck lanes, and access points to ensure trucks can efficiently deliver and pick up ITUs. Efficient access and well-defined lanes are essential to minimize delays and congestion, particularly during peak hours.

- **Handling Area and Cranes**: The handling area is equipped with cranes and reach stackers that move ITUs between trains, trucks, and storage areas. The type of cranes used can vary depending on the terminal size and ITU volume, and may include gantry cranes, mobile cranes, or reach stackers.

- **Storage Areas**: The storage area, also known as the buffer area, is used for temporary storage of ITUs before they are moved to their final destination. The layout and capacity of storage areas must be optimized to ensure ITUs are stored safely and can be easily accessed when needed.

Figure 1.1 illustrates the typical layout of a rail-road intermodal terminal, highlighting the key components such as the rail side, road side, and handling areas.

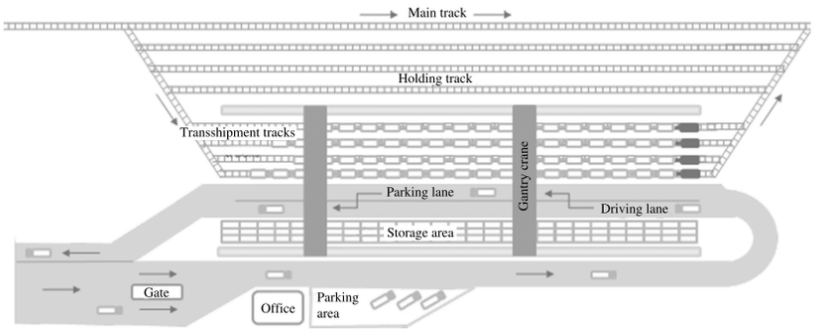


Figure 1.1. Typical Layout of a Rail-road Intermodal Terminal.

Terminal Processes

The operational processes at intermodal terminals can be classified based on the direction of ITU movements, namely, ITU arrival by truck and departure by train, and vice versa.

ITU Arrival by Truck and Departure by Train

In this scenario, trucks arriving at the terminal are directed to the designated unloading area, where ITUs are unloaded using reach stackers or cranes. Once unloaded, ITUs are either stored temporarily in the storage area or moved directly to the rail side, where they are loaded onto waiting wagons for long-distance rail transport.

The efficiency of this process depends on the timely arrival of trucks and trains, and the synchronization between these two modes is essential to minimize waiting times. The availability of sufficient cranes and well-managed truck lanes is also crucial for the efficient transfer of ITUs. Figure 1.3 provides an overview of the steps involved in this process.

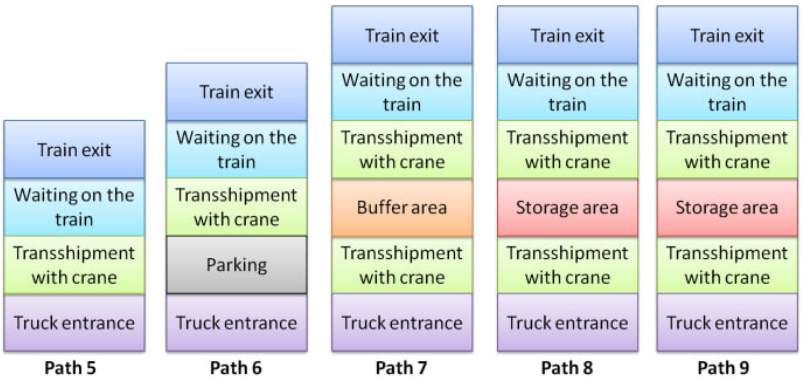


Figure 1.2. Process Flow: ITU Arrival by Truck and Departure by Train.

ITU Arrival by Train and Departure by Truck

When ITUs arrive by train and are destined for delivery by truck, the unloading process begins as soon as the train reaches the terminal. Cranes or reach stackers unload ITUs from the wagons, and they are either stored in the buffer storage area

or directly loaded onto waiting trucks. The key factors affecting the efficiency of this process are the timely arrival of trucks, sufficient crane availability, and adequate storage space to manage any temporary holding needs.

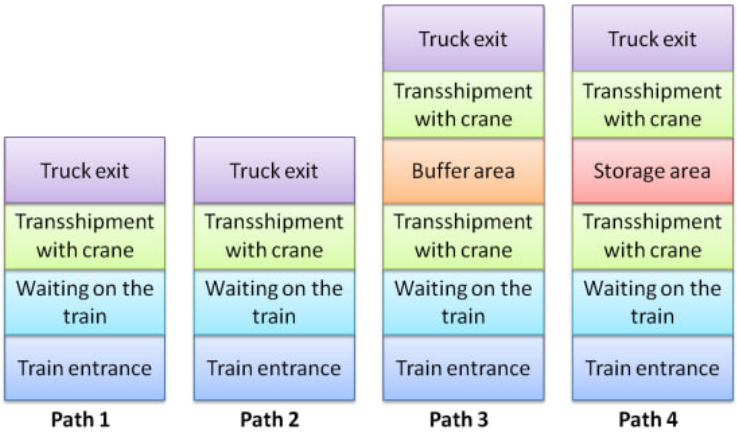


Figure 1.3. Process Flow: ITU Arrival by Train and Departure by Truck.

Train Loading/Unloading

Loading and unloading of trains at intermodal terminals is a critical activity that must be performed with minimal delays to ensure smooth operations. The terminal must ensure that enough handling equipment is available to transfer ITUs from trucks to trains or from trains to storage areas within a short timeframe.

Cranes are typically employed for lifting ITUs onto or off of flat wagons, and these cranes are operated based on the scheduled train arrivals. Efficient train handling requires detailed coordination between terminal operations staff, ensuring that ITUs are positioned at the rail side as soon as the train arrives. Figure 1.4 illustrates the train loading and unloading process at a typical intermodal terminal.

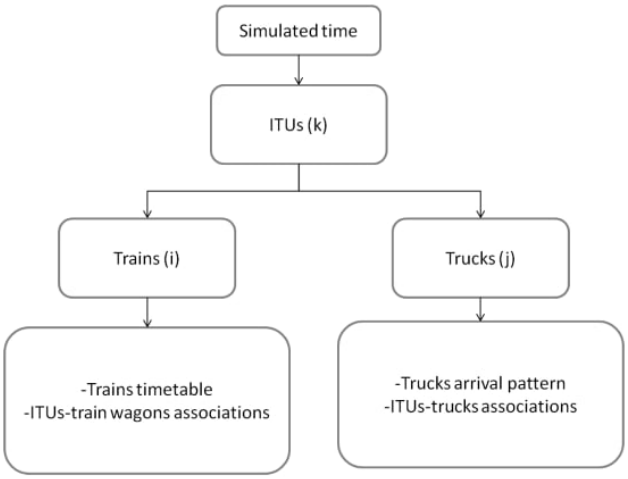


Figure 1.4. Train Loading and Unloading Process at an Intermodal Terminal.

Storage Areas and Processes

Storage areas are an integral component of intermodal terminals, serving as temporary holding zones for ITUs before their final movement. Storage may be required due to a mismatch in timing between truck arrivals and train departures, or vice versa. The design of storage areas must account for easy access, efficient use of available space, and safety considerations for storing containers.

Storage areas are often organized in stacks, with ITUs placed based on their priority for retrieval. Efficient stacking and retrieval mechanisms help minimize unnecessary handling, reducing both time and operational costs. The use of automated systems, such as yard cranes, can further enhance storage efficiency.

Figure 1.5 depicts a typical storage yard layout, including crane pathways for efficient retrieval and handling.

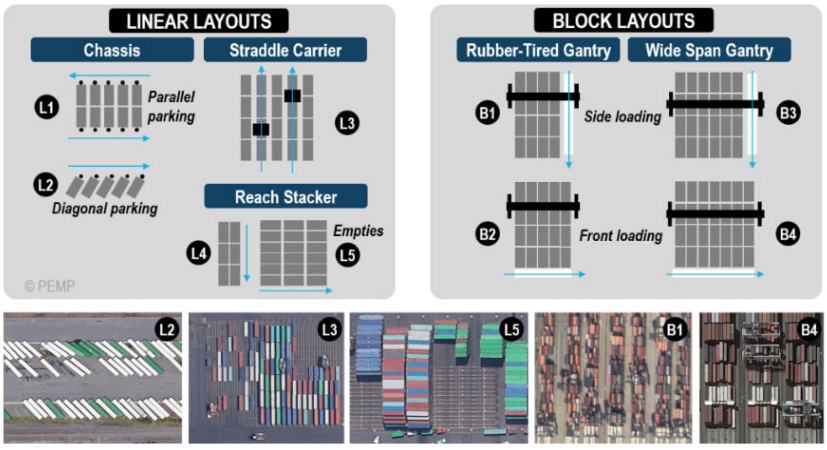


Figure 1.5. Typical Layout of Storage Areas in an Intermodal Terminal[5].

Key Considerations for Efficiency

The efficiency of intermodal terminals depends largely on the smooth coordination of operations between rail and road transportation. Ensuring adequate resource allocation, such as cranes and storage space, managing peak-hour traffic at road gates, and effective synchronization between truck arrivals and train schedules are all crucial factors. Advanced scheduling systems, information sharing platforms, and real-time monitoring can help improve the overall efficiency of these terminals.

The detailed description provided above of a rail-road intermodal terminal emphasizes the importance of layout, equipment, and process coordination to achieve seamless ITU transfers, reduce waiting times, and maximize throughput.

Chapter 2

Literature Review

2.1 Discrete Event Simulation (DES) in Logistics Systems

Discrete Event Simulation (DES) has been widely recognized as a powerful tool for studying and optimizing complex logistics systems. In DES, the operation of a system is modeled as a sequence of discrete events that occur at specific points in time. Each event represents a change in the state of the system, such as the arrival of a truck, the departure of a train, or the operation of a crane. The flexibility of DES allows it to model the dynamic interactions and dependencies between different system components, making it well-suited for analyzing intermodal terminal operations [8].

Early applications of DES in logistics systems focused on transportation networks, production systems, and material handling. As intermodal terminals became more critical for global supply chains, researchers began to use DES to simulate terminal operations, enabling a detailed analysis of resource utilization, scheduling, and synchronization [7].

2.2 Decision Support Systems for Terminal Operations

A decision support tool was presented to optimize two of the most critical activities in intermodal railroad container terminals, in an iterative and integrated framework devoted to the terminal profit improvement. First, the model allows optimizing the freight trains composition, maximizing the company profit, while respecting physical and economic constraints, and placing in the train head/tail containers prosecuting to subsequent destinations. Hence, based on the resulting train composition, the decision support system allows optimizing the containers allocation in the terminal storage yard, in order to maximize the filling level while respecting physical constraints. The model is successfully tested on a real case study, the inland railroad terminal of a leading Italian intermodal logistics company [3].

DES models have also been employed to study the allocation of resources such as cranes and road gates. Boysen et al. (2012) highlighted that poorly managed resources can lead to congestion and underperformance during peak operational times

[2]. The ability of DES to capture such dynamic behavior under varying scenarios has made it a preferred approach for studying resource allocation strategies.

2.3 Synchronization of Truck and Train Arrivals in DES Models

One of the most critical aspects of intermodal terminal operations is the synchronization between truck and train arrivals. DES provides an effective way to model and simulate different synchronization patterns, allowing researchers to investigate their impact on system performance. Tight synchronization, where trucks arrive just before trains, minimizes waiting times for trucks and ensures efficient crane operations. On the other hand, loose synchronization or random arrivals can result in prolonged truck waiting times and inefficient resource utilization [1].

Several studies have explored how DES can be used to evaluate different synchronization scenarios. Park et al. (2009) demonstrated that optimizing truck arrival patterns relative to train schedules can significantly reduce bottlenecks and improve terminal throughput [6]. Simulation models developed using DES allow for the testing of various configurations, enabling terminal operators to choose the most efficient scheduling and resource allocation strategies.

2.4 Resource Allocation Strategies in DES Models

In addition to synchronization, resource allocation is another critical area where DES has been widely applied. DES models enable the simulation of various resource allocation strategies, such as dynamic crane assignment or flexible road gate scheduling, to optimize terminal operations under different conditions.

Huelsz(2015) emphasized that the real-time adjustment of resources based on demand can greatly improve terminal performance, particularly during peak times [4].

