Filip Ivanković, BSc

Development and Simulation of Control Algorithms for a Forklift Control System

MASTER'S THESIS

to achieve the university degree of

Diplom-Ingenieur

Master's degree programme:

Production Science and Management

submitted to

Graz University of Technology

Supervisor

Ass.-Prof. Dipl.-Ing. Dr. techn. Norbert Hafner

Dipl.-Ing. Dominik Stadlthanner

Institute of Logistics Engineering (ITL)

Graz, November 2024

Statutory Declaration (Affidavit)

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

……………………… ………………………………………………..
date (signature)

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Masterarbeit identisch.

Graz, am …………………………… ………………………………………………..
(Unterschrift)

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my supervisors, DI Dominik Stadlthanner and Ass.-Prof. Dipl.-Ing. Dr. techn. Norbert Hafner from the Institute of Logistics Engineering. Their continuous guidance, support, and valuable insights were instrumental in the completion of this thesis. I am truly thankful for their willingness to always be available to answer questions, provide feedback, and offer encouragement throughout this journey.

I would also like to extend my heartfelt thanks to the entire staff at the Institute of Logistics Engineering, where I had the privilege of working for three years. The collaborative environment and support from my colleagues played a key role in both my professional and personal development during that time. Foremost, I am especially grateful to my colleague DI Anto Zelić, who first introduced me to the Institute and sparked my interest in logistics. His guidance as my bachelor's thesis supervisor, along with his encouragement and belief in my potential, has had a lasting impact on my career and journey in this field.

To my parents and my sister, thank you for your unwavering support, love, and encouragement. Your belief in me has been a constant source of strength throughout my academic journey, and I am forever grateful for your patience and understanding. Thank you for always being there, ready to lift me up when I needed it most. I am deeply thankful for everything you have done for me and feel truly blessed to have you in my life.

Lastly, to my beloved wife, my deepest appreciation. Thank you for your endless patience, love, and for being by my side every day. Your support has been invaluable, and I could not have reached this point without your understanding and encouragement. For everything you do, I am truly grateful.

The writing in this thesis has been enhanced with the assistance of ChatGPT (October 2024) by OpenAI (https://chat.openai.com/chat) for structuring and stylistic revisions, and DeepL Translate by DeepL SE (https://www.deepl.com/en/translator) for translation assistance.

Abstract

This thesis presents the development and simulation of control algorithms for Forklift Guidance Systems, aiming to improve the efficiency of forklift operations in warehouse environments. With increasing complexity in logistics and heightened demands for fast and reliable fulfillment, optimizing the allocation of resources such as forklifts is critical. The research centers on three core algorithms: The Sequential Forklift Assignment Algorithm, the Priority-Based Algorithm, and the Auction-Based Routing Algorithm. Each algorithm represents a different approach to key challenges such as minimizing travel distances, reducing empty runs, and balancing workloads among forklifts.

A discrete event simulation model was developed using Tecnomatix Plant Simulation to test the algorithms under various operational conditions. This simulation model, built on the input data of a real-world warehouse system, includes a system boundary that reflects key elements such as warehouse layout, transport orders, forklift specifications, and operational shifts. By automating these inputs, the model generates the warehouse layout and transport order flow, enabling detailed testing of each algorithm.

Key performance indicators (KPIs) were employed to assess algorithm performance, including task completion time, forklift utilization rates, and the on-time fulfillment of transport orders with strict time requirements. These KPIs provide a comprehensive evaluation of each algorithm's ability to handle dynamic workloads. The results show that the Auction-Based Routing Algorithm outperforms the other algorithms, particularly in terms of minimizing empty runs and ensuring timely task fulfillment, which is critical for time-sensitive retrieval orders. The ability of this algorithm to flexibly adjust for different conditions makes it a promising candidate for real-world applications.

In addition to the benchmark simulation results, the study extends the evaluation by applying the Auction-Based Routing Algorithm to a real-world warehouse case. This application validates the simulation outcomes, providing practical insights into the scalability and adaptability of the algorithm.

By demonstrating how simulation models and algorithmic optimizations can enhance operational efficiency, this thesis contributes to both the academic literature on warehouse logistics and the practical development of automated systems. The findings hold significant potential for improving resource management and throughput in modern warehouse operations.

Kurzfassung

Diese Diplomarbeit befasst sich mit der Entwicklung und Simulation von Steuerungsalgorithmen für Staplerleitsysteme zur Verbesserung der Effizienz von Staplereinsätzen in Lagerumgebungen. Angesichts der zunehmenden Komplexität in der Logistik und der gestiegenen Anforderungen an eine schnelle Abwicklung ist die Optimierung der Ressourcenzuweisung entscheidend. Die Forschung konzentriert sich auf drei Algorithmen: den Algorithmus für sequenzielle Staplerzuweisung, den prioritätsbasierten Algorithmus und den auktionsbasierten Routingalgorithmus. Diverse Herausforderungen wie die Minimierung der Fahrstrecken, die Reduktion von Leerfahrten und die gleichmäßige Auslastung der Stapler werden von den einzelnen Algorithmen auf unterschiedliche Weise gelöst.

Ein diskretes Simulationsmodell wurde mit Tecnomatix Plant Simulation entwickelt, um die Algorithmen unter verschiedenen Bedingungen zu testen. Das Modell basiert auf Eingabedaten eines realen Lagersystems und umfasst Schlüsselelemente wie Lagerlayout, Transportaufträge und Betriebsschichten. Automatisierte Eingaben ermöglichen das Generieren von Layout und Transportaufträgen, wodurch eine detaillierte Analyse der Algorithmen möglich wird.

Zur Leistungsbewertung wurden KPIs wie die Aufgabenerledigungszeit, die Staplerauslastung und die pünktliche Erfüllung zeitkritischer Aufträge herangezogen. Die Ergebnisse zeigen, dass der auktionsbasierte Routingalgorithmus besonders bei der Minimierung von Leerfahrten und der rechtzeitigen Erfüllung von Aufträgen die besten Ergebnisse liefert. Seine Flexibilität macht ihn für reale Anwendungen besonders vielversprechend.

Der auktionsbasierte Routingalgorithmus wird anschließend anhand eines realen Lagerszenarios verifiziert. Diese Anwendung bestätigt die Simulationsergebnisse und liefert praktische Einblicke in die Skalierbarkeit und Anpassungsfähigkeit des Algorithmus.

Durch den Nachweis, wie Simulationsmodelle und algorithmische Optimierungen die Betriebseffizienz steigern können, leistet diese Arbeit einen Beitrag sowohl zur wissenschaftlichen Literatur im Bereich der Lagerlogistik als auch zur praktischen Entwicklung automatisierter Systeme. Die Ergebnisse bieten ein erhebliches Potenzial zur Verbesserung des Ressourcenmanagements und des Durchsatzes in modernen Lagerbetrieben.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Automated Guided Vehicle	AGV
Compact Route Distance	CRD
Discrete-Event Simulation	DES
First-Come-First-Served	FCFS
Forklift Guidance System	FGS
Interim Order List	IOL
Key performance indicator	KPI
Mean Time to Repair	MTTR
Order List	OL
Penalty Factor	PF
Parallel station Pallet	PP
Parallel station Stacker	PS
Radio Frequency Identification	RFID
Retrieval Prioritization Factor	RPF
Temporary Order List	TOL
Traveling Salesman Problem	TSP
Technical University of Munich	TUM
Visual Basic for Applications	VBA
Weighting Factor	WF

1 Introduction

1.1 Background

Forklifts are frequently used to transfer and lift various kinds of goods in manufacturing and warehousing processes [1]. As a type of Material Handling Equipment, forklifts have become indispensable in logistics. They are used in warehouses, distribution centers, and other logistics facilities to transfer and hoist heavy loads, including pallets. Forklifts play a vital role in ensuring that goods are moved efficiently and safely within logistics operations [2]. With the rising complexity of supply chain networks and the increased volume of goods handled within warehouses, the efficient use of forklifts is more important than ever.

The continued growth of industries such as manufacturing, construction, and warehousing is one of the main drivers of the forklift market's growth and industrial expansion. Forklifts have become essential for effective material handling in these sectors [3]. The global forklift market is projected to reach USD 57.79 billion in 2024, with an expected Compound Annual Growth Rate of 3.46%, reaching USD 68.01 billion by 2029 [4]. This highlights the growing reliance on forklift systems for warehouse and logistics efficiency. As companies strive to manage increasingly complex warehouse environments, optimizing forklift operations is becoming critical.

In large-scale warehousing facilities where multiple forklifts are in use, several challenges arise, such as coordinating routes, minimizing empty runs, and ensuring the timely completion of transport orders. As the volume of orders with varying priorities increases, the operational efficiency of forklifts becomes a pressing issue. Additionally, safety measures must be maintained to avoid accidents and ensure smooth operations, which calls for more advanced systems to manage these tasks effectively.

One potential solution to these challenges is the implementation of Forklift Guidance System (FGS), which utilizes intelligent algorithms to manage forklift operations. These systems can help minimize empty runs, optimize task assignments, and improve overall warehouse efficiency. However, implementing FGS is a complex and costly undertaking, and the effectiveness of such systems often remains uncertain.

To better understand the impact of FGS and the different algorithms used for task assignment, simulations, particularly Discrete-Event Simulation (DES) models, are employed as a strategic approach to problem-solving. DES models the functioning of a system as a discrete series of events over time. Every event represents a change in the system's state and occurs at a specific moment [5]. These simulations serve as valuable tools for evaluating the potential impact of forklift guidance systems on warehouse logistics, considering factors such as efficiency and resource utilization.

1.2 Problem Statement

FGSs are IT-based systems designed to coordinate the movements of forklifts and other industrial trucks within a warehouse. These systems handle tasks such as the assignment of transport orders, route planning, and process monitoring (Figure 1-1). FGS typically receive transport orders through one or more interfaces, then assign these tasks to available forklifts while monitoring their execution. Forklift operators are guided through the process via screen dialogues or terminals attached to their vehicles, and various technologies such as barcodes, RFID, and real-time location tracking systems are used to track the performance of forklifts, monitor their location, and detect errors [6]. The ability to accurately determine the position of forklifts at any given moment has the potential to improve task allocation and reduce inefficient operations, such as empty runs.

During internal transport within manual warehouses, industrial trucks carry out a variety of transport orders, including storage and retrieval, order picking, provision, and inventory or replenishment [7, p. 315] . These tasks add significant complexity, requiring advanced systems such as FGS to efficiently manage and coordinate the fleet of industrial trucks.

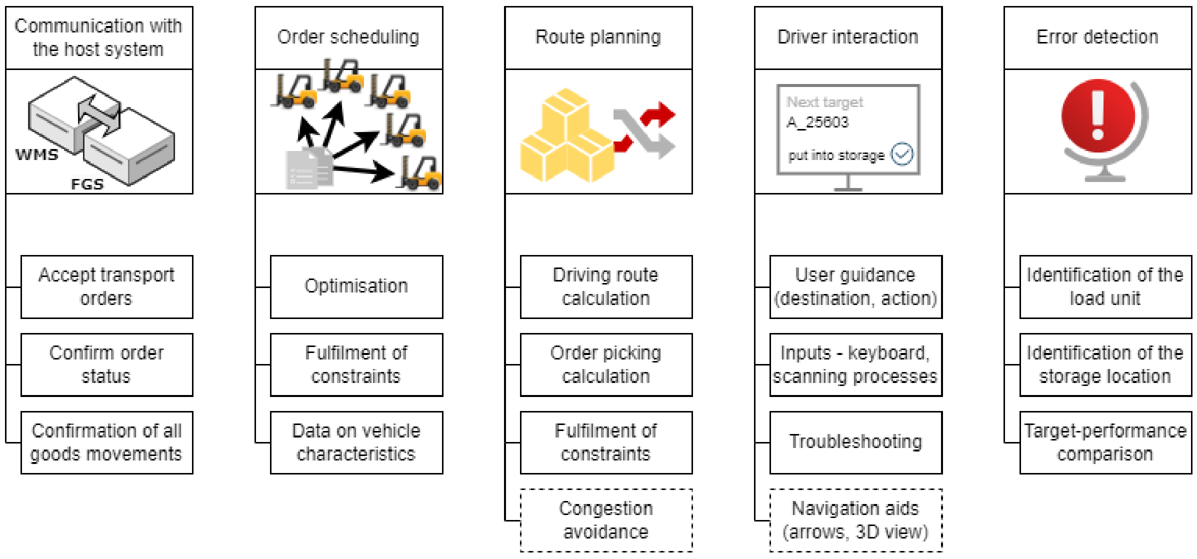

Communication with the host system
WMS
FGS
Accept transport orders
Confirm order status
Confirmation of all goods movements
Order scheduling
Optimisation
Fulfilment of constraints
Data on vehicle characteristics
Route planning
Driving route calculation
Order picking calculation
Fulfilment of constraints
Congestion avoidance
Driver interaction
Next target
A_25603
put into storage
User guidance (destination, action)
Inputs - keyboard, scanning processes
Troubleshooting
Navigation aids (arrows, 3D view)
Error detection
Identification of the load unit
Identification of the storage location
Target-performance comparison

Figure 1-1: Tasks of a FGS [6].

Despite the advantages of FGS, several key challenges remain in optimizing forklift operations. One of the primary goals of FGS is to minimize the number of forklifts required and reduce unnecessary travel distances – particularly the amount of empty runs – while still ensuring tasks are completed on time. In many cases, the system must also ensure an even distribution of workload among the forklifts to avoid overburdening individual machines.

FGS systems typically use control algorithms to assign transport orders. These algorithms fall into two main categories: push-based and pull-based approaches. In a pull-based system, transport orders are assigned only when requested by a forklift, often immediately after the completion of the previous task. In contrast, a push-based system assigns tasks as soon as they are created in the system, such as during the unloading of a truck (e.g., during the receiving process) [8, pp. 228-

231]. Both approaches come with their own challenges in terms of efficiency and task coordination [9].

The specific challenges associated with these algorithms include:

- **Minimizing empty runs**: Empty travel between tasks results in wasted time and reduced efficiency [9].
- **Ensuring timely task completion**: Managing orders with varying priorities while optimizing travel routes is complex [10].
- **Workload balancing**: Ensuring an even distribution of tasks among the fleet to avoid bottlenecks [7, pp. 124-126].
- **Flexibility and responsiveness**: The system must quickly adapt to real-time changes in the warehouse, such as sudden increases in order volume [10].

1.3 Literature Review and Research Gaps

FGS are crucial to optimizing warehouse logistics, and much of the research in this field revolves around the development and evaluation of algorithms for task assignment and route optimization. One of the most notable studies in this area comes from Mirlach et al. at the Technical University of Munich (TUM), where the authors explored various task allocation methods, including an agent-based auction algorithm in FGS to minimize empty runs and ensure efficient task assignment [6].

In their study, Mirlach et al. implemented both pull-based and push-based approaches. The pull-based methods, like First-Come First-Served (FCFS) and Earliest Deadline First, involved forklifts reacting to task availability as it arose, with tasks being assigned based on order arrival or urgency. In contrast, the agent-based anticipatory approach used a push-based system, where tasks were pre-assigned to forklifts based on predictive optimization to improve route efficiency. This push-based system allowed each forklift to operate as an independent agent using a simplified auction mechanism to "bid" on tasks based on a combinatorial auction algorithm.

While the research demonstrated advantages of the agent-based approach, such as reduced empty runs and greater efficiency, it lacked detailed insights into specific simulation conditions and system boundaries. It remains unclear how real-world variability, such as fluctuating order volumes or the dynamic capabilities of forklifts, was considered [6].

This thesis addresses the limitations identified in the research by Mirlach et al. by implementing three distinct algorithms for Forklift Guidance Systems, specifically designed to enhance real-world adaptability. To achieve this, a more comprehensive and detailed simulation model with expanded system boundaries was developed, allowing for a closer approximation to real-world scenarios where transport tasks must be allocated immediately upon generation. By incorporating real-time adaptability and robust system boundaries, this thesis seeks to contribute to a more practical understanding of forklift guidance within dynamic warehouse logistics environments [6].

1.4 Objectives and Research Questions

This thesis focuses on evaluating various algorithms for Forklift Guidance Systems (FGS) using DES. Specifically, it examines how different algorithms optimize forklift operations by reducing the number of forklifts required, minimizing empty runs, and ensuring timely task completion. In addition, the study assesses the real-world applicability of these algorithms by implementing the best-performing algorithm from the benchmark study in a real-world warehouse environment. While the benchmark simulation allows for controlled comparisons of algorithms under different conditions, the real-world use case provides practical insights into the algorithm's effectiveness in actual warehouse operations.

The research questions are as follows:

- **RQ1**: What is the most effective routing algorithm for optimizing forklift operations in terms of task allocation and minimizing empty runs?
- **RQ2**: How does fleet size impact retrieval order performance, and what is the minimum number of forklifts needed to maintain acceptable levels of on-time retrieval orders?
- **RQ3**: How does the choice and accuracy of localization systems impact forklift performance, task allocation efficiency, and overall operational effectiveness?

1.5 Report structure

Chapter 2, "Theoretical Framework," establishes a foundational understanding of warehouse systems, including their definitions, functions, tasks, and material handling systems. This chapter also covers simulation methodologies, focusing on DES and Technomatix Plant Simulation, which form the theoretical basis for analysing warehouse operations and optimization strategies.

Chapter 3, "Methods," transitions to the practical application of the theoretical framework. It addresses Warehouse Layout Design, the Orders Generator, and the Simulation Model Setup. The chapter introduces key algorithmic approaches such as Sequential Forklift Assignment, Combined Criteria, and Optimized Routing. A real-world use case is presented to validate the methods, demonstrating how theoretical concepts are applied to practical situations.

Chapter 4, "Results," presents the findings from the simulations and provides an evaluation of the effectiveness of the algorithms. This chapter details the planning and execution of simulation experiments, offering a thorough analysis of the results. The chapter integrates both the presentation of results and their analysis, linking the applied methods to the outcomes.

Chapter 5, "Summary and Outlook," concludes the thesis by interpreting the results and discussing their implications. This chapter reflects on the research objectives outlined in Section 1.4, summarizes the key contributions of the study, and offers suggestions for future research directions and potential improvements in Forklift Guidance Systems.

2 Theoretical Framework

This section provides the essential theoretical foundation for evaluating and optimizing FGSs through the use of DES. FGS play a crucial role in improving warehouse efficiency, relying on a thorough understanding of warehouse systems and the algorithms that govern task allocation [9].

The theoretical foundation begins with an exploration of the core components of warehouse systems, such as storage systems, conveying technologies, and the specific functions of forklifts, including order picking, storage and retrieval, and transport. Understanding these tasks offers the context needed to evaluate how FGS optimize warehouse processes [11].

Following this, the focus shifts to simulation methodologies, with a focus on DES, which models warehouse operations as a sequence of discrete events. This allows for evaluating strategies to optimize forklift movements and task assignments [12, pp. 28-32]. Technomatix Plant Simulation is introduced as the platform used for visualizing workflows, testing algorithms, and assessing performance under varying conditions [13].

Finally, three key algorithms are presented: **Sequential Forklift Assignment Algorithm**, **Priority-Based Algorithm**, and **Auction-Based Routing Algorithm**, each of which addresses different inefficiencies in forklift operations, such as minimizing unnecessary travel and ensuring timely task completion [6]. This chapter not only lays the theoretical groundwork for understanding these algorithms but also connects them to the practical applications analysed later in the thesis.

2.1 Warehouse systems

2.1.1 Definition and functions of a warehouse

Storage is defined as the planned positioning of good within material flow systems. A warehouse serves as a designated room or area for storing piece goods and/or bulk goods in the form of raw materials, intermediate products or finished good, which is recorded in terms of quantity and/or value. The fundamental processes outlined involve storing storage units, handling the storage and provision of these units in designated locations, and the subsequent retrieval of storage units. Optionally, picking of load units takes place in a warehouse [15, p. 51].

2.1.2 Tasks of a warehouse

According to [15, pp. 52-53], the functions of a warehouse can be the following:

- Asynchronous entries and exits between areas or systems
- Quantity equalisation, e.g. as part of the production of economical batch sizes
- Ensuring the capacity utilisation of cost-intensive production facilities, e.g. in the event of disruptions, supply bottlenecks, traffic problems, etc.
- Utilisation of transport capacities
- Seasonal fluctuations in sales behaviour

- Creation of a high level of delivery service through rapid fulfillment of orders or customer requests or ensuring overall delivery capability
- Storage to increase value (through maturing) or for speculative purposes
- Fulfillment of additional tasks, such as provision for order picking

2.1.3 Types of a warehouse

The selection of the warehouse design and its technical features is a multifaceted process influenced by various factors. These considerations include, but are not limited to, the type of goods meant for storage, the corresponding load carriers, as well as crucial elements such as dwell times within the warehouse and the desired access times, emphasizing the importance of a comprehensive approach to warehouse planning [16, p. 194]. Figure 2-1 shows some of the most important storage types in the form of schematic diagrams.

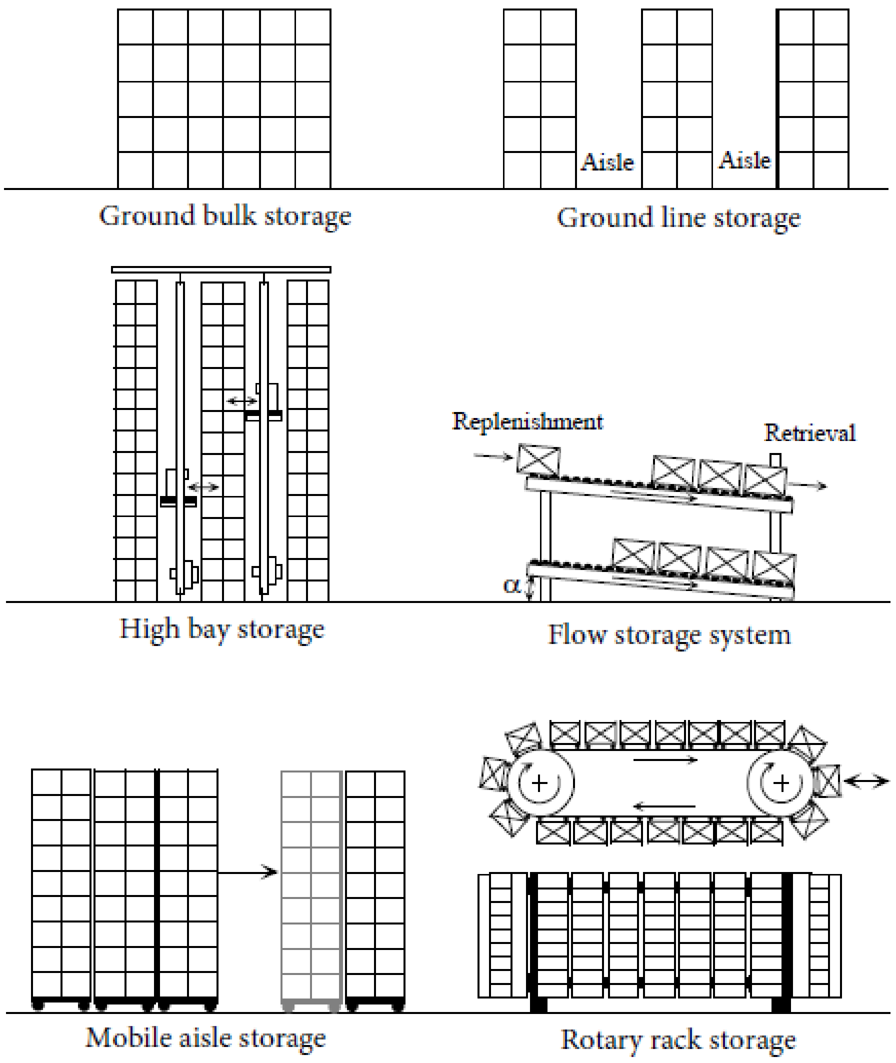


Figure 2-1: Warehouse types with schematic diagrams [16, p. 193]. The high bay storage with pallet racks, as depicted, will be simulated in the benchmark study.

2.1.4 Conveying systems

The general term for the movement or relocation of goods or people through technical means is referred to as transportation. When this spatial relocation occurs within a confined area, such as within a plant or factory, it is specifically referred to as conveying. Conveyors, recognized as work equipment for material flow within a company or plant, serve purposes beyond transportation, encompassing activities such as distribution, collection (during order picking), sorting, and buffering or temporary storage [15, p. 125].

Conveyors are classified into two main groups (Figure 2-2).

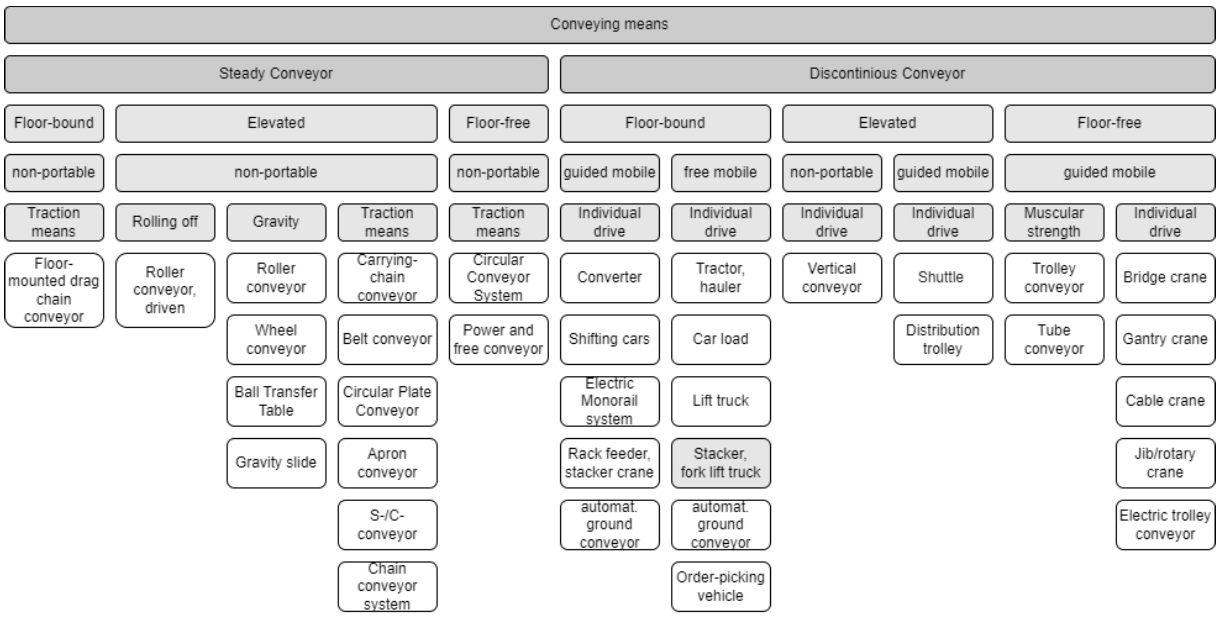


Figure 2-2: Structure of conveying means [15, p. 129].

Compared to steady conveyor technology, discontinuous conveyors are characterised by greater flexibility, and they are being preferred in material flow systems with a high number of transport sources and sinks [15, p. 130].

Discontinuous conveyor

Discontinuous conveyors are characterized by their intermittent conveying nature. This involves the conveying process occurring in individual work cycles, with a typical distinction made between cycle times for load and empty runs. The key feature of discontinuous conveyors lies in their high adaptability to various conveying tasks, providing a flexible solution for diverse material handling scenarios [15, p. 161].

Forklifts

Forklifts are recognized as materials handling devices with a lifting function suitable for picking up or transferring loads of goods stored at floor level, as well as for managing goods stored in racks or stacked configurations. They are considered the most prevalent type of ground conveyors [15, pp. 167-168]. Forklifts utilize two forks attached to the lift mast at the front to pick up goods for transportation.

The lift mast facilitates storage heights of up to nine meters and has a load capacity of up to sixteen tones. Widely deployed in intralogistics systems, forklifts serve as flexible work equipment in various settings, with the majority of goods transported on loading aids such as Euro pallets, chemical pallets, pallet cages, or individual parts [15, pp. 178-179].

Fork lift truck/counterbalanced lift truck			
Driving speed	9-35 km/h	Lift height	up to approx. 9 m
Lifting speed	0,23-0,65 m/s	Load capacity	1-16 t
Drive power	4-120 kW	Stacking height (CCGI/CCGII)	8 and 5 pallets

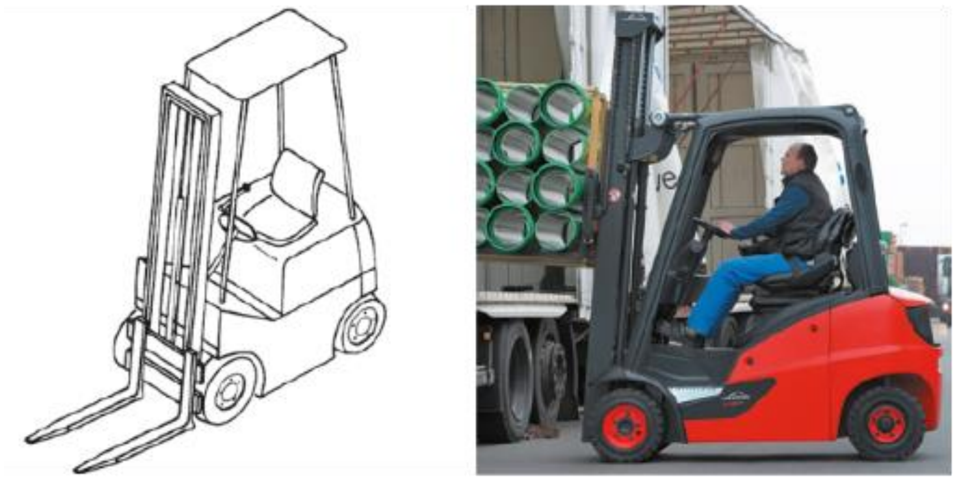


Figure 2-3: Forklift [15, p. 178].

2.2 Simulation

Simulation technology serves as a crucial tool for the strategic planning, seamless implementation, and efficient operation of complex technical systems.

Various economic trends, including

- the rise in product complexity and variety
- escalating quality expectations in connection with high cost pressure
- heightened demands for flexibility
- shorter product life cycles
- shrinking lot sizes
- and intensified competitive pressures

contribute to the necessity for shorter planning cycles. Simulation emerges as a valuable tool in situations where simpler methods no longer provide useful results [17, p. 1].

Simulations play a crucial role in evaluating and forecasting the performance of complex and interconnected operations systems. The use of simulations proves essential in understanding the effects of parameter changes in a controlled environment, offering a more practical and manageable alternative to experimenting directly in real-life systems with uncertain outcomes [18, p. 7].

2.2.1 Definitions

In this section, the terms simulation, system, model, simulation run, experiment, and material flow are explained and defined.

Simulation

Simulation involves replicating the dynamic processes of a real system within a model, with the goal of deriving applicable insights for reality. In a broader context, simulation entails the preparation, implementation, and analysis of specific experiments using a simulation model [17, p. 2].

System

A system is characterized as a separate set of components which are interrelated to one another. Also, the boundaries of the system need to be clarified beforehand [17, p. 2].

Model

A model is a simplified representation of a planned or existing system, including its processes, in another system. It differs from the original only within specified tolerances in important properties [17, p. 2].

Simulation run

A simulation run represents how the system behaves within the simulation model over a specified period [17, p. 2].

Experiment

As defined by VDI 3633, an experiment involves a purposeful empirical examination of a model's behaviour through repeated simulation runs, systematically varying the input parameters [17, p. 2].

Material flow

Following the principles outlined in VDI guideline 3300, material flow refers to the spatial, temporal, and organizational integration of processes related to the extraction, processing, and distribution of goods within specified areas. Companies distinguish between external flow of goods and internal material flows [19, p. 22].

In this context, material flow encompasses all processes within an operational object flow, aligning with procurement, production, and distribution tasks. It involves the movement of raw, auxiliary, and operating materials, semi-finished products, finished products, and tools. The primary objective of material flow is to connect production and assembly units while ensuring effective supply and disposal [19, p. 22].

2.2.2 Simulation Process

According to VDI guideline 3633, the recommended approach includes the following steps [17, pp. 2-6]:

1. Formulate the problem
2. Test of the simulation feasibility
3. Set targets
4. Collect and analyse data
5. Develop a model
6. Perform simulation runs
7. Analyse and interpret results
8. Document the process

The process begins with problem formulation, where the requirements for the simulation are defined. This leads to the creation of a written agreement, such as a technical specification, outlining specific problems to be investigated through simulation.

To assess simulation feasibility, a check is conducted to determine if the defined problems can be effectively examined through simulation. Various factors contribute to this assessment, including:

- the absence of analytical mathematical models (e.g., due to numerous variables)
- high complexity with many factors
- imprecise data
- gradual exploration of system limits
- repeated use of the simulation model

Objectives formulation involves defining the desired outcomes that the simulation should provide. Common simulation objectives include:

- minimizing processing time
- maximizing utilization
- minimizing inventory

These defined objectives are statistically analysed at the conclusion of simulation runs, establishing a required level of detail for the simulation model and determining the range of the simulation study.

Data required for the simulation study can be categorized into

- system load data
- organizational data

technical data

Table 2-1 provides a condensed overview of some collected data elements, offering a glimpse into the structured information needed for the simulation process.

Table 2-1: Data to be collected [17, pp. 3-4].

Technical data	
Factory structural data	Layout
	Means of production
	Transport functions
	Transport routes
	Areas
	Restrictions
Manufacturing data	Use time
	Performance time
	Capacity
Material flow data	Topology
	Conveyors
	Capacities
Accident data	Functional accidents
	Availability
Organizational data	
Working time organization	Break scheme
	Shift scheme
Resource allocation	Worker
	Machines
	Conveyors
Organization	Strategy
	Restrictions
	Incident management
System load data	
Product data	Working plans
	BOM
Job data	Production orders
	Transportation orders
	Volumes
	Dates

After successfully collecting the required data, the model development phase begins, which encompasses the development and testing of the simulation model. This process typically unfolds in two key stages.