2.5 Optimal Control of Resource Allocations in Port Operations

Resource allocation in maritime port operations can be modeled as a dynamic system of queues, where the transfer of containers between ships, trucks, and trains is managed through optimal control strategies. Wanigasekara and Torres (2024) propose an optimal control method for resource allocation in maritime container terminals using a discrete-time queue model. The study highlights how effective resource allocation can reduce operational costs and improve the efficiency of container transfers between different sub-terminals.

The model focuses on balancing precision and computational efficiency by incorporating constraints such as spatial limitations, handling equipment capacities, and processing times for containers. By simulating different resource allocation strategies, the proposed method allows terminals to optimize resource utilization without degrading the quality of service.

Through simulation scenarios, Wanigasekara and Torres demonstrate the benefits of the optimal control strategy in reducing the number of resources required for terminal operations while maintaining the same throughput across ships, trucks, and trains. The study concludes that such a control mechanism can improve the overall management of terminal operations, especially during periods of high container traffic or equipment constraints. This framework can be applied to various terminal settings, including those dealing with intermodal freight transportation.
[9]

2.6 Comparison of DES Tools for Terminal Operations

Several DES tools are available for simulating logistics systems and terminal operations. While `SimPy` and `Salabim` were used in this project, many other tools have been developed for similar purposes. Each tool has its strengths and weaknesses, depending on the complexity of the system being modeled, the level of detail required, and the need for visualization.

- **SimPy**: A Python-based DES library used in this project for simulating terminal operations. SimPy is well-suited for building flexible, customizable models, making it popular for research and educational purposes.
- **Salabim**: Another Python-based library, used here for animating the road gate operations. Salabim offers easy-to-use animation features for visualizing discrete events, providing additional insights into terminal operations.
- **AnyLogic**: A more advanced DES tool that supports multi-method modeling, combining DES with agent-based modeling (ABM) and system dynamics.
- **Arena**: A widely used commercial tool for simulating manufacturing systems and logistics operations.
- **FlexSim**: Known for its graphical user interface, FlexSim is commonly used in production and supply chain simulations.
- **Plant Simulation (Siemens)**: This tool is often used in industrial settings to simulate production lines and logistics systems, offering detailed analysis and visualization capabilities.

The selection of a DES tool depends on the specific requirements of the project. In this thesis, `SimPy` was chosen for its open-source flexibility, and `Salabim` for its animation capabilities, enabling both efficient simulation and visualization of terminal operations.

2.7 Gaps in the Literature and Future Research Opportunities

While DES has proven to be a valuable tool for simulating intermodal terminal operations, there are still gaps in the literature. For instance, the integration of

real-time data into DES models is a relatively unexplored area that could enhance the adaptability of terminal operations. Additionally, more work is needed to investigate how DES can be used in conjunction with machine learning algorithms to create predictive models that optimize terminal performance .

Future research could focus on the development of hybrid models that combine DES with other simulation techniques, such as agent-based modeling or system dynamics, to provide a more comprehensive understanding of terminal operations under varying conditions. Exploring the effects of stochastic variability, such as random delays in truck or train arrivals, would also help to improve the robustness of resource allocation strategies.

Chapter 3

Methodology

3.1 Overview of Methodological Approach

The primary objective of this study is to simulate and analyze the operations of an intermodal terminal using Discrete Event Simulation (DES). DES allows for the modeling of terminal operations as a series of events occurring at discrete points in time, which provides insights into resource utilization, waiting times, and operational efficiency under various configurations. This section describes the methodology used to develop, implement, and analyze the DES model of the terminal, along with the specific simulation tools and data sets used.

The simulation framework focuses on the integration of resource allocation strategies, synchronization of truck and train arrivals, and the evaluation of key performance indicators (KPIs) such as truck waiting times, crane utilization, and throughput. The goal is to identify optimal operational strategies that improve terminal efficiency.

3.2 Problem Statement

Intermodal terminals are critical in the global supply chain as they facilitate the transfer of cargo between different transportation modes. However, challenges often arise in the following areas:

Synchronization: Aligning truck arrivals with train schedules to minimize delays.

Resource Utilization: Efficient allocation and usage of cranes and storage areas.

Performance Metrics: Reducing truck waiting times while increasing throughput.

3.2.1 Research Question:

"How do different levels of truck-train synchronization and resource allocation affect the operational efficiency and capacity utilization of intermodal terminals?"

3.3 Discrete Event Simulation (DES) Framework

The DES framework models the intermodal terminal as a system where key resources—cranes, road gates, storage yards—are used to handle the transfer of Intermodal Transport Units (ITUs) between trucks and trains. The primary events modeled in the system include truck arrivals, ITU loading and unloading, and train departures. The system state changes at each event, with updates to resource availability, queue lengths, and processing times.

3.3.1 Simulation Tools

To implement the DES model, we used two Python-based simulation libraries:

- **SimPy**: A process-based discrete-event simulation library used to simulate the operation of the terminal. SimPy was chosen for its flexibility and ease of use in modeling queue-based systems, making it ideal for this application.
- **Salabim**: Used to create animations of the simulated operations, focusing specifically on road gate activities. The animation provides a visual representation of the flow of trucks through the terminal, helping to validate and better understand the behavior of the system.

3.4 Rationale for Selecting SimPy and Salabim

The choice of suitable simulation tools is vital for ensuring both the accuracy and reliability of any discrete event simulation (DES) study. For this research, SimPy was selected as the core simulation engine, and Salabim was used for visualization purposes, particularly for modeling road gate operations. The reasoning behind these decisions is outlined below:

3.4.1 SimPy for Core Simulation

1. Open-source and Python-based: SimPy is an open-source library that runs on Python. This aligns with the growing use of open-source tools in both academia and industry, enhancing transparency and reproducibility of research. Additionally, Python's extensive ecosystem allows for easy integration with other data analysis and scientific libraries.
2. Flexibility and Customization: SimPy offers a high level of flexibility, enabling the development of custom simulation models. This was particularly important for our study, as intermodal terminal operations involve complex interactions that would be difficult to model with more rigid commercial software.
3. Event-driven Architecture: SimPy's event-driven structure makes it ideal for simulating terminal operations, where system behavior is driven by discrete events such as truck arrivals, crane activities, and train departures.
4. Scalability: SimPy is capable of efficiently handling large-scale simulations, which is crucial for modeling busy intermodal terminals involving numerous entities and events.

5. Active Community and Documentation: With a strong user community and extensive documentation, SimPy supports troubleshooting and allows for continuous model enhancement.

3.4.2 Salabim for Visualization

1. Complements SimPy: While SimPy is highly effective for building the core simulation, it lacks visualization features. Salabim fills this gap by providing powerful animation capabilities, making it ideal for visualizing complex processes like road gate operations.

2. Python Integration: Since Salabim is also built on Python, it integrates seamlessly with SimPy, ensuring smooth interaction between the simulation logic and visualization.

3. Tailored for Logistics: Salabim is particularly suited for logistics and transport simulations, offering functionalities that match our focus on intermodal terminal operations.

4. Real-time Animation: Salabim's ability to animate the simulation in real-time helps visually validate the model, providing valuable insights into system behavior and potential bottlenecks.

3.4.3 Comparison with Other Tools

Although other simulation software like AnyLogic, Arena, or FlexSim provide comprehensive simulation environments with built-in visualization, the SimPy-Salabim combination was chosen for several reasons:

Cost-effectiveness: Both SimPy and Salabim are open-source, making them a budget-friendly option compared to commercial software, which is especially beneficial in academic research.

Customization: The Python-based approach allows for more customization and integration with external analytical tools, which may not be as straightforward in closed-source commercial software.

Learning Curve and Transferability: For users already familiar with Python, learning SimPy and Salabim is relatively simple. The skills acquired are also applicable to other Python-based projects in data science and simulation.

Focus on Specific Needs: While tools like AnyLogic offer broader modeling paradigms (e.g., agent-based modeling, system dynamics), our focus on discrete event simulation for terminal operations was better served by SimPy and Salabim's specialized features.

Integration with Data Analysis: Python's ecosystem allows for seamless integration with data analysis libraries like Pandas and visualization tools such as Matplotlib, enabling comprehensive analysis of simulation outputs.

3.5 Simulation Model Architecture

The simulation model consists of several key components, each modeled as a class in Python, representing specific terminal operations:

ITU (Intermodal Transport Unit): Represents cargo units moved through the terminal.

Train: Models the arrival and departure of trains.

StorageYard: Simulates storage areas, including buffer and long-term storage.

Truck: Represents trucks arriving to deliver or pick up ITUs.

RoadGate: Handles truck entry and exit.

CraneOperation: Simulates crane movements for loading/unloading ITUs.

TrainUnloading and TrainLoading: Manage the process of unloading/loading ITUs from/to trains.

MainSimulation: Coordinates overall operations, including the initialization and running of multiple scenarios.

3.6 Key Parameters and Assumptions

The simulation model incorporates several adjustable parameters that govern terminal operations:

T1 (6 hours): Time window for direct transshipment.

T2 (24 hours): Time window to determine buffer vs. long-term storage.

Crane Operation Time: 4 minutes per crane movement.

Storage Yard Capacity: Long-term (1000 ITUs), buffer (200 ITUs).

Maximum Stack Height: 2 ITUs.

3.6.1 Assumptions:

Trucks can carry only one ITU at a time. Trains have fixed capacity and schedules. Crane operations are performed sequentially.

3.7 Input Datasets for Simulation

The input datasets are essential for accurately representing the dynamics of intermodal terminal operations within the simulation model. These datasets are designed to simulate truck arrival patterns and train schedules, both of which are critical for testing various synchronization scenarios and their impact on terminal performance. The data provides the foundation for exploring different operational configurations, helping assess key performance metrics such as truck waiting time, crane utilization, and throughput. In this section, we present a detailed explanation of the input datasets used in the study.

3.7.1 Truck Arrival Datasets

The truck arrival datasets simulate the timing and volume of trucks entering the terminal, which is vital for capturing real-world variability. Different randomness models were applied to generate varying truck arrival patterns across multiple scenarios. For example, the `truck1_gaussian.csv` dataset models truck arrivals using a Gaussian (normal) distribution, where trucks are more likely to arrive around a central time, with fewer arrivals deviating from that time. This distribution simulates less predictable patterns, with slight randomness in truck arrivals, enabling the evaluation of terminal performance under such conditions.

In contrast, the `truck1_poisson.csv` dataset applies a Poisson distribution, which is suitable for modeling random arrival events with a specific average rate. This introduces more variability in arrival times, representing cases where truck arrivals are less predictable but follow an average rate over time. This helps assess the terminal's ability to handle more stochastic arrival patterns.

Additionally, the `truck2_modified_randomness.csv` dataset incorporates higher levels of randomness, modeling scenarios with even less predictability in truck arrival times. This dataset enables testing how the terminal responds to truck arrivals that are not coordinated with the train schedules, allowing for the examination of resource utilization and system flexibility in the face of uncertainty.

Other datasets, such as `truck3_gaussian.csv` and `truck3_poisson.csv`, are similar to the first two datasets but simulate conditions with higher truck loads and more complex operational setups. The `truck4.csv` dataset serves as a baseline, likely representing real-world data or a simplified arrival pattern, providing a point of comparison for testing various synchronization and loading conditions.

3.7.2 Train Schedule Datasets

The train schedules define the timing of train arrivals and departures at the terminal and play a key role in determining the synchronization between trucks and trains. Synchronization refers to how closely truck arrivals are aligned with the train schedules. For instance, the `train1.csv` dataset represents a tightly synchronized schedule, where truck arrivals are closely timed with train arrivals. This tight synchronization ensures minimal waiting times, as trucks arrive just before trains are ready to load or unload.

In scenarios with looser synchronization, such as those using the `train2.csv` dataset, the alignment between truck and train arrivals is less precise. Trucks may arrive well before or after trains, creating potential inefficiencies, such as increased waiting times and lower resource utilization. The `train3.csv` dataset represents a more complex scenario, with a higher volume of Intermodal Transport Units (ITUs) and greater variability in train arrivals. This dataset is paired with different truck datasets to test how varying synchronization affects the terminal's ability to handle larger volumes of traffic.

3.7.3 Scenario Configurations

The input datasets are combined into six distinct scenarios, each testing the effects of different truck arrival patterns and train schedules on terminal performance. In

Scenario 1, the `truck1_gaussian.csv` dataset is paired with `train1.csv` to simulate a tightly synchronized terminal, where trucks arrive in close coordination with the train schedules. This scenario focuses on testing the efficiency of the terminal when truck arrivals are predictable and aligned with train arrivals, minimizing idle times and optimizing resource use.