1. Derivation of an iconic model from the conceptual model
2. Transfer to software model

In the first stage, an iconic model is derived from the conceptual model. This involves gaining a comprehensive understanding of the simulated system and making decisions regarding the desired level of simulation accuracy. Based on these considerations, choices are made about which aspects to simplify. The initial modelling stage consists of two main activities:

- system analysis (breakdown)
- abstraction (generalization)

System analysis dissolves the complexity of the system by breaking it down into meaningful elements aligned with investigation targets, while abstraction reduces specific system attributes to form an essential and limited image of the original system, often achieved through reduction (elimination of irrelevant details) and generalization (simplification of essential details).

The second stage involves transferring the model into a software model. This encompasses building and testing the simulation model based on the derived iconic model. The results of the modelling phase are included in the model documentation to facilitate future adjustments.

Once the model is created, simulation runs can commence. Depending on the objectives of the simulation study, experiments are conducted according to a predefined test plan. This plan outlines individual experiments, specifying output data, model arguments, objectives, and expected results. It is crucial to define a time span for the simulation experiments based on the findings of test runs. Detailed documentation of input and output data, along with the underlying parameters of the simulation model, is maintained for each experiment.

Values that undergo changes in the modelled system are derived from simulation results. The accurate interpretation of these results significantly influences the success of the simulation study.

For documentation purposes, a project report format is recommended, providing an overview of the study's timing and thoroughly documenting the work undertaken during the simulation study.

2.2.3 DES - Discrete Event Simulation

Simulation models are classified based on several dimensions, offering a framework for understanding their characteristics [20, pp. 13-14]:

- Static vs. Dynamic
  - Static models focus on a single point in time or exclude the temporal aspect, such as Monte Carlo simulation.
  - Dynamic models capture the temporal behaviour of the system, providing insights into its evolution over time, as seen in simulations of production plants.
- Deterministic vs. Stochastic
  - Deterministic models lack random components, offering a deterministic representation (e.g., chemical reactions).
  - Stochastic models incorporate random events, influencing system behaviour, as seen in queueing systems.
- Continuous vs. Discrete
  - Continuous models involve continuous changes in system states, often described using differential equation systems.
  - Discrete models feature system state changes occurring at distinct points in time, exemplified in systems such as warehouse management system.

In the domain of production and logistics, computer models play a crucial role in replicating the dynamic behaviours of plants (e.g., factories, construction sites, warehouses) and processes, such as projects. Employing computer models that incorporate stochastic components, DES emerges as a predominant modelling approach. DES is an event-oriented technique that characterizes system behaviour by detailing state changes triggered by events, such as the arrival of orders at a machine or the completion of specific process steps [20, p. 14].

This distinctive simulation technique portrays the operation of a system as a sequential series of discrete events unfolding over time. Each event occurs at a precise moment, signifying a change in the system's state. Notably, between consecutive events, no changes are assumed, allowing the simulation to seamlessly transition in time from one event to the next. DES models exhibit the unique characteristics of being both stochastic and dynamic, with changes occurring only at discrete times [5].

2.2.4 Application of DES in FGS

DES is commonly employed to model and optimize resource allocation and operational efficiency in systems such as FGS [12]. In the dynamic and complex environment of warehouse operations, DES allows for the simulation of key processes such as task allocation, forklift routing, and overall resource management. By breaking down warehouse operations into discrete, measurable events, DES provides valuable insights into how different strategies can be applied to optimize forklift movement and task execution.

One of the key advantages of using DES in FGS is its ability to evaluate multiple scenarios under controlled conditions. By simulating various operational setups, DES enables researchers and managers to test the impact of different strategies on key performance indicators (KPIs), such as system throughput, resource utilization, and task completion times. The ability to model multiple scenarios without interrupting actual warehouse operations makes DES particularly valuable for testing system changes and identifying bottlenecks [14].

However, DES has certain limitations when applied to FGS. One major drawback is the difficulty of accurately modelling human factors in warehouse operations. Forklift operators may experience fatigue, take breaks, or encounter unexpected delays - factors that are not easily captured in simulation models [20]. This limitation creates a gap between simulation results and real-world performance, particularly in scenarios where human intervention plays a significant role.

Additionally, DES struggles to fully account for the dynamic and stochastic nature of warehouse environments. Sudden shifts in order volume, changes in task priority, or equipment breakdowns are common in real-world operations, but these variables are difficult to simulate with precision [18, p. 26].

Such unpredictability can lead to discrepancies between the optimal strategies identified in simulation and the actual performance of the system in practice.

Despite these limitations, DES remains a valuable tool for modelling and improving FGS. Its ability to simulate, evaluate, and refine task allocation strategies makes it indispensable for optimizing warehouse operations. However, it is important to recognize that DES results should be interpreted with caution when applying them to real-world scenarios. In practice, simulation outcomes need to be adjusted to account for the dynamic and human-driven elements of warehouse environments.

2.2.5 Tecnomatix Plant Simulation

Tecnomatix Plant Simulation, a component of Siemens PLM Software's digital manufacturing solutions, is a discrete-event process simulation software tailored for modelling complex production and logistics systems. It is highly applicable to the study of FGSs, providing a digital environment in which users can simulate forklift operations and task allocation strategies. Users have the capability to develop object-oriented and hierarchical simulation models, incorporating sophisticated control strategies to analyse system characteristics, identify potential vulnerabilities, and optimize performance metrics such as resource utilization, throughput, and material handling operations [21]. With its user-friendly interface, Plant Simulation facilitates the creation of models of factories encompassing business, logistics, and production processes [22].

The software facilitates experimentation and scenario analysis, enabling the testing of various event- or plant-related scenarios, both during operation and in the pre-planning phase, to assess system behaviour and efficiency [22]. Through interactive 2D and 3D visualization, valuable insights can be gained into facility layout integration and the visualization of throughput, utilization, and bottlenecks with integrated charts. Moreover, Plant Simulation provides extensive analysis tools, including neural networks, genetic algorithms, and an experiment manager, to support decision-making and performance optimization [21].

In the context of FGS, Plant Simulation's digital modelling capabilities offer significant advantages for manufacturing and warehouse planning, enabling analysis of material flow, resource utilization, and logistics at different levels of production and task allocation. By creating digital models of logistics systems, companies can optimize performance, run experiments, and conduct what-if scenarios without disrupting existing production systems. This allows manufacturers to detect and rectify problems early, minimize investment costs in production lines, and optimize the performance and energy usage of existing systems prior to implementation [22]. This makes it an effective tool for improving task allocation strategies and optimizing forklift operations.

However, it is essential to remember that simulation results should be interpreted carefully, especially when applying them to dynamic and human-driven environments such as warehouses. Uncertainties in actual human behaviour or the stochastic nature of real operations can affect the accuracy of the model's outcomes, even if all available data has been taken into account [23]. While Plant Simulation allows for the modelling of scheduled breaks and downtimes, unexpected events such as unscheduled breaks or sudden system failures are difficult to model accurately without detailed real-world data.

2.3 Forklift guidance system

As introduced in the Problem Statement, FGSs are IT-based solutions aimed at optimizing order disposition and forklift movements within warehouse operations. Their primary goal is to allocate tasks efficiently, minimize unnecessary travel (such as empty runs) and improve overall system efficiency. Building on this foundation, this section will expand on the operational goals of FGSs and the challenges they address [6].

The core objective of FGSs is order disposition - the efficient assignment and distribution of tasks to available resources, such as forklifts, to execute transport orders effectively. This involves real-time coordination of transport tasks, such as storage, retrieval, and replenishment, ensuring that resources are deployed optimally across the warehouse [11]. FGSs are designed to achieve these goals by focusing on order disposition and sequencing, which is crucial for reducing inefficiencies in warehouse logistics [11].

FGSs operate using either push-based or pull-based control algorithms. Push-based systems assign tasks to forklifts as soon as they are created, such as during truck unloading or when new orders are generated. In contrast, pull-based systems assign tasks only when a forklift requests a new task, typically after completing its previous assignment. Each approach presents unique challenges in practical implementation [8, pp. 228-231]. For this study, the emphasis will be on push-based systems, which are more suited to dynamic, real-world use cases where transport orders must be allocated immediately to available forklifts [6].

Despite the potential advantages of FGSs, several persistent challenges remain:

- **Empty Runs**: As outlined in Section 1.2, reducing empty runs - when forklifts travel without a load - is a major challenge. Even with advanced algorithms, fully eliminating empty runs is practically impossible due to the dynamic and fluctuating nature of warehouse environments, which can disrupt task and resource alignment [9].
- **Task Allocation Complexity**: Simple algorithms such as FCFS can lead to unbalanced workloads, with some forklifts becoming overburdened while others remain idle. Advanced algorithms, such as combinatorial auction algorithms, which have been explored in studies such as that of Mirlach et al. [6], offer more sophisticated task allocation but are computationally intensive, especially in large-scale warehouse environments [14].
- **Adaptability to Dynamic Conditions**: Warehouse operations are subject to sudden changes in order volumes and task priorities. Maintaining system balance and avoiding bottlenecks in these real-time scenarios remains a significant challenge for FGSs [6].
- **Integration of Localization Systems**: The integration of real-time location tracking systems in FGS enables precise tracking of forklift positions, which can enhance task allocation by improving spatial awareness. However, research indicates that the overall impact on system efficiency may be marginal, with improvements in reducing empty runs being relatively

small [6]. This presents a trade-off between the cost and complexity of implementing localization systems and the relatively modest operational gains they provide.

In conclusion, while FGSs offer significant potential for optimizing forklift operations through improved task allocation, several challenges, including minimizing empty runs, addressing task allocation complexity, and adapting to dynamic warehouse environments, must be resolved. This thesis will focus on evaluating different algorithms, particularly push-based approaches, in both simulated and real-world contexts to address these operational inefficiencies.

3 Methods

Chapter 3 outlines the methodological framework used to evaluate and optimize the performance of FGS within warehouse operations. The primary focus is to describe the steps taken to design, implement, and test algorithmic approaches aimed at improving forklift task allocation, reducing empty runs, and ensuring overall warehouse operations. Simulation-based approaches are frequently employed in logistics to model complex environments and evaluate optimization strategies before implementation in real-world operations [9].

Building on the benchmark study framework, introduced earlier and inspired by the work of Mirlach et al. [6], this research modifies the methodology to evaluate different algorithms. The goal is to perform a comprehensive analysis of task allocation efficiency within warehouse operations.

This chapter begins by presenting the warehouse layout design and the simulation environment used to replicate real-world warehouse scenarios. The warehouse layout, operational data, and shortest path distance matrix were generated using Microsoft Excel (MS Excel) and Microsoft Visual Basic for Applications (VBA). The Plant Simulation software was then used to build the model, enabling automatic generation of various layouts and simulate task allocation strategies. Additionally, the Orders Generator created random orders and shifts using statistical distributions, ensuring that the model realistically represented warehouse operations. The simulation results were evaluated by calculating KPIs, such as system throughput, resource utilization, and task efficiency.

Following the layout design and data generation, this research investigates how different task allocation algorithms perform within the simulation model. Each algorithm is assessed for its effectiveness in optimizing task distribution and reduce inefficiencies such as empty runs.

The final section introduces a real-world use case, validating the proposed methods by comparing simulated results with operational data from a functioning warehouse. This practical application provides insights into the efficacy of the methods in real-world scenarios, supporting the overall goal of optimizing FGS in dynamic environments. Validating simulation results with real-world data is crucial to ensuring that proposed optimizations are both feasible and effective in practice.

The subsequent sections explore each methodological component in detail.

3.1 Warehouse Layout Design

To facilitate the simulation process, it was imperative to establish a defined warehouse layout. A grid layout was chosen, recognizing its manageability and balance between simplicity and practical implementation. This layout choice establishes system boundaries while remaining feasible for real-world applications.

The chosen grid layout is two-dimensional (2D) and comprises intersecting vertical and horizontal lines, creating a structured environment for forklift navigation. In the simulated high bay storage model, a fleet of forklifts, with quantities adjustable according to the parameters of each study, navigates this layout along predefined grid lines. The forklifts are limited to two directional movements, mirroring actual warehouse constraints.

The layout includes designated nodes, each with specific x and y coordinates, where the high bay pallet racks are strategically positioned. Each forklift is equipped to perform pickups and deliveries at these nodes (pallet racks), providing a clear framework for the simulation and establishing a realistic foundation for evaluating the performance of various algorithms and strategies within a dynamic warehouse environment.

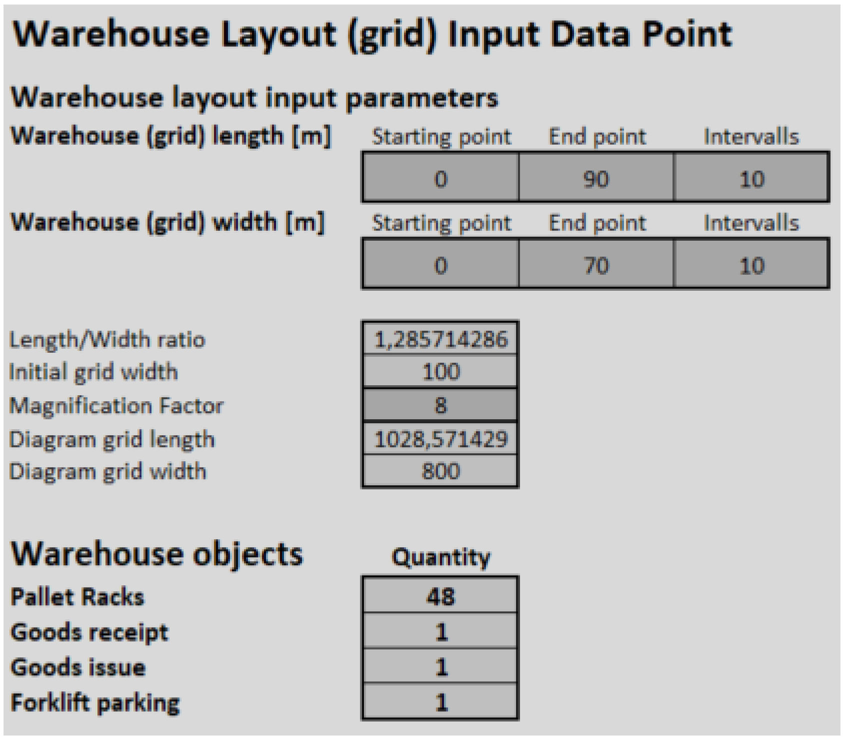
3.1.1 Warehouse Layout Generation

To acquire essential data for Technomatix Plant Simulation and generate the grid layout, MS Excel, supported by VBA, was utilized.

A Warehouse Layout Generator was developed, allowing the input of specific parameters for creating various grid layouts and the corresponding data. This data is then implemented into Plant Simulation for further modelling.

As illustrated in Figure 3-1, the process begins with setting the grid length, establishing the starting and end points, along with intervals (in meters). In this instance, a warehouse with a length of 90 meters and 10-meter intervals was created, following the same setup for the width (set at 70 meters with 10-meter intervals). The magnification factor, set to 8 in this case, influences various values, such as the length-to-width ratio, diagram grid length, and diagram grid width. Initial grid width was manually set at 100 in this case. The magnification factor is used in order to have a proportional grid layout in Excel.

The length-to-width ratio is calculated by dividing the warehouse length's endpoint by the warehouse width's endpoint. The diagram grind length is determined by multiplying the magnification factor by the initial grid width and then dividing by the length-to-width ratio. Similarly, the diagram grid width is calculated by multiplying the magnification factor by the initial grid width and then dividing by the length-to-width ratio.


Warehouse Layout (grid) Input Data Point

Warehouse layout input parameters

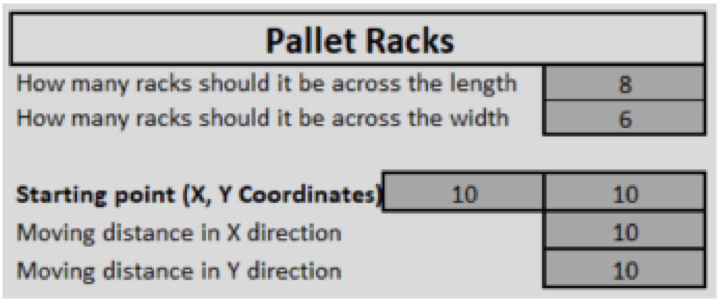
	Starting point	End point	Intervalls
Warehouse (grid) length [m]	0	90	10
Warehouse (grid) width [m]	0	70	10

Length/Width ratio	1,285714286
Initial grid width	100
Magnification Factor	8
Diagram grid length	1028,571429
Diagram grid width	800

Warehouse objects	**Quantity**
Pallet Racks	**48**
Goods receipt	**1**
Goods issue	**1**
Forklift parking	**1**

Figure 3-1: Warehouse Grid Layout – Input Data Interface for Specifying Warehouse-Specific Parameters.

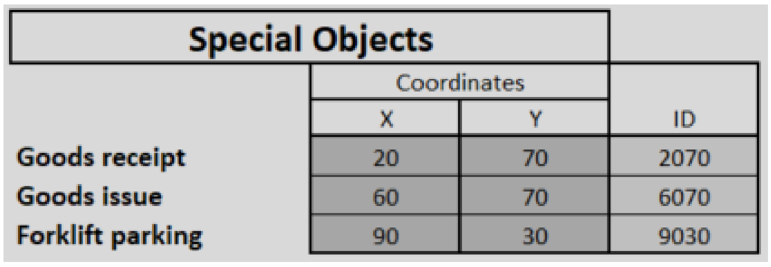
The Pallet Racks section specifies the number of racks distributed throughout the warehouse according to its length and width (see Figure 3-2). The starting point for the first rack is defined by x and y coordinates (in this case, 10 and 10), and the movement distances in both directions are established (here, 10 and 10 meters).


Pallet Racks

How many racks should it be across the length		8
How many racks should it be across the width		6
Starting point (X, Y Coordinates)	10	10
Moving distance in X direction		10
Moving distance in Y direction		10

Figure 3-2: Warehouse Grid Layout – Pallet Racks Interface for Specifying Quantity and Coordinates.

Subsequently, Special Objects, such as goods receipt, goods issue, and parking areas for forklifts, are then defined manually, with each assigned specific x and y coordinates to establish their precise locations within the warehouse (see Figure 3-3.)



Special Objects	Coordinates		
	X	Y	ID
Goods receipt	20	70	2070
Goods issue	60	70	6070
Forklift parking	90	30	9030

Figure 3-3: Warehouse Grid Layout – Special Objects Interface for Specifying Coordinates.

After defining all input parameters, VBA is used to generate a visual representation of the warehouse grid layout, as shown in Figure 3-4. This visualization displays the positions of objects and possible movement paths along horizontal and vertical axes. Additionally, a list of generically named pallet racks is generated based on the input parameters, combining the initial digits of the x and y coordinates for effective naming (see Figure 3-5). The Rack IDs are similarly created by combining x and y coordinates, which is essential for subsequent programming tasks. Special objects are also assigned distinct IDs based on their x and y coordinates (see Figure 3-3 for details).

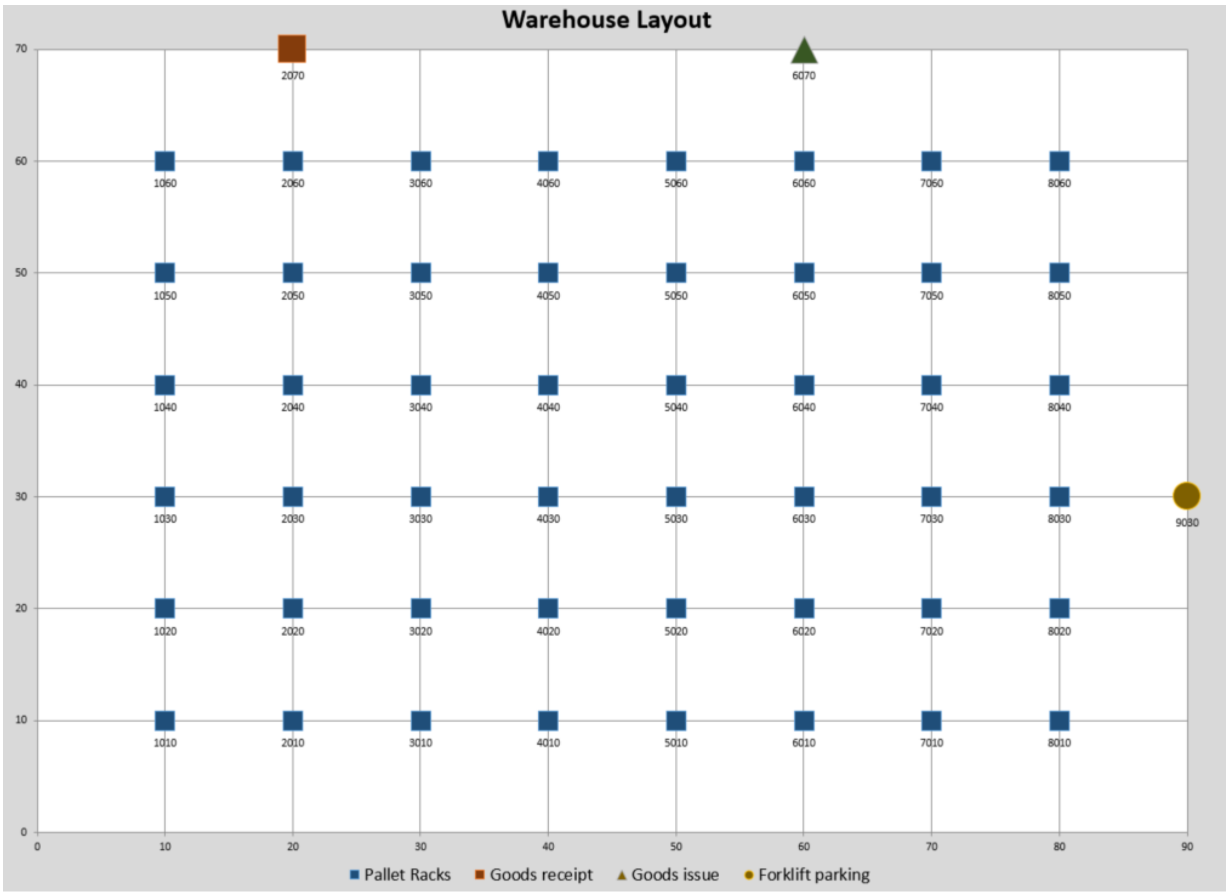
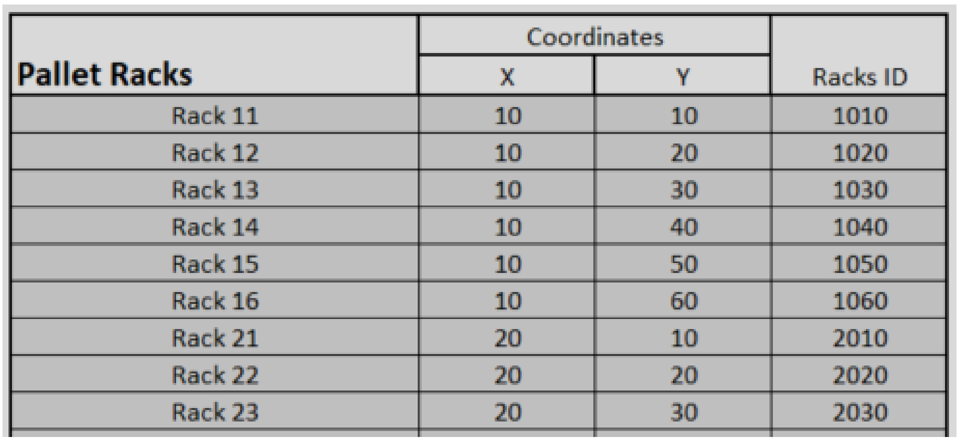


Figure 3-4: Warehouse Layout – Visual Representation of Grid and Object Placement.



Pallet Racks	Coordinates		Racks ID
	X	Y	
Rack 11	10	10	1010
Rack 12	10	20	1020
Rack 13	10	30	1030
Rack 14	10	40	1040
Rack 15	10	50	1050
Rack 16	10	60	1060
Rack 21	20	10	2010
Rack 22	20	20	2020
Rack 23	20	30	2030

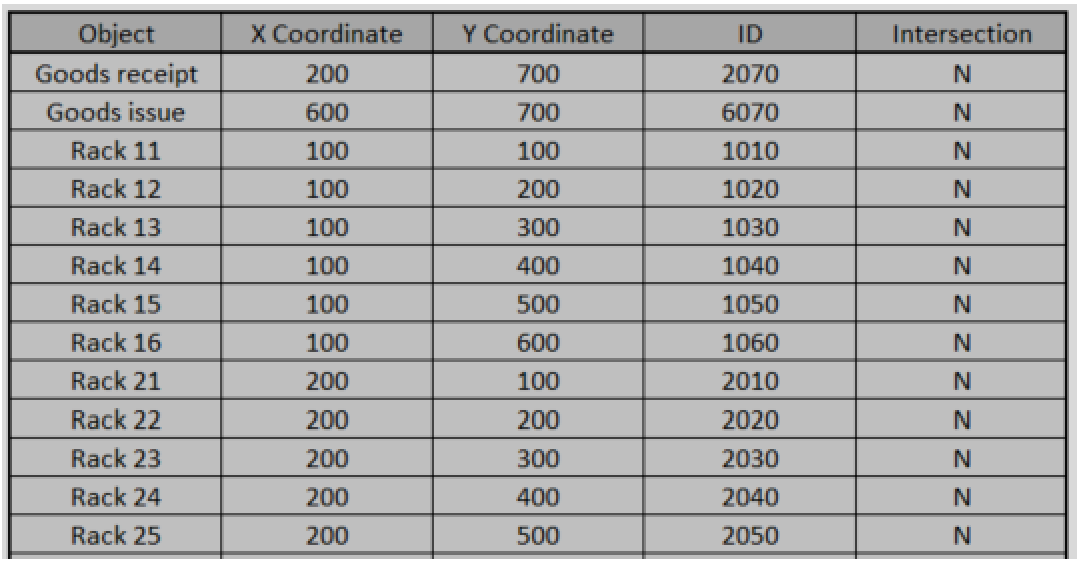
Figure 3-5: Pallet Racks Overview – Coordinates and Racks ID for Precise Positioning within Grid (snippet).

3.1.2 Input Data for Plant Simulation

Upon completing the layout grid and visualizing it in Excel, the next crucial step involves preparing the data for seamless integration into Plant Simulation.

Model Data

Once the layout is deemed suitable, the extraction of model data is required, encompassing all grid-located objects – both special objects (apart from the forklift parking) and pallet racks. To facilitate the transfer to Plant Simulation, the x and y coordinates are multiplied by 10 ensuring the coordinates are suitable for implementation in Plant Simulation. Alongside coordinates, essential information includes object IDs and the indication of whether each object is an intersection, a detail that will be relevant in the future real-life use case discussed later in this thesis (see Figure 3-6).



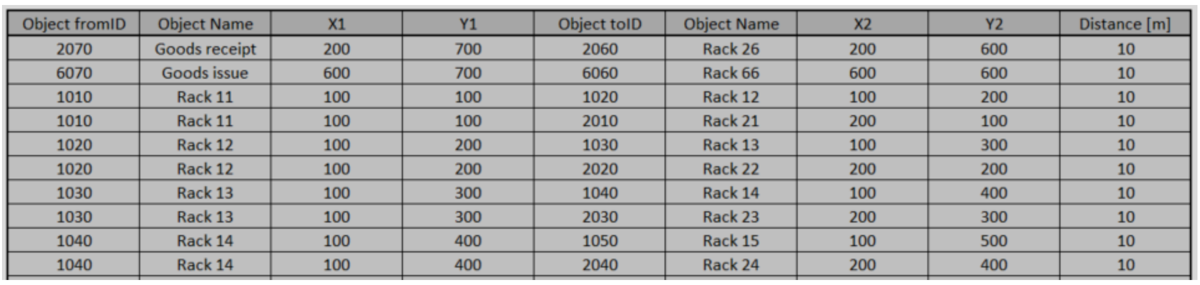
Object	X Coordinate	Y Coordinate	ID	Intersection
Goods receipt	200	700	2070	N
Goods issue	600	700	6070	N
Rack 11	100	100	1010	N
Rack 12	100	200	1020	N
Rack 13	100	300	1030	N
Rack 14	100	400	1040	N
Rack 15	100	500	1050	N
Rack 16	100	600	1060	N
Rack 21	200	100	2010	N
Rack 22	200	200	2020	N
Rack 23	200	300	2030	N
Rack 24	200	400	2040	N
Rack 25	200	500	2050	N

Figure 3-6: Model Data Overview - Coordinates, IDs, and Intersection Indicators for Precise Positioning of Warehouse Objects within Grid Layout (snippet).

Road Network Data

With the model data gathered to represent all objects, the focus shifts to the road network data – a compilation of all plausible connections between neighbouring model objects. For example, in Figure 3-4, the goods receipt (ID 2070) has a single neighbour, pallet rack ID 2060, while pallet rack ID 1010 has two neighbours: pallet rack ID 1020 and ID 2010.

Figure 3-7 illustrates the road network data, encompassing details on neighbouring objects necessary for implementation in Plant Simulation. For a comprehensive overview of the data included, refer to the figure.

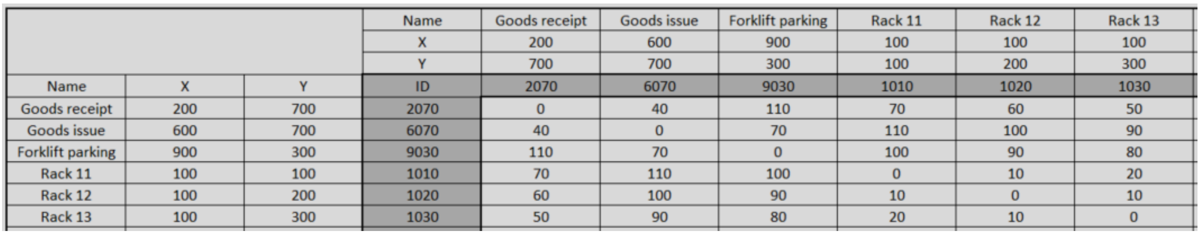


Object fromID	Object Name	X1	Y1	Object toID	Object Name	X2	Y2	Distance [m]
2070	Goods receipt	200	700	2060	Rack 26	200	600	10
6070	Goods issue	600	700	6060	Rack 66	600	600	10
1010	Rack 11	100	100	1020	Rack 12	100	200	10
1010	Rack 11	100	100	2010	Rack 21	200	100	10
1020	Rack 12	100	200	1030	Rack 13	100	300	10
1020	Rack 12	100	200	2020	Rack 22	200	200	10
1030	Rack 13	100	300	1040	Rack 14	100	400	10
1030	Rack 13	100	300	2030	Rack 23	200	300	10
1040	Rack 14	100	400	1050	Rack 15	100	500	10
1040	Rack 14	100	400	2040	Rack 24	200	400	10

Figure 3-7: Road Network Data - Overview of Data for Implementation in Plant Simulation (snippet).

Shortest Path Distance Matrix

To complete the requirements for Plant Simulation integration, a fully populated Shortest Path Distance Matrix is essential. A distance matrix is a square matrix representing the shortest paths between all pairs of objects, even when a direct connection is absent [24]. Unlike an adjacency matrix, which only indicates immediate connections with entries of 1 for direct links and 0 otherwise, the Shortest Path Distance Matrix ensures that each entry reflects the shortest path distance across the network, taking indirect routes into account. In this matrix, the distance between the same objects is inherently zero. Using VBA, the distances between all objects are calculated and systematically incorporated into this matrix. This matrix is critical in Plant Simulation, as it enables route optimization for transportation orders by providing accurate distances between objects for efficient task allocation and routing.



			Name	Goods receipt	Goods issue	Forklift parking	Rack 11	Rack 12	Rack 13
			X	200	600	900	100	100	100
			Y	700	700	300	100	200	300
Name	X	Y	ID	2070	6070	9030	1010	1020	1030
Goods receipt	200	700	2070	0	40	110	70	60	50
Goods issue	600	700	6070	40	0	70	110	100	90
Forklift parking	900	300	9030	110	70	0	100	90	80
Rack 11	100	100	1010	70	110	100	0	10	20
Rack 12	100	200	1020	60	100	90	10	0	10
Rack 13	100	300	1030	50	90	80	20	10	0

Figure 3-8: Shortest Path Distance Matrix, detailing the calculated shortest path distances between various objects in the warehouse layout. Each entry in the matrix represents the shortest distance between pairs of objects, incorporating indirect routes where necessary. Diagonal entries represent distances between the same objects and are inherently zero (snippet).

3.2 Orders Generator

After gathering the essential data for constructing the layout in Plant Simulation, the next step involves generating transport orders. This study simulates a one-week timeframe, covering five working days, with each day split into two shifts totalling 16 working hours (06:00-14:00 and 14:00-23:00). The weekly volume of transport orders is set at 28,500, aligning closely with the volume observed in the real-world use case while being scaled down to match the layout size and number of forklifts. Additionally, the proportion of transport orders across three categories has been maintained at a similar percentage to the real-world scenario.

These transport orders are categorized as follows:

- **Retrieval orders** – Involving the removal of pallets from the warehouse to designated factory production departments.
- **Storage orders** – Representing pallets arriving at the goods receipt area for subsequent storage within the warehouse.
- **Restorage orders** – Entailing the relocation of items within the warehouse to a different storage location.

Retrieval orders include a precise requirement time (latest completion), while storage and restorage orders are generated without a strict requirement time and with a lower priority compared to retrieval orders. For these order types, the time of order creation (trigger time) is synonymous with the requirement time, making it acceptable, if necessary, to exceed the requirement times.

Using the real-world case as a reference, Table 3-1 shows the distribution of transport orders by type.

Table 3-1: Transport order type overview.

Transport order type	#	%
Retrieval	19693	69,10
Storage	5000	17,54
Restorage	3807	13,36
Overall	28500	100,00

To facilitate order generation, VBA was employed under the following framework. Orders are created from Monday, October 16, 2023, to Friday, October 20, 2023, within a daily time frame of 06:00 to 22:00. To closely emulate real-world data, a combination of discrete uniform and normal distribution was chosen. This combination was selected to accurately reflect patterns observed in the real-world use case, where storage orders predominantly occur in the morning, while retrieval and restorage orders are distributed throughout the day.