Scenario 2 uses `truck1_poisson.csv` and `train1.csv`, maintaining tight synchronization but with more randomness in truck arrivals. This setup allows exploration of how the terminal handles variability in truck arrivals while keeping train schedules constant. Scenario 3 introduces `truck2_modified_randomness.csv` with `train2.csv`, simulating a loosely synchronized terminal with higher unpredictability in truck arrivals and less coordination with train schedules. This scenario assesses the impact of less structured operations on waiting times and crane utilization.

Scenarios 4 and 5 involve higher truck loads, using `truck3_gaussian.csv` and `truck3_poisson.csv`, respectively, paired with `train3.csv` to test how the terminal manages larger volumes under different synchronization conditions. These scenarios help to evaluate the trade-offs between resource allocation and synchronization when handling more significant demands. Finally, Scenario 6 combines `truck4.csv` with `train3.csv`, providing a comparative baseline to analyze how synchronization and varying truck arrival patterns influence terminal efficiency.

3.7.4 Key Variables

The datasets capture several key variables essential for analyzing terminal performance. **Truck arrival times** are a primary variable, representing different arrival patterns and allowing the assessment of waiting times, crane utilization, and throughput under varying conditions. The **train schedules** define when trains arrive and depart, impacting the degree of synchronization between truck and train operations. The **number of ITUs** being handled, along with the **number of cranes and road gates** available in each scenario, further influences how well the terminal adapts to different operational loads and configurations.

3.8 Scenario Design

The scenarios in the simulation model are constructed to evaluate the effects of several critical variables on the performance of intermodal terminal operations. These variables include the degree of synchronization between truck arrivals and train schedules, resource configurations, and varying levels of ITU (Intermodal Transport Units) traffic. Each of these aspects is explored to understand their impact on key performance indicators (KPIs) such as truck waiting times, crane utilization, and overall terminal throughput.

3.8.1 Truck-Train Synchronization

This variable focuses on assessing how the synchronization between truck arrivals and train departures influences terminal performance. Three levels of synchronization are tested:

- **Tight Synchronization**: In this case, truck arrivals are closely timed with train arrivals, minimizing idle times and maximizing the efficiency of loading and unloading operations. This scenario is designed to test whether near-perfect coordination can optimize resource utilization and reduce waiting times.
- **Loose Synchronization**: Here, truck arrivals are less strictly coordinated with train schedules. Trucks may arrive significantly earlier or later than trains, introducing potential inefficiencies. This scenario helps determine the extent to which loose synchronization impacts terminal operations, including increased truck waiting times and reduced throughput.
- **Random Synchronization**: In this scenario, truck arrivals follow a more unpredictable pattern, with no clear alignment to train schedules. The randomness simulates real-world uncertainties, where delays and disruptions affect the synchronization between different transport modes. This setup tests the system's robustness and flexibility under less controlled conditions.

The aim of testing these synchronization levels is to analyze how closely coordinated operations (tight synchronization) compare to more irregular or loosely coordinated schedules, in terms of improving terminal performance.

3.8.2 Resource Configurations

The resource configurations in the simulation model are designed to explore the effects of different allocations of terminal resources—cranes and road gates—on overall operational efficiency. Specifically, the configurations aim to evaluate how variations in the number of cranes and road gates impact key performance indicators such as truck waiting times, crane utilization rates, and overall terminal throughput. Multiple combinations of resources are considered:

- **Number of Cranes**: Scenarios include configurations with 1, 2, and, in some cases, more cranes to investigate how increasing crane availability affects system performance. The simulation seeks to determine whether adding more cranes leads to higher throughput and lower waiting times or whether it results in diminishing returns due to potential underutilization or over-allocation. In high-demand scenarios, multiple cranes are expected to share the workload more efficiently, while in lower-demand scenarios, fewer cranes may be sufficient without sacrificing performance.
- **Number of Road Gates**: The number of road gates is varied between 1 and 2 to assess the impact on truck processing times. A higher number of gates is anticipated to reduce queuing at the terminal entrance, as multiple trucks can be processed simultaneously. However, it is important to examine whether the increase in infrastructure, such as adding an extra gate, significantly improves terminal operations or whether the benefit is marginal. The simulation considers scenarios where trucks arrive in both synchronized and unsynchronized patterns relative to train schedules, testing the gate configurations under different levels of coordination and congestion.

- **Combined Resource Configurations**: Several scenarios combine different crane and road gate setups (e.g., 1 crane with 1 road gate, 2 cranes with 2 road gates, etc.) to explore how a balanced or unbalanced allocation of resources affects terminal performance. For example, increasing the number of cranes without adjusting the number of road gates may lead to bottlenecks at the terminal entrance, while adding more gates without increasing crane capacity may create inefficiencies in ITU handling. This approach helps to identify the most effective combination of cranes and road gates to optimize system flow under varying traffic loads.
- **Dynamic Resource Allocation**: Additionally, the model evaluates scenarios where resource allocation (cranes and road gates) is dynamically adjusted based on queue lengths or operational demands. For instance, cranes may be reallocated based on the length of the truck queue at a particular time, or additional road gates may be opened during peak hours. This dynamic allocation is tested to measure its ability to reduce idle times and respond effectively to changing traffic conditions.

Through these various resource configurations, the model aims to determine how the interplay between crane availability, road gate infrastructure, and truck-train synchronization influences overall terminal efficiency. The goal is to identify optimal resource allocations that maximize throughput and minimize waiting times without incurring unnecessary operational costs.

3.8.3 ITU Load Variations

This variable introduces different levels of ITU traffic to evaluate the terminal's performance under varying demand conditions. Two distinct load scenarios are tested:

- **Low ITU Load**: A scenario with a smaller number of ITUs is simulated to represent periods of low demand. This allows for testing how efficiently the terminal operates with fewer resources engaged and whether it can maintain high performance without resource overcommitment.
- **High ITU Load**: A high-traffic scenario, simulating peak demand with a large number of ITUs, is used to assess the terminal's capacity to handle increased volumes. The performance is measured in terms of potential bottlenecks, waiting times, and resource saturation, testing whether the system can scale up effectively under heavy load conditions.

These scenarios provide insights into the terminal's sensitivity to load variations and help determine whether performance bottlenecks emerge when demand exceeds typical operational capacity.

3.9 Key Performance Indicators (KPIs)

To assess the performance of the terminal under different scenarios, several KPIs are used:

- **Average Truck Waiting Time**: The average time a truck waits at the terminal before its ITU is loaded or unloaded.
- **Crane Utilization Rate**: The percentage of time cranes are actively working, indicating the efficiency of crane operations.
- **Throughput**: The total number of ITUs processed by the terminal during the simulation.
- **Percentage of Direct Transshipments**: The proportion of ITUs that are directly transferred between trucks and trains without being stored in the yard.
- **Storage Yard Utilization**: The percentage of storage yard capacity that is utilized during the simulation.

3.10 Model Validation and Verification

The model was validated using real-world data provided by a case study of a terminal operation. The input parameters, such as truck arrival times, crane operation times, and storage capacities, were compared with actual terminal data to ensure that the model accurately reflects real-world operations. Verification of the model was achieved through extensive testing, including sensitivity analysis to ensure that the model behaves as expected under different input conditions.

3.10.1 Sensitivity Analysis

Sensitivity analysis was conducted by varying key parameters such as truck arrival frequency, crane speed, and ITU processing time. This analysis highlights the factors that most influence terminal performance and identifies potential bottlenecks in the system.

3.11 Simulation Execution and Analysis

The DES model was executed for each scenario, simulating one week operational period. For each scenario, the KPIs were recorded and analyzed to compare the effects of different synchronization patterns, resource configurations, and ITU load levels. A thorough analysis of the results provided insights into the optimal operational strategies for the terminal.

Results are presented using line graphs, bar charts, and scatter plots to visualize trends and relationships between KPIs under different scenarios. For example, graphs of truck waiting times across different crane configurations help to identify the optimal number of cranes required to minimize delays. Heatmaps were also used to represent resource utilization patterns, showing peak times of operation and identifying bottlenecks.

3.12 Case Study: Piedimonte San Germano Terminal

To validate the proposed simulation model and explore its practical applications, a case study was conducted using data representative of a typical intermodal terminal. This section details the specific parameters employed in the simulation, encompassing structural characteristics, demand scenarios, and operational assumptions.

Structural Parameters

The physical and operational attributes of the simulated terminal were defined as follows:

- **Operational Days per Year:** 270
- **Daily Operational Hours:** 24

  Infrastructure

  - Transshipment Tracks: 2 tracks, each 500 meters in length

  Equipment

  - Cranes: 2 cranes
  - Crane Operation Time: 4 minutes per move
  - Crane Allocation: One crane allocated per track

  Operational Strategies

  - Train Loading/Unloading: Sequential process, progressing from the first to the last wagon

  Rail Operations

  - Train Entry Time: 30 minutes
  - Train Exit Time: 60 minutes
  - Interval between Unloading and Loading: 10 minutes

  Road Operations

  - Road Gates: 1 gate
  - Gate Service Time: 3 minutes per truck

  Capacity

  - Truck Parking: 50 spaces
  - Storage Area: 1000 ITU spaces
  - Buffer Area: Integrated with the main storage area

Demand Scenarios

The simulation encompassed the following demand parameters:

- **Temporal Scope**
  - Simulation Duration: 7 days (168 hours)

  ITU Characteristics
  - Total ITUs: 200
  - Composition: 70
  - Length Distribution: 30
  - Weight Range: 5-30 tons (randomly distributed)
  - Special Categories: 5

  Train Operations
  - Daily Train Frequency: 4 trains (2 arrivals, 2 departures)
  - Routes: Terminal A Terminal B
  - Schedule:
    - Arrivals: 07:00 and 09:00
    - Departures: 18:00 and 20:00
  - Track Utilization: Alternating between available tracks
  - Train Capacity: 50 ITUs

Management Assumptions

The simulation incorporated the following operational parameters:

- Synchronization Parameters:
  - W1 (Early Synchronization Limit): -5 minutes
  - W2 (Late Synchronization Limit): +5 minutes
- ITU Handling Thresholds:
  - T1 (Parking to Buffer Transition): 3 hours
  - T2 (Buffer to Storage Transition): Not applicable (integrated storage)
  - T3 (Waiting ITU Allocation): Not applicable (integrated storage)
- Storage Yard Utilization Thresholds:
  - g1 (First Level): 0.5 (50
  - g2 (Second Level): 1.0 (100
  - g3 (Third Level): Not applicable (maximum stack height is 2)

3.13 Road Gate Operation Modeling Using Salabim

One of the critical components of intermodal terminal operations is the road gate, where trucks arrive, are processed, and either enter or leave the terminal. To better visualize and understand this process, an animation was developed using the `Salabim` library in Python, which specializes in discrete-event simulation (DES).

The animation simulates the arrival, waiting, and processing of trucks at the road gate. Each truck is represented as a color-coded object, where:

- **Green**: Pickup
- **Blue**: Delivery
- **Orange**: Both pickup and delivery

The trucks queue at the road gate, and their movements are visualized as they proceed through the gate. This animation provides an intuitive understanding of the dynamics at the gate, particularly the bottlenecks that can arise during peak operation times.

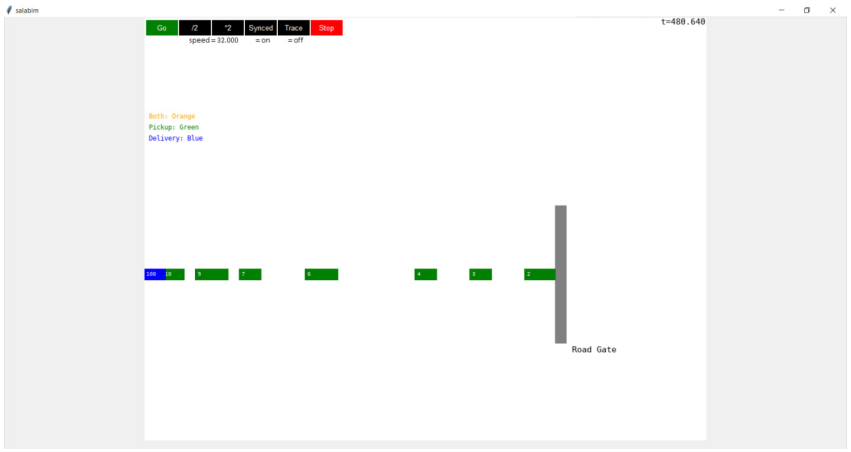


Figure 3.1. Salabim Animation of the Road Gate Operation for Scenario X

The animation was run across different scenarios that were tested during the simulation. By comparing animations for different configurations (e.g., varying numbers of gates or trucks), it becomes easier to understand the impact of synchronization on gate performance. The animation also illustrates truck dwell times and the overall flow efficiency at the road gate under various configurations.