Each transport order generated in this process is assigned a unique order ID. The order trigger time (moment of order creation) is essential for simulation purposes. The attributes ID_from and ID_to designate the origin and destination of each order, respectively. The order restriction is initially marked as N (not specified), subject to change in subsequent phases. The order type and requirement time are defined, with the requirement time set one hour after the trigger time for retrieval orders.

Figure 3-9 illustrates the average number of transport orders per hour across a typical day, categorized by order type. This distribution provides insights into the hourly demand for each order type, which is crucial for optimizing resource allocation and scheduling within the warehouse.

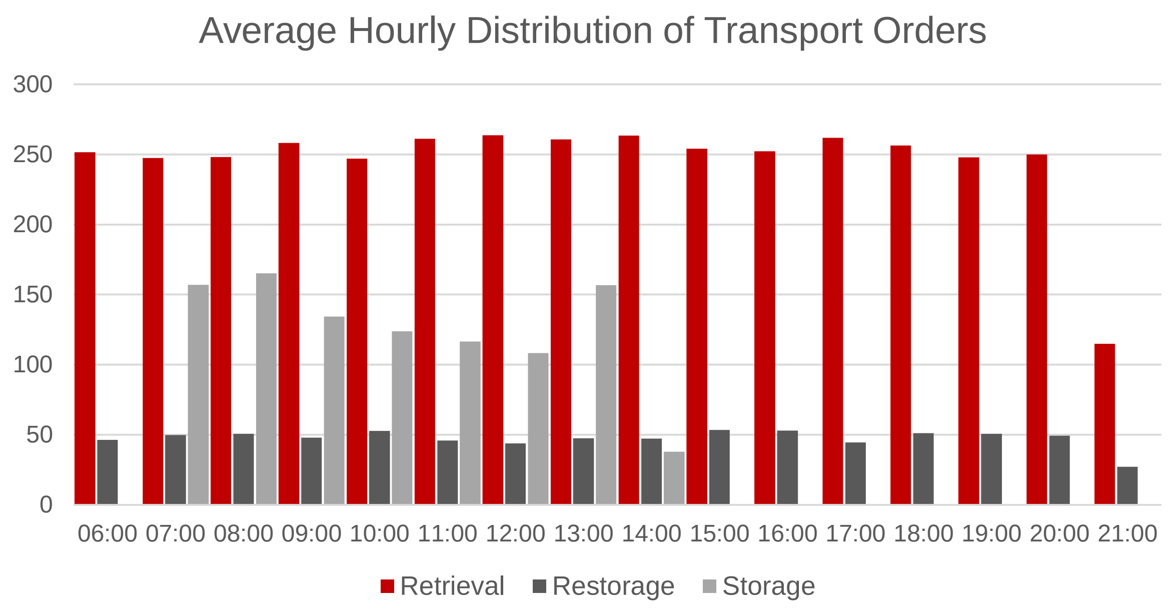


Figure 3-9: Average Hourly Distribution of Transport Orders by Type.

3.2.1 Discrete uniform distribution

Over a five-day span, 20,000 transport orders were generated, equating to 4,000 daily orders issued every 14 seconds starting at 06:00. Each order received a unique order ID, and random row numbers were employed, with random rows selected from the list of warehouse layout rack IDs (refer to Figure 3-5), to populate ID_to and ID_from. The order type was determined using the Rnd function (randomly generates values between 0 and 1), where values under 0.8125 generated retrieval orders; otherwise, orders were categorized as restorage. For retrieval orders, ID_to was set to 8030, the location for goods issue.

3.2.2 Normal distribution

In addition to uniform distribution, storage and retrieval orders were generated using normal distribution:

- **Storage orders**: Set to 1,000 daily with a standard deviation of 0.0833 (2/24), spanning the hours from 07:00 to 14:30, simulating truck deliveries. A random number between 8 and 25 determined the batch size per truck, representing varied pallet quantities. ID_from was set to 2070 (goods receipt location), while ID_to was assigned based on random rack IDs in the warehouse layout.

- **Retrieval orders**: Spread across the week with 3,500 total, averaging 700 per day. The standard deviation was again set at 0.0833, with orders generated between 06:00 and 21:00 to mimic pallet retrievals from storage. To simulate real-world conditions, random batch sizes between 1 and 4 were applied to similar retrieval tasks. ID_to was designated as 6070 (goods issue), and ID_from was randomly chosen from the rack IDs.

This phase of the order generation process concludes here, preparing essential data (illustrated in Figure 3-9) for further use in Plant Simulation as input and order data.

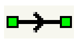
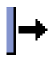
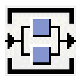
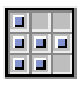
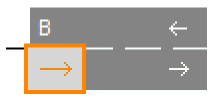
3.3 Simulation Model Setup

The simulation model serves as a pivotal component in this research objective to optimize warehouse operations. This section provides an in-depth look into the methodology behind the simulation model setup, covering the foundational aspects essential for comprehension. Detailed insights are provided on the individual Plant Simulation classes utilized in this study, with standard modules from Plant Simulation adapted for specific research purposes. These modules are categorized into three primary classes: material flow, resource flow, and information flow objects.

3.3.1 Model Components

This section provides a comprehensive overview of the simulation model's built-in objects systematically categorized into material flow, information flow, resource flow, user interface, and mobile unit (MU) objects. Table 3-2 furnishes a detailed insight into the roles and fundamental characteristics of each component.

Table 3-2: Technomatix Plant Simulation used built-in objects [25].

	Symbol	Meaning
Material Flow Objects		
Connector		The Connector establishes material flow connections between two objects in the same network. It indicates the direction of the connection with an arrowhead on the connection line.
Event Controller		The Event Controller coordinates and synchronises the events that take place during the simulation run. It allows functions to start, stop and reset the simulation.
Source		The source produces MUs (mobile units), as they are specified. It can produce different part types one after the other or in a mixed sequence. It is possible to set up a method for determining the production times and a method for determining the types of MUs to be produced.
Parallel Station		The parallel station has several workstations on which the incoming MUs are processed. Its properties are the same as those of a single station with several processing stations (Incoming parts are received for processing. It accepts a workpiece from its predecessor and passes it on in turn to one of its successors after the set-up and processing time has elapsed). An input control system can be used to determine the processing station to which the mobile object is transferred.
Store		The warehouse receives MUs and stores them until they are removed using a method.
Two Lane Track		With the Two Lane Track, transport routes with two lanes on which traffic runs in opposite directions can be modelled. It also provides the automatic route search function. The transporter is the only MU that can drive on the two lane track.

Resource Objects		
AGV Pool		With the AGV Pool, AGVs can be created. This allows to model automated guided vehicle systems (AGVs) or other types of transporters in the plant.
Shift Calendar		The shift calendar is used to define the different shifts during which the facility operates. As many shifts as needed can be defined. A shift pauses one or more machines in the model by setting the Boolean attributes Pause and Unscheduled of the corresponding material flow object.
Information Flow Objects		
Method	M	Controls can be programed with the object Method, which are then called up and started by other objects. Plant Simulation then executes these controls during the simulation run.
Variable	n=1	The Variable object serves as a global variable accessible to other objects and methods throughout a simulation run. It represents an item which stores a quantity. For example, they are used to store data over an extended period of time during a simulation run.
Data Table		The Data Table is a tabular structure consisting of two or more columns. Accessing specific cells is done using their index, determined by the row and column numbers.
User Interface Objects		
Html Report	HTML	Reports can be generated using the Html Report object, displayed by Plant Simulation as an HTML page.
Moving Units (MUs)		
Transporter		The Transporter is an active mobile material flow object with self-propelling capabilities, enabling independent movement on the length-oriented Two Lane Track in both forward and reverse directions. Moreover, it has the capability to load and transport various items such as Parts, Containers, and other Transporters.

Moving Units (MUs)		
Container	▦	The Container serves as a mobile material flow object designed for transporting other MUs. It can be used to model pallets, bins, boxes, etc.

3.3.2 Layout Implementation

Prior to testing the different FGS algorithms, the warehouse layout must be generated using the modelling methods as well as the imported model data and road network data. This process involves creating two layout modules: storage compartments and the road network with appropriate connectors.

The storage compartments, along with goods issues, goods receipt, are generated using two types of parallel stations: Parallel Station – Pallet (PP) and Parallel Station – Stacker (PS), with representative connectors. The reasoning behind modelling the storage compartments in this way will be explained in the following sections (see

Table 3-2 and 3.3.5). Multiple storage spaces can be consolidated under a single station, allowing for greater flexibility in layout design.
This model incorporates 48 storage compartments (stations), one goods issue area, one goods receipt area, and a parking lot for forklifts. The parking lot is represented by a store object.

Connections between these entities – storage compartments, goods issue, goods receipt, and the forklift parking lot – are established through two-lane tracks. These tracks are represented as straight lines, both vertically and horizontally, providing a simplified representation of routes between two stations. This representation assumes that the curves found in real cases have been accounted for, with their paths approximated as straight lines set 10 meters apart.

The forklift is characterized as a transporter object that can navigate the two-lane tracks within the layout, facilitating movement from the parking lot to various warehouse stations. The forklift's loading space is represented as a store, a matrix-oriented loading area with loading locations across the X and Y dimensions, onto which the transporter places loaded items, specifically pallets in this scenario. Forklift speed is set at 2 m/s, maintaining a constant speed with no acceleration. The number of forklifts, defined within the Forklift Pool (AGV Pool), varies as it is an optimization target. Initial trials indicated that with a certain number of forklifts, all transport orders were completed on time with each algorithm. Consequently, the number of forklifts was gradually reduced in subsequent tests to assess performance under different resource constraints.

The shift calendar is integral to the model, outlining operational shifts for forklifts, including any designated breaks. In this model, shifts extend from 06:00 to 14:00 and from 14:00 to 22:00, with breaks omitted for initial model simplicity. However, breaks will be included in the real-world case study to allow for a more comprehensive simulation.
Figure 3-10 presents a detailed representation of the warehouse layout, illustrating forklifts engaged in pallet transfer operations. A close review of the storage compartments is evident, showing their construction using two parallel stations. Additionally, a sample waiting transport order (pallet) is included.

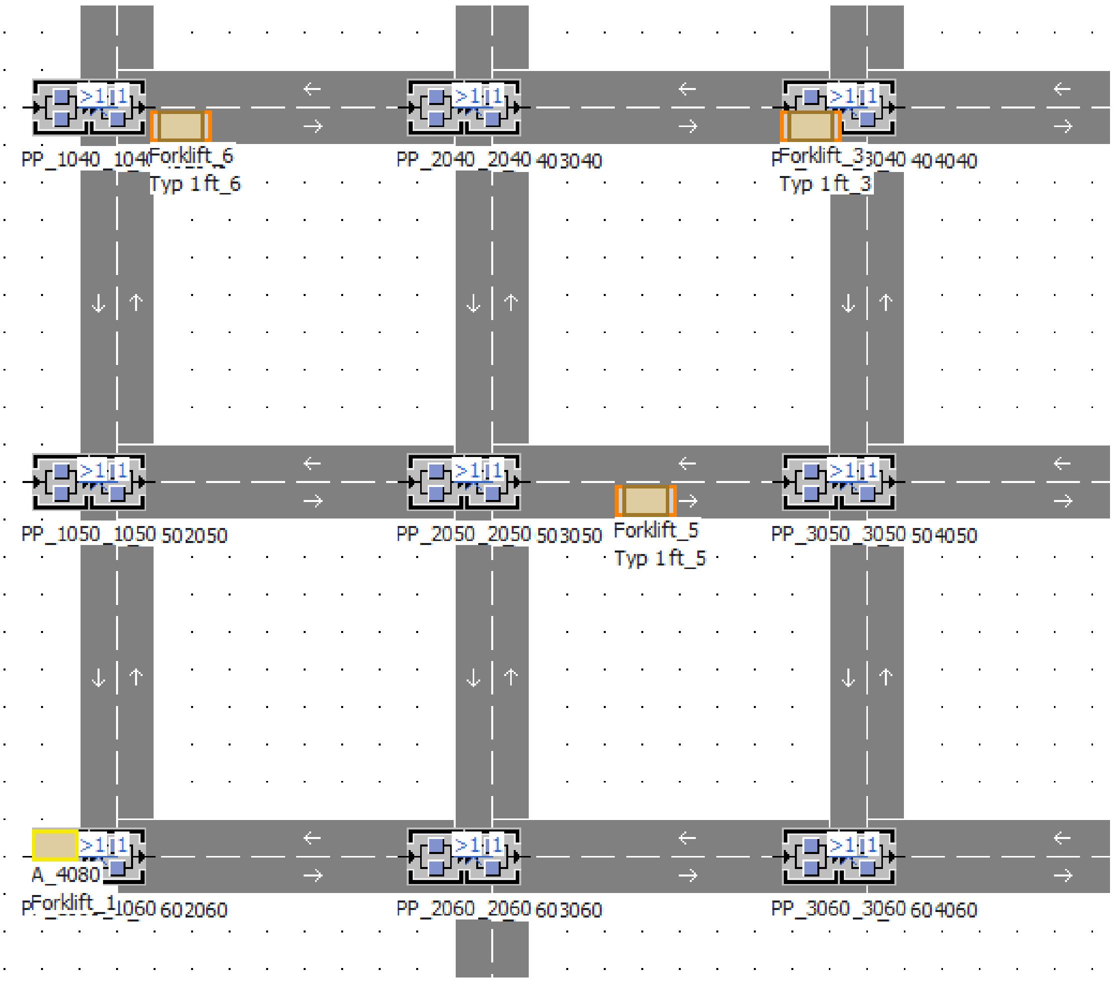


Figure 3-10: Detailed Warehouse Layout – This figure illustrates the warehouse layout, showing the pathways for forklift movement and storage compartments positioned at each intersection (node). Each storage compartment consists of two parallel stations: PP and PS. The pathways are designed as two-lane tracks to facilitate bidirectional forklift navigation. The orange-trimmed yellow rectangles represent forklifts, while the yellow-trimmed rectangles denote pallets waiting to be picked up.

3.3.3 Model Functionality

The preceding paragraph introduced the model creation process and the use of user objects. These objects act as model building blocks, derived from material flow or MU objects within the class library through duplication. By duplicating, each object can be given uniform attributes that can be quickly modified if necessary. Table 3-3 offers an exhaustive list of the user objects employed, with descriptions outlining adaptations made to suit the model. This serves as a precursor to the subsequent sections, where we the simulation logic and transport order execution are described.

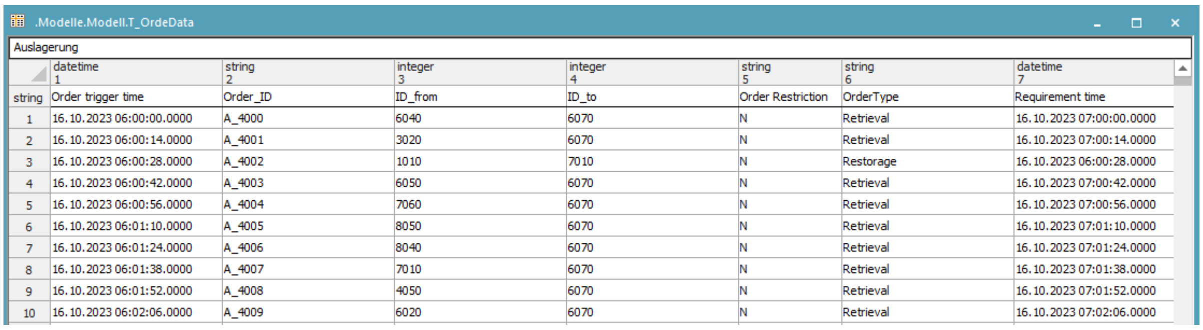
Table 3-3: Technomatix Plant Simulation User Objects.

Name	Function	User-defined attr.
S_Pallet	All transport orders are stored in a delivery list containing the order information. The time at which an order is placed by the FGS is determined by the "order trigger time" time stamp. When this time stamp is reached in the simulation, a pallet in the source (S_Pallet) is generated and transferred to the respective PP. Several orders/pallets can be generated simultaneously at the same time and transferred to the respective PPs.	- Delivery list
Origin		
Source		
Rel. Method		
M_S_OrderGeneration		
Name	Function	User-defined attr.
PP (Parallel station Pallet)	Upon the generation of a transport order (refer to S_Pallet), it is promptly transferred to the corresponding PP, and remains there waiting pickup by a forklift.	- Capacity of 10,000 pallets
Origin		
Parallel station		
Rel. Method		
M_PP_Incoming		
Name	Function	User-defined attr.
PS (Parallel station Stacker)	Upon the entry of a forklift into the parallel station stacker (PS), the scenario dictates either the loading of a pallet from the PP onto the forklift, unloading a pallet from the forklift, or transferring a pallet within the station.	- Capacity of 7 pallets - Forklifts have a handling (processing) time of 30 seconds
Origin		
Parallel station		
Rel. Method		
M_PS_Outgoing		
Name	Function	User-defined attr.
Forklift	As soon as a transport order is generated and the transfer of the pallet to the corresponding PP happens, the order details are dispatched and a selection of a forklift based on the respective FGS algorithm is initiated. The designated forklift initially receives the transport order pickup location. Additional information is provided to the forklift during the pickup process.	- Constant speed of 2 m/s - No acceleration - Loading area capacity of 1 (pallet) - Match (Order ID) - Call order - Destination list etc.
Origin		
Transporter		
Rel. Method		
CostCalcualtion (User-def. attr.)		

Name	Function	User-defined attr.
ForkliftParking	In the initialization phase, forklifts are relocated to their designated parking lots and remain paused until their first deployment. Throughout the simulation, forklifts move to the designated parking lots in case there are no assigned orders.	- Capacity of 100
Origin		
Store		
Rel. Method		
None		
Name	Function	User-defined attr.
Track	The route network is represented through interconnected two-lane tracks. These tracks serve to connect two endpoints, each with a specified length, in this case, 10 meters, based on the layout data.	- Track width 2 meters
Origin		
Two Lane Track		
Rel. Method		
None		
Name	Function	User-defined attr.
Forklift Pool	The forklift pool generates the entire forklift fleet during initialisation of the model. Forklift quantity is calculated based on the information stored in the table T_ForkliftPool.	
Origin		
AGV Pool		
Rel. Method		
None		

3.3.4 Simulation Initialisation

In the initialization phase of a simulation run, the event manager sets the simulation duration. When the simulation begins and before the first event occurs, the init method is invoked. Within this method, forklifts are created, named, and assigned to designated parking lots, where they are initially paused. Subsequently, order data is transferred from T_OrderData (see Figure 3-11) to T_DeliveryList (see Figure 3-12) with a predefined format from the source user object. The simulation then starts, and orders are processed according to the "Order trigger time" timestamp in the delivery list. The algorithms under investigation are responsible for task allocation, directing forklifts to execute their assigned orders according to the specific logic and prioritization defined by each algorithm. Once all orders (28,500) are completed, a cessation method is triggered.

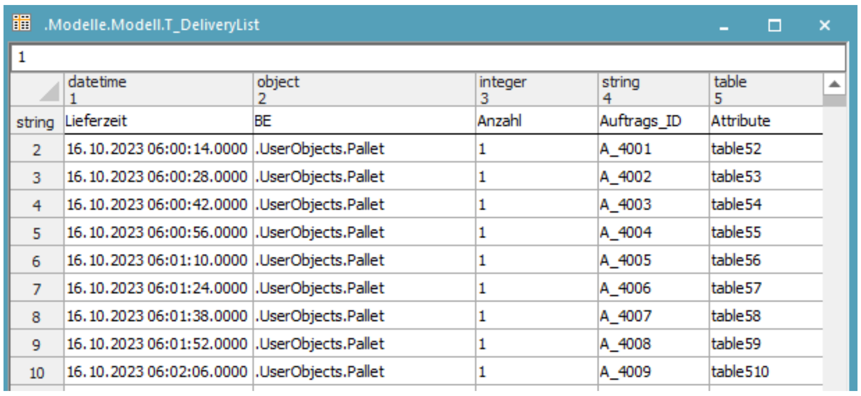


.Modelle.Modell.T_OrdeData

Auslagerung

	datetime 1	string 2	integer 3	integer 4	string 5	string 6	datetime 7
string	Order trigger time	Order_ID	ID_from	ID_to	Order Restriction	OrderType	Requirement time
1	16.10.2023 06:00:00.0000	A_4000	6040	6070	N	Retrieval	16.10.2023 07:00:00.0000
2	16.10.2023 06:00:14.0000	A_4001	3020	6070	N	Retrieval	16.10.2023 07:00:14.0000
3	16.10.2023 06:00:28.0000	A_4002	1010	7010	N	Restorage	16.10.2023 06:00:28.0000
4	16.10.2023 06:00:42.0000	A_4003	6050	6070	N	Retrieval	16.10.2023 07:00:42.0000
5	16.10.2023 06:00:56.0000	A_4004	7060	6070	N	Retrieval	16.10.2023 07:00:56.0000
6	16.10.2023 06:01:10.0000	A_4005	8050	6070	N	Retrieval	16.10.2023 07:01:10.0000
7	16.10.2023 06:01:24.0000	A_4006	8040	6070	N	Retrieval	16.10.2023 07:01:24.0000
8	16.10.2023 06:01:38.0000	A_4007	7010	6070	N	Retrieval	16.10.2023 07:01:38.0000
9	16.10.2023 06:01:52.0000	A_4008	4050	6070	N	Retrieval	16.10.2023 07:01:52.0000
10	16.10.2023 06:02:06.0000	A_4009	6020	6070	N	Retrieval	16.10.2023 07:02:06.0000

Figure 3-11: T_OrderData - Overview of Generated Orders from 3.2 (snippet).



.Modelle.Modell.T_DeliveryList

1

	datetime 1	object 2	integer 3	string 4	table 5
string	Lieferzeit	BE	Anzahl	Auftrags_ID	Attribute
2	16.10.2023 06:00:14.0000	.UserObjects.Pallet	1	A_4001	table52
3	16.10.2023 06:00:28.0000	.UserObjects.Pallet	1	A_4002	table53
4	16.10.2023 06:00:42.0000	.UserObjects.Pallet	1	A_4003	table54
5	16.10.2023 06:00:56.0000	.UserObjects.Pallet	1	A_4004	table55
6	16.10.2023 06:01:10.0000	.UserObjects.Pallet	1	A_4005	table56
7	16.10.2023 06:01:24.0000	.UserObjects.Pallet	1	A_4006	table57
8	16.10.2023 06:01:38.0000	.UserObjects.Pallet	1	A_4007	table58
9	16.10.2023 06:01:52.0000	.UserObjects.Pallet	1	A_4008	table59
10	16.10.2023 06:02:06.0000	.UserObjects.Pallet	1	A_4009	table510

Figure 3-12: T_DeliveryList - Preformatted Order Data (snippet)

3.3.5 Execution of transport orders

As soon as an order is triggered, the source object (S_Pallet) generates a pallet (transport order) containing all necessary information. The source entrance control uses the M_Q_Orders method to transfer the pallet (transport order) to the respective PP, retrieving the relevant information from the pallet. Upon the pallet's entry into the PP, the "M_PP_Incoming" method is activated, performing two key tasks. First, the method invokes additional procedures to handle forklift selection. Second, it facilitates the transfer of order information from the pallet to the designated forklift.

When a forklift arrives at a PS, the "M_PS_Outgoing" method is triggered to determine whether the forklift is loaded or empty. If the forklift is empty, it proceeds to load a pallet, specifying the destination (sink of the order). If it is already loaded, the forklift unloads the pallet. After this, the forklift either proceeds to its next destination (if orders remain on its processing list) or returns to its designated parking area.

3.4 Algorithms

In warehouse management and logistics, efficiently allocating resources is crucial for optimizing operations. Algorithms, as systematic step-by-step procedures, play a significant role in achieving this efficiency. FGSs, designed to streamline task allocation and resource optimization, depend heavily on algorithms to manage the assignment and routing of transport orders.

An algorithm, in the context of this study, refers to a set of rules and procedures designed to determine the assignment of transport orders to forklifts in a simulated warehouse environment [26, pp. 5-15] . The ability of FGS to quickly and efficiently allocate transport orders to available forklifts ensures smooth operations and minimizes costly inefficiencies, such as empty runs and imbalanced workload distribution.

Transport orders are assigned immediately upon entry into the system, allowing forklifts to handle multiple transport orders at the same time. In such cases, these transport orders are processed systematically, aiming for the optimal sequence. This sequencing is critical for minimizing delays and ensuring tasks are completed efficiently, one of the core challenges in warehouse operations. Algorithms play a key role in optimizing this process by determining the best way to assign, prioritize, and route tasks in real time, ultimately improving system throughput and reducing resource waste. This creates an opportunity for optimization.

The following sections explore and analyse three specific algorithms employed for FGS within the simulated warehouse model. Each algorithm represents a distinct strategy for tackling the challenges associated with order assignment, considering factors such as forklift availability, task complexity, and minimization of travel distance. These algorithms are essential for improving the operational efficiency of FGS, where timely and accurate task assignment can significantly impact overall warehouse performance. Each algorithm offers a unique approach to addressing the challenges related to order assignment, taking into account factors such as forklift availability, order priorities, and operational efficiency.

For this research, three algorithms, which will be detailed in the following chapter, are tested within the simulation model outlined earlier. After evaluating their performance in the simulated environment, the best-performing algorithm will be applied to a real-world use case, utilizing actual warehouse data and layouts for further validation.

3.4.1 Sequential Forklift Assignment Algorithm (A1)

The Sequential Forklift Assignment Algorithm follows a FCFS approach for assigning transport orders to forklifts. Numerical identifiers, ranging from Forklift-1 onward, determine the order in which forklifts receive assignments. Each incoming transport order is assigned to the next available forklift according to this predefined sequence.

This algorithm is simple, fast, and computationally inexpensive, making it straightforward to implement [8, pp. 320-322]. However, while it is efficient in execution, this approach does not account for dynamic factors such as task priorities or distances.

FCFS method serves as a baseline for comparison with more advanced algorithms in this study, providing insight into the potential performance gains that can be achieved by incorporating more sophisticated optimization techniques.

3.4.2 Priority-Based Algorithm (A2)

The Priority-Based Algorithm introduces a more nuanced approach, integrating two critical factors – routing distance and order backlog – to determine task assignment. The order list is the queue of transport orders assigned to a forklift, where orders wait while the forklift completes its current task. Routing distance is the computed distance between the sink of the last order on the order list and the source of the new transport order. In contrast, the order backlog represents the quantity of pending orders in the order list.

This algorithm evaluates different scenarios by applying various weightings to these two factors. By adjusting the weightings, this approach explores the trade-offs between minimizing travel distance and addressing the backlog of pending orders, offering insights into the most effective balance for optimizing forklift operations.

The following table provides an overview of the different Weightings Factors (WF) that will be tested in this study.

Table 3-4: Priority-Based Algorithm WF overview.

	WF Routing distance (WF_{RD})	WF Orders backlog (WF_{OB})
A2a – Sim. study case 1	1.0	0.0
A2b – Sim. study case 2	0.8	0.2
A2c – Sim. study case 3	0.6	0.4
A2d – Sim. study case 4	0.4	0.6
A2e – Sim. study case 5	0.2	0.8
A2f – Sim. study case 6	0.0	1.0

The Priority-Based Algorithm assigns the transport order to the forklift with the smallest Priority. The Priority is calculated by first determining the Routing Distance and Orders Backlog for each forklift, followed by identifying the maximum values for both metrics across the forklift fleet.

To calculate the Forklift Distance Weighting (F_{DW}), if the maximum routing distance is greater than zero, it is determined by dividing the routing distance by the maximum routing distance; otherwise, it is simply equal to the routing distance. Similarly, the Forklift Orders Weighting (F_{OW}) is calculated by dividing the orders backlog by the maximum orders backlog if the maximum orders backlog is not zero; if it is zero, it remains equal to the orders backlog.

The Priority (P) for each forklift is calculated based on a combination of the Routing Distance and Orders Backlog, weighted according to predefined factors (F_{DW} and F_{OW}). The equation below illustrates how the Priority value is determined for each forklift, taking into account the WFs for both routing distance (WF_{RD}) and order backlog (WF_{OB}). The forklift with the lowest Priority value is then selected to handle the next transport order.

$$P = WF_{RD} * F_{RDW} + WF_{OB} * F_{OW}$$

These various weightings allow the algorithm to adapt under different conditions, revealing how each factor impacts the performance of the system. This approach provides insights into the algorithm's adaptability and effectiveness under different operational scenarios.

3.4.3 Auction-Based Routing Algorithm (A3)

The Auction-Based Routing Algorithm addresses a complex set of challenges within the warehouse environment, emphasizing the reduction of empty runs, ensuring on-time order completion, and potentially minimizing the overall fleet size of forklifts. Secondary objectives are the consideration of the total transport distance of all forklifts and their even utilisation. This algorithm draws inspiration from the auction-based system proposed by Mirlach et al. [6]. This study adapts their approach, with several key modifications to better suit the dynamic nature of forklift guidance in real-time operations. These modifications, such as the penalty factor and routing optimizations, will be explained in the process of detailing the algorithm below.

An essential mathematical perspective underlying this problem is the well-known Traveling Salesman Problem (TSP), which involves finding the shortest possible route for a salesman to visit a set of locations and return to the starting point. The TSP is a foundational problem in optimization and logistics because it addresses the challenge of minimizing travel distance or time across multiple locations, making it directly applicable to forklift routing in a warehouse environment [27, pp. 1-5]. In the realm of TSP solution approaches, a fundamental distinction can be made between exact and heuristic methods. While exact methods, aim for precise solutions, they become impractical for large problem instances due to computational inefficiencies. Heuristic methods, on the other hand, offer approximate solutions that are computationally more efficient. These approaches allow the algorithm to find near-optimal solutions within a reasonable timeframe, making them suitable for dynamic and complex warehouse environments.

The challenges posed by warehouse logistics, particularly in the context of forklift routing optimization, often demand practical and efficient solutions. Heuristic approaches become indispensable in such scenarios, providing reasonably optimal solutions within a reasonable timeframe. As the simulation deals with a dynamic and intricate warehouse environment, the Auction-Based Routing Algorithm employs heuristic techniques to deliver effective and timely results.

In the real-world use case, due to the high lifting heights required in high bay storage areas, only forklifts with sufficient lifting capabilities are suitable for handling storage operations in these zones. Typically, the distinction between lifting heights is not made for individual tasks; instead, assignments are allocated based on the storage area. Consequently, there are two types of forklifts: Type 1 forklifts, which are versatile and capable of handling all transport orders due to their extended lifting reach, and Type 2 forklifts, which have restricted lifting capabilities and are only suitable for certain areas within the warehouse.

When a transport order is placed, the system initially checks whether there is an order restriction. If such order restrictions are in place, the assignment of the order is limited to Type 1 forklifts, which are capable of reaching the required heights. In this case, the forklift pool – the set of forklifts available to undertake the order – excludes Type 2 forklifts, reducing the pool size. In the absence of any order restrictions, all forklifts, including Type 2, are eligible to undertake the transport order, resulting in a larger available pool of forklifts.

For this simulation study, however, only Type 1 forklifts are modeled to simplify the model. Unlike the real-world use case, this simulation includes all forklifts in the fleet within the forklift pool, allowing any forklift to handle any order, as lifting height restrictions are disregarded.

Subsequently, each forklift in the relevant forklift pool submits an acquisition cost, representing the takeover expenses associated with the assignment of the order. The order is then assigned to the most cost-effective forklift. Figure 3-13 illustrates a flowchart of the optimized routing algorithm.

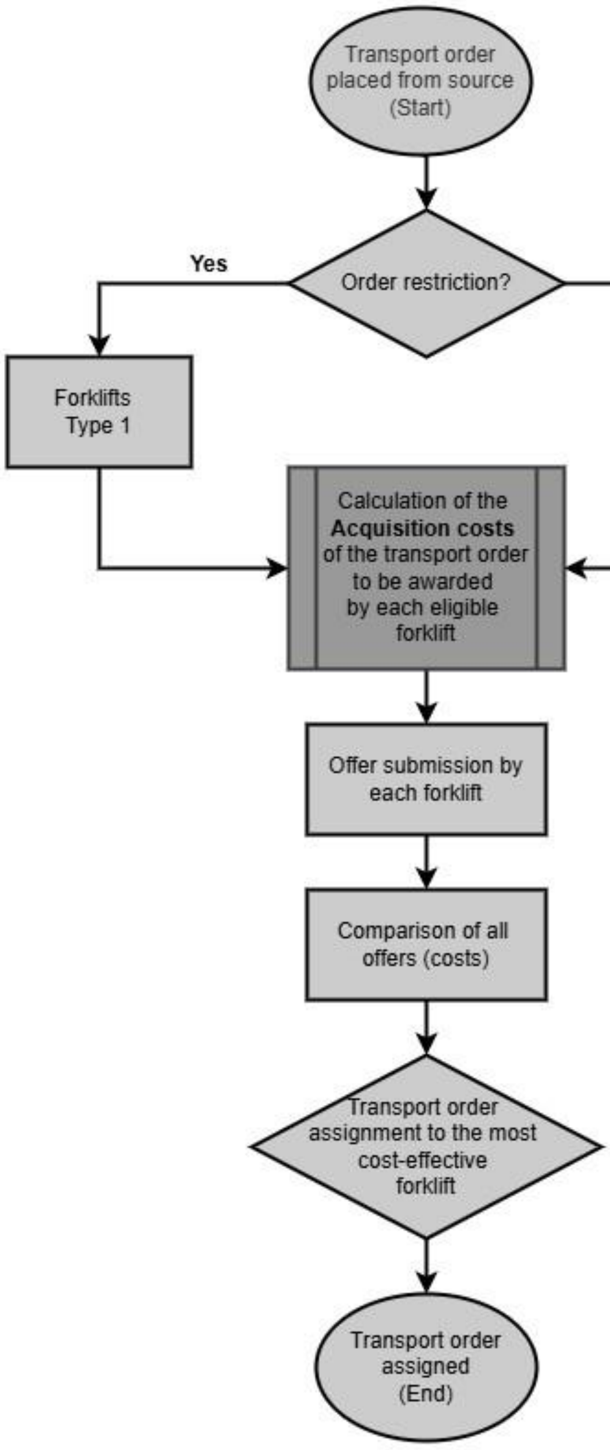

Transport order placed from source (Start)
Order restriction?
Yes
No
Forklifts Type 1
Forklifts Type 1 and 2
Calculation of the **Acquisition costs** of the transport order to be awarded by each eligible forklift
Offer submission by each forklift
Comparison of all offers (costs)
Transport order assignment to the most cost-effective forklift
Transport order assigned (End)

Figure 3-13: Auction-Based Routing Algorithm (A3) Transport order Assignment.

To compute the acquisition costs, each forklift generates an optimized tour (order sequence) for all open transport orders, incorporating the pending transport orders from the order list, along with the new transport order to be assigned. The tour creation process not only emphasizes minimizing transport distance but also takes into account order requirement times and priorities, ensuring the timely processing of transport orders. The acquisition costs signify the difference in total transport distance between the tour with the new transport order to be assigned and the tour without it. The calculation of distances, relies on the shortest path distance matrix discussed in Section 3.1.2. Assuming route-optimized tours for both scenarios, the additional route necessary to accommodate the new order is quantified.

Figure 3-14 provides a detailed breakdown of the acquisition cost calculation across four different cases, each of which is distinguished by the number of current transport orders and the length of the order list (OL). The main objective is to determine the additional cost incurred by introducing a new transport order, with more complex scenarios factoring in additional considerations such as existing transport orders and OLs.

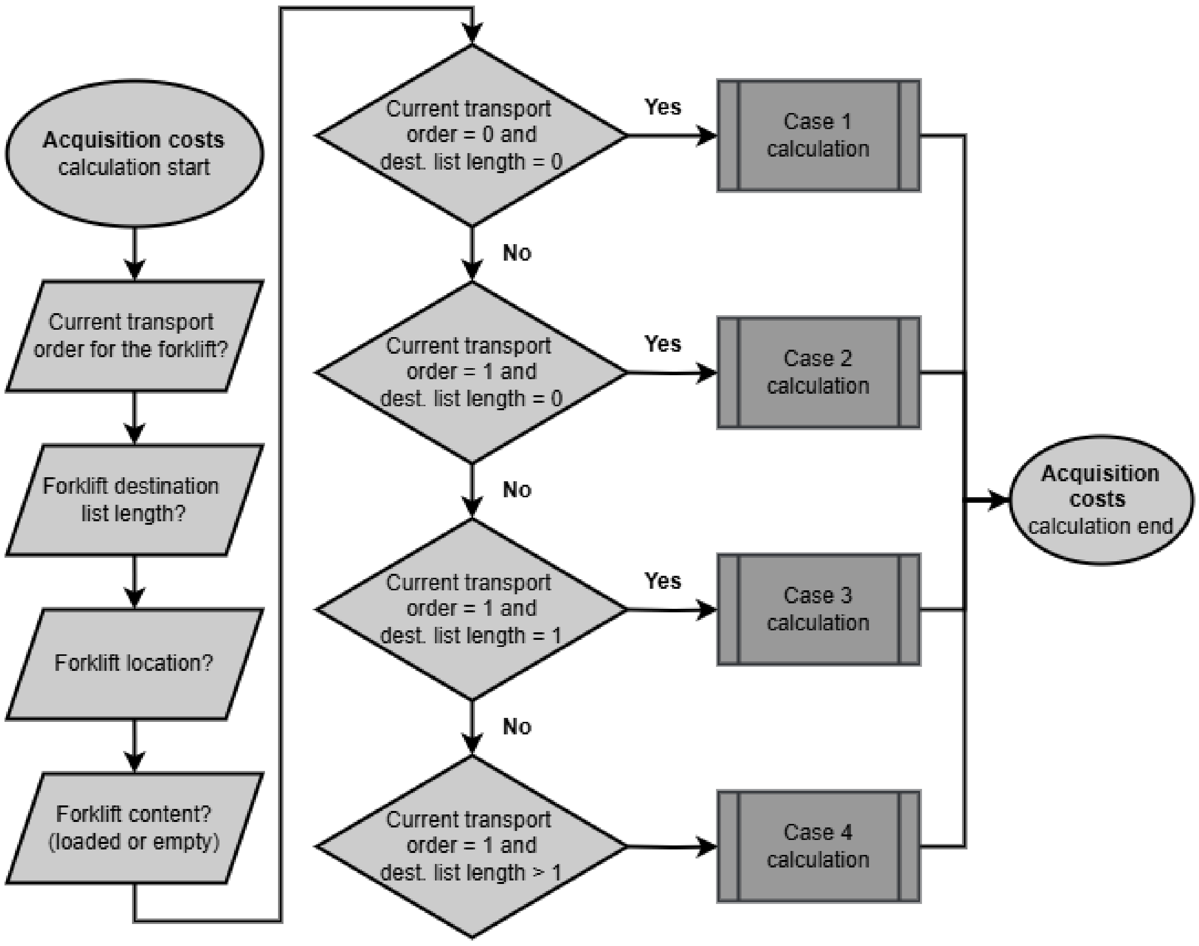

Acquisition costs calculation start
Current transport order for the forklift?
Forklift destination list length?
Forklift location?
Forklift content? (loaded or empty)
Current transport order = 0 and dest. list length = 0
Yes
No
Current transport order = 1 and dest. list length = 0
Yes
No
Current transport order = 1 and dest. list length = 1
Yes
No
Current transport order = 1 and dest. list length > 1
Case 1 calculation
Case 2 calculation
Case 3 calculation
Case 4 calculation
Acquisition costs calculation end

Figure 3-14: Acquisition Cost Calculation Process for Task Assignment.

In Case 1, the simplest scenario, there are no current transport orders assigned to the forklift, and the OL length is zero. As shown in Figure 3-15, the acquisition cost in this case is determined purely based on the distance for the new transport order. Since there are no pre-existing tasks or destinations, the system calculates the acquisition cost solely by evaluating the distance from the forklift's current location to the source and destination of the new order. This straightforward calculation ensures minimal complexity; as no additional factors such as task backlog or order restrictions need to be considered.

In this case, as well as in all subsequent cases, the transport order type is checked before calculating the acquisition costs. As noted in Section 3.2, retrieval transport orders are associated with a requirement time or latest completion time.

For these orders, the system calculates the order fulfillment duration and compares it with the requirement time. If the system determines that the order cannot be fulfilled within the specified time limit, the acquisition of the order is denied. This denial occurs in Cases 1, 2, and 3 when the time restriction cannot be met. In Case 4, the process becomes more complex due to additional factors involved in the decision-making process.

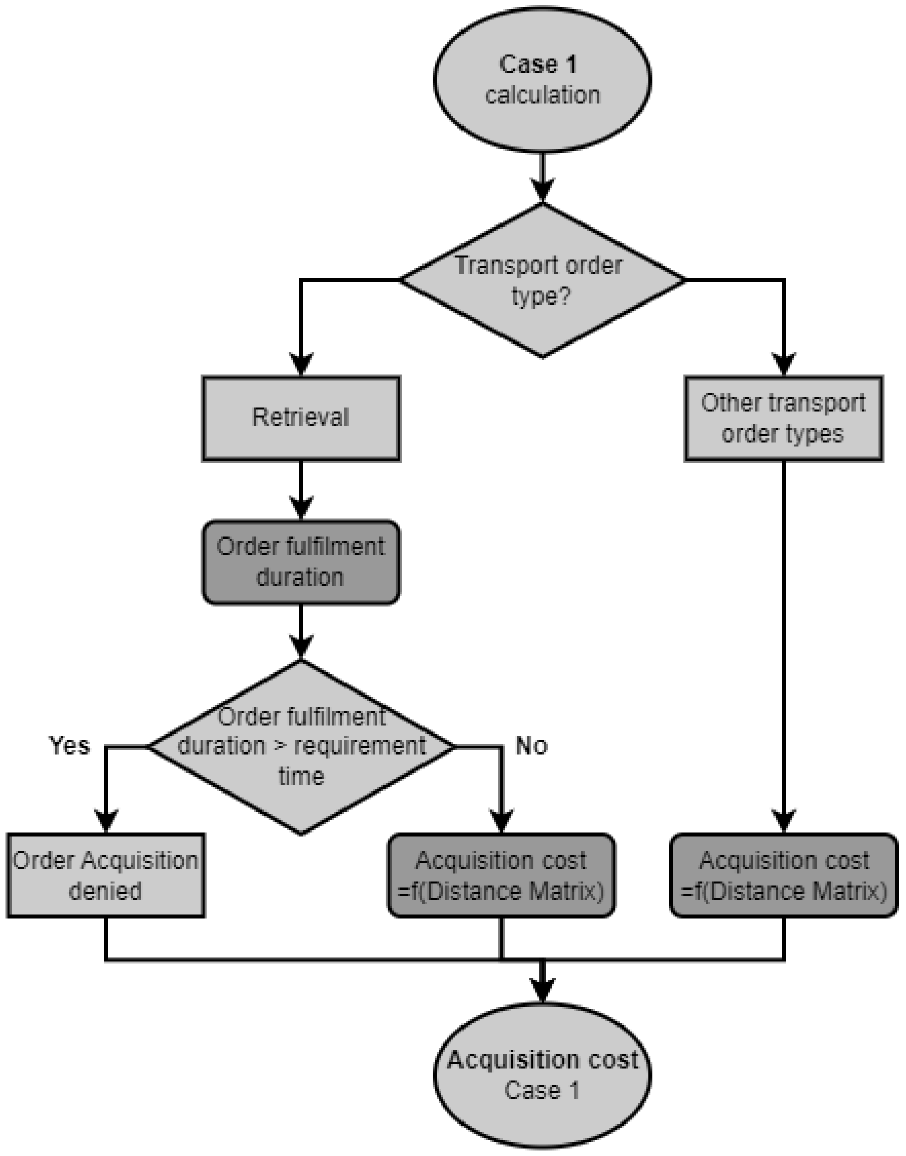

Case 1 calculation
Transport order type?
Retrieval
Other transport order types
Order fulfilment duration
Order fulfilment duration > requirement time
Yes
No
Order Acquisition denied
Acquisition cost =f(Distance Matrix)
Acquisition cost =f(Distance Matrix)
Acquisition cost Case 1

Figure 3-15: Acquisition Cost Calculation for Case 1 - No Existing Transport Orders.

For Case 2, a single existing transport order is already assigned to the forklift, but the OL length remains empty. As shown in Figure 3-16, the acquisition cost calculation must now account for the distance from the final destination of the current task to the source of the new transport order. This introduces slightly more complexity compared to Case 1, as the system must calculate the additional travel required to complete both tasks in sequence.

After verifying that the requirement time for any retrieval transport orders can be met, the system calculates the new acquisition cost. This cost reflects the difference between the total distance of the new tour, which includes the new or-der, and the existing transport cost for the current order alone.

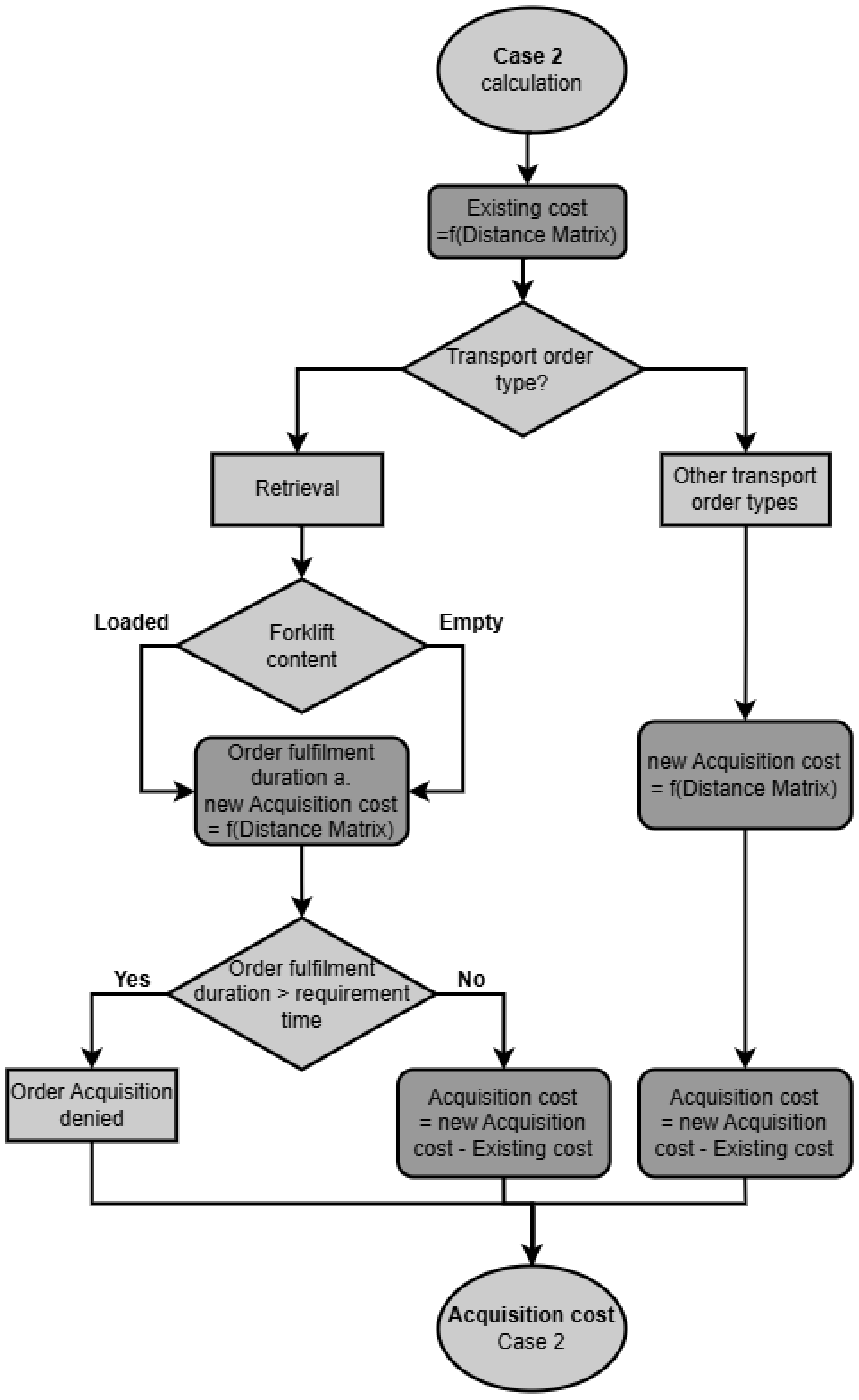

Case 2 calculation
Existing cost =f(Distance Matrix)
Transport order type?
Retrieval
Other transport order types
Loaded
Forklift content
Empty
Order fulfilment duration a. new Acquisition cost = f(Distance Matrix)
new Acquisition cost = f(Distance Matrix)
Yes
Order fulfilment duration > requirement time
No
Order Acquisition denied
Acquisition cost = new Acquisition cost - Existing cost
Acquisition cost = new Acquisition cost - Existing cost
Acquisition cost Case 2

Figure 3-16: Acquisition Cost Calculation for Case 2 - Single Existing Transport Order.

In Case 3, the complexity increases further as there is now one current transport order, and an additional order on the OL. This means that after completing the current task, the forklift already has another task waiting in its queue. As shown in Figure 3-17, the acquisition cost must now consider the distance needed not only to fulfill the current transport order but also to accommodate the next order on the OL. In other words, the system evaluates the additional distance that the forklift must travel to complete the new transport order, while also considering the current and pending order.

As in the previous cases, the system first verifies that the requirement time for any retrieval transport orders is met before proceeding. If the time restriction cannot be fulfilled, the acquisition of the new task is denied. Otherwise, the new acquisition cost is calculated by determining the difference between the total transport cost for the updated route, which includes the new order, and the cost for completing the existing transport orders. When the acquisition costs are calculated, the new transport order is temporarily added to the interim order list (IOL).

The concept of the IOL is critical here, as it holds tasks temporarily until the order assignment is finalized. Once the acquisition cost for each forklift has been determined (see Figure 3-13), the system compares the calculated acquisition costs for all forklifts and assigns the new task to the forklift with the lowest acquisition cost. For the selected forklift, the IOL becomes finalized, and the new transport order is officially added to the forklift's OL, replacing the temporary status of the task.

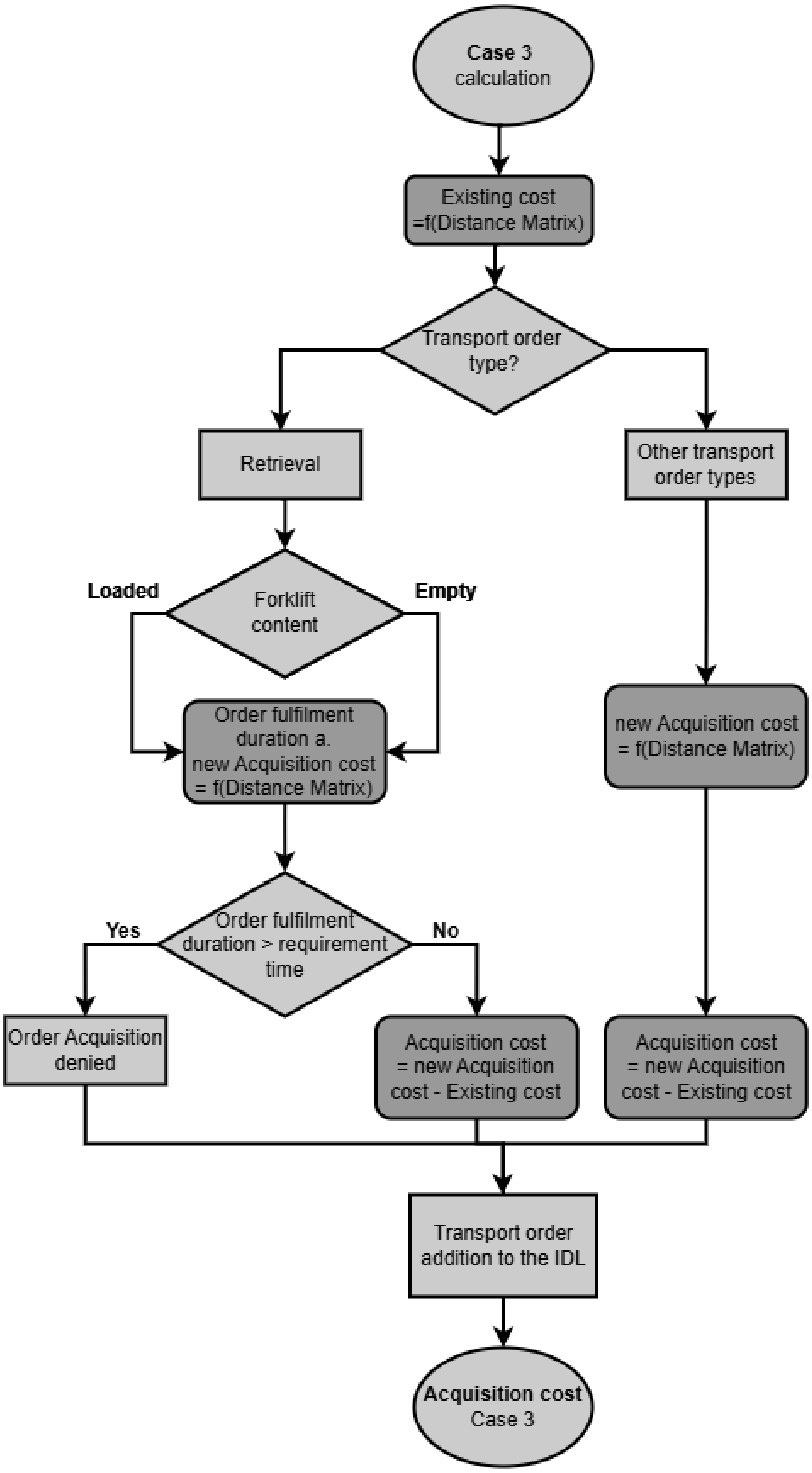

Case 3 calculation
Existing cost =f(Distance Matrix)
Transport order type?
Retrieval
Other transport order types
Loaded
Forklift content
Empty
Order fulfilment duration a. new Acquisition cost = f(Distance Matrix)
new Acquisition cost = f(Distance Matrix)
Yes
Order fulfilment duration > requirement time
No
Order Acquisition denied
Acquisition cost = new Acquisition cost - Existing cost
Acquisition cost = new Acquisition cost - Existing cost
Transport order addition to the IDL
Acquisition cost Case 3

Figure 3-17: Acquisition Cost Calculation for Case 3 - Two Existing Transport Orders.

In Case 4, the complexity increases significantly, as the forklift must handle at least three transport orders: the current transport order, the orders already present in the OL, and the new order being assigned. To determine the acquisition cost accurately, the algorithm must consider not only the next task but the entire sequence of tasks on the OL and the new transport order, integrating these into a Temporary Order List (TOL) for the subsequent optimization process.

This process involves a greedy heuristic that evaluates and prioritizes each task in the Temporary Order List (TOL) by minimizing travel distance and taking time constraints into account, particularly for retrieval orders nearing their requirement time. In this context, the greedy approach sequentially selects the most immediately optimal task, focusing on the one that requires the shortest additional travel from the last completed task, as in the nearest-neighbour algorithm. The Nearest Neighbour algorithm, widely used in routing problems due to its simplicity and computational efficiency, is a suitable choice for forklift guidance systems because it selects the next task by choosing the transport order closest to the last completed task. Although it does not guarantee an optimal solution, its speed and practicality in real-time systems make it highly effective for this use case, where quick decision-making is essential [26, pp. 1106-1140].

For retrieval orders and tasks with a requirement time, the travel distance is scaled accordingly to elevate the priority of time-sensitive tasks. This method enables the algorithm to determine the most efficient sequence for fulfilling tasks from the TOL, optimizing the overall route while maintaining timely order completion.

During each iteration, the algorithm selects the transport order with the shortest effective distance, known as the Compact Route Distance (CRD) (see Case 4 flowchart, Figure 3-18), and places it in the Intermediate Order List (IOL). The IOL acts as an interim list where the best-suited sequence of tasks is constructed. Once an order is added to the IOL, it is removed from the TOL, and the next iteration begins, identifying the next best-suited order. This process continues until all tasks from the TOL have been transferred to the IOL in the optimal sequence.

Once the IOL is finalized, the acquisition cost is calculated by comparing the total cost of the new route – which includes the newly assigned transport order – with the initial existing cost for fulfilling the previously assigned order. This final acquisition cost reflects the additional travel and time required to integrate the new task into the forklift's existing route, allowing for an efficient and dynamic task allocation.

To prioritize time-sensitive retrieval orders, the algorithm applies a Retrieval Prioritization Factor (RPF) to scale the travel distance associated with each task. The RPF is applied specifically to retrieval transport orders that are nearing their requirement time. When less than 45 minutes remain until the deadline (from the time the order is placed), the algorithm adjusts the travel distance using a scaling factor $f<1$, which artificially reduces the distance for these retrieval orders, making them more likely to be prioritized.

The theoretical travel distance is calculated as follows in the following equation.

$$d_{th} = d \cdot \frac{t_{DL}}{t_{limit}}$$

Here, d_{th} represents the scaled distance, d is the actual traveling distance, t_{DL} is the remaining time from order placement to the requirement time, and t_{limit} is the predefined time threshold of 45 minutes. This adjustment ensures that time-sensitive retrieval orders are prioritized, even if they require a longer travel distance than other tasks. If retrieval orders cannot be completed on time, d_{th} assumes negative values, prioritizing these orders ahead of all others.

To achieve balanced task distribution across the forklift fleet, a Penalty Factor (PF) is integrated into the acquisition cost calculation. The PF addresses the potential imbalance in task assignments, which can arise when certain forklifts, particularly those already in the route network, continuously receive new tasks due to their favourable starting positions. Forklifts stationed at the periphery, often in parking areas, typically have longer travel distances to task locations and therefore receive fewer assignments. Without adjustments, this leads to an imbalance where some forklifts carry a disproportionately high workload while others remain underutilized.

The PF functions as a corrective measure, increasing acquisition costs for forklifts with higher workloads or longer tours, thus encouraging a more even distribution of tasks across the fleet. By penalizing acquisition costs according to the quantity and duration of overdue transport orders, the PF helps ensure that each forklift remains appropriately utilized, regardless of initial starting position.

PF applies specifically to storage, restorage, and empty transport orders by comparing the expected completion time of each fulfilled order to its requirement time. For these types of orders, the requirement time is synonymous with the time of order creation (trigger time), placing it in the past, and a threshold value is assigned based on the order type. No additional costs are accrued until the task surpasses this threshold; only then do penalties accumulate according to the time overrun.

To calculate the PF, acquisition costs for a transport order are multiplied by a factor that correlates with both the quantity and duration of time overruns for orders, such as storage, restorage, and empty transport orders. For these order types, penalties are incurred once a predefined threshold is surpassed.

On the other hand, retrieval orders differ in that they have a specified requirement time set one hour after the order's trigger time, meaning no initial time overrun exists. Instead, the PF for retrieval orders is based on the remaining time until the retrieval deadline. As this time shortens, the effective travel distance is scaled accordingly, raising the task's priority and ensuring that time-sensitive retrieval orders are completed on schedule.

When a tour extends for a long duration, more time overruns are likely, leading to a higher penalty factor and thus higher acquisition costs. This incentivizes forklifts with fewer orders or shorter tours to present a more favourable acquisition cost, even if their route is less optimal for the new order. The PF formula is provided in the following equation

$$PF = 1 + \sum_{i \in Tour} k \cdot t_{Diff_i}$$

Here, k is a constant that reflects the priority level of each transport order type, and t_{Diff_i} represents the time overrun for each order within the tour.

Only positive time differences contribute to the PF, meaning orders fulfilled within their requirement time do not increase the PF. Thresholds vary according to order type, with penalties only applied once the time overrun exceeds the specified threshold.

The threshold values for different transport order types in this study are as follows:

- **Restorage and Empty Orders**: Penalties apply only if completion exceeds 10 minutes beyond the requirement time.
- **Storage Transport Orders**: A threshold of up to 60 minutes beyond the requirement time.
- **Retrieval Transport Orders**: A negative threshold of -15 minutes, with penalties applying if completion is later than 15 minutes before the requirement time.

The PF approach explains why longer tours incur higher acquisition costs. Longer tours inherently involve more transport orders, which increases the likelihood that individual time differences t_{Diff_i} exceed zero. As tour duration grows, so do the time overruns for each order, amplifying the total acquisition cost.

In selecting factor k, the priority of each transport order type is considered. In this study, k is set at 0.1 for immediate requirement orders (such as storage and restorage) and 0.5 for retrieval orders, reflecting their higher urgency.

The SimTalk code for implementing the acquisition cost calculations for Cases 1 to 4 of the Auction-Based Routing Algorithm, which forms the central component of this thesis and represents the most intricate and crucial element of the algorithm, can be found in Appendix A – Acquisition Costs SimTalk Code for the Auction-Based Routing Algorithm. This calculation plays a pivotal role in optimizing the routing process, with the detailed implementation available in the appendix

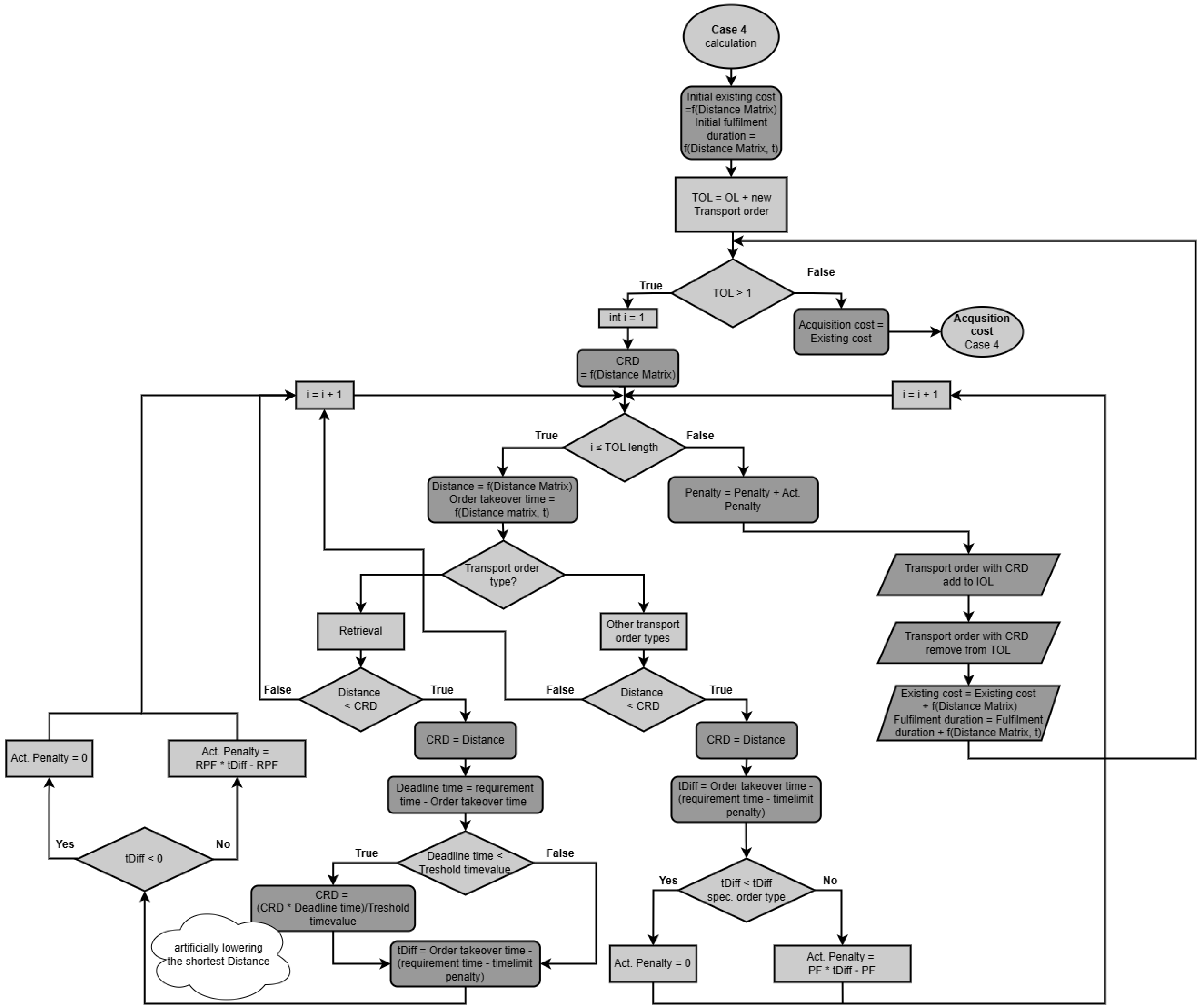

Case 4 calculation
Initial existing cost =f(Distance Matrix) Initial fulfilment duration = f(Distance Matrix, t)
TOL = OL + new Transport order
TOL > 1
True
False
int i = 1
Acquisition cost = Existing cost
Acqusition cost Case 4
CRD = f(Distance Matrix)
i = i + 1
i = i + 1
i ≤ TOL length
True
False
Distance = f(Distance Matrix) Order takeover time = f(Distance matrix, t)
Penalty = Penalty + Act. Penalty
Transport order type?
Transport order with CRD add to IOL
Transport order with CRD remove from TOL
Existing cost = Existing cost + f(Distance Matrix) Fulfilment duration = Fulfilment duration + f(Distance Matrix, t)
Retrieval
Other transport order types
False
Distance < CRD
True
False
Distance < CRD
True
CRD = Distance
CRD = Distance
Deadline time = requirement time - Order takeover time
tDiff = Order takeover time - (requirement time - timelimit penalty)
Act. Penalty = 0
Act. Penalty = RPF * tDiff - RPF
Yes
tDiff < 0
No
True
Deadline time < Treshold timevalue
False
CRD = (CRD * Deadline time)/Treshold timevalue
artificially lowering the shortest Distance
tDiff = Order takeover time - (requirement time - timelimit penalty)
Yes
tDiff < tDiff spec. order type
No
Act. Penalty = 0
Act. Penalty = PF * tDiff - PF

Figure 3-18: Acquisition Cost Calculation for Case 4 - Three or more Existing Transport Orders.

3.5 Simulation Evaluation

The simulation evaluation focuses on analysing the performance of the algorithms used in the FGS through specific KPIs, which serve as essential metrics to measure system efficiency, effectiveness, and optimization. In the simulation study, various model parameters were systematically defined and adjusted to evaluate how well each algorithm manages forklift operations under different conditions. For each algorithm, simulations were conducted with varying numbers of forklifts, starting with 16 and progressively reducing the count by one until the final run with 10 forklifts. This approach provides insight into each algorithm's performance in different resource scenarios, highlighting their applicability in real-world warehouse operations.

Each variant of the simulation involved modifying key model parameters while keeping others constant, allowing for consistent comparisons across scenarios. An Excel-based tool was used to automate the processing of simulation output and KPI calculations, offering a streamlined method of evaluating various operational scenarios. For each simulation run, KPIs were recorded and analysed to assess the effectiveness of the algorithms in optimizing forklift operations. The KPIs used to evaluate the simulation results include:

On-Time Retrieval Orders

This KPI measures the percentage of retrieval orders completed before their required time, known as the requirement time. Given that retrieval orders are critical tasks in the warehouse, their timely completion is essential for maintaining operational efficiency. The effectiveness of each algorithm is partly evaluated based on its ability to prioritize and manage these time-sensitive orders.