This animation allows for visual insight into the simulated road gate operation process. It showcases how factors such as truck arrival patterns (Poisson, Gaussian, or random), the number of road gates, and overall terminal resource management directly affect operational efficiency at the gate.

Chapter 4

Results and Discussion

4.1 Analysis of Examined Scenarios

In this section, we present an in-depth analysis of the six scenarios examined in the simulation study. These scenarios differ based on the degree of synchronization between truck arrivals and train schedules, the arrival distribution type, and the demand levels. A comparison of their key results provides insights into how these factors affect intermodal terminal performance.

4.1.1 Scenario 1: Tight Synchronization with Gaussian Arrival Distribution

In Scenario 1, truck arrivals were tightly synchronized with train schedules and followed a **Gaussian distribution**. This high level of coordination between trucks and trains resulted in relatively low average truck waiting times (around 2.5 to 4.5 minutes, depending on resource configuration) and a moderate yard utilization rate of 12.79%. The crane utilization rate varied based on the number of available cranes, but it was generally lower with more cranes in operation due to improved resource balancing. The percentage of direct transshipments was 7.86%, indicating that tight synchronization, despite a Gaussian distribution, contributed to better system performance, though there was still room for improvement in transshipment efficiency.

4.1.2 Scenario 2: Tight Synchronization with Poisson Arrival Distribution

In Scenario 2, truck arrivals also maintained **tight synchronization** with train schedules but followed a **Poisson distribution**, introducing more randomness compared to Scenario 1. Despite this increased variability, the system performed similarly, with average truck waiting times remaining around 2.5 to 4.5 minutes, depending on the configuration. The storage yard utilization rate was slightly higher at 13.17%, suggesting that the terminal could handle the additional variability without significant impact. The percentage of direct transshipments increased to 8.43%, indicating that tight synchronization was effective in mitigating the effects of random truck arrivals, leading to improved efficiency in handling operations.

4.1.3 Scenario 3: Loose Synchronization with Modified Randomness

Scenario 3 introduced **loose synchronization** with **modified randomness** in truck arrivals, meaning there was less precise alignment between truck arrivals and train schedules. This led to slightly increased yard utilization at 13.83%, which was higher than the previous scenarios, but the system struggled to maintain efficiency in terms of direct transshipments, which decreased to 7.14%. The random arrival patterns and reduced synchronization negatively impacted the efficiency of direct transfers, even though overall storage utilization improved. This scenario illustrates the challenges of managing a less structured environment.

4.1.4 Scenario 4: Tight Synchronization with Gaussian Arrival Distribution Under Higher Load

In Scenario 4, truck arrivals followed a **Gaussian distribution** but under a **higher load** condition, with tight synchronization maintained between truck and train schedules. This scenario demonstrated significantly improved performance metrics, with the storage yard utilization rate increasing to 25.37%. The percentage of direct transshipments also increased to 8.14%, highlighting the benefits of tighter coordination under higher demand conditions. Additionally, crane utilization was optimized when more cranes were available, showing that resource allocation played a critical role in handling the increased workload efficiently.

4.1.5 Scenario 5: Tight Synchronization with Poisson Arrival Distribution Under Higher Load

Scenario 5 tested **tight synchronization** between truck arrivals and train schedules, with trucks arriving according to a **Poisson distribution** under a **higher load**. Despite the higher demand and stochastic nature of Poisson arrivals, the system performed well, with a storage yard utilization rate of 25.00%. The most significant improvement was seen in direct transshipments, which increased to 10.71%, reflecting the terminal's ability to handle increased volumes while maintaining high efficiency in direct transfers. This scenario shows that tight synchronization can offset the randomness of arrivals, even under higher demand.

4.1.6 Scenario 6: High-Demand Scenario with Tight Synchronization

In Scenario 6, the terminal was pushed to its maximum capacity under a **high-demand scenario** with **tight synchronization** between truck and train operations. The storage yard utilization rate peaked at 27.04%, demonstrating the terminal's ability to manage a high volume of ITUs. However, the percentage of direct transshipments decreased to 7.21%, indicating that while the terminal could handle the increased load, there was a trade-off in maintaining synchronization for direct transshipment efficiency. This scenario highlights the challenge of balancing high demand with the need for precise coordination in order to optimize terminal performance.

4.1.7 Summary of Results

Across the scenarios, the results show that **tight synchronization** generally leads to better terminal performance in terms of truck waiting times, crane utilization, and direct transshipments. Scenarios 1 and 2, with Gaussian and Poisson distributions under tight synchronization, revealed that a more predictable arrival pattern (Gaussian) led to slightly better performance, but the terminal was still able to maintain high efficiency even with the randomness introduced by Poisson arrivals.

In **higher load scenarios** (Scenarios 4 and 5), tight synchronization proved essential in managing increased demand. The combination of tighter alignment between truck arrivals and train schedules, coupled with optimized resource configurations, significantly improved yard utilization and transshipment efficiency. However, in the **high-demand scenario** (Scenario 6), the terminal reached its operational limits, resulting in a drop in direct transshipment efficiency despite tight synchronization.

4.1.8 Discussion of Key Findings

The six scenarios explored in this study demonstrate the critical impact of synchronization, arrival distributions, and demand levels on terminal performance:

- **Synchronization as a Key Efficiency Driver:** Scenarios with **tight synchronization** (Scenarios 1, 2, 4, 5, and 6) consistently exhibited improved performance metrics. These scenarios achieved higher storage yard utilization rates and a greater percentage of direct transshipments compared to those with **loose synchronization** (Scenario 3). This demonstrates the importance of aligning truck arrivals closely with train schedules to minimize idle times and improve overall terminal efficiency.

- **Impact of Arrival Distributions:** The distribution of truck arrivals also influenced performance. Scenarios with **Gaussian distributions** (Scenarios 1 and 4) resulted in slightly lower variability in yard utilization and direct transshipments, providing a more predictable arrival pattern. On the other hand, **Poisson distributions** (Scenarios 2 and 5) introduced higher variability in arrivals, but under tight synchronization, they led to improved efficiency, particularly in Scenario 5 where direct transshipments reached 10.71%. This suggests that while stochastic arrivals may increase variability, tight synchronization can help offset the effects and enhance overall operational outcomes.

- **Handling High Demand:** Scenario 6 highlighted the terminal's capability to manage increased demand effectively, but it also exposed limitations in flexibility, particularly in direct transshipments. High yard utilization rates (27.04%) demonstrated efficient space usage under tight synchronization, yet the percentage of direct transshipments decreased to 7.21%. This reflects the operational pressures and challenges of maintaining synchronization and handling large volumes simultaneously, suggesting a trade-off between handling peak loads and maintaining synchronization efficiency.

- **Resource Utilization Insights:** Crane utilization rates varied across scenarios and configurations. In scenarios with **tight synchronization** (Scenarios 1, 2, 4, 5, and 6), crane utilization percentages were lower when more cranes were available, indicating a more efficient distribution of the workload. This suggests that tight synchronization, combined with optimal resource allocation, can reduce bottlenecks and improve the overall balance of workload across available cranes. It underscores the importance of matching resource availability with synchronization strategies to maximize efficiency.

The findings indicate that improving synchronization between truck arrivals and train schedules is essential for enhancing terminal performance. Furthermore, selecting appropriate arrival distributions and optimizing demand levels can help balance efficient yard space utilization, reduce waiting times, and allocate resources effectively.

4.2 Average Truck Waiting Time

The table of average truck waiting time across different resource configurations clearly demonstrates that increasing the number of **road gates** significantly reduces truck waiting times, while adding more cranes alone has a limited impact. For instance, when only one gate is available, the waiting time remains high at **4.5 minutes**, regardless of the number of cranes in operation. This indicates that a single road gate creates a bottleneck that cannot be alleviated by increasing crane numbers.

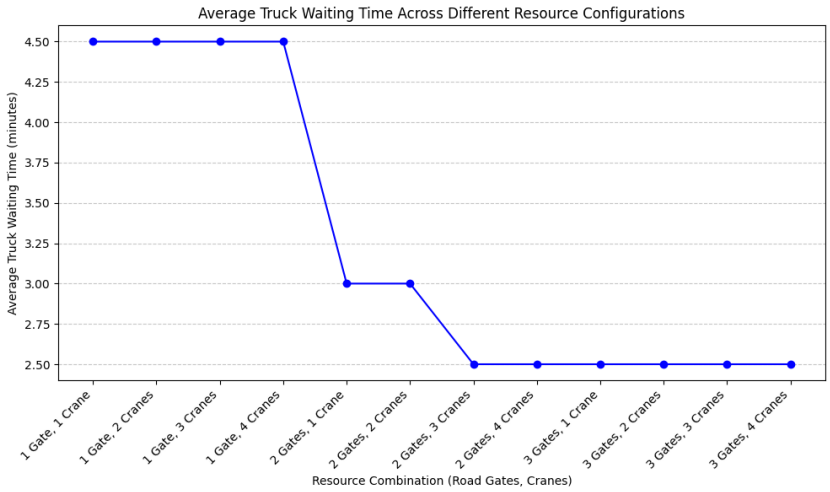
However, when a second or third gate is added, waiting times decrease dramatically. Adding a second gate reduces the waiting time to **3.0 minutes**, and the addition of a third gate further reduces it to **2.5 minutes**. This trend highlights that road gate availability is a critical factor in terminal efficiency, particularly in minimizing truck waiting times.

The chart and table also show that increasing the number of cranes, without a corresponding increase in road gates, leads to **diminishing returns**. For optimal terminal performance, the focus should be on expanding gate capacity in combination with crane resources to balance the workload effectively and reduce overall truck waiting times.

Therefore, terminal management should prioritize increasing the number of road gates alongside cranes to address bottlenecks, ensure smoother traffic flow, and optimize resource utilization. This approach would result in significantly shorter truck waiting times, maximizing operational efficiency across the terminal.

Table 4.1. Average Truck Waiting Time (minutes) Across Resource Combinations

Config.	1G, 1C	1G, 2C	1G, 3C	1G, 4C	2G, 1C	2G, 2C	2G, 3C	2G, 4C	3G, 1C	3G, 2C
Wait Time (min)	4.5	4.5	4.5	4.5	3.0	3.0	3.0	3.0	2.5	2.5
Config.	3G, 3C	3G, 4C								
Wait Time (min)	2.5	2.5								



4.3 Crane Utilization Rate

Crane utilization rates vary significantly across the different scenarios, reflecting the impact of synchronization and resource allocation on overall terminal efficiency. In **Scenario 1**, crane utilization is the highest, primarily due to the limited synchronization and fewer available resources, as shown in Figure 6.2. The lack of synchronization leads to more frequent use of cranes, but this does not necessarily translate to higher efficiency.

In **Scenarios 4 and 5**, with tighter synchronization between truck arrivals and train schedules (using Gaussian and Poisson distributions), crane utilization becomes more balanced. This improved balance suggests that resources are being allocated more efficiently, reducing idle times and distributing the workload more evenly across the available cranes.

Scenario 6, which involves the highest ITU traffic under tight synchronization and maximum operational load, shows increased crane utilization. The high demand in this scenario pushes the terminal to its capacity, leading to more frequent crane operations. Although crane utilization is higher, it is also important to note that this scenario highlights the need for balanced resource allocation to prevent overuse and potential bottlenecks in the system.

Overall, the results indicate that while high crane utilization can be a sign of resource maximization, it should be balanced with proper gate allocation and synchronization strategies to ensure operational efficiency without overburdening specific resources.