Empty Run Ratio

The Empty Run Ratio refers to the percentage of time forklifts spend moving without a load, which is a primary source of inefficiency in warehouse operations. Empty runs consume time and energy without directly contributing to productivity, leading to increased operational costs and reduced resource utilization. Reducing these unnecessary movements is crucial for optimizing forklift operations, as it improves resource utilization, reduces fuel consumption, and enhances overall efficiency. In this study, the empty run percentage is based on time rather than distance, providing insight into how effectively forklifts are utilized during their active hours. Lower percentages in this KPI indicate greater operational efficiency and productivity improvements [28].

Distance Travelled

This metric tracks the cumulative distance covered by the forklift fleet during task execution in the simulation. Minimizing unnecessary travel is essential for reducing equipment wear and tear, fuel consumption, and energy usage, which collectively enhance productivity and operational savings. Algorithms that optimize routing and task assignment can significantly lower the total distance travelled, indicating more efficient task allocation and routing. The Distance Travelled KPI is calculated by summing the total distance covered by forklifts over the simulation period, with lower values reflecting more optimized operations. Mathematically, this is equivalent to the empty run ratio, as only the empty runs can be optimized within the system.

The simulation evaluation, conducted through the KPIs mentioned above, provides valuable insights into system efficiency and optimization, highlighting the strengths and weaknesses of each algorithm. By analysing these KPIs across different simulation runs and for each algorithm, the study identifies which approach is most effective at reducing inefficiencies and improving overall warehouse operations. This analysis is crucial for determining the algorithm that best balances on-time order fulfilment, minimal empty runs, and optimized travel distances, even as the number of forklifts is reduced. The ultimate goal is to find the algorithm that ensures robust performance under varying operational demands.

3.6 Real-world Use Case

In this section, we apply the best-performing algorithm from the simulation study to a real-world warehouse scenario. The primary objective is to assess how effectively the chosen algorithm can optimize forklift task allocation and reduce inefficiencies within a real-life operation. By utilizing real-world data, the study aims to validate the results obtained in the simulated environment and determine whether the FGS can significantly enhance logistical processes and improve overall warehouse performance.

The simulation model is applied to a large-scale, internationally recognized company with a complex logistical operation. While specific details about the company are withheld, key information about the warehouse layout, operational workflow, and performance metrics are provided to demonstrate the algorithm's practical application and potential impact.

3.6.1 Real-World Warehouse System Modelling

The real-world warehouse system is modelled using a similar approach to the simulation study but with greater detail and accuracy to reflect the actual conditions and constraints of the warehouse environment. The model is created following the same procedure, using real data to generate the layout design, road network, and distance matrix. The model replicates the warehouse structure, including workstations, storage areas, and transport routes, while accounting for operational specifics.

In contrast to the simulation study, which operated under idealized conditions, the real-world model incorporates additional complexities such as workers' scheduled breaks, introducing a layer of variability that reflects human factors and their impact on overall efficiency. The actual transport orders and shift patterns in this model are derived from real operational data, enhancing the realism and accuracy of the simulation. This allows for a more detailed evaluation of the FGS and the selected algorithm's performance in optimizing task allocation, reducing empty runs, and improving overall operational efficiency.

The warehouse infrastructure includes approximately 200 storage locations, with dedicated areas for goods receipt and goods issue, ensuring a systematic flow of goods within the facility. Unlike the simplified paths used in the simulation study, the real-world warehouse layout features more detailed considerations, such as path intersections and other complexities.

The forklift fleet is composed of two distinct types of forklifts with varying capabilities. The differences between Type 1 and Type 2 forklifts have been explained in Section 3.4.3. This heterogeneous fleet configuration is designed to meet the diverse demands of warehouse operations.

During a three-week period (Monday to Saturday), the warehouse processes a significant volume of transport orders, totalling 178,610 orders. The warehouse operates on a two-shift system, with one morning shift and one afternoon shift, both incorporating scheduled breaks for the workforce. In addition to the transport orders considered in the simulation study (retrieval, storage, and restorage), the real-world use case also includes the handling of empty pallets. Empty pallet handling involves material flow in the opposite direction of goods issue and is crucial for managing pallet storage. Table 3-5 provides an overview of the transport order types processed in the real-world use case.

Table 3-5: Real-world use case transport order types overview.

Transport order type	#	%
Retrieval	112014	62,71
Storage	13026	7,29
Restorage	17001	9,52
Empty pallet handling	36569	20,47
Overall	178610	100,00

Similarly, to the benchmark study, an analysis was conducted to assess the potential for reducing the number of forklifts while still ensuring the timely completion of retrieval orders. This study also aimed to identify the point at which a further reduction in the number of forklifts would result in system collapse due to operational overload.

3.6.2 Localisation system analysis

In addition to validation, the study also examines the impact of a localization system on the efficiency of the FGS. For the real-world use case study, two localization systems will be considered: a coarse localization system, as well as a fine localization system.

The coarse localization system determines the forklift's position based on the last completed transport order's destination, referred to as the "last sink." Even if the forklift is en route to a new task or parking area, the system assumes the forklift remains at this last known position. While this method is straightforward and simpler to implement, it may introduce inaccuracies in acquisition cost calculations, as it does not account for the forklift’s actual location in transit.

The fine localization system, on the other hand, updates the forklift’s position at every node within the warehouse's road network, including intersections. This more detailed system provides frequent updates on the forklift’s actual location, enabling more accurate task allocation and cost calculation. By continuously tracking the forklift’s movement in near real-time, the system can optimize routes more effectively, minimizing unnecessary travel and enhancing overall operational efficiency.

In the benchmark simulation study, the fine localization system was used, where the forklift’s position was frequently updated as it moved through the warehouse's road network. This system provided sufficient accuracy for the benchmark simulation environment. However, in the real-world use case, both coarse and fine localization methods will be analysed to evaluate their impact on acquisition cost calculations and overall system performance.

4 Results

This chapter presents the outcomes of both the simulation study and the real-world use case. It begins by describing the simulation model and its evaluation process, followed by a detailed analysis of the performance of different algorithms tested in the study. The chapter also includes an overall comparison of the algorithms to determine the best performer. Finally, the results of the real-world use case are discussed, including the evaluation of the localization systems and the application of the selected algorithm in a real warehouse environment.

4.1 Simulation Model

The simulation model developed for this study provides an automated framework for evaluating various FGS algorithms. It is designed to simulate warehouse operations based on key input data, such as warehouse layout, transport orders, and operational parameters such as shifts and forklift types. Through this system, it is possible to automatically generate the warehouse layout, simulate transport orders, and evaluate the results using an MS Excel-based evaluation tool built with VBA logic.

The model allows for the automatic creation of the warehouse layout, which in this case includes racks (serving as workstations), parking lots for forklifts, and areas for goods receipt and issue. Based on the input data described in Chapter 3, the model generates the layout automatically. Figure 4-1 illustrates the 2D layout generated for the simulation study, based on the input data described in Chapter 3. Although there is an option to model the layout in 3D, this study focuses on 2D representation for simplicity and efficiency. A more detailed view of the layout itself, as well as the objects and methods used for creating this model, can be found in Appendix B – Detailed Layout Overview.

Different modules in the simulation include a shift calendar, event manager, forklift pool, and methods for algorithm implementation, as well as tables used for input and output data processing. After the simulation run is completed, the output data is exported in table form and processed using an Excel-based evaluation tool. This tool automatically generates detailed reports, offering insights into the performance of different FGS algorithms.

The evaluation tool simplifies the reporting process by generating KPIs and visual reports through a series of automated steps, ensuring that the results are ready for analysis with minimal manual effort. Appendix C – MS Excel Evaluation Toolprovides a closer look at the interface of the MS Excel tool and includes sample reports generated from the simulation.

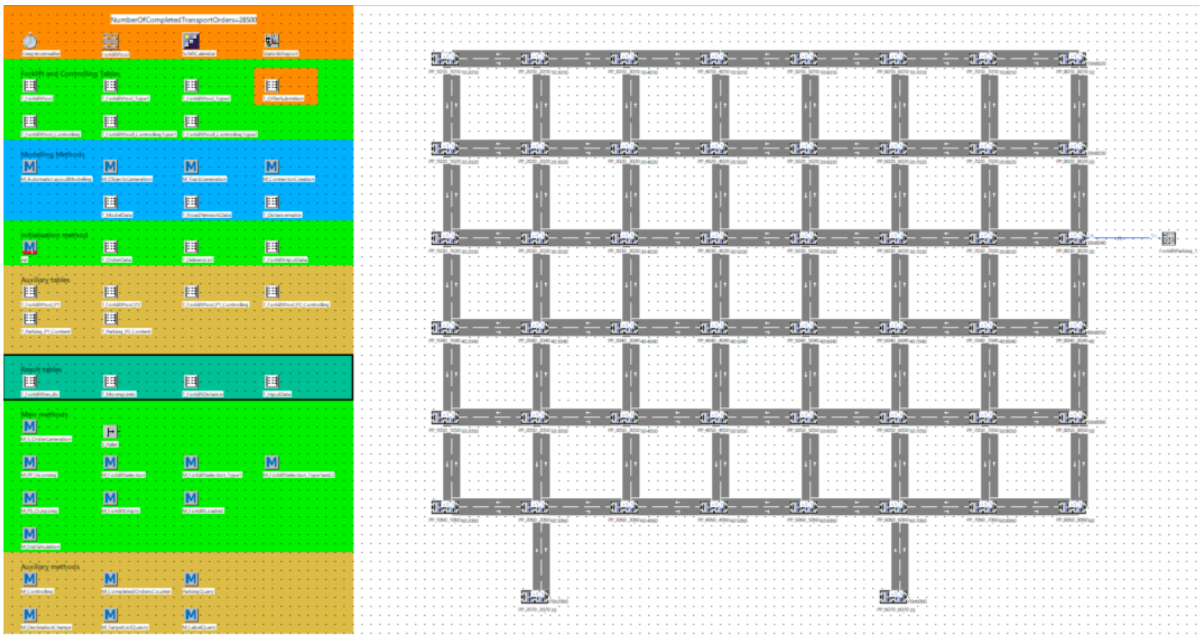


Figure 4-1: Simulation Model Layout – Overview showing the complete layout of the simulation model. The right side displays the generated warehouse layout, with the organized grid structure and pathways. On the left side are the objects used in constructing the model, categorized by function and color-coded for clarity, representing all essential components necessary to build and operate the simulation.

The purpose of this simulation model is to evaluate various FGS algorithms and identify the best-performing one in terms of efficiency and optimization of forklift operations. Once the most suitable algorithm is determined, it will be applied to real-world data to test its effectiveness in practical scenarios. Another key objective is to examine the potential of the FGS to reduce the number of forklifts required in the warehouse, aiming to cut costs associated with labour, maintenance, and operational inefficiencies.

4.2 Analysis and Evaluation of the Simulation Study

This chapter presents the results of the simulation study for each algorithm. As outlined in Section 3.5, KPIs were recorded and analysed for each simulation run to assess the effectiveness of each algorithm in optimizing warehouse operations. This analysis provides a comparative view of each algorithm's performance under varying conditions.

4.2.1 A1 - Sequential Forklift Assignment Algorithm Evaluation

The A1 Sequential Forklift Assignment Algorithm demonstrates strong performance in terms of on-time retrievals when 14 or more forklifts are available, maintaining 100% on-time retrievals. As the name suggests, the on-time retrieval metric specifically considers only retrieval orders with a defined requirement time, as opposed to restorage and storage orders, which have a lower priority and no strict requirement times. However, as the number of forklifts drops to 13, the performance shows a slight decline, with retrievals dropping just below perfect levels. At 12 forklifts, there is a sharp drop to 66.34%, and the algorithm struggles significantly with 11 forklifts, achieving only 6.4% of retrievals on time. This pattern, as shown in Table 4-1 and Figure 4-2, makes it evident that the algorithm cannot maintain acceptable performance with fewer than 13 forklifts (see Table 4-1 and Figure 4-2).

In terms of empty runs, as depicted in Table 4-1, the percentage fluctuates between 50% and 60% as the number of forklifts is reduced, indicating some minor improvement in resource utilization. However, these fluctuations are relatively modest and do not offset the substantial decline in retrieval performance as the fleet size decreases.

As shown in Figure 4-2, although the total distance travelled decreases as the number of forklifts is reduced, the average distance per forklift increases significantly, highlighting the extra operational burden placed on each forklift. This underscores the challenges of maintaining efficiency when the fleet size is constrained.

Overall, the A1 algorithm is effective when there are 14 or more forklifts, but its performance degrades rapidly as the fleet size is reduced. While the algorithm manages to slightly reduce empty runs, it cannot maintain high on-time retrieval rates with fewer forklifts, which limits its applicability in resource-constrained environments.

Table 4-1: A1 Sequential Forklift Assignment Algorithm Results

A1 - Sequential Forklift Assignment Algorithm									
Number of forklifts	On time Retrieval orders		Late Retrievals	Empty runs	Distance travelled		Completed orders	Completed Retrievals	Missing Retrievals
	[%]	[-]	[-]	[%]	total [km]	avg. [km]			
16 Forklifts	100,00%	19693	0	59,61%	4004,6	250,3	28500	19693	0
15 Forklifts	100,00%	19693	0	58,29%	3879,5	258,6	28500	19693	0
14 Forklifts	100,00%	19693	0	56,46%	3718,2	265,6	28500	19693	0
13 Forklifts	99,74%	19641	52	53,98%	3518,9	270,7	28500	19693	0
12 Forklifts	66,34%	13065	6628	50,55%	3275,9	273,0	28500	19693	0
11 Forklifts	6,40%	1188	17385	49,62%	3069,8	279,1	27173	18573	1120
10 Forklifts	4,21%	713	16215	49,59%	2790,9	279,1	24723	16928	2765

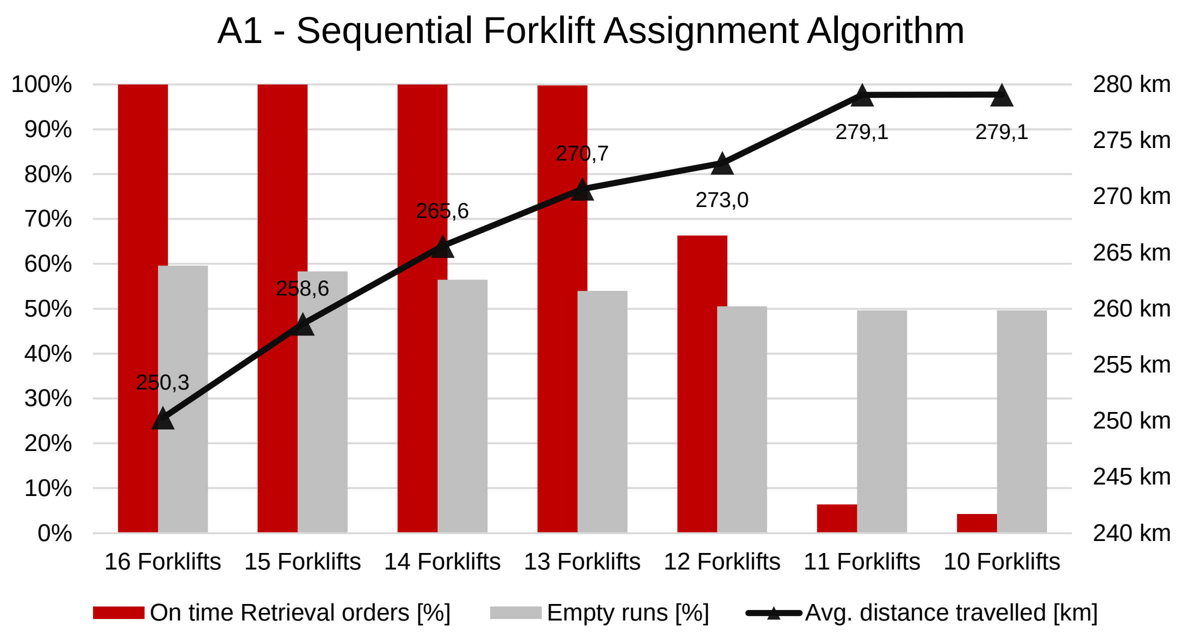


Figure 4-2: A1 Sequential Forklift Assignment Algorithm maintains 100% on-time retrievals with 14 or more forklifts but shows a rapid performance decline as the fleet size decreases. The algorithm achieves modest improvements in empty run reduction.

4.2.2 A2a - Priority-Based Algorithm Evaluation - Case 1

In this scenario, the A2a Priority-Based Algorithm emphasizes minimizing the routing distance as the primary factor in assigning transport orders. While it places the highest priority on reducing travel distance (weighted at 100%), the orders backlog is given no explicit weighting in this case (0%). This prioritization focuses on optimizing forklift routes to minimize travel time, potentially at the expense of addressing delayed or pending orders.

Compared to the baseline A1 Sequential Forklift Assignment Algorithm, A2a delivers better on-time retrieval performance even though it ignores deadlines and only considers routing distance. As shown in Table 4-2 and Figure 4-3, the algorithm performs effectively with on-time retrievals remaining at 100% when there are 14 or more forklifts available. However, as the number of forklifts decreases, the algorithm's performance begins to deteriorate noticeably. At 12 forklifts, on-time retrievals drop to 90.65%, and with only 11 forklifts, the performance falls dramatically, with just 16.39% of retrievals completed on time. At 10 forklifts, the on-time rate is critically low at 4.64%.

The empty run percentage improves slightly compared to A1, averaging between 46% and 50%, suggesting more efficient use of forklifts in terms of load distribution. However, this marginal improvement does not compensate for the significant drop in on-time retrievals, particularly when the fleet size is reduced.

The average distance travelled per forklift increases as the number of forklifts decreases, reflecting the added burden on each forklift as the fleet size shrinks. While this algorithm is more effective at minimizing travel distance than A1, the rising operational burden highlights the limitations of relying solely on routing distance for task allocation without considering the backlog of pending orders.

In summary, the A2a algorithm provides a slight improvement in empty runs and distance efficiency compared to A1, but its performance still falters significantly when fewer than 12 forklifts are in operation, primarily due to its inability to maintain high on-time retrieval rates in resource-constrained scenarios.

Table 4-2: A2a – Priority-Based Algorithm – Case 1 Results

A2a - Priority-Based Algorithm (**Case 1** → 1.0 Routing distance, 0.0 Orders backlog)									
Number of forklifts	On time Retrieval orders		Late Retrievals	Empty runs	Distance travelled		Completed orders	Completed Retrievals	Missing Retrievals
	[%]	[-]	[-]	[%]	total [km]	avg. [km]			
16 Forklifts	100,00%	19693	0	49,61%	3215,1	200,9	28500	19693	0
15 Forklifts	99,96%	19685	8	49,26%	3193,2	212,9	28500	19693	0
14 Forklifts	100,00%	19693	0	48,69%	3157,2	225,5	28500	19693	0
13 Forklifts	100,00%	19693	0	48,06%	3119,1	239,9	28500	19693	0
12 Forklifts	90,65%	17852	1841	46,96%	3054,2	254,5	28500	19693	0
11 Forklifts	16,39%	3161	16122	46,25%	2964,4	269,5	28002	19283	410
10 Forklifts	4,64%	806	16559	46,29%	2699,0	269,9	25456	17365	2328

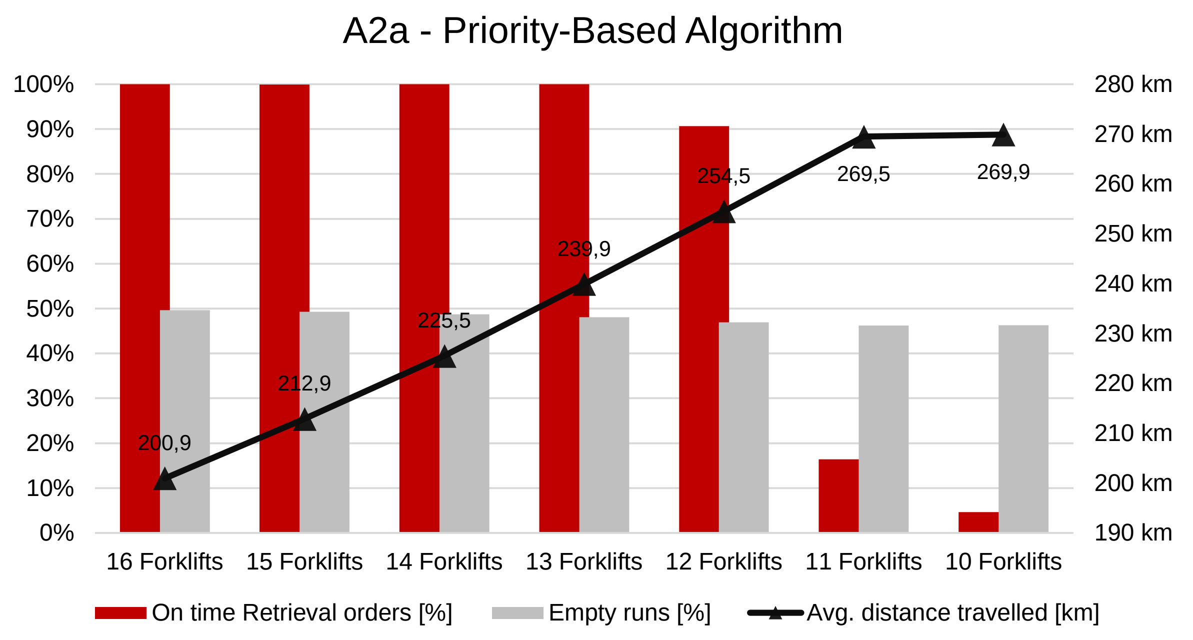


Figure 4-3: A2a – Priority-Based Algorithm – Case 1 maintains 100% on-time retrievals with 14 or more forklifts, outperforming the A1 algorithm in retrieval efficiency even without prioritizing deadlines. The algorithm shows slight improvement in empty run reduction, with percentages between 46% and 50%.

4.2.3 A2b - Priority-Based Algorithm Evaluation - Case 2

In this case, the A2b Priority-Based Algorithm assigns transport orders based on a balance of 80% routing distance and 20% orders backlog. Compared to A2a, which prioritized routing distance solely, this adjustment reflects an effort to strike a balance between minimizing travel distance and managing the pending orders backlog.

As shown in Table 4-3 and **Fehler! Verweisquelle konnte nicht gefunden werden.**Figure 4-4, A2b exhibits similar performance to A2a when there are 14 or more forklifts, consistently achieving 100% on-time retrievals. However, with fewer forklifts, A2b demonstrates a slightly better performance than A2a. For example, at 12 forklifts, the on-time retrievals remain 90.94%, slightly higher than A2a's 90.65%, and at 11 forklifts, the rate is 16.41%, again slightly better than A2a. Nevertheless, the performance drops significantly at 10 forklifts, where only 4.62% of retrievals are completed on time—still low but comparable to A2a.

The empty runs show slight improvements compared to A2a, staying within the 46-49% range across different fleet sizes. This more consistent performance can be attributed to the orders backlog consideration, which distributes tasks more evenly across the fleet.

In terms of average distance travelled, the results follow a similar trend as A2a, with the distance per forklift increasing as the number of forklifts decreases. However, the orders backlog prioritization helps control the travel distances, making A2b marginally more efficient when fewer forklifts are in operation.

Overall, A2b provides a slightly more balanced performance compared to A2a due to its inclusion of the orders backlog weighting. It offers minor improvements in on-time retrievals and empty runs, especially when the number of forklifts is reduced. However, as its predecessor, it still faces significant challenges in maintaining acceptable performance under conditions of limited resources.

Table 4-3: A2b – Priority-Based Algorithm – Case 2 Results

A2b - Priority-Based Algorithm (Case 2 → 0.8 Routing distance, 0.2 Orders backlog)									
Number of forklifts	On time Retrieval orders		Late Retrievals	Empty runs	Distance travelled		Completed orders	Completed Retrievals	Missing Retrievals
	[%]	[-]	[-]	[%]	total [km]	avg. [km]			
16 Forklifts	100,00%	19693	0	49,64%	3217,1	201,1	28500	19693	0
15 Forklifts	100,00%	19693	0	49,28%	3194,3	213,0	28500	19693	0
14 Forklifts	100,00%	19693	0	48,73%	3159,8	225,7	28500	19693	0
13 Forklifts	100,00%	19693	0	48,03%	3117,2	239,8	28500	19693	0
12 Forklifts	90,94%	17909	1784	46,97%	3054,9	254,6	28500	19693	0
11 Forklifts	16,41%	3164	16116	46,24%	2964,3	269,5	27998	19280	413
10 Forklifts	4,62%	802	16564	46,23%	2699,2	269,9	25466	17366	2327

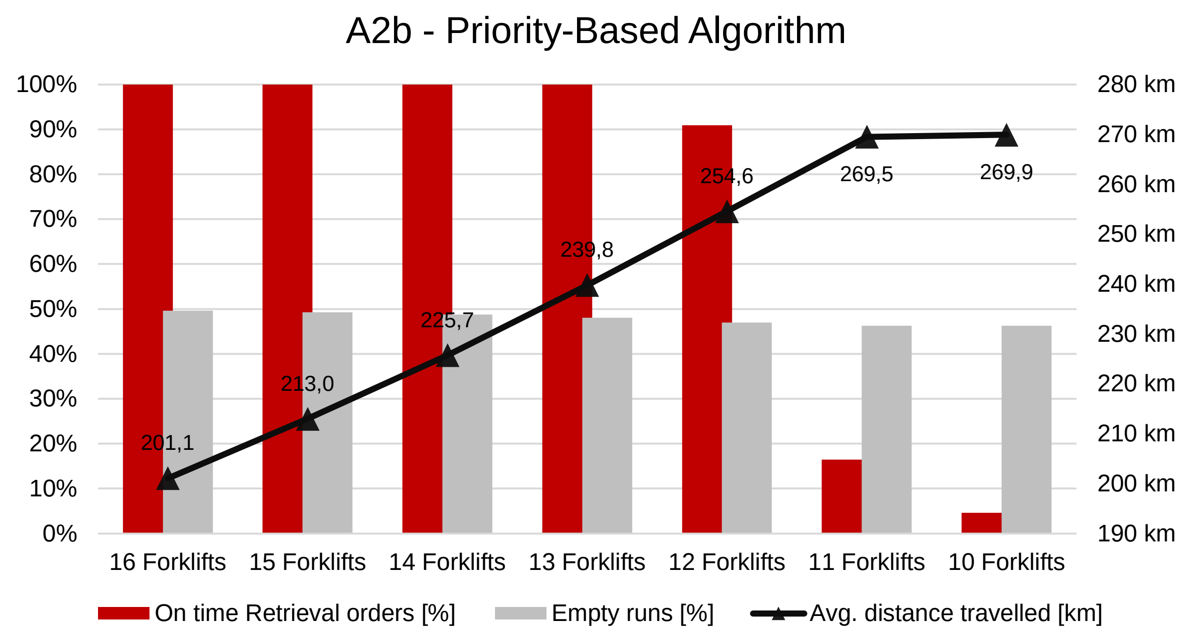


Figure 4-4: A2b – Priority-Based Algorithm – Case 2 achieves 100% on-time retrievals with 14 or more forklifts and shows slightly improved performance over A2a in scenarios with fewer forklifts, due to a balance between routing distance and orders backlog. The algorithm demonstrates modest gains in empty run reduction, and maintains similar average travel distances per forklift.

4.2.4 A2c - Priority-Based Algorithm Evaluation - Case 3

The A2c Priority-Based Algorithm shifts the weightings further toward the orders backlog, with a configuration of 60% routing distance and 40% orders backlog. This increased consideration of the backlog aims to address the delays caused by prioritizing only routing distance, making the algorithm more responsive to pending tasks.

As shown in Table 4-4 and Figure 4-5, A2c maintains 100% on-time retrieval rates with 16, 15, and 14 forklifts, similar to the performance of the earlier algorithms. However, the more balanced weighting starts to show improvements in on-time retrievals as the number of forklifts is reduced. For example, with 12 forklifts, A2c achieves 91.47% on-time retrievals, which is higher than A2a and A2b at the same fleet size. With 11 forklifts, the on-time retrieval rate is 16.25%, which is actually slightly lower than both A2a and A2b. This indicates that while the algorithm attempts to manage the orders backlog, it does not necessarily result in better performance with fewer resources. At 10 forklifts, the performance still drops significantly to 4.63%, similar to the previous algorithms, indicating that even with a 40% backlog focus, the system struggles with reduced resources.

The empty run percentage slightly increases for A2c compared to A2b, reflecting the impact of prioritizing orders backlog more heavily. This leads to a minor sacrifice in distance efficiency to better manage task distribution across the fleet.

Regarding distance travelled, A2c's emphasis on orders backlog yields a moderate increase in the total distance travelled, as forklifts are sometimes routed to complete tasks based on backlog prioritization rather than optimal travel paths.

Overall, A2c demonstrates similar performance to A2a and A2b, with no significant improvements when the fleet size is reduced below 12 forklifts. While it continues to encounter challenges in resource-constrained scenarios, the increased emphasis on backlog provides marginal improvements in on-time retrieval rates and task distribution over earlier cases. However, this shift also leads to slightly increased empty runs and travel distances, underscoring the trade-offs involved in balancing routing distance with backlog considerations.

Table 4-4: A2c – Priority-Based Algorithm – Case 3 Results

A2c - Priority-Based Algorithm (**Case 3** → 0.6 Routing distance, 0.4 Orders backlog)									
Number of forklifts	On time Retrieval orders		Late Retrievals	Empty runs	Distance travelled		Completed orders	Completed Retrievals	Missing Retrievals
	[%]	[-]	[-]	[%]	total [km]	avg. [km]			
16 Forklifts	100,00%	19693	0	50,06%	3244,1	202,8	28500	19693	0
15 Forklifts	100,00%	19693	0	49,64%	3216,4	214,4	28500	19693	0
14 Forklifts	100,00%	19693	0	48,93%	3172,5	226,6	28500	19693	0
13 Forklifts	100,00%	19693	0	48,10%	3121,5	240,1	28500	19693	0
12 Forklifts	91,47%	18014	1679	47,02%	3057,8	254,8	28500	19693	0
11 Forklifts	16,25%	3134	16149	46,25%	2964,2	269,5	28000	19283	410
10 Forklifts	4,63%	804	16560	46,28%	2698,9	269,9	25463	17364	2329

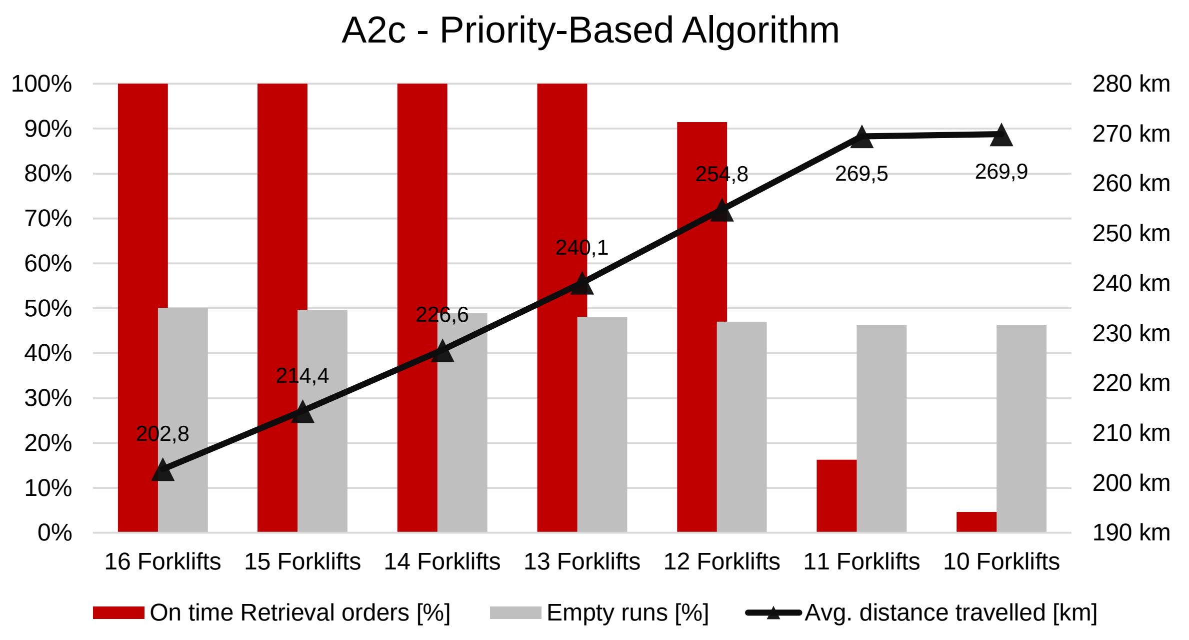


Figure 4-5: A2c – Priority-Based Algorithm – Case 3 balances routing distance and orders backlog with a 60/40 weighting. This configuration maintains 100% on-time retrievals with 14 or more forklifts and achieves slightly better on-time retrieval rates than A2a and A2b at 12 forklifts. However, the increased focus on backlog slightly raises empty run percentages and total travel distance, illustrating the trade-offs in prioritizing task backlog over minimal travel paths.

4.2.5 A2d - Priority-Based Algorithm Evaluation - Case 4

The A2d Priority-Based Algorithm shifts the weight further towards the orders backlog, with 40% routing distance and 60% orders backlog. This increased focus aims to improve the handling of pending orders, while still considering travel distance as a secondary factor.

As shown in Table 4-5 and Figure 4-6, A2d maintains strong performance with 100% on-time retrievals when 14 or more forklifts are available. However, as the number of forklifts is reduced, the on-time retrieval rate begins to decline, dropping to 91.06% at 12 forklifts. A more severe decline occurs with 11 forklifts, where only 15.91% of retrievals are completed on time. When the number of forklifts reaches 10, the on-time retrieval rate falls further to 4.59%, indicating that the algorithm struggles significantly under resource constraints.

In terms of empty runs, the percentage fluctuates between 46% and 50%, showing a modest improvement in resource utilization as the number of forklifts decreases. The reduction is gradual, suggesting that the algorithm manages to keep empty travel relatively stable, even under tighter conditions.

While the average distance per forklift increases as the fleet size is reduced, this trend is consistent with other priority-based algorithms. The reduced number of forklifts must cover the same workload, resulting in longer trips for each vehicle.

Overall, A2d demonstrates better performance in balancing the backlog and routing distance compared to previous cases, but its on-time retrieval performance drops sharply with fewer than 12 forklifts. The empty runs remain more stable, but the overall gains are modest. This suggests that while the algorithm is somewhat better at handling resource constraints, it still faces significant challenges with lower fleet sizes.

Table 4-5: A2d – Priority-Based Algorithm – Case 4 Results

A2d - Priority-Based Algorithm (**Case 4** → 0.4 Routing distance, 0.6 Orders backlog)									
Number of forklifts	On time Retrieval orders		Late Retrievals	Empty runs	Distance travelled		Completed orders	Completed Retrievals	Missing Retrievals
	[%]	[-]	[-]	[%]	total [km]	avg. [km]			
16 Forklifts	100,00%	19693	0	50,81%	3292,0	205,7	28500	19693	0
15 Forklifts	100,00%	19693	0	50,20%	3252,3	216,8	28500	19693	0
14 Forklifts	100,00%	19693	0	49,25%	3192,2	228,0	28500	19693	0
13 Forklifts	100,00%	19693	0	48,25%	3130,7	240,8	28500	19693	0
12 Forklifts	91,06%	17933	1760	47,07%	3060,3	255,0	28500	19693	0
11 Forklifts	15,91%	3068	16210	46,27%	2965,0	269,5	27995	19278	415
10 Forklifts	4,59%	797	16565	46,30%	2699,4	269,9	25462	17362	2331

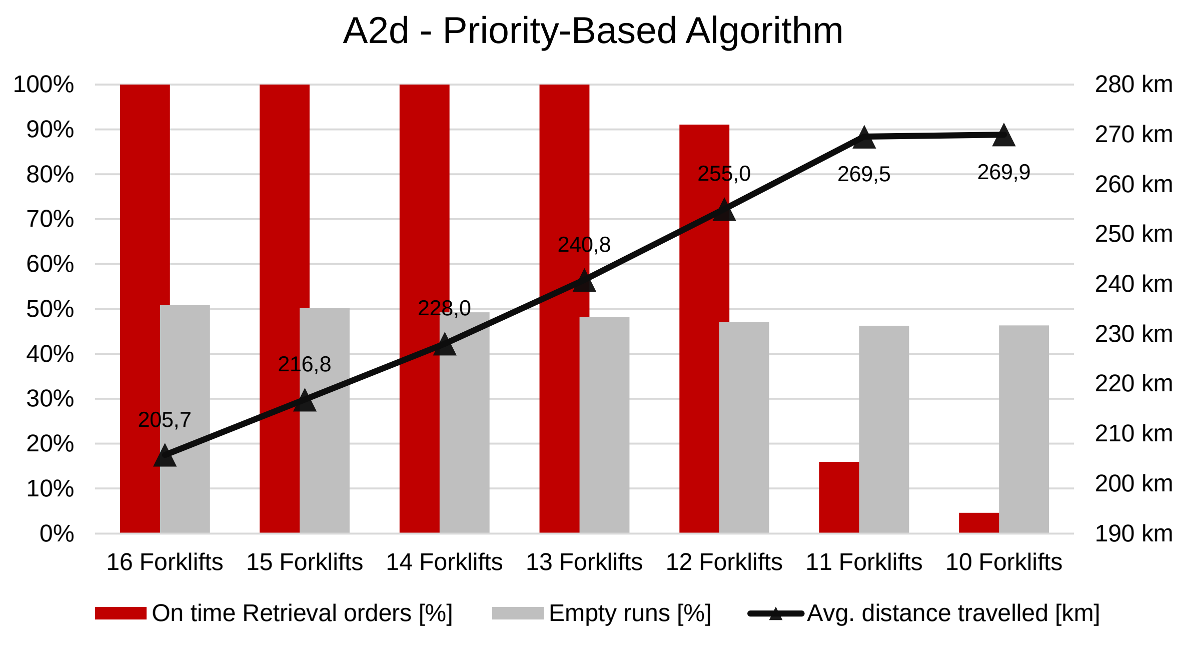


Figure 4-6: A2d – Priority-Based Algorithm – Case 4 with a 40/60 weighting between routing distance and orders backlog, A2d maintains 100% on-time retrievals with 14 or more forklifts, but performance declines sharply as fleet size reduces. The algorithm faces challenges in maintaining high on-time retrieval rates when fewer than 12 forklifts are available.

4.2.6 A2e - Priority-Based Algorithm Evaluation - Case 5

The A2e Priority-Based Algorithm shifts the focus significantly towards the orders backlog, with 20% routing distance and 80% orders backlog. This configuration emphasizes prioritizing tasks based on accumulation rather than minimizing travel distance.

As seen in Table 4-6 and Figure 4-7, the algorithm performs well with 100% on-time retrievals when 14 or more forklifts are available. However, at 12 forklifts, the on-time retrieval rate drops slightly to 88.93%, showing the algorithm's reduced effectiveness as the fleet size decreases. This worsens at 11 forklifts, where the on-time retrieval rate falls sharply to 15.62%. With only 10 forklifts, the performance declines further to 4.56% on-time retrievals.

In terms of empty runs, the values fluctuate between 46% and 51%, showing steady performance in managing empty travel. This indicates that A2e maintains its efficiency in reducing empty runs, even as the fleet size diminishes.

As with other algorithms, the average distance per forklift increases as the number of forklifts decreases. This reflects the additional workload being distributed among fewer forklifts, a trend consistent across all cases.

Compared to A2d and A2c, A2e performs similarly in terms of empty runs but struggles slightly more with on-time retrievals once the fleet size is reduced. This indicates that increasing the weight of the orders backlog beyond a certain point (80%) does not yield significant improvements in performance.

Table 4-6: A2e – Priority-Based Algorithm – Case 5 Results

A2e - Priority-Based Algorithm (Case 5 → 0.2 Routing distance, 0.8 Orders backlog)									
Number of forklifts	On time Retrieval orders		Late Retrievals	Empty runs	Distance travelled		Completed orders	Completed Retrievals	Missing Retrievals
	[%]	[-]	[-]	[%]	total [km]	avg. [km]			
16 Forklifts	100,00%	19693	0	51,01%	3306,2	206,6	28500	19693	0
15 Forklifts	100,00%	19693	0	50,54%	3274,8	218,3	28500	19693	0
14 Forklifts	100,00%	19693	0	49,61%	3215,2	229,7	28500	19693	0
13 Forklifts	100,00%	19693	0	48,51%	3146,1	242,0	28500	19693	0
12 Forklifts	88,93%	17513	2180	47,22%	3069,4	255,8	28500	19693	0
11 Forklifts	15,62%	3011	16264	46,29%	2965,4	269,6	27992	19275	418
10 Forklifts	4,56%	791	16571	46,33%	2700,2	270,0	25454	17362	2331

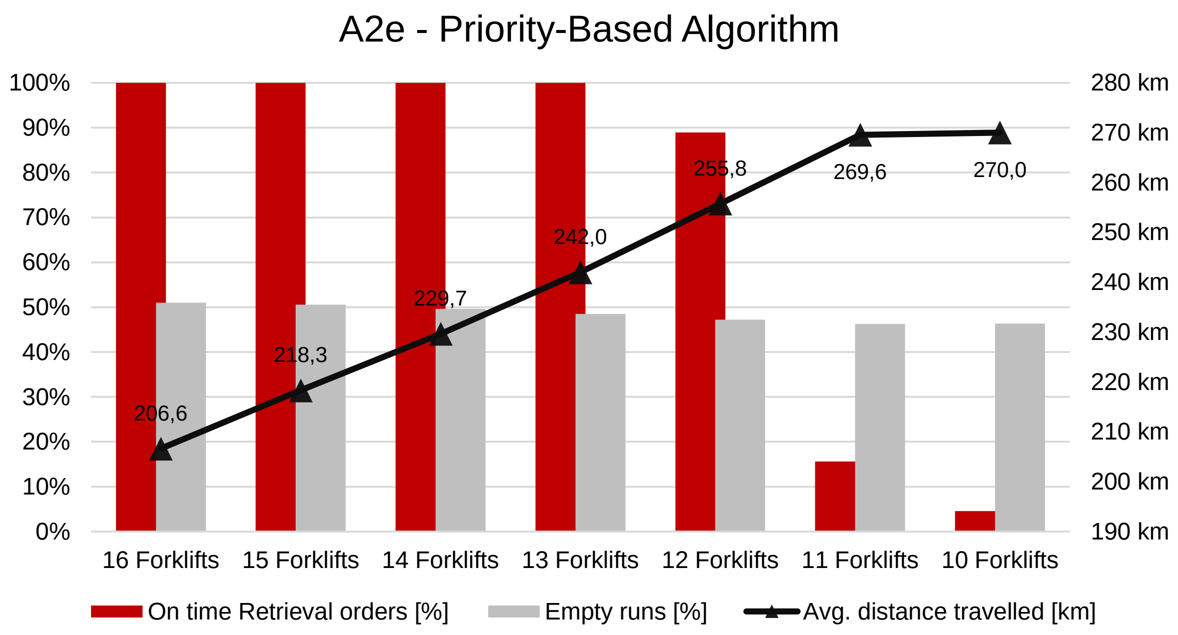


Figure 4-7: A2e – Priority-Based Algorithm – Case 5 with an 80% weighting on orders backlog and 20% on routing distance, A2e maintains 100% on-time retrievals with 14 or more forklifts but sees a notable decline as fleet size decreases. The empty run percentage remains steady between 46% and 51%, but the algorithm struggles with retrieval timeliness when fleet size is reduced, indicating limited benefits from heavily prioritizing the orders backlog.

4.2.7 A2f - Priority-Based Algorithm Evaluation - Case 6

The A2f algorithm fully prioritizes the order backlog (100%) over routing distance (0%), placing maximum focus on minimizing task accumulation at the expense of routing efficiency.

As shown in Table 4-7 and Figure 4-8, the algorithm maintains 100% on-time retrievals with 16, 15, 14, and 13 forklifts. However, the performance significantly deteriorates when the number of forklifts drops below 13. At 12 forklifts, the on-time retrieval rate declines sharply to 80.22%, and when the fleet is reduced further to 11 forklifts, the rate falls drastically to 9.60%. With 10 forklifts, only 4.36% of retrieval orders are completed on time, showing a significant inability to handle reduced resources.

In terms of empty runs, the algorithm begins with 51% at 16 forklifts and fluctuates slightly, ending with 47.54% at 10 forklifts. This suggests that A2f manages empty travel moderately well, but the improvements in this area are not sufficient to offset the severe decline in retrieval performance as the fleet size decreases.

Overall, A2f demonstrates that while prioritizing the orders backlog can be effective with larger fleets, it struggles considerably when the number of forklifts drops below 12. The sharp reduction in on-time retrievals underscores the limitations of ignoring routing distance, particularly in scenarios with constrained resources.

Table 4-7: A2f – Priority-Based Algorithm – Case 6 Results

A2f - Priority-Based Algorithm (Case 6 → 0.0 Routing distance, 1.0 Orders backlog)									
Number of forklifts	On time Retrieval orders		Late Retrievals	Empty runs	Distance travelled		Completed orders	Completed Retrievals	Missing Retrievals
	[%]	[-]	[-]	[%]	total [km]	avg. [km]			
16 Forklifts	100,00%	19693	0	51,03%	3306,9	206,7	28500	19693	0
15 Forklifts	100,00%	19693	0	50,58%	3277,7	218,5	28500	19693	0
14 Forklifts	100,00%	19693	0	49,79%	3226,5	230,5	28500	19693	0
13 Forklifts	100,00%	19693	0	49,05%	3179,5	244,6	28500	19693	0
12 Forklifts	80,22%	15798	3895	47,95%	3112,8	259,4	28500	19693	0
11 Forklifts	9,60%	1826	17195	47,48%	3001,2	272,8	27695	19021	672
10 Forklifts	4,36%	750	16459	47,54%	2732,1	273,2	25187	16459	3234

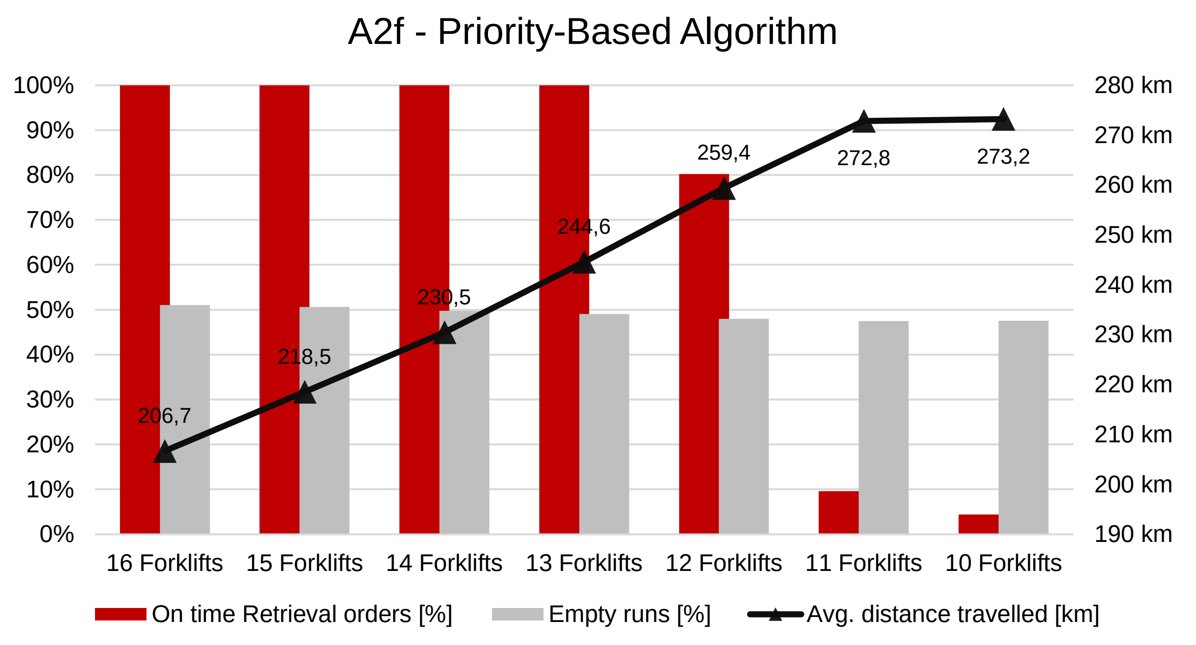


Figure 4-8: A2f – Priority-Based Algorithm – Case 6 with 100% weighting on orders backlog and 0% on routing distance, A2f maintains 100% on-time retrievals with 13 or more forklifts but shows significant declines when fleet size drops below this threshold. Empty run percentages fluctuate slightly, remaining around 47-51%, but the heavy backlog focus limits the algorithm's effectiveness in smaller fleets, underscoring the drawbacks of deprioritizing routing efficiency.

4.2.8 A3 - Auction-Based Routing Algorithm Evaluation

The A3 algorithm, utilizing an auction-based routing method, demonstrates impressive performance across various fleet sizes. As shown in Table 4-8 and Figure 4-9, it achieves 100% on-time retrievals with 16 to 13 forklifts, maintaining optimal task completion rates even as the number of forklifts decreases. When the fleet is reduced to 12 forklifts, the on-time retrieval rate remains strong at 95.99%, showing resilience in managing retrieval orders. However, with 11 forklifts, the on-time retrieval rate drops to 76.71%, and further declines to 52.26% with 10 forklifts, indicating the algorithm faces challenges when resources are more constrained.

In terms of empty runs, the A3 algorithm performs exceptionally well, starting at 48.36% with 16 forklifts and decreasing to 43.00% with 10 forklifts. This steady reduction in empty runs highlights the algorithm's effectiveness in optimizing task assignments, ensuring forklifts spend more time handling tasks and less time traveling without a load.

As the number of forklifts decreases, the total distance travelled naturally declines, reflecting the higher operational efficiency achieved by the A3 algorithm as the fleet size shrinks.

In summary, A3 is the most efficient algorithm tested in this study, striking an effective balance between reducing empty runs and maintaining high on-time retrieval rates. Even with fewer forklifts, it manages to distribute the workload efficiently, making it the top-performing algorithm for optimizing forklift task allocation. A3 demonstrates strong potential for real-world application due to its ability to handle varying operational conditions while maintaining efficiency.

Table 4-8: A3 – Auction-Based Routing Algorithm Results

A3 - Auction-Based Routing Algorithm									
Number of forklifts	On time Retrieval orders		Late Retrievals	Empty runs	Distance travelled		Completed orders	Completed Retrievals	Missing Retrievals
	[%]	[-]	[-]	[%]	total [km]	avg. [km]			
16 Forklifts	100,00%	19693	0	48,36%	3137,0	196,1	28500	19693	0
15 Forklifts	100,00%	19693	0	47,86%	3107,3	207,2	28500	19693	0
14 Forklifts	100,00%	19693	0	47,35%	3077,2	219,8	28500	19693	0
13 Forklifts	100,00%	19693	0	46,31%	3017,6	232,1	28500	19693	0
12 Forklifts	95,99%	18903	790	44,90%	2940,2	245,0	28500	19693	0
11 Forklifts	76,71%	15102	4585	45,73%	2867,8	260,7	28475	19687	6
10 Forklifts	52,26%	9696	8858	43,00%	2592,8	259,3	26343	18554	1139

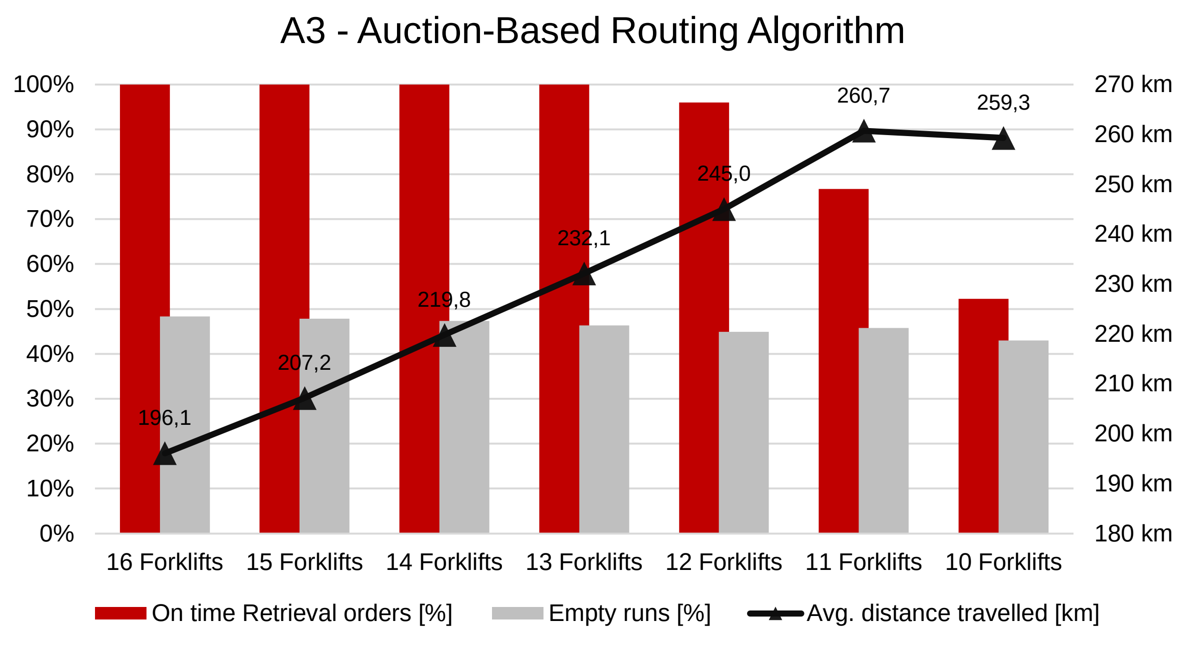


Figure 4-9: A3 – Auction-Based Routing Algorithm Results

4.3 Algorithm Results Comparison

In this section, the performance of all algorithms is compared across key metrics to evaluate their efficiency and effectiveness in optimizing forklift operations. The analysis focuses on three critical KPIs: on-time retrieval orders, empty runs, and total distance travelled. By examining these metrics side by side, it can be assessed which algorithm best balances timely order completion, minimizes unnecessary travel, and reduces overall system inefficiencies. This comparison provides valuable insights into each algorithm's strengths and weaknesses, helping to identify the most suitable approach for real-world implementation in FGS.

4.3.1 On time retrieval orders

The comparison of on-time retrieval rates for all algorithms, as seen in Table 4-9 and Figure 4-10, reveals significant differences in performance as the number of forklifts decreases. Each algorithm behaves differently under reduced fleet sizes, showing varying levels of efficiency in maintaining on-time retrievals.

- **A1 (Sequential Forklift Assignment Algorithm):** The A1 algorithm performs well, with 100% on-time retrievals when 14 or more forklifts are used. However, performance drops off sharply once the number of forklifts falls below 13. The on-time retrieval rate declines to 66.34% at 12 forklifts and plummets to just 4.21% with 10 forklifts. This indicates that A1 is not well-suited for situations where resource availability is constrained, as it struggles to manage the workload with fewer forklifts.
- **A2a to A2f (Priority-Based Algorithms):** The A2 algorithms, which vary based on different weightings between routing distance and order backlog, perform similarly well with 100% on-time retrievals when using 14 or more forklifts. However, their performance begins to degrade as the fleet size decreases. Notably, the A2d (40% routing distance, 60% orders backlog)

and A2e (20% routing distance, 80% orders backlog) cases, which offer a more balanced priority between routing distance and order backlog, maintain relatively higher performance when the fleet size is reduced to 12 forklifts, achieving on-time retrieval rates between 88.93% and 91.06%. Despite this, all A2 algorithms experience significant declines when the fleet size falls below 12 forklifts, with A2f, which prioritizes backlog entirely, performing the worst with only 4.36% on-time retrievals at 10 forklifts.

- **A3 (Auction-Based Routing Algorithm)**: The A3 algorithm consistently outperforms all others, maintaining 100% on-time retrievals even with 13 forklifts. At 12 forklifts, A3 still achieves a high on-time retrieval rate of 95.99%, demonstrating superior resilience compared to other algorithms. When the fleet size is reduced to 11 forklifts, A3 achieves an on-time retrieval rate of 76.71%, far exceeding the performance of any other algorithm. Even at 10 forklifts, A3 manages 52.26% on-time retrievals, which, although lower than ideal, is significantly higher than any other algorithm's performance under similar conditions.

A3’s superior performance across different fleet sizes indicates its robustness in maintaining critical retrieval operations, particularly as resource availability decreases. This algorithm especially demonstrates its strengths when the system is pushed to its limits, such as with 12, 11, or even 10 forklifts. This makes it the most reliable choice for scenarios where operational flexibility is required.

In this study’s context, an on-time retrieval rate of 95% or higher can be considered a highly effective benchmark for warehouse performance. This standard reflects the fact that real-world warehouse operations often face inevitable disruptions, such as delays, waiting times, or maintenance issues, which make consistently perfect retrievals unattainable. A3’s ability to maintain this level of performance with 12 forklifts demonstrates its suitability for optimizing forklift task allocation in challenging environments where resource constraints are common.

Table 4-9: On time Retrieval Orders - Algorithm Comparison

Number of forklifts	A1	A2a	A2b	A2c	A2d	A2e	A2f	A3
16 Forklifts	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%
15 Forklifts	100,00%	99,96%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%
14 Forklifts	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%
13 Forklifts	99,74%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%
12 Forklifts	66,34%	90,65%	90,94%	91,47%	91,06%	88,93%	80,22%	95,99%
11 Forklifts	6,40%	16,39%	16,41%	16,25%	15,91%	15,62%	9,60%	76,71%
10 Forklifts	4,21%	4,64%	4,62%	4,63%	4,59%	4,56%	4,36%	52,26%

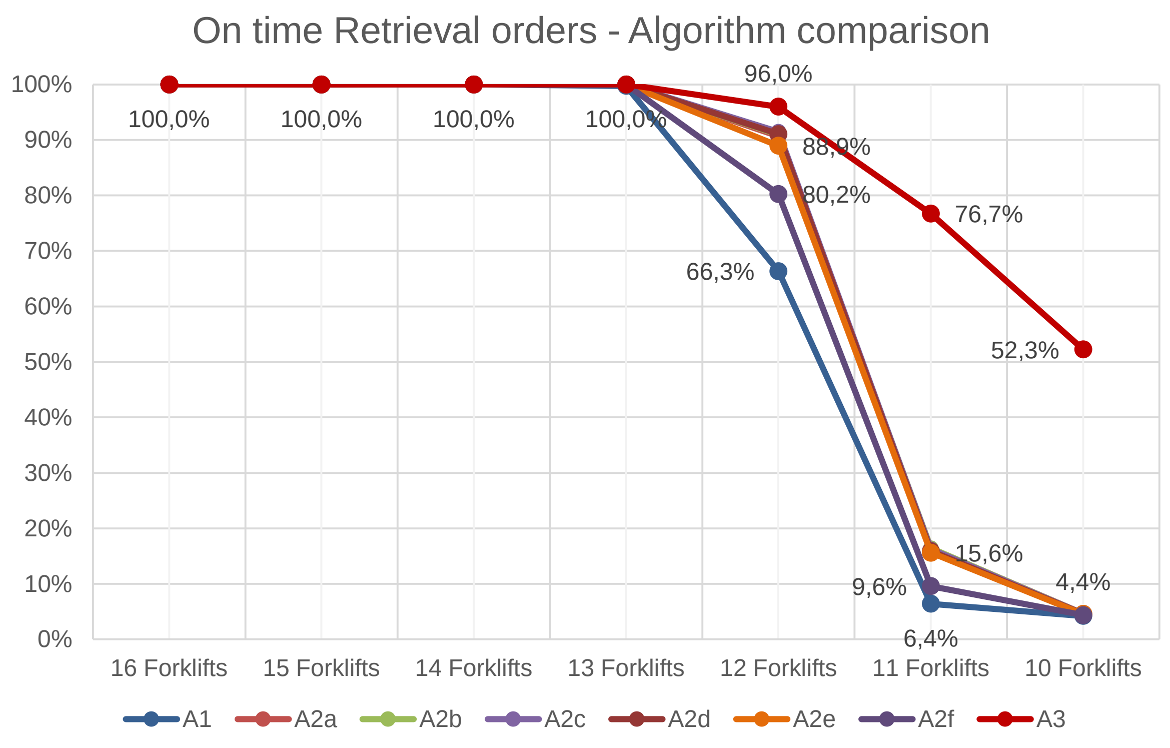


Figure 4-10: On time Retrieval Orders - Algorithm Comparison. A3 demonstrates superior robustness in maintaining high on-time retrieval rates across various fleet sizes, especially when resources are constrained to 12 forklifts or fewer, highlighting its effectiveness in high-demand, flexible operational environments.

4.3.2 Empty runs

Table 4-10 and Figure 4-11 present a comparison of the percentage of empty runs across all algorithms, highlighting their ability to reduce unproductive travel as the fleet size changes.

- **A1 (Sequential Forklift Assignment Algorithm):** A1 shows the highest percentage of empty runs, starting at 59.6% with 16 forklifts and gradually decreasing to 49.6% at 10 forklifts. While it achieves a reduction, the algorithm's ability to minimize empty runs remains limited, suggesting inefficiency in task allocation as fewer forklifts are available.
- **A2a to A2f (Priority-Based Algorithms):** The A2 algorithms perform better than A1, starting at around 50% empty runs for 16 forklifts and showing gradual improvements as the fleet size decreases. The algorithms that balance routing distance and order backlog, such as A2d and A2e, deliver the best results in reducing empty runs. In contrast, A2f, which emphasizes the backlog more heavily, shows slightly higher empty runs, indicating a trade-off between prioritizing order fulfillment and reducing unnecessary travel.
- **A3 (Auction-Based Routing Algorithm):** A3 consistently outperforms the other algorithms, with empty runs starting at 48.4% for 16 forklifts and decreasing to 43.0% for 10 forklifts. This highlights A3's superior ability to allocate tasks dynamically, resulting in more efficient travel even as the fleet size is reduced.

When considering 100% on-time retrievals, A3 again stands out with the lowest percentage of empty runs at 46.3%, compared to A1's highest rate of 54.0%. This suggests that A3 handles task allocation more efficiently, even when aiming for perfect retrieval performance. Furthermore, in scenarios where 95% on-time retrievals are acceptable, A3 further reduces empty runs to 44.9%, reinforcing its dominance in minimizing unproductive travel.

In conclusion, A3 proves to be the most effective algorithm in reducing empty runs across all fleet sizes. While the A2 algorithms, particularly A2d and A2e, demonstrate decent efficiency, they still fall short of A3's performance, making A3 the top choice for optimizing forklift operations with minimal unproductive travel.

Table 4-10: Empty Runs - Algorithm Comparison.

Number of forklifts	A1	A2a	A2b	A2c	A2d	A2e	A2f	A3
16 Forklifts	59,6%	49,6%	49,6%	50,1%	50,8%	51,0%	51,0%	48,4%
15 Forklifts	58,3%	49,3%	49,3%	49,6%	50,2%	50,5%	50,6%	47,9%
14 Forklifts	56,5%	48,7%	48,7%	48,9%	49,3%	49,6%	49,8%	47,4%
13 Forklifts	54,0%	48,1%	48,0%	48,1%	48,3%	48,5%	49,1%	46,3%
12 Forklifts	50,6%	47,0%	47,0%	47,0%	47,1%	47,2%	48,0%	44,9%
11 Forklifts	49,6%	46,3%	46,2%	46,3%	46,3%	46,3%	47,5%	45,7%
10 Forklifts	49,6%	46,3%	46,2%	46,3%	46,3%	46,3%	47,5%	43,0%
100% on time retrievals	54,0%	48,1%	48,0%	48,1%	48,3%	48,5%	49,1%	46,3%

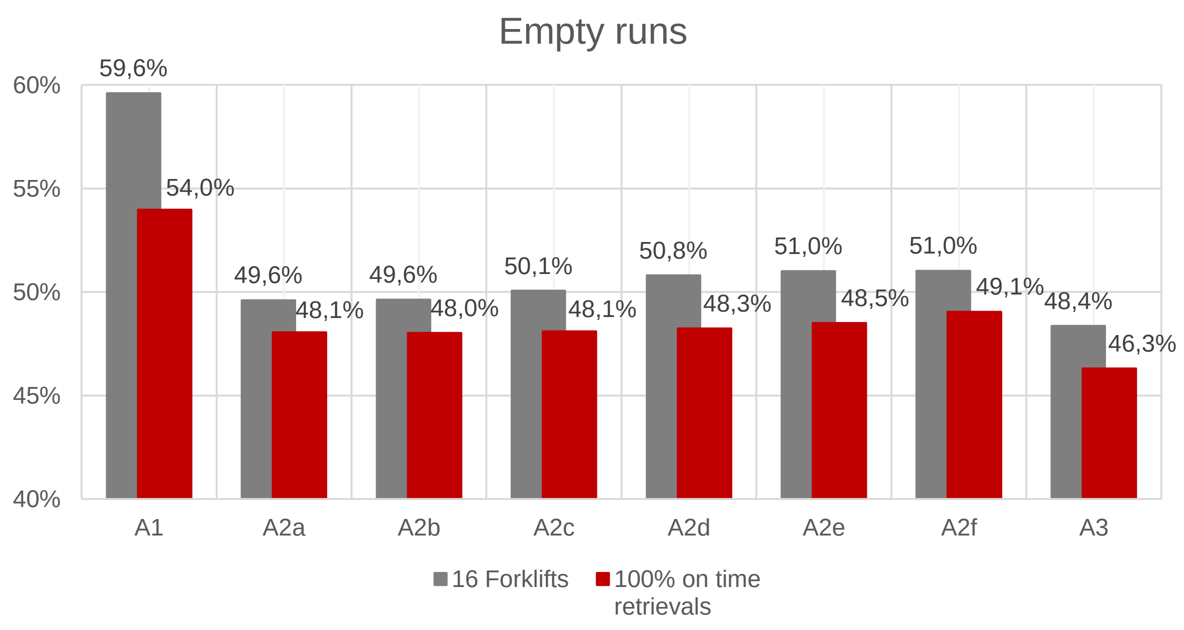


Figure 4-11: Empty Runs - Algorithm Comparison illustrates the empty run percentages for each algorithm with 16 forklifts and the lowest fleet size for which each algorithm still achieved 100% on-time retrievals.

4.3.3 Travelled distance

The results in
Table 4-11 and Figure 4-12 highlight the differences in total distance travelled by forklifts across the various algorithms. This metric is crucial for evaluating how efficiently each algorithm minimizes unnecessary travel and optimizes routes.

As discussed in Section 3.5, travelled distance correlates strongly with empty runs. Thus, the trends observed in empty run percentages are similarly reflected in the travelled distances for each algorithm. Algorithms with lower empty runs also show reduced travel distances, indicating their effectiveness in route optimization and minimizing non-productive travel time.

In summary, A3 proves to be the most effective algorithm in minimizing travelled distance across all fleet sizes, particularly when achieving 100% on-time retrievals. The A2 algorithms, especially A2d and A2e, also perform well, but A1 continues to lag behind, with significantly higher travel distances, making it the least efficient option for route optimization.