Table 4.2. Crane Utilization Rates (%) Across Scenarios and Resource Combinations

Scenario	1 Road Gate, 1 Crane	1 Road Gate, 2 Cranes	1 Road Gate, 3 Cranes	1 Road Gate, 4 Cranes	2 Road Gates, 1 Crane	2 Road Gates, 2 Cranes	2 Road Gates, 3 Cranes	2 Road Gates, 4 Cranes	3 Road Gates, 1 Crane	3 Road Gates, 2 Cranes	3 Road Gates, 3 Cranes	3 Road Gates, 4 Cranes
Scenario 1	27.78	13.89	9.26	6.94	27.78	13.89	9.26	6.94	27.78	13.89	9.26	6.94
Scenario 2	27.78	13.89	9.26	6.94	27.78	13.89	9.26	6.94	27.78	13.89	9.26	6.94
Scenario 3	27.78	13.89	9.26	6.94	27.78	13.89	9.26	6.94	27.78	13.89	9.26	6.94
Scenario 4	55.56	27.78	18.52	13.89	55.56	27.78	18.52	13.89	55.56	27.78	18.52	13.89
Scenario 5	55.56	27.78	18.52	13.89	55.56	27.78	18.52	13.89	55.56	27.78	18.52	13.89
Scenario 6	55.56	27.78	18.52	13.89	55.56	27.78	18.52	13.89	55.56	27.78	18.52	13.89

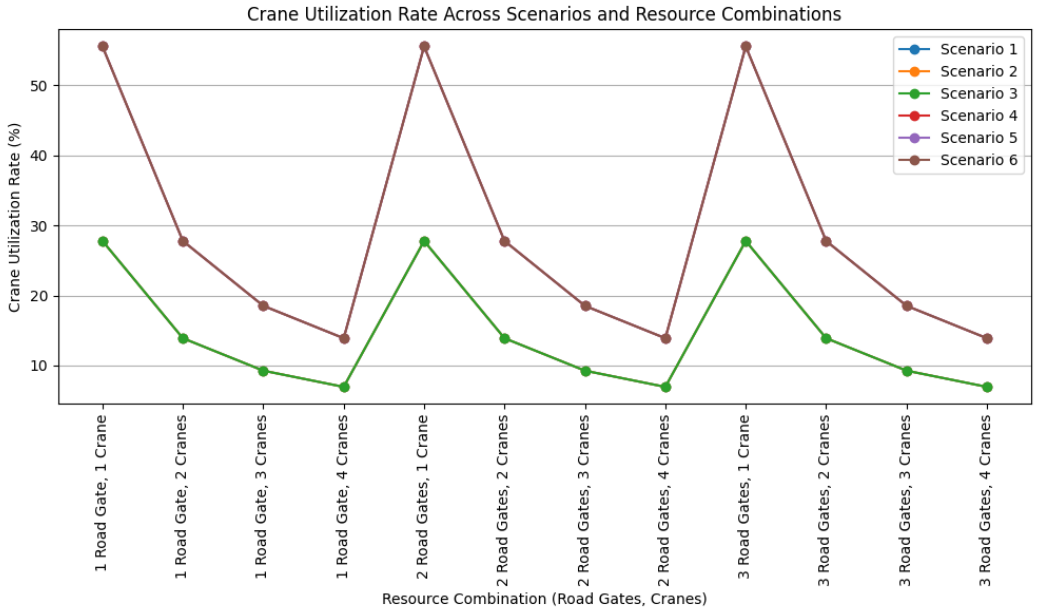


Figure 4.1. Crane Utilization Rate Across Scenarios and Resource Combinations. The chart illustrates the impact of different resource combinations on crane utilization across the six scenarios. It highlights how increasing the number of cranes reduces crane utilization rates, particularly in high-load scenarios.

Table 4.2 illustrates crane utilization rates across six different scenarios under various combinations of road gates and cranes. The data shows that crane utilization decreases as the number of cranes increases, suggesting that the workload is distributed more evenly when more cranes are available, thereby reducing the utilization rate of each crane. For example, in **Scenario 4**, crane utilization drops from **55.56%** with one crane to **13.89%** with four cranes. Scenarios 4, 5, and 6, which represent higher ITU loads and varying levels of synchronization, consistently demonstrate higher crane utilization compared to Scenarios 1, 2, and 3, particularly

when fewer cranes are employed. This trend indicates that increased ITU demand leads to greater crane activity, resulting in higher utilization when crane resources are limited.

Additionally, the number of road gates does not significantly impact crane utilization, as the primary factor influencing crane usage is the number of cranes available in conjunction with overall ITU demand. However, while additional road gates may help reduce truck waiting times, they do not directly affect crane workload. These findings suggest that a balance should be struck between the number of cranes and expected ITU demand—higher crane utilization rates may signal potential bottlenecks, while very low utilization could indicate underutilized resources. These insights offer valuable guidance for making strategic decisions on optimal resource allocation in intermodal terminals.

4.3.1 Practical Implications

The crane utilization rate, as represented in Table 4.2, provides important practical implications for decision-making in intermodal terminal management. Specifically, this metric can guide the optimal allocation of cranes to balance efficiency and resource use effectively.

When the crane utilization rate is too high, it suggests that cranes are heavily utilized, which could lead to potential bottlenecks during peak periods, delaying operations. On the other hand, very low utilization rates may imply that resources are underutilized, resulting in unnecessary costs without adding operational benefits. Therefore, an optimal number of cranes should be considered to handle the expected workload during both regular and peak times.

For example, the data in **Scenario 4** shows that crane utilization drops significantly as the number of cranes increases from one to four, suggesting that there is a diminishing return on adding more cranes beyond a certain point. Decision-makers can use this insight to avoid overinvestment in cranes, ensuring they strike a balance between having enough capacity for peak demand and minimizing costs during normal operations.

In practice, terminal operators should carefully assess the demand patterns and synchronize resource allocation accordingly. Higher crane utilization might be acceptable if the terminal experiences frequent peak operations, but during regular times, adding too many cranes could lead to idle resources. By understanding and acting on this balance, terminal managers can optimize crane allocation, reduce operational costs, and improve overall efficiency.

4.4 Storage Yard Utilization

The storage yard utilization rate, as depicted in Table 4.3, highlights the impact of different truck-train synchronization scenarios on yard efficiency. **Scenarios 4, 5, and 6**, which involve increased ITU loads or different synchronization patterns, exhibit significantly higher utilization rates compared to **Scenarios 1, 2, and 3**. The highest utilization rate is observed in **Scenario 6** (High Demand), which demonstrates the effect of increased ITU volumes on yard congestion and storage requirements.

Scenarios 1 and 2, featuring Gaussian and Poisson truck arrival patterns respectively, display lower utilization rates. These scenarios suggest that when truck arrivals are spread out and loosely synchronized, the demand on yard capacity is reduced. This is consistent with lower yard utilization when synchronization is less stringent, allowing for smoother, more gradual use of storage space without overwhelming the system.

Overall, the findings indicate that higher synchronization levels, particularly under increased demand, contribute to higher yard utilization rates, with **Scenario 6** clearly showcasing the challenges associated with handling peak volumes and the importance of proper yard capacity management.

Table 4.3. Storage Yard Utilization Rate (%) Across Different Synchronization Scenarios

Synchronization Scenario	Storage Yard Utilization Rate (%)
Scenario 1 (Gaussian, Train 1)	12.79
Scenario 2 (Poisson, Train 1)	13.17
Scenario 3 (Modified Randomness, Train 2)	13.83
Scenario 4 (Gaussian, Train 3)	25.37
Scenario 5 (Poisson, Train 3)	25.00
Scenario 6 (High Demand, Train 3)	27.04

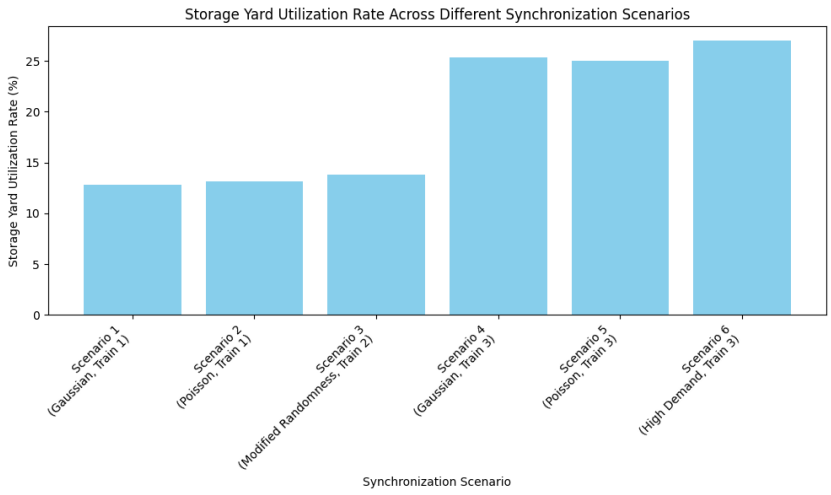


Figure 4.2. Storage Yard Utilization Rate (%) Across Different Synchronization Scenarios. The chart illustrates the storage yard utilization across the six different scenarios, highlighting significant differences based on truck-train synchronization patterns.

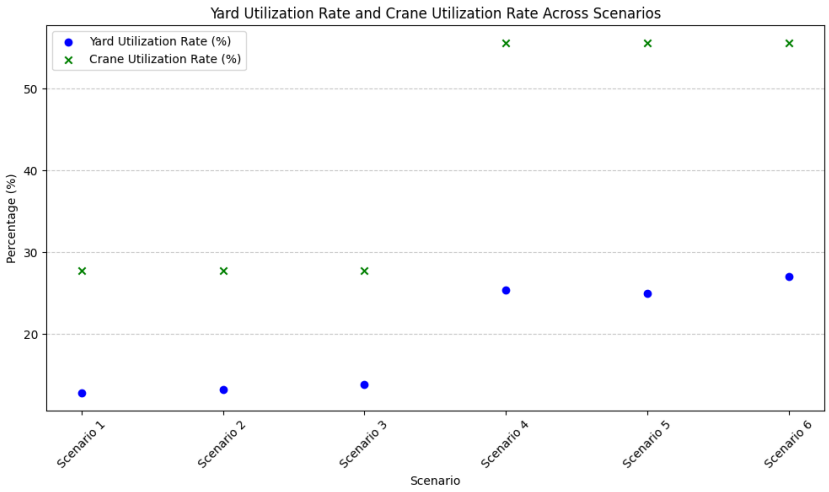


Figure 4.3. Scatter Plot of Yard Utilization vs. Crane Utilization across Scenarios

The scatter plot in Figure 4.3 illustrates the relationship between yard utilization and crane utilization across different scenarios. Each point represents a specific scenario, showing its corresponding crane and yard utilization rates. The plot highlights how variations in crane availability and yard usage are correlated. Generally, higher crane utilization is associated with higher yard utilization, indicating that increased yard activity is matched by greater crane operations. This visualization provides valuable insights into the interaction between crane usage and yard utilization, helping to better understand how terminal resources are managed under different operating conditions.

4.5 Percentage of Direct Transshipments

The percentage of direct transshipments varies across the different scenarios, reflecting the impact of truck-train synchronization and demand levels on the efficient handling of ITUs. **Scenario 5**, which uses a Poisson distribution for truck arrivals, exhibits the highest percentage of direct transshipments at **10.71%**, indicating that the synchronization in this scenario is particularly effective for facilitating direct transfers.

In contrast, **Scenario 6**, which operates under high demand conditions, shows a relatively lower percentage of direct transshipments at **7.21%**. This suggests that the increased volume of ITUs in Scenario 6 leads to a greater reliance on intermediate storage, thereby reducing the number of direct transfers.

Overall, the results indicate that higher synchronization generally correlates with a higher percentage of direct transshipments, improving operational efficiency by minimizing the need for storage and handling. These findings underscore the importance of maintaining optimal synchronization to maximize direct transshipments and enhance terminal performance.

Table 4.4. Percentage of Direct Transshipments Across Different Synchronization Scenarios

Synchronization Scenario	Percentage of Direct Transshipments (%)
Scenario 1 (Gaussian, Train 1)	7.86
Scenario 2 (Poisson, Train 1)	8.43
Scenario 3 (Modified Randomness, Train 2)	7.14
Scenario 4 (Gaussian, Train 3)	8.14
Scenario 5 (Poisson, Train 3)	10.71
Scenario 6 (High Demand, Train 3)	7.21

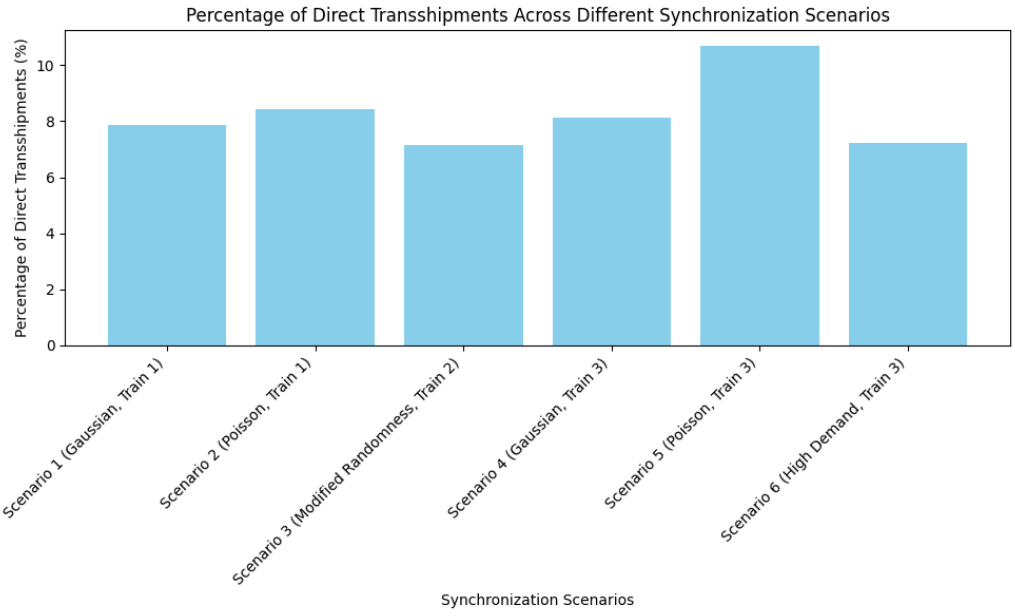


Figure 4.4. Percentage of Direct Transshipments Across Different Synchronization Scenarios. Scenario 5, employing a Poisson distribution for truck arrivals, shows the highest direct transshipments rate, indicating improved synchronization and efficiency. In contrast, high demand in Scenario 6 shows reduced direct transshipments.

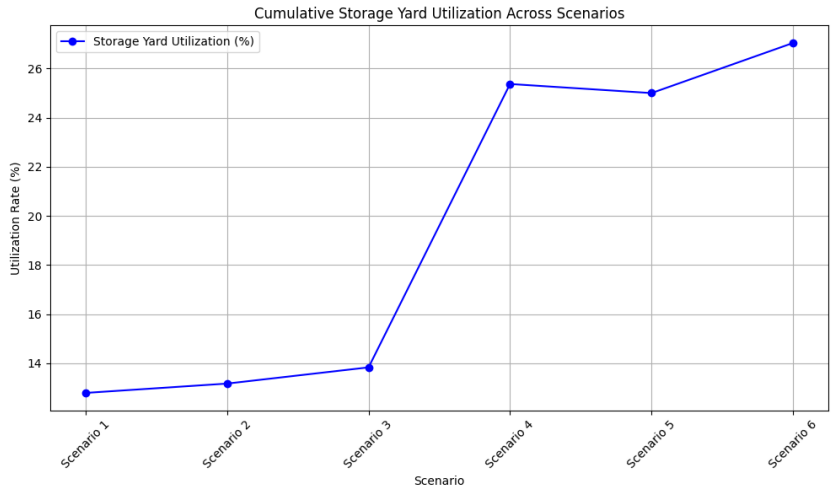


Figure 4.5. Cumulative Storage Yard Utilization Rate (%) Across Different Synchronization Scenarios. The chart shows how increased synchronization levels and demand significantly impact yard utilization, with Scenario 6 reaching the highest utilization level.

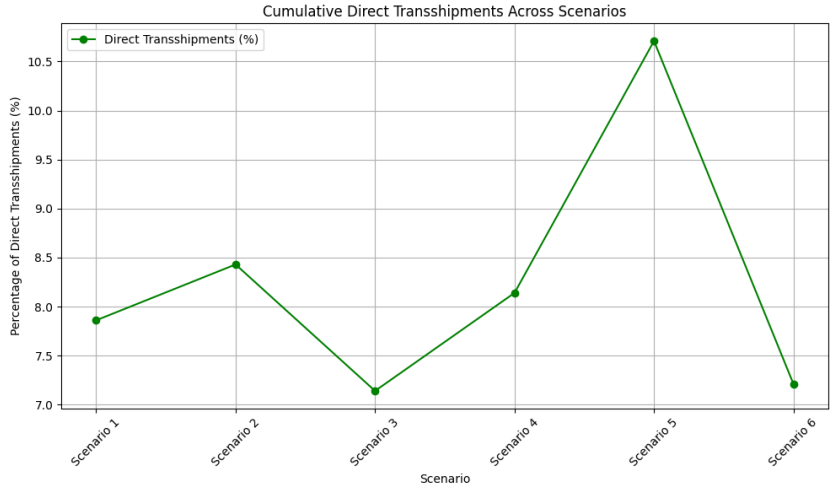


Figure 4.6. Cumulative Direct Transshipments Rate (%) Across Different Synchronization Scenarios. The chart illustrates the proportion of ITUs transshipped directly from trucks to trains, with Scenario 5 achieving the highest direct transshipment rate due to improved synchronization.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

This study investigated the optimization of intermodal terminal operations through discrete event simulation, focusing on resource utilization and truck-train synchronization. By developing a comprehensive simulation model using Python, SimPy, and Salabim libraries, we tested multiple scenarios ranging from simple resource allocations to complex arrival patterns, including Poisson, Gaussian, and randomized distributions.

Our findings highlight the significant impact of synchronization between truck arrivals and train schedules on terminal efficiency. Tighter synchronization led to substantial improvements in terminal performance, with **Scenario 5** (tight synchronization and Poisson-distributed arrivals) showing the highest percentage of direct transshipments at **10.71%**. This indicates that implementing advanced scheduling systems could substantially enhance terminal efficiency by improving synchronization.

Resource allocation also played a crucial role in terminal performance. Increasing the number of cranes from one to four reduced crane utilization rates by up to 75% in high-demand scenarios. While this suggests that additional resources can better distribute the workload, it also highlights the risk of underutilization if not balanced with demand. Therefore, we recommend adopting dynamic resource allocation strategies to optimize crane usage based on real-time demand.

The study revealed that Poisson-distributed truck arrivals, when tightly synchronized with train schedules, resulted in higher operational efficiency compared to Gaussian distributions, particularly in terms of direct transshipments and storage yard utilization. This finding suggests that terminal operators should consider using Poisson distribution models for scheduling truck arrivals to maximize efficiency.

In the high-demand scenario (**Scenario 6**), we observed a trade-off between handling higher volumes and maintaining efficient direct transfers. While this scenario achieved a **27.04%** storage yard utilization rate, it also saw a reduction in direct transshipments to **7.21%**, highlighting the challenge of balancing high throughput with operational efficiency. This underscores the need for strategies to balance handling high volumes with maintaining synchronization.

Our analysis also identified road gates as a critical bottleneck in terminal opera-

tions. Increasing the number of road gates from one to three reduced average truck waiting times by **44%**, suggesting that investment in additional road gate capacity should be prioritized to improve overall terminal throughput and reduce delays.

These findings have important implications for terminal operators and logistics planners. They underscore the importance of implementing advanced scheduling systems and real-time data integration to enhance synchronization between different transport modes. Moreover, the results suggest that terminal designs should incorporate flexibility to adapt to varying demand levels and synchronization requirements.

5.2 Future Work

While this research provides valuable insights, several areas warrant further investigation. Future work could explore the integration of machine learning algorithms with the simulation model, enabling predictive analytics for demand forecasting and dynamic resource allocation. Incorporating real-time data from operational terminals could enhance the model's accuracy and provide more up-to-date simulations.

Expanding the scope to simulate a network of intermodal terminals could offer insights into system-wide optimizations and the cascading effects of operational changes. Moreover, incorporating environmental factors into the simulation, such as calculating CO2 emissions based on terminal operations, could provide valuable data for sustainability efforts in the logistics sector. Integrating economic factors, including operational costs and revenue streams, could offer a more comprehensive view of terminal performance and guide investment decisions.

Further research could also explore stochastic variability, such as equipment breakdowns or extreme weather events. Developing a user-friendly interface for the simulation tool could make it more accessible to practitioners. Comparative studies with other terminals globally, the development of optimization algorithms for resource configurations and scheduling strategies, and the exploration of blockchain technology for enhancing security and traceability in freight movements are additional research directions that could enhance this work.

The simulation model developed in this study serves as a valuable tool for analyzing and optimizing intermodal terminal operations. Its ability to test various scenarios offers insights that can guide decision-making in resource allocation, scheduling, and infrastructure planning. As global trade continues to evolve, advancements in simulation and optimization techniques will play a crucial role in shaping more efficient, resilient, and sustainable freight transportation systems.

In conclusion, this research contributes significantly to our understanding of intermodal terminal operations and provides a foundation for future studies in this critical area of logistics and supply chain management. By continuing to refine and expand upon these simulation models, we can work towards creating more efficient, cost-effective, and environmentally sustainable intermodal transportation systems.

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my advisor, **Prof. Gaetano Fusco**, for his invaluable guidance, encouragement, and support throughout this project. His expertise and insights were instrumental in shaping the direction of this thesis.

I am also extremely grateful to my co-advisor, **Prof. Chiara Colombaroni**, for her constructive feedback and continuous support. Her detailed reviews and suggestions greatly improved the quality of my work.

A special thank you goes to **Dr. Mostafa Mohammadi**, whose advice and mentorship throughout this process were indispensable. His knowledge of simulation and intermodal terminal operations contributed significantly to the successful completion of this project. I would also like to extend my heartfelt appreciation to **Dr. Golman Rahmanifar** for her support and valuable discussions during my research process.

I would like to extend my appreciation to the Faculty of Civil and Industrial Engineering at *Sapienza University of Rome* for providing me with the resources and opportunities to pursue this research.

Finally, I would like to thank my family and friends for their unwavering encouragement and support, particularly during challenging times. Their belief in me has been a constant source of motivation, and I am truly grateful for their love and patience.

To everyone who has contributed in one way or another to this journey, thank you.

Bibliography

[1] Basallo-Triana, M. J., Bravo-Bastidas, J. J., Contreras, I., Cordeau, J.-F., and Vidal-Holguín, C. J. Intermodal hub network design with generalized capacity constraints and non-synchronized train–truck operations. *Transportation Research Part B: Methodological*, **174** (2023), 102770.

[2] Boysen, H. E. Øresund and fehmarnbelt high-capacity rail corridor standards updated. *Journal of Rail Transport Planning & Management*, **4** (2014), 44.

[3] Dotoli, M., Epicoco, N., Falagario, M., Seatzu, C., and Turchiano, B. A decision support system for optimizing operations at intermodal railroad terminals. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, **47** (2016), 487.

[4] Huelsz Prince, A. Capacity factors in intermodal road-rail terminals. (2015).

[5] Notteboom, T., Pallis, A., and Rodrigue, J.-P. *Port Economics, Management and Policy*. Routledge, New York (2022). ISBN 9780367331559. 218 illustrations.

[6] Park, N. K., Dragovic, B., and Kim, J. Y. Dynamic equipment deployment at a container terminal: transfer system based on real-time positioning. *Journal of Mechanical Engineering*, **55** (2009), 83.

[7] Semini, M., Fauske, H., and Strandhagen, J. O. Applications of discrete-event simulation to support manufacturing logistics decision-making: a survey. In *Proceedings of the 2006 Winter Simulation Conference*, pp. 1946–1953. IEEE (2006).

[8] Tako, A. A. and Robinson, S. The application of discrete event simulation and system dynamics in the logistics and supply chain context. *Decision support systems*, **52** (2012), 802.

[9] Wanigasekara, C. and Torres, F. S. Optimal control of resource allocations in maritime port operations using queue models. In *2024 SICE International Symposium on Control Systems (SICE ISCS)*, pp. 48–55. IEEE (2024).

[10] Woxenius, J., Roso, V., and Lumsden, K. The dry port concept–connecting seaports with their hinterland by rail. In *ICLSP Conference Proceedings*, pp. 305–319. Citeseer (2004).