Table 4-11: Travelled Distance - Algorithm Comparison

Number of forklifts	A1	A2a	A2b	A2c	A2d	A2e	A2f	A3
16 Forklifts	250,29	200,94	201,07	202,76	205,75	206,64	206,68	196,06
15 Forklifts	258,64	212,88	212,95	214,43	216,82	218,32	218,51	207,15
14 Forklifts	265,59	225,51	225,70	226,61	228,01	229,66	230,47	219,80
13 Forklifts	270,69	239,93	239,79	240,12	240,83	242,00	244,58	232,12
12 Forklifts	273,00	254,52	254,57	254,82	255,03	255,79	259,40	245,02
11 Forklifts	279,07	269,49	269,48	269,48	269,54	269,58	272,84	260,71
10 Forklifts	279,09	269,90	269,92	269,89	269,94	270,02	273,21	259,28
100% on time retrievals	270,69	239,93	239,79	240,12	240,83	242,00	244,58	232,12

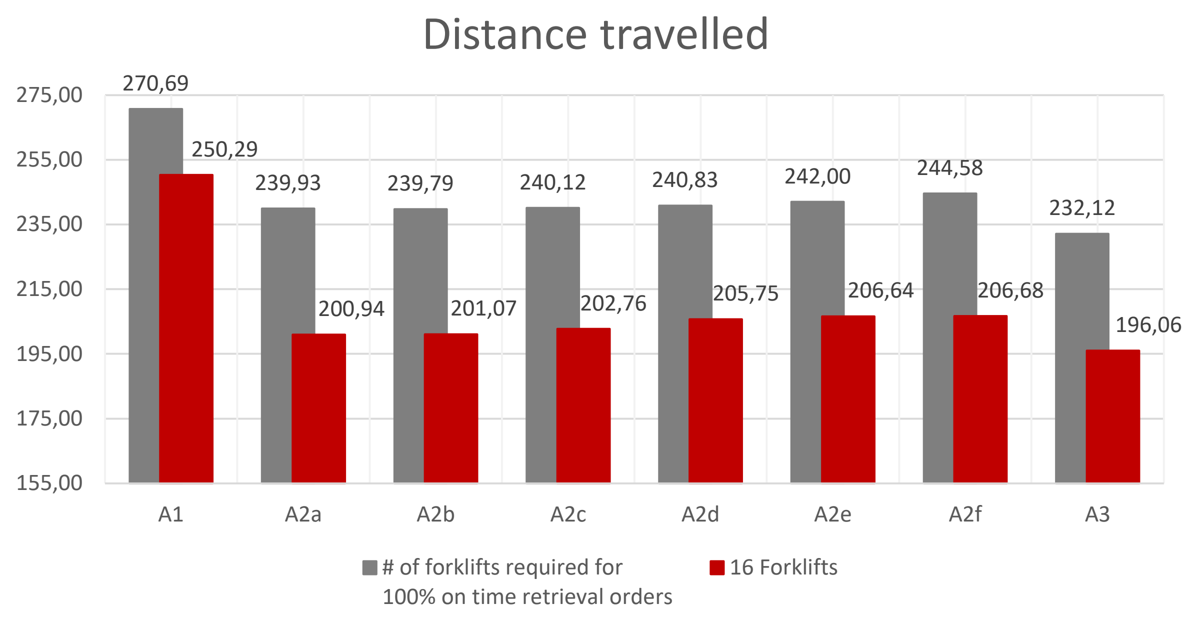


Figure 4-12: Travelled Distance - Algorithm Comparison illustrates the total distance travelled for each algorithm with 16 forklifts and the lowest fleet size for which each algorithm still achieved 100% on-time retrievals. This comparison highlights the efficiency of each algorithm in minimizing travel distance while maintaining full retrieval performance.

In summary, across all three key performance indicators - on-time retrieval orders, empty runs, and travelled distance - the A3 Auction-Based Routing Algorithm consistently outperforms the other algorithms. It maintains higher efficiency in resource utilization, reduces unnecessary travel, and ensures timely completion of critical tasks, even as the number of available forklifts decreases. Based on these results, A3 is identified as the optimal solution for optimizing forklift task allocation in dynamic and resource-constrained environments. As a result, the A3 algorithm will be applied in the subsequent real-world use case to evaluate its effectiveness in a practical warehouse setting.

4.4 Real-world Use Case results

This section presents the results of applying the chosen algorithm and localization system to a real-world warehouse layout. The performance of two different localization systems is compared, and the impact of reducing the forklift fleet size is examined. The analysis focuses on key performance indicators such as on-time retrieval orders and empty runs, along with potential cost savings that could result from using fewer forklifts. The insights gained from these results will guide the implementation of the most suitable solution for real-world warehouse operations.

4.4.1 Forklift Fleet Size Reduction

In this section, we analyse the results of the parameter study using the A3 algorithm, where the number of forklifts was incrementally reduced from 42 to 30. The following table presents the forklift fleet size reduction in the parameter study, illustrating how the number of forklifts varied across different scenarios.

Table 4-12: Forklift fleet size reduction in the parameter study.

Parameter study	Total Number of Forklifts	Number of Type 1 Forklifts	Number of Type 2 Forklifts
Simulation Study 1	42	35	7
Simulation Study 2	41	35	6
Simulation Study 3	37	30	7
Simulation Study 4	36	30	6
Simulation Study 5	34	28	6
Simulation Study 6	32	27	5
Simulation Study 7	30	25	5

The objective was to determine the minimum number of forklifts required to maintain acceptable operational efficiency, with a particular focus on on-time retrieval orders and the time overruns of late retrievals. This study was motivated by potential cost savings and operational optimization, as reducing fleet size can decrease expenses related to fuel, maintenance, and labour, provided performance remains within acceptable limits.

Figure 4-13 presents the results, displaying both on-time retrieval rates and time overruns for late retrievals across different fleet sizes. The upper portion of Figure 4-13 illustrates the trend in on-time retrieval orders as the fleet size decreases, showing strong performance with 36 or more forklifts, where on-time retrieval rates stay close to 99%. However, there is a notable decline as fleet size is further reduced. The lower section of Figure 4-14 highlights the time overruns of late retrieval orders, using Box plots created with Seaborn - a Python-based statistical data visualization library [29].

Each plot's largest box represents the central 50% of data points, with the median shown as a middle line. Boxes above and below the median contain the next 25% of data, with outliers represented as points beyond the boxes [30]. The width of each box represents the frequency of data points within each section, much like a histogram, indicating the distribution of retrieval times. As the fleet size decreases, the width of the boxes broadens, signifying a higher number of late retrievals [30]. The 'n' values within each plot indicate the total number of retrieval orders not completed on time, providing insight into how retrieval performance degrades with fewer forklifts.

In total, 178,610 transport orders were processed during the simulation, with 112,014 of those being retrieval orders. This high volume of data provides a robust foundation for analysing the effects of reducing the forklift fleet on system performance.

As shown in Figure 4-13, on-time retrieval rates remain high (above 99%) when the fleet size is 37 forklifts or more. For 36 forklifts, the rate still holds at a reasonable 98.72%. However, when the number of forklifts is reduced below 34, the system starts to collapse. At 33 forklifts, the on-time retrieval rate drops to 97.35%, and with only 30 forklifts, it plummets to 80.26%.

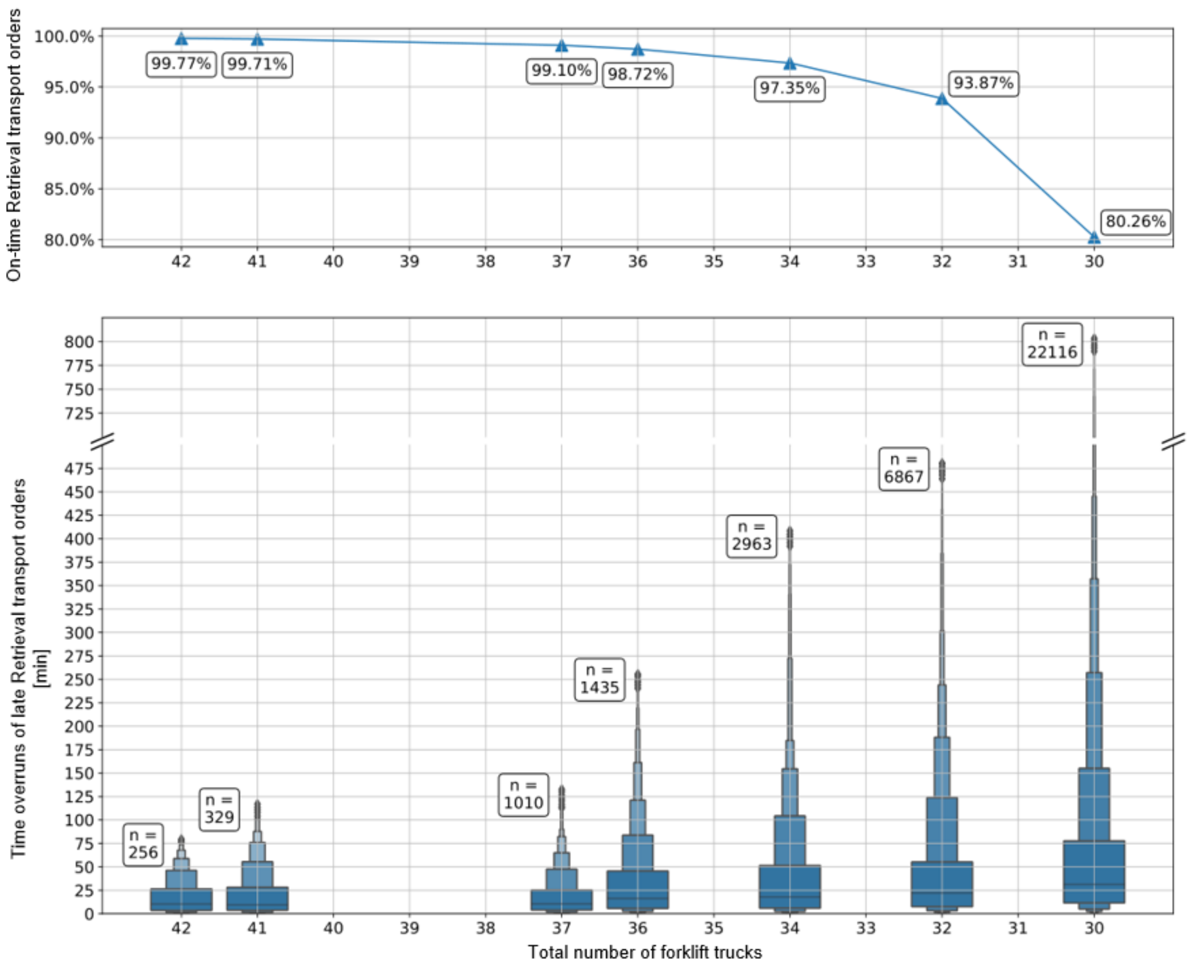


Figure 4-13: On-Time Retrieval Orders and Time Overruns of Late Retrievals for Different Fleet Sizes (Box plot created via Seaborn [29]). Number of transport orders = 178,610, Number of retrieval orders = 112,014.

In practical terms, reducing the fleet size to 36 forklifts is a viable solution. While the on-time retrieval rate is no longer at 100%, a rate of 97.35% is still considered operationally acceptable. In real-world scenarios, a 100% on-time retrieval rate is rarely achievable due to factors such as forklift maintenance, human error, or unexpected disruptions. Achieving over 97% on-time retrievals would still allow the warehouse to operate efficiently while benefiting from the cost savings associated with a reduced forklift fleet.

It’s important to note that this simulation study, while valuable, does not account for all potential real-world variables. Factors such as peak operational hours, unexpected equipment failures, or warehouse layout changes could affect the performance of a reduced fleet size. Therefore, while 36 forklifts appear to be a plausible solution based on the simulation results, close monitoring and periodic adjustments would be required in a real-world setting to maintain this level of performance.

4.4.2 Impact of Localization Systems on Forklift Efficiency

Table 4-13 presents the results of comparing the coarse and fine localization systems using the A3 Auction-Based Routing Algorithm with 42 forklifts. Both systems deliver identical on-time retrieval rates of 99.77%, showing that in terms of punctuality, there is no discernible difference between the two systems. However, when analysing the percentage of empty runs, the fine localization system shows a slight improvement, with 43.17% empty runs compared to 45.05% for the coarse system.

Table 4-13: Comparison of Coarse and Fine Localization Systems with A3 Auction-Based Routing Algorithm.

Localization System Comparison with A3 - Auction-Based Routing Algorithm		
Criteria	Coarse Localization System	Fine Localization System
Number of forklifts	42	42
On time retrieval orders	99,77%	99,77%
Empty runs	45,05%	43,17%

Despite this slight improvement, the overall difference between the two localization systems is minimal. The minor distinction between them lies in how each system calculates the forklift's position. The coarse system assumes the forklift remains at the last known position after completing a task ("last sink"), which can result in minor inaccuracies during task allocation. The fine localization system, on the other hand, updates the forklift’s position at each node in the warehouse network, providing more accurate location information. While this finer granularity could theoretically enhance efficiency, the difference in performance between the two systems is minimal, as reflected in the data.

Furthermore, as shown in Figure 4-14, the potential for optimization through more precise localization, such as real-time location tracking, is extremely limited. Real-time location tracking was considered but ultimately not investigated further, as the slight differences observed between the coarse and fine systems suggest that any additional performance gains from real-time location tracking would likely be negligible. Only forklifts without an active task would directly benefit from real-time location tracking, which accounts for just 1% of cases. For the remaining 99%, forklifts are either en route to fulfill an order or parked, meaning their exact location is either irrelevant or already integrated into task allocation.

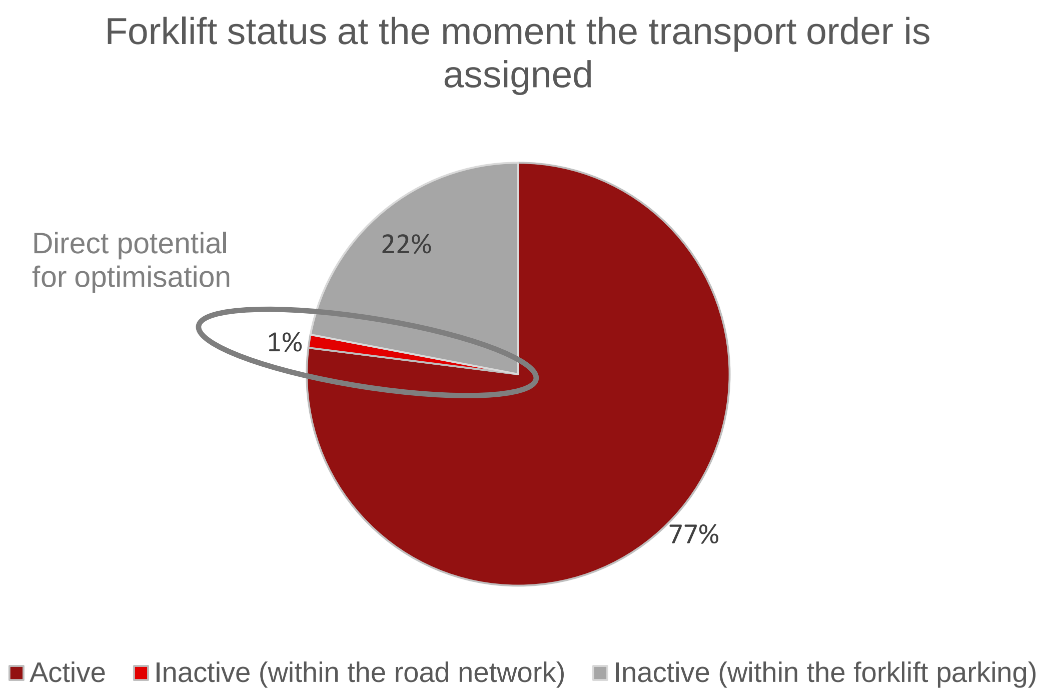


Figure 4-14: Direct Optimization Potential of Forklifts During Task Allocation.

Even for forklifts with active tasks, the difference in calculated travel times resulting from different localization systems would only amount to a few seconds or, at most, a couple of minutes. Given the much larger uncertainties in real-world operations (e.g., unexpected delays, maintenance issues), this slight discrepancy in calculated travel times is considered negligible.

5 Summary and Outlook

5.1 Summary

This thesis focuses on developing an optimized FGS for warehouse operations, exploring various routing algorithms and localization systems to improve efficiency. The primary goal was to minimize empty runs, reduce travel distances, and enhance on-time retrieval rates. By comparing several algorithms and localization systems, the study sought to identify the best-performing solution that could be practically implemented in a warehouse environment. The study involved both a benchmark simulation and a real-world use case to evaluate the performance of different algorithms and localization systems under varying conditions.

The fine localization system, which was used in the benchmark simulation, was selected for the real-world use case as well. It provided a better balance between accuracy and implementation feasibility compared to the coarse localization system.

Through extensive simulations, the auction-based routing algorithm (A3) was found to outperform other algorithms, demonstrating strong performance even as the number of forklifts was reduced. In the benchmark simulation, the fleet of 16 forklifts could be reduced to 12, while still maintaining acceptable on-time retrieval values. In the real-world use case, the forklift fleet was reduced from 42 to 34, while still achieving high operational efficiency.

Research Questions:

RQ1: What is the most effective routing algorithm for optimizing forklift operations in terms of task allocation and minimizing empty runs?

- **Answer**: The A3 Auction-Based Routing Algorithm proved to be the most effective, offering the best balance between task allocation, minimizing empty runs, and maintaining high on-time retrieval rates.

RQ2: How does fleet size impact retrieval order performance, and what is the minimum number of forklifts needed to maintain acceptable levels of on-time retrieval orders?

- **Answer**: In the benchmark simulation, the fleet size could be reduced from 16 to 12 forklifts while maintaining valid on-time retrieval rates. In the real-world use case, the fleet was successfully reduced from 42 to 36 forklifts, maintaining high operational efficiency.

RQ3: How does the choice and accuracy of localization systems impact forklift performance, task allocation efficiency, and overall operational effectiveness?

- **Answer**: The fine localization system proved to be the most effective solution, providing more precise updates on forklift positions compared to the coarse system. However, the difference in performance between coarse and fine localization was relatively small, suggesting that the additional precision of real-time tracking would yield negligible improvements. Consequently, knowing the exact location of forklifts in real time does not significantly enhance task allocation efficiency, making real-time tracking unnecessary for effective forklift performance and overall operational effectiveness.

5.2 Outlook

While this study has successfully identified the optimal FGS algorithm and the most suitable localization system, there are several opportunities for further research and improvement. The findings demonstrate the effectiveness of the auction-based A3 algorithm, but there is considerable potential to refine both the model and the broader system in ways that would improve its applicability in more complex, real-world environments.

One area where this could be realized is in the development of more sophisticated algorithms. The A3 algorithm, which performed best in this study, has proven its capacity to manage task allocation efficiently, even as the number of forklifts was reduced. However, future research could investigate alternative optimization techniques or hybrid algorithms. For example, variations of the TSP, nearest neighbour algorithms, or reinforcement learning as a promising approach could be explored to push the system's efficiency further. Algorithm refinement should focus on improving adaptability, especially under varied operational conditions such as changing demand patterns or diverse warehouse layouts.

Beyond algorithmic improvements, the simulation model itself could benefit from greater complexity and realism. While the study used a 2D warehouse layout for simplicity, transitioning to a 3D model would offer substantial benefits in terms of visual clarity and operational accuracy. A 3D model better reflects the real-world warehouse environment, especially in industries where high-bay storage systems are used. This shift would not only enhance the simulation's accuracy but also make the system more marketable, as potential customers could see a more realistic representation of the warehouse logistics in action. Incorporating the vertical axis, especially for tasks involving the retrieval of goods at different heights, would allow for more precise calculation of handling times, which could vary based on task complexity. This approach would also enable order restrictions to be applied to each individual task rather than by zone, enhancing flexibility and accuracy in task allocation.

Additionally, the handling time, which was modelled as a constant value in this study (e.g., 30 seconds per task), could be adjusted to better reflect real-world variability. In practice, handling times are influenced by multiple factors, including the type of task, the height of the storage, and the weight of the load. Future simulations could implement dynamic handling time distributions to introduce variability into the process, providing a more accurate representation of operational complexity. Furthermore, battery management for forklifts, including the need for regular charging and the downtime associated with recharging, should be incorporated into future models to improve operational accuracy. Maintenance requirements and downtime were also not included in the current model. Introducing parameters such as Mean Time to Repair (MTTR) and simulating regular maintenance schedules could allow for a more realistic view of forklift availability and the overall efficiency of the warehouse system.

In the context of real-world application, human factors and operational uncertainties also present an opportunity for enhancing the model. Warehouse operations are rarely as predictable as a simulation environment; real-world factors such as human error, forklift operator behaviour, and the learning curves associated with using new systems need to be accounted for. Simulating human factors would add another layer of realism, helping to predict how well the system might perform when subject to real-world constraints. Additionally, the current simulation assumed constant speeds and handling efficiency, but actual forklift performance is often affected by variables such as acceleration, speed fluctuations, and congestion in the warehouse. Incorporating these aspects would improve the system's robustness, making it more adaptable to different scenarios.

Another consideration is the inclusion of multiple-item transports. Currently, such decisions are left to the discretion of forklift operators and are largely dependent on the stack ability of load carriers. In the simulation, it was assumed that all storage and empty load transports with the same source, destination, and creation timestamp would be grouped into pairs. Accounting for these types of transport bundling in future simulations could provide a more accurate reflection of operational choices and improve the model's applicability in real-world scenarios.

In addition to refining the internal workings of the simulation, expanding the scope of future studies could provide further insights. Testing the system under different layouts, transport order structures, or operational peaks would help determine its flexibility and scalability. A design of experiments approach could be applied to systematically vary input parameters, such as the number of storage locations, to examine how these factors influence performance differences between algorithms. Benchmark studies might include varying warehouse configurations or industries with unique logistical challenges. Additionally, exploring the impact of introducing multiple operational shifts or simulating peak periods would allow for evaluation of system performance under higher workloads, offering a more comprehensive understanding of its robustness.

In summary, while this study has demonstrated the potential for significant cost reductions and efficiency improvements by implementing a well-optimized Forklift Guidance System and localization approach, it also highlights areas where further refinement is needed. The findings show that the A3 algorithm can deliver strong results, even in reduced fleet sizes, and that the fine localization system offers sufficient accuracy without the need for real-time tracking. However, the pursuit of greater realism and adaptability in the model, combined with ongoing algorithmic improvements, will be crucial for future studies aiming to bridge the gap between simulation-based solutions and their successful application in complex, real-world environments.

Bibliography

[1] L. d'Apolito and H. Hong, "Forklift truck performance simulation and fuel consumption estimation," *Journal of Engineering, Design and Technology,* vol. 18, no. 3, pp. 689-703, 2020.

[2] MHE Bazar, "The Role of Forklifts in the Logistics Industry," 14 April 2023. [Online]. Available: https://www.linkedin.com/pulse/role-forklifts-logistics-industry-mhe-bazar/. [Accessed 10 April 2024].

[3] Research and Markets, „Global Forklift Market - Global Industry Size, Share, Trends, Opportunity, and Forecast, 2018-2028,“ October 2023. [Online]. Available: https://www.researchandmarkets.com/research/qtwzpq/world_forklift?w=12. [Zugriff am 10 April 2024].

[4] Mordor Intelligence, "Forklift Trucks Market Size & Share Analysis - Growth Trends & Forecasts (2024 - 2029)," [Online]. Available: https://www.mordorintelligence.com/industry-reports/forklift-trucks-market. [Accessed 10 April 2024].

[5] S. Prateek, "Discrete-Event Simulation," *International Journal of Scientific & Technology Research,* vol. 4, no. 4, pp. 136-140, April 2015.

[6] M. Mirlach, W. Günthner, A. Ulbrich und K. Beckhaus, „Auftragszuteilungsverfahren für Staplerleitsysteme,“ in *17. Flurförderzeugtagung 2013*, Düsseldorf, 2013.

[7] W. Günthner and M. ten Hompel, Internet der Dinge in der Intralogstik, vol. 1, Berlin: Springer-Verlag, 2010.

[8] W. J. Hopp and M. L. Spearman, Factory Physics: Foundations of Manufacturing Management, 2 ed., McGraw-Hill, 2001, p. 698.

[9] B. Rouwenhorst, Reuter, S. V. B, G. J. van Hotum and W. H. M. Zijm, "Warehouse Design and Control: Framework and Literature Review," *European Journal of Operational Research,* vol. 122, no. 3, pp. 515-533, 2000.

[10] J. Van den Berg and W. Zijm, "Models for warehouse management: Classification and examples," *International Journal of Production Economics,* vol. 59, no. 1-3, pp. 519-528, 1999.

[11] N. Hafner, "Staplerleitsysteme: Anwendungen, Funktionen und Technik (Teil 01)," *Staplerworld,* January 2012.

[12] J. Banks, J. S. Carson, B. L. Nelson and D. M. Nicol, Discrete-Event System Simulation, vol. 4, Pearson Prentice Hall, NJ: Upper Saddle River, 2005.

[13] Siemens Digital Industries Software, "Tecnomatix Plant Simulation," [Online]. Available: https://plm.sw.siemens.com/en-US/tecnomatix/products/plant-simulation-software/. [Accessed 16 April 2024].

[14] M. ten Hompel, T. Schmidt and J. Dregger, Materialflusssysteme, 4 ed., Dortmund: Springer-Verlag, 2018, p. 407.

[15] D. Arnold and K. Furmans, Materialfluss in Logistiksystemen, 7 ed., Karlsruhe: Springer-Verlag, 2019, p. 418.

[16] S. Bangsow, Manufacturing Simulation with Plant Simulation and SimTalk, Berlin: Springer-Verlag, 2010, p. 297.

[17] S. Robinson, Simulation: The Practice of Model Development and Use, Warwick: John Wiley & Sons, 2004, p. 316.

[18] H. Martin, Transport- und Lagerlogistik, 9 ed., Hamburg: Springer-Verlag, 2013, p. 549.

[19] L. März, W. Krug, O. Rose und G. Weigert, Simulation und Optimierung in Produktion und Logistik, Karlsruhe: Springer-Verlag, 2010, p. 220.

[20] N. Hafner, "Staplerleitsysteme: Anwendungen, Funktionen und Technik (Teil 02)," *Staplerworld,* February 2012.

[21] "Tutorial - Chapter 1: Getting Started with Plant Simulation," [Online]. Available: https://plant-simulation.de/schulungen/tutorial/tutorial-chapter1/. [Accessed 15 September 2024].

[22] "Produktüberblick," [Online]. Available: https://plant-simulation.de/produktueberblick/. [Accessed 15 September 2024].

[23] "Use plant simulation and throughout optimization to improve manufacturing performance," [Online]. Available: https://resources.sw.siemens.com/it-IT/fact-sheet-plant-simulation-and-throughout-optimization-to-improve-manufacturing. [Accessed 15 September 2024].

[24] G. Weyenberg and R. Yoshida, "Reconstructing the Phylogeny: Computational Methods," in *Algebraic and Discrete Mathematical Methods for Modern Biology*, Cambridge, MA, Academic Press, 2015, pp. 293-319.

[25] "Tecnomatix Plant Simulation Help," Siemens, [Online]. Available: https://docs.sw.siemens.com/en-US/doc/297028302/PL20190919115137952.PlantSimulationENU/Help_Start_Page. [Accessed 15 September 2024].

[26] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, Introduction to Algorithms, 3 ed., Cambridge, Massachusetts: The MIT Press, 2022, p. 1291.

[27] D. L. Applegate, R. E. Bixby, V. Chvátal und W. J. Cook, The Traveling Salesman Problem: A Computational Study, Princeton University Press, 2006, p. 606.

[28] J. Gu, G. Marc and M. L. F., "Research on warehouse design and performance evaluation: A comprehensive review," *European Journal of Operational Research,* no. 203 (3), pp. 539-549, 16 June 2010.

[29] M. L. Waskom, "seaborn: statistical data visualization," *Journal of Open Source Software,* no. 6 (60), p. 3021.

[30] H. Heike, K. Kafadar and H. Wickham, "Letter-value plots: Boxplots for large data," *The American Statistican,* 2011.