Appendix A

Appendix: Code Listings

A.1 Road Gate Operation Code in Salabim

Listing A.1. Road Gate Simulation Code using Salabim

```
import salabim as sim
import pandas as pd

# Load the data from the Excel file
file_path_correct = r'C:\Users\RayanegostaR\Desktop\New folder (8)\truck2.xlsx'
truck_data_excel = pd.read_excel(file_path_correct)

# Extract the relevant data from the Excel file
truck_data_excel['arrival_time_minutes'] = truck_data_excel['Orario di arrivo'].apply(lambda x: int(x) * 60 + int((x - int(x)) * 100))

# Map the truck purposes from the Excel file
purpose_mapping = {0: 'pickup', 1: 'delivery', 2: 'both'}
truck_data_excel['purpose'] = truck_data_excel['ritiro(0)/consegna(1)/entrambi(2)'].map(purpose_mapping)

# Limit to the first 100 trucks
truck_data_excel = truck_data_excel.head(100)

# Simulation environment setup
sim.yieldless(False)
env = sim.Environment(trace=False)
env.animate(True)  # Turn on animation
env.animate_debug(False)  # Turn off debug mode for animation

# Constants
CHECKIN_TIME = 4  # Fixed check-in time in minutes as per the document
SCREEN_WIDTH = 800
SCREEN_HEIGHT = 600
CENTER_Y = SCREEN_HEIGHT // 2
GATE_X = SCREEN_WIDTH - 50
GATE_Y = CENTER_Y  # Set the gate's Y position
TRUCK_HEIGHT = 20
TRUCK_WIDTH = 40  # Width of the truck
```

```
33 TRUCK_SPEED = 2  # Speed at which trucks move (pixels per simulation
       step)
34 GATE_WIDTH = 20  # Width of the gate
35 GATE_HEIGHT = 250  # Height of the gate
36
37 # RoadGate Component
38 class RoadGate(sim.Component):
39     def __init__(self, truck_data, *args, **kwargs):
40         super().__init__(*args, **kwargs)
41         self.truck_data = truck_data
42         self.checkin_server = sim.Resource(capacity=1,
               name="Check-in Server")
43
44         # Animation for the road gate
45         self.gate_animation = sim.AnimateRectangle(
46             spec=(0, 0, GATE_WIDTH, GATE_HEIGHT),
47             fillcolor='gray',
48             x=GATE_X,
49             y=GATE_Y - GATE_HEIGHT // 2  # Center the gate vertically
50         )
51
52         # Label for the road gate
53         self.gate_label = sim.AnimateText(
54             text='Road Gate',
55             x=GATE_X + GATE_WIDTH + 10,
56             y=GATE_Y - GATE_HEIGHT // 2 - 20,
57             textcolor='black',
58             fontsize=15
59         )
60
61         # Legend for truck colors
62         self.legend_delivery = sim.AnimateText(
63             text='Delivery: Blue',
64             x=10,
65             y=SCREEN_HEIGHT - 60,
66             textcolor='blue',
67             fontsize=12
68         )
69
70         self.legend_pickup = sim.AnimateText(
71             text='Pickup: Green',
72             x=10,
73             y=SCREEN_HEIGHT - 40,
74             textcolor='green',
75             fontsize=12
76         )
77
78         self.legend_both = sim.AnimateText(
79             text='Both: Orange',
80             x=10,
81             y=SCREEN_HEIGHT - 20,
82             textcolor='orange',
83             fontsize=12
84         )
85
86     def process(self):
87         for i, row in self.truck_data.iterrows():
```

```
88              print(f"Creating Truck {i + 1} at time
                    {row['arrival_time_minutes']} minutes")
89              Truck(self.checkin_server, CHECKIN_TIME,
                    row['arrival_time_minutes'], row['purpose'], row['ITU
                    corrispondente'])
90              yield self.hold(1)  # Yield for 1 minute between truck
                    generations to prevent overlapping
91
92  # Truck Component
93  class Truck(sim.Component):
94      def __init__(self, checkin_server, checkin_time, arrival_time,
            purpose, itu_code, *args, **kwargs):
95          super().__init__(*args, **kwargs)
96          self.checkin_server = checkin_server
97          self.checkin_time = checkin_time
98          self.arrival_time = arrival_time
99          self.purpose = purpose
100         self.itu_code = itu_code
101         self.x = 0  # Starting position on the x-axis
102         self.y = CENTER_Y - TRUCK_HEIGHT // 2  # Vertical center
                position
103
104         # Truck animation
105         self.truck_animation = sim.AnimateRectangle(
106             spec=(0, 0, TRUCK_WIDTH, TRUCK_HEIGHT),
107             fillcolor='blue' if self.purpose == 'delivery' else
                    'green' if self.purpose == 'pickup' else 'orange',
108             x=lambda: self.x,  # Lambda function to update x position
109             y=self.y  # Static y position
110         )
111
112         # Label for the ITU code on the truck
113         self.itu_label = sim.AnimateText(
114             text=lambda: str(self.itu_code),
115             x=lambda: self.x + 5,  # Position the label inside the
                    truck
116             y=lambda: self.y + 5,  # Center vertically within the
                    truck
117             textcolor='white',
118             fontsize=10
119         )
120
121     def process(self):
122         yield self.hold(self.arrival_time)  # Wait until the truck's
                arrival time
123
124         # Move the truck towards the gate
125         while self.x < GATE_X - TRUCK_WIDTH:
126             self.x += TRUCK_SPEED  # Increment position based on a
                    fixed speed
127             yield self.hold(0.1)  # Hold for a short time to create
                    a smooth movement
128
129         yield self.request(self.checkin_server)  # Request the
                check-in server
130         yield self.hold(self.checkin_time)  # Simulate the check-in
                process
```

```
131         self.release()  # Release the check-in server
132
133 # Instantiate the RoadGate with truck data
134 road_gate = RoadGate(truck_data_excel)
135
136 # Run the simulation for one day (1440 minutes)
137 env.run(1440)
```

A.2 Intermodal Terminal Simulation in SimPy

Listing A.2. Intermodal Terminal Simulation Code using SimPy

```
1  import simpy
2  import pandas as pd
3
4  # Load the Excel files
5  itu_df = pd.read_csv('ITU3.csv')
6  train_df = pd.read_csv('train3.csv')
7  truck_df = pd.read_csv('truck4.csv')
8  T1 = 3 * 60  # Convert hours to minutes
9  T2 = 24 * 60 # Convert hours to minutes
10
11
12
13 # Define the ITU class
14 class ITU:
15     def __init__(self, env, id, priority=1):
16         self.env = env
17         self.id = id
18         self.priority = priority
19         self.train_arrival_time = None
20         self.train_departure_time = None
21         self.unloaded = False
22         self.stored = False
23         self.loaded = False
24         self.storage_start_time = None
25         self.storage_end_time = None
26
27
28 # Define the Train class
29 class Train:
30     def __init__(self, env, id, itus, arrival_time, departure_time):
31         self.env = env
32         self.id = id
33         self.itus = itus
34         self.arrival_time = arrival_time
35         self.departure_time = departure_time
36
37
38 # Define the StorageYard class
39 class StorageYard:
40     def __init__(self, env, long_term_capacity=1000,
       buffer_capacity=200, max_stack=2):
41         self.env = env
42         self.long_term_capacity = long_term_capacity
```

```
43          self.buffer_capacity = buffer_capacity
44          self.max_stack = max_stack
45          self.long_term_storage = {}
46          self.buffer_storage = {}
47          self.itu_entry_times = {}
48          self.total_itus_stored = 0
49          self.max_utilization = 0
50          self.itu_entry_times = {}
51
52
53      def store_itu(self, itu, storage_type):
54          if storage_type == 'buffer':
55              storage = self.buffer_storage
56              capacity = self.buffer_capacity
57          else:
58              storage = self.long_term_storage
59              capacity = self.long_term_capacity
60
61          available_slot = next((slot for slot in storage if
                len(storage[slot]) < self.max_stack), None)
62          if available_slot is not None:
63              storage[available_slot].append(itu)
64          elif len(storage) < capacity:
65              new_slot = len(storage) + 1
66              storage[new_slot] = [itu]
67          else:
68              print(f"No {storage_type} space available for ITU
                    {itu.id} at time {self.env.now:.2f}")
69              return
70
71          itu.stored = True
72          itu.storage_start_time = self.env.now
73          self.update_utilization()
74          self.itu_entry_times[itu.id] = self.env.now
75          print(f"ITU {itu.id} stored in {storage_type} at time
                {self.env.now}")
76
77      def retrieve_itu(self, itu, storage_type):
78          storage = self.buffer_storage if storage_type == 'buffer'
                else self.long_term_storage
79          for slot in storage:
80              if itu in storage[slot]:
81                  storage[slot].remove(itu)
82                  itu.storage_end_time = self.env.now
83          entry_time = self.itu_entry_times.pop(itu.id, None)
84          if entry_time is not None:
85              dwell_time = self.env.now - entry_time
86              self.total_dwell_time += dwell_time
87              self.total_itus_stored += 1
88              print(f"ITU {itu.id} retrieved from {storage_type} at
                    time {self.env.now}, dwell time: {dwell_time}")
89
90
91      def update_utilization(self):
92          long_term_utilization = sum(len(stack) for stack in
                self.long_term_storage.values()) /
                (self.long_term_capacity * self.max_stack)
```

```
93          buffer_utilization = sum(len(stack) for stack in
                self.buffer_storage.values()) / (self.buffer_capacity *
                self.max_stack)
94          total_utilization = (long_term_utilization *
                self.long_term_capacity + buffer_utilization *
                self.buffer_capacity) / (self.long_term_capacity +
                self.buffer_capacity)
95          if total_utilization > self.max_utilization:
96              self.max_utilization = total_utilization
97
98      def get_average_dwell_time(self):
99          return self.total_dwell_time / self.total_itus_stored if
                self.total_itus_stored > 0 else 0
100
101     def get_utilization_rate(self):
102         return self.max_utilization
103
104
105 # Define the Truck class
106 class Truck:
107     def __init__(self, env, id, arrival_time, purpose, road_gate):
108         self.env = env
109         self.id = id
110         self.arrival_time = arrival_time
111         self.purpose = purpose
112         self.road_gate = road_gate
113         self.waiting_time = 0
114         self.service_start_time = None
115         self.env.process(self.run())
116
117     def run(self):
118         yield self.env.timeout(self.arrival_time)
119         arrive_time = self.env.now
120         with self.road_gate.resource.request() as req:
121             yield req
122             self.service_start_time = self.env.now
123             self.waiting_time = self.service_start_time - arrive_time
124             self.road_gate.total_waiting_time += self.waiting_time
125             self.road_gate.total_trucks += 1
126             yield self.env.timeout(3)  # Assuming 3 minutes service
                    time
127         print(f'Truck {self.id} waited for {self.waiting_time}
                minutes')
128
129
130 # Define the RoadGate class
131 class RoadGate:
132     def __init__(self, env, gate_id, truck_data):
133         self.env = env
134         self.gate_id = gate_id
135         self.truck_arrivals = {}
136         self.total_waiting_time = 0
137         self.total_trucks = 0
138         self.truck_data = truck_data
139         self.resource = simpy.Resource(env, capacity=1)  # Add this
                line
140         self.process_truck_data()
```

```
141         self.env.process(self.run())
142
143     def process_truck_data(self):
144         print("Processing truck data in RoadGate:")
145         for _, truck in self.truck_data.iterrows():
146             itu_id = str(int(truck['ITU corrispondente']))  # Ensure
                it's a string without decimal
147             arrival_time = self.hours_to_minutes(truck['Orario di
                arrivo'])
148             print(f"Processing truck for ITU {itu_id}, arrival time:
                {arrival_time}")
149             self.truck_arrivals[itu_id] = arrival_time
150
151     def hours_to_minutes(self, time):
152         return int(float(time) * 60)
153
154     def get_truck_arrival_time(self, itu):
155         arrival_time = self.truck_arrivals.get(str(itu.id), None)
156         print(f"Getting truck arrival time for ITU {itu.id}:
            {arrival_time}")
157         return arrival_time
158
159     def run(self):
160         for _, truck in self.truck_data.iterrows():
161             itu_id = str(int(truck['ITU corrispondente']))
162             arrival_time = self.hours_to_minutes(truck['Orario di
                arrivo'])
163             yield self.env.timeout(arrival_time)
164             self.env.process(Truck(self.env, itu_id, arrival_time,
165                                    truck['ritiro(0)/consegna(1)/entrambi(2)'],
                                       self).run())
166         yield self.env.timeout(0)
167     def hours_to_minutes(self, time):
168         return int(float(time) * 60)
169
170     def get_truck_arrival_time(self, itu):
171         arrival_time = self.truck_arrivals.get(itu.id, None)
172         print(f"Getting truck arrival time for ITU {itu.id}:
            {arrival_time}")
173         return arrival_time
174
175     def average_truck_waiting_time(self):
176         return self.total_waiting_time / self.total_trucks if
            self.total_trucks > 0 else 0
177
178
179 # Define the CraneOperation class
180 class CraneOperation:
181     def __init__(self, env, num_cranes=3):
182         self.env = env
183         self.num_cranes = num_cranes
184         self.cranes = [simpy.Resource(env, capacity=1) for _ in
            range(num_cranes)]
185         self.total_operations = 0
186         self.operating = {crane: False for crane in self.cranes}
187         self.last_operation_end_time = 0
188
```

```
189
190     def operate(self, operation_time):
191         with self.cranes[0].request() as req:
192             yield req
193             if self.env.now > self.last_operation_end_time:
194                 idle_time = self.env.now -
                        self.last_operation_end_time
195                 self.total_idle_time += idle_time
196                 print(f"Crane idle for {idle_time} minutes")
197             yield self.env.timeout(operation_time)
198             self.total_operations += 1
199             self.last_operation_end_time = self.env.now
200
201     def get_utilization_rate(self, simulation_time):
202         return (self.total_operations * 4) / (simulation_time *
                self.num_cranes) if simulation_time > 0 else 0
203
204
205 # Define the TrainUnloading class
206 class TrainUnloading:
207     def __init__(self, env, road_gates, storage_yard,
            crane_operation, T1, T2):
208         self.env = env
209         self.road_gates = road_gates  # Now it's a list of road gates
210         self.storage_yard = storage_yard
211         self.crane_operation = crane_operation
212         self.T1 = T1
213         self.T2 = T2
214         self.direct_transshipments = 0
215
216     def determine_path(self, itu):
217         truck_times = [gate.get_truck_arrival_time(itu) for gate in
                self.road_gates]
218         truck_time = min(time for time in truck_times if time is not
                None)
219         if truck_time is None:
220             print(f"ITU {itu.id}: No truck time, using storage")
221             return "storage"
222
223         train_time = itu.train_arrival_time
224         time_difference = abs(truck_time - train_time)
225         print(f"ITU {itu.id}: Truck time: {truck_time}, Train time:
                {train_time}, Time difference: {time_difference}")
226
227         if time_difference <= self.T1:
228             print(f"ITU {itu.id}: Direct transshipment (within T1 =
                    {self.T1} minutes)")
229             return "direct"
230         elif time_difference <= self.T2:
231             print(f"ITU {itu.id}: Using buffer (within T2 =
                    {self.T2} minutes)")
232             return "buffer"
233         else:
234             print(f"ITU {itu.id}: Using storage (time difference >
                    T2)")
235             return "storage"
236
```

```
237     def unload_train(self, train):
238         for itu in train.itus:
239             if not itu.unloaded:
240                 path = self.determine_path(itu)
241                 yield self.env.process(self.execute_unloading(itu,
                        path))
242                 itu.unloaded = True
243 
244     def execute_unloading(self, itu, path):
245         if path == "direct":
246             yield self.env.process(self.direct_transshipment(itu))
247         elif path == "buffer":
248             yield self.env.process(self.buffer_storage(itu))
249         else:
250             yield self.env.process(self.long_term_storage(itu))
251 
252     def direct_transshipment(self, itu):
253         yield self.env.process(self.crane_operation.operate(4))  #
                Assume 4 minutes for direct transfer
254         self.direct_transshipments += 1
255         print(f"ITU {itu.id} directly transshipped at time
                {self.env.now:.2f}")
256 
257     def buffer_storage(self, itu):
258         yield self.env.process(self.crane_operation.operate(4))
259         self.storage_yard.store_itu(itu, 'buffer')
260         print(f"ITU {itu.id} stored in buffer area at time
                {self.env.now:.2f}")
261 
262     def long_term_storage(self, itu):
263         yield self.env.process(self.crane_operation.operate(4))
264         self.storage_yard.store_itu(itu, 'long_term')
265         print(f"ITU {itu.id} stored in long-term storage at time
                {self.env.now:.2f}")
266 
267 class TrainLoading:
268     def __init__(self, env, road_gates, storage_yard,
            crane_operation, T1, T2):
269         self.env = env
270         self.road_gates = road_gates  # Now it's a list of road gates
271         self.storage_yard = storage_yard
272         self.crane_operation = crane_operation
273         self.T1 = T1
274         self.T2 = T2
275         self.direct_transshipments = 0
276 
277     def determine_path(self, itu):
278         truck_times = [gate.get_truck_arrival_time(itu) for gate in
                self.road_gates]
279         truck_time = min(time for time in truck_times if time is not
                None)
280         if truck_time is None:
281             print(f"ITU {itu.id}: No truck time, using storage")
282             return "storage"
283 
284         train_time = itu.train_departure_time
285         time_difference = abs(truck_time - train_time)
```

```
286         print(f"ITU {itu.id}: Truck time: {truck_time}, Train time:
                {train_time}, Time difference: {time_difference}")
287
288         if time_difference <= self.T1:
289             print(f"ITU {itu.id}: Direct transshipment (within T1 =
                    {self.T1} minutes)")
290             return "direct"
291         elif time_difference <= self.T2:
292             print(f"ITU {itu.id}: Using buffer (within T2 =
                    {self.T2} minutes)")
293             return "buffer"
294         else:
295             print(f"ITU {itu.id}: Using storage (time difference >
                    T2)")
296             return "storage"
297
298     def load_train(self, train):
299         for itu in train.itus:
300             if not itu.loaded:
301                 path = self.determine_path(itu)
302                 yield self.env.process(self.execute_loading(itu,
                        path))
303                 itu.loaded = True
304
305     def execute_loading(self, itu, path):
306         if path == "direct":
307             yield self.env.process(self.direct_transshipment(itu))
308         elif path == "buffer":
309             yield self.env.process(self.buffer_retrieval(itu))
310         else:
311             yield self.env.process(self.storage_retrieval(itu))
312
313     def direct_transshipment(self, itu):
314         yield self.env.process(self.crane_operation.operate(4))
315         self.direct_transshipments += 1
316         print(f"ITU {itu.id} directly loaded onto train at time
                {self.env.now:.2f}")
317
318     def buffer_retrieval(self, itu):
319         yield self.env.process(self.crane_operation.operate(4))
320         self.storage_yard.retrieve_itu(itu, 'buffer')
321         print(f"ITU {itu.id} retrieved from buffer area at time
                {self.env.now:.2f}")
322
323     def storage_retrieval(self, itu):
324         yield self.env.process(self.crane_operation.operate(4))
325         self.storage_yard.retrieve_itu(itu, 'long_term')
326         print(f"ITU {itu.id} retrieved from long-term storage at
                time {self.env.now:.2f}")
327
328 # Define the MainSimulation class
329 def load_data():
330     # Load data from CSV files
331     itu_df = pd.read_csv('ITU3.csv')
332     train_df = pd.read_csv('train3.csv')
333     truck_df = pd.read_csv('truck4.csv')
334     print("Sample of loaded truck data:")
```

```
335     print(truck_df.head())
336     return itu_df, train_df, truck_df
337 class MainSimulation:
338     T1 = 6 * 60  # minutes, time window for direct transshipment
339     T2 = 24 * 60  # minutes, time window for buffer area vs
            long-term storage
340 
341     def __init__(self, env, itu_df, train_df, truck_df,
            num_road_gates=1, num_cranes=1):
342         self.env = env
343         self.storage_yard = StorageYard(env,
                long_term_capacity=1000, buffer_capacity=200, max_stack=2)
344         self.crane_operation = CraneOperation(env,
                num_cranes=num_cranes)
345         self.itu_data = itu_df
346         self.truck_data = truck_df
347         self.train_data = train_df
348         self.num_road_gates = num_road_gates
349         self.direct_transshipments = 0
350         self.total_itus_handled = 0
351 
352         # Initialize road gates first
353         self.road_gates = [RoadGate(env, i+1, truck_df) for i in
                range(num_road_gates)]
354 
355         # Now initialize train schedule
356         self.trains = self.initialize_train_schedule(train_df)
357 
358         # Start the simulation process
359         self.env.process(self.run())
360 
361     def hours_to_minutes(self, time):
362         return int(float(time) * 60)
363 
364     def initialize_train_schedule(self, train_df):
365         trains = []
366         for day in range(7):  # Repeat for 7 days (one week)
367             day_offset = day * 1440  # 1440 minutes per day
368             for _, train in train_df.iterrows():
369                 train_id = train['Unnamed: 0']
370                 is_arrival = 'Arrivo' in train_id
371 
372                 if is_arrival:
373                     train_arrival_time =
                            self.hours_to_minutes(train['Orario di
                            arrivo']) + day_offset
374                     train_departure_time = None
375                 else:
376                     train_arrival_time = None
377                     train_departure_time =
                            self.hours_to_minutes(train['Orario di
                            partenza']) + day_offset
378 
379                 # Collect ITU IDs from relevant columns
380                 itu_ids =
                        train.iloc[6:].dropna().astype(int).tolist()
381 
```

```
382                 # Create ITU objects
383                 itus = [ITU(self.env, str(itu_id)) for itu_id in
                        itu_ids]
384
385                 # Set train arrival and departure times for each ITU
386                 for itu in itus:
387                     itu.train_arrival_time = train_arrival_time
388                     itu.train_departure_time = train_departure_time
389                     if str(itu.id) not in
                            self.road_gates[0].truck_arrivals:
390                         print(f"Warning: No truck arrival time found
                                for ITU {itu.id}")
391
392                 train_obj = Train(self.env, train_id, itus,
                        train_arrival_time, train_departure_time)
393                 trains.append(train_obj)
394                 self.env.process(self.unload_and_load_train(train_obj))
395         return trains
396
397     def unload_and_load_train(self, train):
398         if train.arrival_time is not None:
399             train_unloading = TrainUnloading(self.env,
                    self.road_gates, self.storage_yard,
                    self.crane_operation, self.T1, self.T2)
400             yield
                    self.env.process(train_unloading.unload_train(train))
401
402         if train.departure_time is not None:
403             yield self.env.timeout(30)  # Delay between unloading
                    and loading
404             train_loading = TrainLoading(self.env, self.road_gates,
                    self.storage_yard, self.crane_operation, self.T1,
                    self.T2)
405             yield self.env.process(train_loading.load_train(train))
406
407         self.total_itus_handled += len(train.itus)
408         if train.arrival_time is not None:
409             self.direct_transshipments +=
                    train_unloading.direct_transshipments
410         if train.departure_time is not None:
411             self.direct_transshipments +=
                    train_loading.direct_transshipments
412
413
414     def run(self):
415         for gate in self.road_gates:
416             yield self.env.process(gate.run())
417
418     def calculate_kpis(self, print_results=True):
419         total_itus = self.total_itus_handled
420         percentage_direct = (self.direct_transshipments /
                total_itus) * 100 if total_itus > 0 else 0
421
422         kpis = {
423             "Average Truck Waiting Time":
                    sum(gate.average_truck_waiting_time() for gate in
                    self.road_gates) / len(self.road_gates),
```

```
424             "Storage Yard Utilization Rate":
                    self.storage_yard.get_utilization_rate() * 100,
425             "Total Crane Operations":
                    self.crane_operation.total_operations,
426             "Crane Utilization Rate":
                    self.crane_operation.get_utilization_rate(self.env.now)
                    * 100,
427             "Throughput (Total ITUs handled)":
                    self.total_itus_handled,
428             "Percentage of Direct Transshipments": percentage_direct
429         }
430
431         if print_results:
432             print("\n--- KPI Report ---")
433             for key, value in kpis.items():
434                 print(f"{key}: {value:.2f}")
435         return kpis
436
437
438 def main():
439     road_gate_scenarios = [1, 2, 3]  # Number of road gates to
            simulate
440     crane_scenarios = [1, 2, 3, 4]  # Number of cranes to simulate
441
442     results = {}
443
444     for num_gates in road_gate_scenarios:
445         for num_cranes in crane_scenarios:
446             print(f"\nRunning simulation with {num_gates} road
                    gate(s) and {num_cranes} crane(s)")
447             env = simpy.Environment()
448             itu_df, train_df, truck_df = load_data()
449             main_simulation = MainSimulation(env, itu_df, train_df,
                    truck_df,
450                                              num_road_gates=num_gates,
451                                              num_cranes=num_cranes)
452             env.run(until=10080)  # Run the simulation for one week
                    (10,080 minutes)
453             kpis = main_simulation.calculate_kpis()
454
455             # Store results
456             scenario_key = (num_gates, num_cranes)
457             results[scenario_key] = kpis
458
459     # Print or analyze results
460     for (gates, cranes), kpis in results.items():
461         print(f"\nResults for {gates} road gate(s) and {cranes}
                crane(s):")
462         for kpi, value in kpis.items():
463             print(f"{kpi}: {value}")
464
465 if __name__ == "__main__":
466     main()
```