Appendix

A – Acquisition Costs SimTalk Code for the Auction-Based Routing Algorithm

```
param  X_von:integer, Y_nach:integer, Auftrag:string, Auftragsart:string,
Bedarfszeit:datetime, ÜbernahameKosten:integer->integer

var LängeZielliste:integer
var AktuellerAuftrag:integer
var Standort:object
var StandortPkt:string
var StandortName:string
var X_now:integer
var Y_now:integer
var X_0:integer
var X_1:integer
var Y_1:integer
var ÜbernahmeKosten:real
var ÜbernahmeKostenAlt:real

var PS_X:object
var PS_Y:object
PS_X:="PS_"+to_str(X_von)
PS_Y:="PS_"+to_str(Y_nach)

--Aufträge, die eine Auslagerung sind, sind zeitkritisch und bei den Auf-
trägen muss auch die Bedarfszeit berücksichtigt werden.
var tAktl:datetime:=Ereignisverwalter.AbsZeit

if Auftragsart="Auslagerung"
    var t:datetime:=tAktl
    var dauer:time:=0
    var dauerAbs:datetime:=t+dauer
end

--Zuerst wird die Situation am Gabelstapler bestimmt (ob der Gabelstapler
einen Auftrag gerade erledigt, bzw. ob der Gabelstapler Aufträge auf sei-
ner Zielliste hat).
if  self.~.Zielort=void or self.~.Zielort=ForkliftParking_1 //or
self.~.Zielort=ForkliftParking_2
    AktuellerAuftrag:=0
else
    AktuellerAuftrag:=1
end

--Jetzt wird ein Update gemacht, falls in der Zwischenzeit ein Auftrag er-
ledigt worden ist und der Auftrag ist dann aus der Zielliste entfernt.
--schau M_ZiellisteAbfrage
if  self.~.Zielliste[2,1]=void
    self.~.Zielliste.entferneZeile(1)
end

if  self.~.Zielliste.yDim=0
    LängeZielliste:=0
else
    LängeZielliste:=self.~.Zielliste.yDim
end

--Aktueller Standort vom Gabelstapler:
```

```
var StandortAktl:object
var StandortAktlName:string
var StandortAktlPt:string

StandortAktl:=self.~.Standort
StandortAktlName:=self.~.Standort.Name

if  StandortAktl=ForkliftParking_1
    StandortAktlPt:="9030"
//elseif    StandortAktl=ForkliftParking_2
    //StandortAktlPt:="xxxx"
/*elseif strlen(StandortAktlName)<5
    StandortAktlPt:=strRcopy(StandortAktlName,2)
elseif strlen(StandortAktlName)=5
    if strRpos("P_", StandortAktlName)=1
        StandortAktlPt:=strRcopy(StandortAktlName,3)
    else
        StandortAktlPt:=strRcopy(StandortAktlName,2)
    end*/
elseif strlen(StandortAktlName)=6
    if strRpos("P_", StandortAktlName)=1
        StandortAktlPt:=strRcopy(StandortAktlName,4)
    /*else
        StandortAktlPt:=strRcopy(StandortAktlName,3)*/
    end
/*elseif strlen(StandortAktlName)=7
    StandortAktlPt:=strRcopy(StandortAktlName,3)
    if strRpos("n", StandortAktlPt)=1
        StandortAktlPt:=strRcopy(StandortAktlName,2)
    else
        StandortAktlPt:=strRcopy(StandortAktlName,3)
    end
elseif strlen(StandortAktlName)=8
    StandortAktlPt:=strRcopy(StandortAktlName,3)*/
elseif strlen(StandortAktlName)=7
    if strRpos("PS_", StandortAktlName)=1
        StandortAktlPt:=strRcopy(StandortAktlName,4)
    elseif strRpos("PP_", StandortAktlName)=1
        StandortAktlPt:=strRcopy(StandortAktlName,4)
    end
    StandortAktlPt:=strRcopy(StandortAktlName,4)
elseif strlen(StandortAktlName)=10
    StandortAktlPt:=strRcopy(StandortAktlName,4)

end

--AktuellerStandort
X_0:=str_to_num(StandortAktlPt)

--Jetzt wird überprüft, ob der Gabelstapler eine Palette auf sich hat oder
nicht.
--Diese Information wird wichtig für die Berechnung der Dauer.
--Falls ohne Palette (leer) --> Gabelstapler geht von Standort X0 bis
Quelle, dann zur Senke, etc.
--Falls mit Palette (beladen) --> Gabelstapler geht vom Standort X0 zur
Senke, etc.

var Status:string
if self.~.Inhalt=void
    Status:="GabelstaplerLeer"
else
    Status:="GabelstaplerBeladen"
end
```

```
--4 verschiedene Fälle:

--Fall 1:
        --Gabelstapler und Zielliste sind leer.
        --Gabelstapler hat keinen Zielort, bzw. der Zielort ist der Park-
platz.
        --AktuellerAuftrag = 0 und LängeZielliste = 0.
if AktuellerAuftrag=0 and LängeZielliste=0
--Der aktuelle Standort ist schon bekannt - X0.
    StandortName:=self.~.Standort.Name
    Standort:=self.~.Standort
    --Ist die Auftragsart eine Auslagerung, der Auftrag ist zeitkritisch.
    if Auftragsart="Auslagerung"
        dauer:=((T_Distanzmatrix[#X_0,#X_von]+T_Distanz-
matrix[#X_von,#Y_nach])/2)+2*30
        dauerAbs:=t+dauer
        --Im Fall, dass die Auftragserledigung Dauer länger ist als die
Bedarfszeit, wird der Auftrag nicht angenommen.
        --Die ÜbernahmeKosten bekommen einen Wert von 1.000.000.
        if dauerAbs>Bedarfszeit
            ÜbernahmeKosten:=1000000
        else
            ÜbernahmeKosten:=T_Distanzmatrix[#X_0,#X_von]
        end
    --Falls es sich um keine Auslagerung handelt, die Bedarfszeit wird
nicht betrachtet und die ÜbernahmeKosten werden aus der Distanzmatrix be-
rechnet.
    else
        ÜbernahmeKosten:=T_Distanzmatrix[#X_0,#X_von]
    end

--Fall 2:
        --Gabelstapler hat einen Auftrag, Zielliste ist leer.
        --Gabelstapler hat einen Zielort.
        --AktuellerAuftrag = 1 und LängeZielliste = 0
elseif AktuellerAuftrag=1 and LängeZielliste=0
    X_now:=self.~.Aufruf
    Y_now:=self.~.AktuellerAuftragNach
    --Berechnung der bestehenden ÜbernahmeKosten (bei der jetzigen Situa-
tion am Gabelstapler ohne den neuen Auftrag).
        if Status="GabelstaplerLeer"
            ÜbernahmeKostenAlt:=T_Distanzmatrix[#X_0,#X_now]+T_Distanz-
matrix[#X_now,#Y_now]
        else --Status="GabelstaplerBeladen"
            ÜbernahmeKostenAlt:=T_Distanzmatrix[#X_0,#Y_now]
        end
    --Berechnung der ÜbernahmeKosten sowie der entsprechenden Dauer im
Fall, dass der Auftrag auf den Gabelstapler angenommen wird.
        --Ist die Auftragsart eine Auslagerung, der Auftrag ist zeitkri-
tisch.
        if Auftragsart="Auslagerung"
            if Status="GabelstaplerLeer"
                ÜbernahmeKosten:=T_Distanzmatrix[#X_0,#X_now]+T_Distanz-
matrix[#X_now,#Y_now]+T_Distanzmatrix[#Y_now,#X_von]
                dauer:=((T_Distanzmatrix[#X_0,#X_now]+T_Distanz-
matrix[#X_now,#Y_now]+T_Distanzmatrix[#Y_now,#X_von]+T_Distanz-
matrix[#X_von,#Y_nach])/2)+4*30
                dauerAbs:=t+dauer
            else --Status="GabelstaplerBeladen"
                ÜbernahmeKosten:=T_Distanzmatrix[#X_0,#Y_now]+T_Distanz-
matrix[#Y_now,#X_von]
```

```
                dauer:=((T_Distanzmatrix[#X_0,#Y_now]+T_Distanz-
matrix[#Y_now,#X_von]+T_Distanzmatrix[#X_von,#Y_nach])/2)+3*30
                dauerAbs:=t+dauer
            end
            --Im Fall, dass die Auftragserledigung Dauer länger ist als
die Bedarfszeit, wird der Auftrag nicht angenommen.
            --Die ÜbernahmeKosten bekommen einen Wert von 1.000.000.
            if dauerAbs>Bedarfszeit
                ÜbernahmeKosten:=1000000
            else
                --Kostendifferenz zwischen den  ÜbernahmeKosten mit dem
neuen Auftrag und den alten  ÜbernahmeKosten ohne ihn (ÜbernahmeKosten-
Alt).
                ÜbernahmeKosten:=ÜbernahmeKosten-ÜbernahmeKostenAlt
            end
        --Falls es sich um keine Auslagerung handelt.
        else
            if Status="GabelstaplerLeer"
                ÜbernahmeKosten:=T_Distanzmatrix[#X_0,#X_now]+T_Distanz-
matrix[#X_now,#Y_now]+T_Distanzmatrix[#Y_now,#X_von]
            else --Status="GabelstaplerBeladen"
                ÜbernahmeKosten:=T_Distanzmatrix[#X_0,#Y_now]+T_Distanz-
matrix[#Y_now,#X_von]
            end
        --Kostendifferenz zwischen den  ÜbernahmeKosten mit dem neuen Auf-
trag und den alten  ÜbernahmeKosten ohne ihn (ÜbernahmeKostenAlt).
        ÜbernahmeKosten:=ÜbernahmeKosten-ÜbernahmeKostenAlt
        end

    --Der Auftrag wird auf die ZiellisteVergabe hinzugefügt, und im Fall,
dass dieser Gabelstapler am günstigsten wird, und der ausgewählt wird,
dann wird die  ZiellisteVergabe auf die Zielliste überschrieben.
    self.~.ZiellisteVergabe[1,1]:=PS_X
    self.~.ZiellisteVergabe[2,1]:=Auftrag
    self.~.ZiellisteVergabe[3,1]:=X_von
    self.~.ZiellisteVergabe[4,1]:=PS_Y
    self.~.ZiellisteVergabe[5,1]:=Y_nach
    self.~.ZiellisteVergabe[6,1]:=Bedarfszeit
    self.~.ZiellisteVergabe[7,1]:=Auftragsart

--Fall 3:
        --Gabelstapler hat einen Auftrag, Zielliste hat einen Auftrag.
        --AktuellerAuftrag = 1 und LängeZielliste = 1
elseif AktuellerAuftrag=1 and LängeZielliste=1
    X_now:=self.~.Aufruf
    Y_now:=self.~.AktuellerAuftragNach
    X_1:=self.~.Zielliste[3,1] --Quelle von den einzigen Auftrag auf der
Zielliste
    Y_1:=self.~.Zielliste[5,1] --Senke von den einzigen Auftrag auf der
Zielliste
    --Berechnung der bestehenden ÜbernahmeKosten (bei der jetzigen Situa-
tion am Gabelstapler (ein aktueller Auftrag und ein Auftrag auf der Ziel-
liste), ohne den neuen Auftrag).
        if Status="GabelstaplerLeer"
            ÜbernahmeKostenAlt:=T_Distanzmatrix[#X_0,#X_now]+T_Distanz-
matrix[#X_now,#Y_now]+T_Distanzmatrix[#Y_now,#X_1]+T_Distanz-
matrix[#X_1,#Y_1]
        else --Status="GabelstaplerBeladen"
            ÜbernahmeKostenAlt:=T_Distanzmatrix[#X_0,#Y_now]+T_Distanz-
matrix[#Y_now,#X_1]+T_Distanzmatrix[#X_1,#Y_1]
        end
    --Berechnung der ÜbernahmeKosten sowie der entsprechenden Dauer im
Fall, dass der Auftrag auf den Gabelstapler angenommen wird.
```

```
        --Ist die Auftragsart eine Auslagerung, der Auftrag ist zeitkri-
tisch.
        if Auftragsart="Auslagerung"
            if Status="GabelstaplerLeer"
                ÜbernahmeKosten:=T_Distanzmatrix[#X_0,#X_now]+T_Distanz-
matrix[#X_now,#Y_now]+T_Distanzmatrix[#Y_now,#X_1]+T_Distanz-
matrix[#X_1,#Y_1]+T_Distanzmatrix[#Y_1,#X_von]
                dauer:=((T_Distanzmatrix[#X_0,#X_now]+T_Distanz-
matrix[#X_now,#Y_now]+T_Distanzmatrix[#Y_now,#X_1]+T_Distanz-
matrix[#X_1,#Y_1]+T_Distanzmatrix[#Y_1,#X_von]+T_Distanz-
matrix[#X_von,#Y_nach])/2)+6*30
                dauerAbs:=t+dauer
            else --Status="GabelstaplerBeladen"
                ÜbernahmeKosten:=T_Distanzmatrix[#X_0,#Y_now]+T_Distanz-
matrix[#Y_now,#X_1]+T_Distanzmatrix[#X_1,#Y_1]+T_Distanz-
matrix[#Y_1,#X_von]
                dauer:=((T_Distanzmatrix[#X_0,#Y_now]+T_Distanz-
matrix[#Y_now,#X_1]+T_Distanzmatrix[#X_1,#Y_1]+T_Distanz-
matrix[#Y_1,#X_von]+T_Distanzmatrix[#X_von,#Y_nach])/2)+5*30
                dauerAbs:=t+dauer
            end
            --Im Fall, dass die Auftragserledigung Dauer länger ist als
die Bedarfszeit, wird der Auftrag nicht angenommen.
            --Die ÜbernahmeKosten bekommen einen Wert von 1.000.000.
            if dauerAbs>Bedarfszeit
                ÜbernahmeKosten:=1000000
            else
                ÜbernahmeKosten:=ÜbernahmeKosten-ÜbernahmeKostenAlt
            end
        --Falls es sich um keine Auslagerung handelt.
        else
            if Status="GabelstaplerLeer"
                ÜbernahmeKosten:=T_Distanzmatrix[#X_0,#X_now]+T_Distanz-
matrix[#X_now,#Y_now]+T_Distanzmatrix[#Y_now,#X_1]+T_Distanz-
matrix[#X_1,#Y_1]+T_Distanzmatrix[#Y_1,#X_von]
            else --Status="GabelstaplerBeladen"
                ÜbernahmeKosten:=T_Distanzmatrix[#X_0,#Y_now]+T_Distanz-
matrix[#Y_now,#X_1]+T_Distanzmatrix[#X_1,#Y_1]+T_Distanz-
matrix[#Y_1,#X_von]
            end
        --Kostendifferenz zwischen den  ÜbernahmeKosten mit dem neuen Auf-
trag und den alten  ÜbernahmeKosten ohne ihn (ÜbernahmeKostenAlt).
        ÜbernahmeKosten:=ÜbernahmeKosten-ÜbernahmeKostenAlt
        end

        --Die Zielliste mit einem Eintrag wird auf die ZiellisteVergabe
kopiert.
        self.~.Zielliste.kopiereInhaltNach(self.~.ZiellisteVergabe)
        --Der neue Auftrag wird auf die ZiellisteVergabe hinzugefügt.
        var intZähler1:integer:=self.~.ZiellisteVergabe.yDim+1
        self.~.ZiellisteVergabe[1,intZähler1]:=PS_X
        self.~.ZiellisteVergabe[2,intZähler1]:=Auftrag
        self.~.ZiellisteVergabe[3,intZähler1]:=X_von
        self.~.ZiellisteVergabe[4,intZähler1]:=PS_Y
        self.~.ZiellisteVergabe[5,intZähler1]:=Y_nach
        self.~.ZiellisteVergabe[6,intZähler1]:=Bedarfszeit
        self.~.ZiellisteVergabe[7,intZähler1]:=Auftragsart

--Fall 4:
        --Gabelstapler hat einen Auftrag, Zielliste hat zwei oder mehr
Aufträge.
        --AktuellerAuftrag = 1 und LängeZielliste > 1
elseif AktuellerAuftrag=1 and LängeZielliste>1
```

```
    X_now:=self.~.Aufruf
    Y_now:=self.~.AktuellerAuftragNach
    X_1:=self.~.Zielliste[3,1] --Quelle von den einzigen Auftrag auf der
Zielliste
    Y_1:=self.~.Zielliste[5,1] --Senke von den einzigen Auftrag auf der
Zielliste
    --Berechnung der bestehenden ÜbernahmeKosten (bei der jetzigen Situa-
tion am Gabelstapler mit der jetzigen Auftrgsverteilung auf der Ziel-
liste, ohne den neuen Auftrag).
    var ÜbernahmeKostenInit:real:=0
    var ÜbernahmeKostenInitT1:real:=0
    var ÜbernahmeKostenInitT2:real:=0
    var dauerInit:time
    var dauerInitT1:time
    var dauerInitT2:time
    var dauerAbsInit:datetime
    var dauerAbsInitT1:datetime
    --var dauerAbsInitT2:datetime

    LängeZielliste:=self.~.Zielliste.yDim

    --Die bestehenden ÜbernhameKosten setzen sich zusammen aus den Kosten,
von aktuellen Standort X_0 bis zu den ersten Auftrag auf der Zielliste
(seiner Senke). UND
    --Den Rest der Kosten auf der Zielliste (von Senke vom ersten Auftrag
auf der Zielliste bis zum ende derselben Zielliste).

    --Initiale ÜbernahmeKosten, sowie die entsprechendene Dauer von aktu-
ellen Standort X_0 bis Senke von 1. Auftrag auf der Zielliste.
        if Status="GabelstaplerLeer"
            ÜbernahmeKostenInit:=T_Distanzmatrix[#X_0,#X_now]+T_Distanz-
matrix[#X_now,#Y_now]+T_Distanzmatrix[#Y_now,#X_1]+T_Distanz-
matrix[#X_1,#Y_1]
            dauerInit:=((T_Distanzmatrix[#X_0,#X_now]+T_Distanz-
matrix[#X_now,#Y_now]+T_Distanzmatrix[#Y_now,#X_1]+T_Distanz-
matrix[#X_1,#Y_1])/2)+4*30
            dauerAbsInit:=tAktl+dauerInit

            ÜbernahmeKostenInitT1:=ÜbernahmeKostenInit
            dauerInitT1:=dauerInit
            dauerAbsInitT1:=dauerAbsInit
        else --Status="GabelstaplerBeladen"
            ÜbernahmeKostenInit:=T_Distanzmatrix[#X_0,#Y_now]+T_Distanz-
matrix[#Y_now,#X_1]+T_Distanzmatrix[#X_1,#Y_1]
            dauerInit:=((T_Distanzmatrix[#X_0,#Y_now]+T_Distanz-
matrix[#Y_now,#X_1]+T_Distanzmatrix[#X_1,#Y_1])/2)+3*30
            dauerAbsInit:=tAktl+dauerInit

            ÜbernahmeKostenInitT1:=ÜbernahmeKostenInit
            dauerInitT1:=dauerInit
            dauerAbsInitT1:=dauerAbsInit
        end
    --Initiale ÜbernahmeKosten, sowie die entsprechendene Dauer für die
Zielliste ab den zweiten Auftrag (T2) + die vorherigen Initial Übernahme-
Kosten
        var intZähler2:integer

        for intZähler2:=2 to LängeZielliste
            var A:integer:=self.~.Zielliste[3,intZähler2]   --Quelle n auf
der Zielliste
            var B:integer:=self.~.Zielliste[5,intZähler2]   --Senke n auf
der Zielliste
            var C:integer:=self.~.Zielliste[5,intZähler2-1]     --Senke n-
```

```
1 auf der Zielliste

            ÜbernahmeKostenInitT2:=T_Distanzmatrix[#A,#C]
            dauerInitT2:=((T_Distanzmatrix[#A,#C]+T_Distanz-
matrix[#A,#B])/2)+2*30

            ÜbernahmeKostenInit:=ÜbernahmeKostenInit+ÜbernahmeKostenInitT2
            dauerInit:=dauerInit+dauerInitT2
            dauerAbsInit:=dauerAbsInit+dauerInitT2
        next

        --ZiellisteTemporär ist eine Kopie der Zielliste.
        self.~.Zielliste.kopiereInhaltNach(self.~.ZiellisteTemporär)
        --Der neue Auftrag wird auf die ZiellisteTemporär hinzugefügt.
        var LängeZiellisteTemporär:integer
        LängeZiellisteTemporär:=self.~.ZiellisteTemporär.yDim+1 --Länge-
ZiellisteTemporär+1

        self.~.ZiellisteTemporär[1,LängeZiellisteTemporär]:=PS_X
        self.~.ZiellisteTemporär[2,LängeZiellisteTemporär]:=Auftrag
        self.~.ZiellisteTemporär[3,LängeZiellisteTemporär]:=X_von
        self.~.ZiellisteTemporär[4,LängeZiellisteTemporär]:=PS_Y
        self.~.ZiellisteTemporär[5,LängeZiellisteTemporär]:=Y_nach
        self.~.ZiellisteTemporär[6,LängeZiellisteTemporär]:=Bedarfszeit
        self.~.ZiellisteTemporär[7,LängeZiellisteTemporär]:=Auftragsart

        --REIHENFOLGE-OPTIMIERUNGSPROZESS

        --ZiellisteVergabe wird mit den Aufträgen aus ZiellisteTemporär
befüllt.
        --Der 1. Auftrag ist fixiert, und er wird überschrieben.
        self.~.ZiellisteTemporär.kopiereBereich-
Nach({1,1}..{7,1},self.~.ZiellisteVergabe, 1,1)
        --Die Methode kopiereBereichNach kopiert den Bereich der mit
<Pfad> bezeichneten Quelltabelle in einen Bereich einer Zieltabelle.
        --<Pfad>.kopiereBereichNach(Quellbereich:listrange, Zieltab-
elle:any, ZielSpalte:integer, ZielZeile:integer)

        --Der 1. Auftrag wird aus der ZiellisteTemporär gelöscht.
        self.~.ZiellisteTemporär.entferneZeile(1)

        var intZähler3:integer
        var intZähler4:integer
        var letzteSenke:integer
        var letzteBedarfszeit:datetime
        var kürzesteDistanz:real
        var maximaleDistanz:real
        var nähersterAuftragID:string
        var näehersteQuelle:integer
        var näehersteSenke:integer
        var näehersteQuelleObj:object
        var näehersteSenkeObj:object
        var näehersteAuftragIDZeile:integer
        var näehersteBedarfszeit:datetime
        var näehersteAuftragsart:string

        --Neue ÜbernahmeKosten bestimmen

        --Der konstante Anteil der Kosten ist schon ausgerechnet und ent-
sprechend gespeichert worden.
        --Initiale ÜbernahmeKosten, sowie die entsprechendene Dauer von
aktuellen Standort X_0 bis Senke von 1. Auftrag auf der Zielliste.
        --Diese Werte sind konstant, da die von aktuellen Standort bis hin
```

```
zu Senke des ersten Auftrags auf der Zielliste berechnet worden sind.
        --Und dieser Teil bleibt bei der Reihenfolgenoptimerung unverän-
dert!
        /*print ÜbernahmeKostenInitT1
        print dauerInitT1
        print dauerAbsInitT1*/

        var ÜbernahmeKostenNeu:real:=ÜbernahmeKostenInitT1
        var dauerNeu:time:=dauerInitT1
        var dauerAbsNeu:datetime:=dauerAbsInitT1

        var penalty:real:=1
        var penaltyAktl:real
        var PENALTY_FAKTOR:real:=0.1

        --Variablen, die für Aufträge, die Auslagerungen sind, wichtig
sind.
        var PENALTY_FAKTOR_AUSLAGERUNGEN:real:=0.5
        var Deadlinezeit:time                               --Deadlinezeit
ist gleich der Bedarfszeit substrahiert von der aktuellen Zeit und Auftra-
gerledigungsdauer.
        var ZEITLIMIT:time:=str_to_time("00:45:00.0000")    --ZEITLIMIT
wird auf 30 Minuten gestellt.
        var ZEITLIMIT_PENALTY:time:=str_to_time("00:15:00.0000")
        var tDiff:time
        var tDiffSekunden:real

        --Variablen, für die Zeitdifferenzen.
        var tDiffAUSLAGERUNG:time:=str_to_time("00:00:00.0000")
        VAR tDiffEINLAGERUNG:time:=str_to_time("01:00:00.0000")
        var tDiffUMLAGERUNGoderLEERGUT:time:=str_to_time("00:10:00.0000")

        while self.~.ZiellisteTemporär.yDim > 0 loop
        --While Schleife wird durchgeführt, solange die Länge der Ziel-
listeTemporär > 0 ist.
            intZähler3:=self.~.ZiellisteVergabe.yDim            --Die
letzte Reihe auf der ZiellisteVergabe
            letzteSenke:=self.~.ZiellisteVergabe[5,intZähler3]  --Die
letzte Senke  auf der ZiellisteVergabe
            maximaleDistanz:=(.Modelle.Modell.T_Distanz-
matrix.max({1,1}..{51,51})*1.4)+1 -- max. mögliche Distanz aus der Dis-
tanzmatrix (+20%)
            kürzesteDistanz:=maximaleDistanz

                --For Schleife wird durchgeführt von 1 bis zur Länge der
ZiellisteTemporär
                for intZähler4:=1 to self.~.ZiellisteTemporär.yDim
                    var D:integer:=self.~.ZiellisteTemporär[3,intZähler4]
--Quellen auf der ZiellisteTemporär
                    var E:integer:=letzteSenke
--letzte Senke aus ZiellisteVergabe
                    var F:integer:=self.~.ZiellisteTemporär[5,intZähler4]
--Senken auf der ZiellisteTemporär

                    var AuftragsBedarfszeit:=self.~.ZiellisteTempo-
rär[6,intZähler4]     --Die aktuelle Bedarfszeit des Auftrags
                    var AktlAuftragsart:=self.~.ZiellisteTempo-
rär[7,intZähler4]         --Die aktuelle Auftragsart des Auftrags
                    var Distanz:real
--Abstand zw. der möglichen neuen Quelle und der letzten Senke in Ziellis-
teVergabe
                    var AuftragsÜbernahmeAbs:datetime
                    var AuftragsÜbernahme:time
                    Distanz:=.Modelle.Modell.T_Distanzmatrix[#D,#E]
                    AuftragsÜbernahme:=((T_Distanzmatrix[#E,#D]+T_Distanz-
matrix[#D,#F])/2)+2*30
                    AuftragsÜbernahmeAbs:=dauerAbsNeu+AuftragsÜbernahme

                    if AktlAuftragsart="Einlagerung" or AktlAuftrags-
art="Umlagerung"
                        Distanz:=Distanz*1.4
                    elseif AktlAuftragsart="Auslagerung"
                        Distanz:=Distanz
                    end

                    if AktlAuftragsart="Auslagerung"
                        if Distanz < kürzesteDistanz
                            kürzesteDistanz:=Distanz
                            nähersteAuftragIDZeile:=intZähler4

                            Deadlinezeit:=AuftragsBedarfszeit-Auftrags-
ÜbernahmeAbs
                            if Deadlinezeit <= ZEITLIMIT
                                kürzesteDistanz:=(kürzesteDistanz*Dead-
linezeit)/ZEITLIMIT
                            end

                            tDiff:=AuftragsÜbernahmeAbs-(AuftragsBedarfs-
zeit-ZEITLIMIT_PENALTY)
                            tDiffSekunden:=time_to_num(tDiff)
                            if tDiff < tDiffAUSLAGERUNG
                                penaltyAktl:=0
                            else
                                penaltyAktl:=PENALTY_FAKTOR_AUSLAGERUN-
GEN*(tDiffSekunden/60)-PENALTY_FAKTOR_AUSLAGERUNGEN
                            end
                        end
                    elseif AktlAuftragsart="Einlagerung"
                        if Distanz< kürzesteDistanz
                            kürzesteDistanz:=Distanz
                            nähersteAuftragIDZeile:=intZähler4
                            tDiff:=AuftragsÜbernahmeAbs-AuftragsBedarfs-
zeit
                            tDiffSekunden:=time_to_num(tDiff)
                            if tDiff < tDiffEINLAGERUNG
                                penaltyAktl:=0
                            else
                                penaltyAktl:=PENALTY_FAKTOR*(tDiffSekun-
den/60)-PENALTY_FAKTOR
                            end
                        end
                    elseif AktlAuftragsart="Umlagerung" or AktlAuftrags-
art="Leergut"
                        if Distanz < kürzesteDistanz
                            kürzesteDistanz:=Distanz
                            nähersteAuftragIDZeile:=intZähler4
                            tDiff:=AuftragsÜbernahmeAbs-AuftragsBedarfs-
zeit
                            tDiffSekunden:=time_to_num(tDiff)
                            if tDiff < tDiffUMLAGERUNGoderLEERGUT
                                penaltyAktl:=0
                            else
                                penaltyAktl:=PENALTY_FAKTOR*(tDiffSekun-
den/60)-PENALTY_FAKTOR
                            end
                        end
                    end
                next
                penalty:=penalty+penaltyAktl

                self.~.ZiellisteTemporär.kopiereBereichNach({1,näherste-
AuftragIDZeile}..{7,nähersteAuftragIDZeile},self.~.ZiellisteVergabe,
1,intZähler3+1)
                --Die Methode kopiereBereichNach kopiert den Bereich der
mit <Pfad> bezeichneten Quelltabelle in einen Bereich einer Zieltabelle.
                --<Pfad>.kopiereBereichNach(Quellbereich:listrange,
Zieltabelle:any, ZielSpalte:integer, ZielZeile:integer)
                self.~.ZiellisteTemporär.entferneZeile(nähersteAuftragID-
Zeile)

                --Die ÜbernahmeKosten müssen aktualisiert werden.
                var intZähler5:integer:=self.~.ZiellisteVergabe.yDim    --
Die letzte Reihe auf der ZiellisteVergabe
                var G:integer:=self.~.ZiellisteVergabe[3,intZähler5]
--letzte Quelle auf ZiellisteVergabe
                var H:integer:=self.~.ZiellisteVergabe[5,intZähler5]
--letzte Senke aus ZiellisteVergabe
                var I:integer:=self.~.ZiellisteVergabe[5,intZähler5-1]
--vorletzte Senke aus ZiellisteVergabe

                ÜbernahmeKostenNeu:=ÜbernahmeKostenNeu+T_Distanz-
matrix[#I,#G]
                dauerNeu:=dauerNeu+((T_Distanzmatrix[#I,#G]+T_Distanz-
matrix[#G,#H])/2)+2*30
                dauerAbsNeu:=dauerAbsNeu+dauerNeu

        end

        if ÜbernahmeKosten=1000000
            ÜbernahmeKosten:=ÜbernahmeKosten
        else
            ÜbernahmeKosten:=(ÜbernahmeKostenNeu-ÜbernahmeKostenInit)*pe-
nalty
        end
end

result:=ÜbernahmeKosten
```

B – Detailed Layout Overview

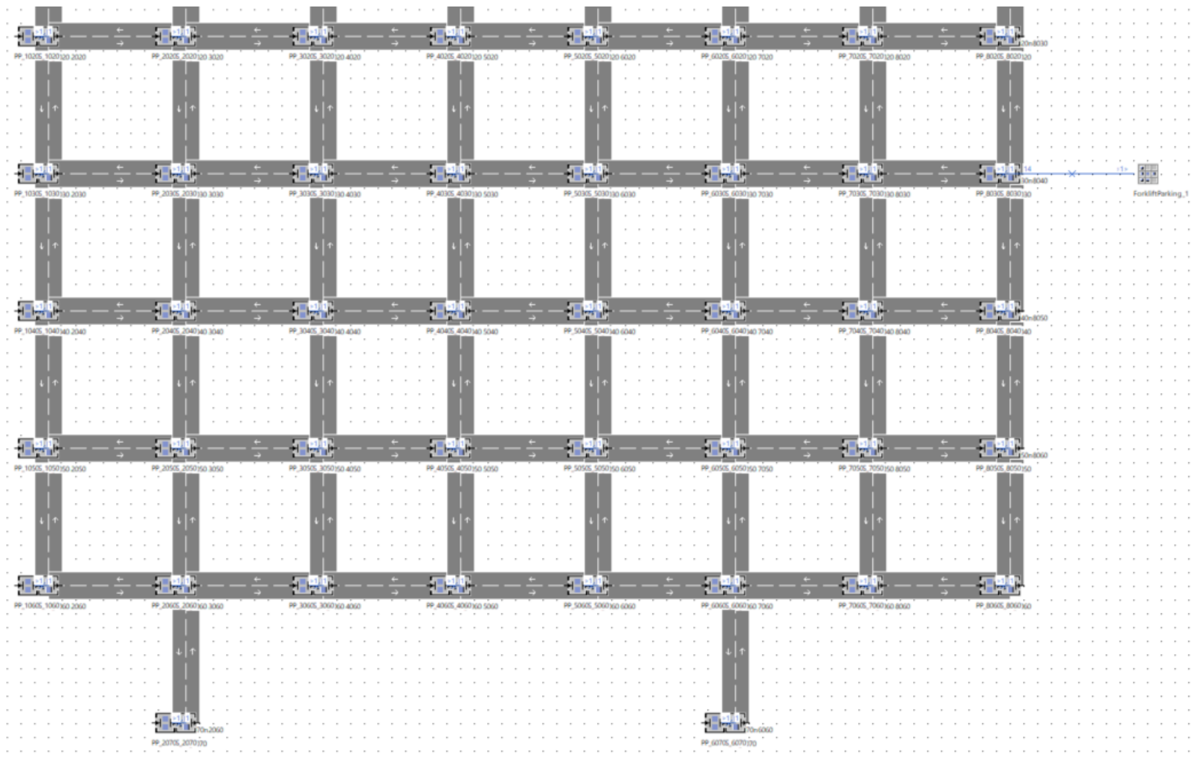


Figure B- 1: Detailed Simulation Layout.

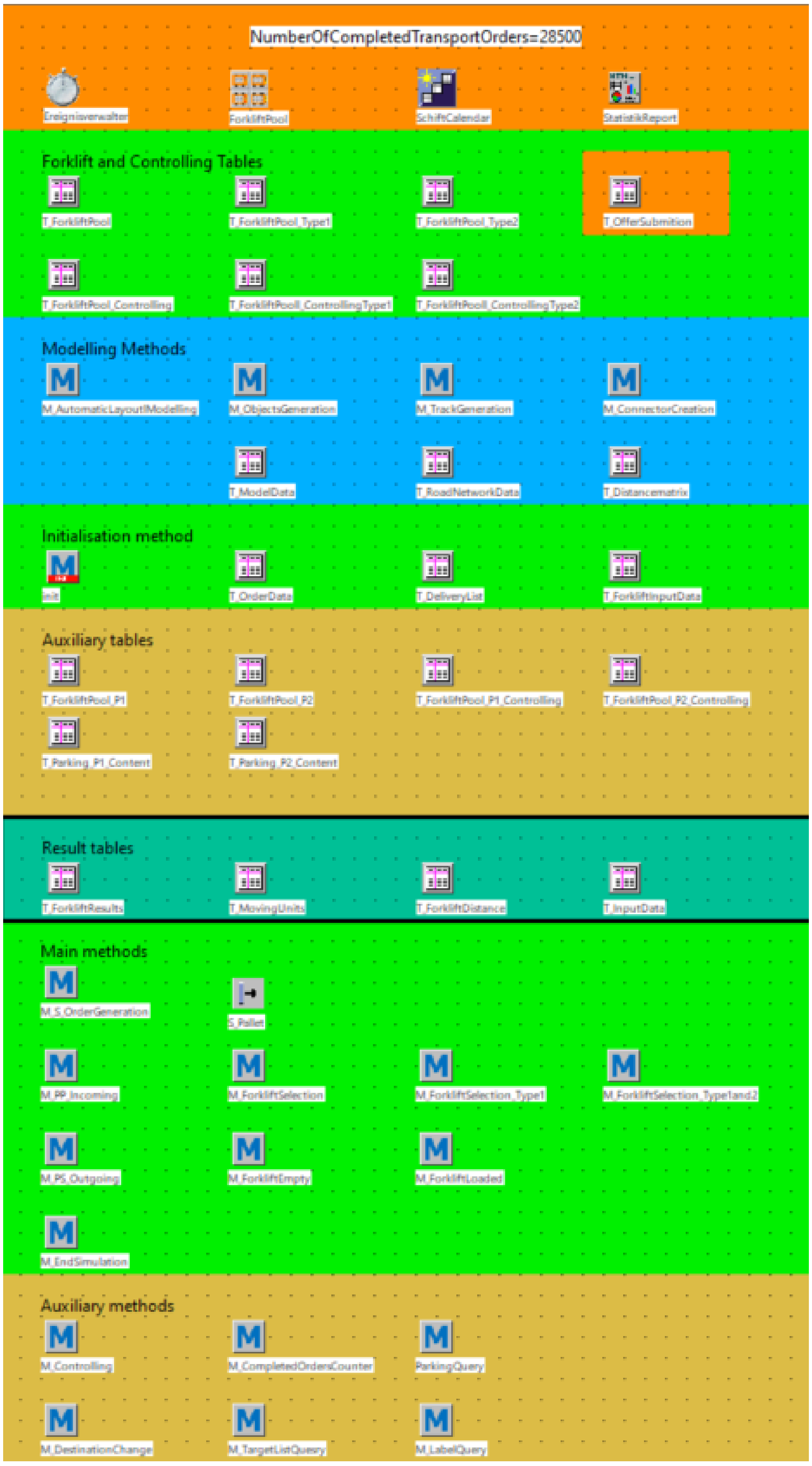


Figure B- 2: Detailed Simulation Layout - used User Objects.

C – MS Excel Evaluation Tool

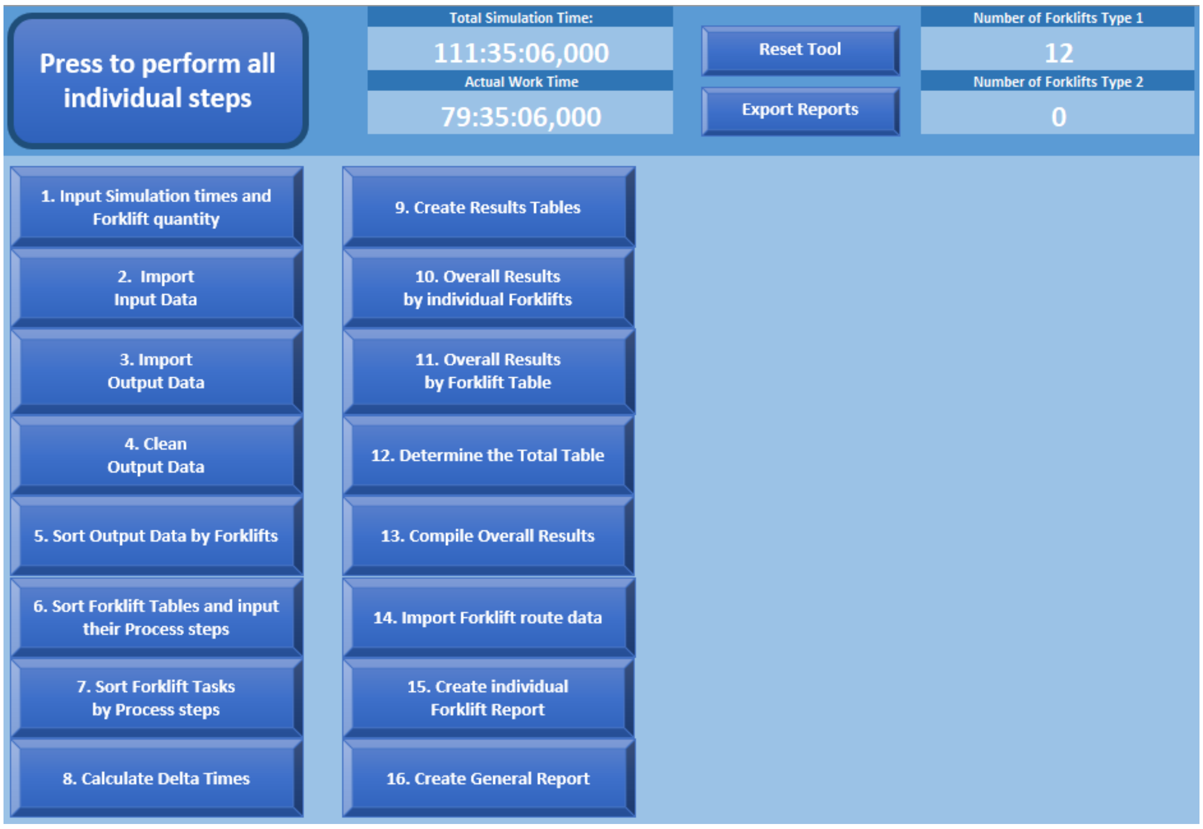
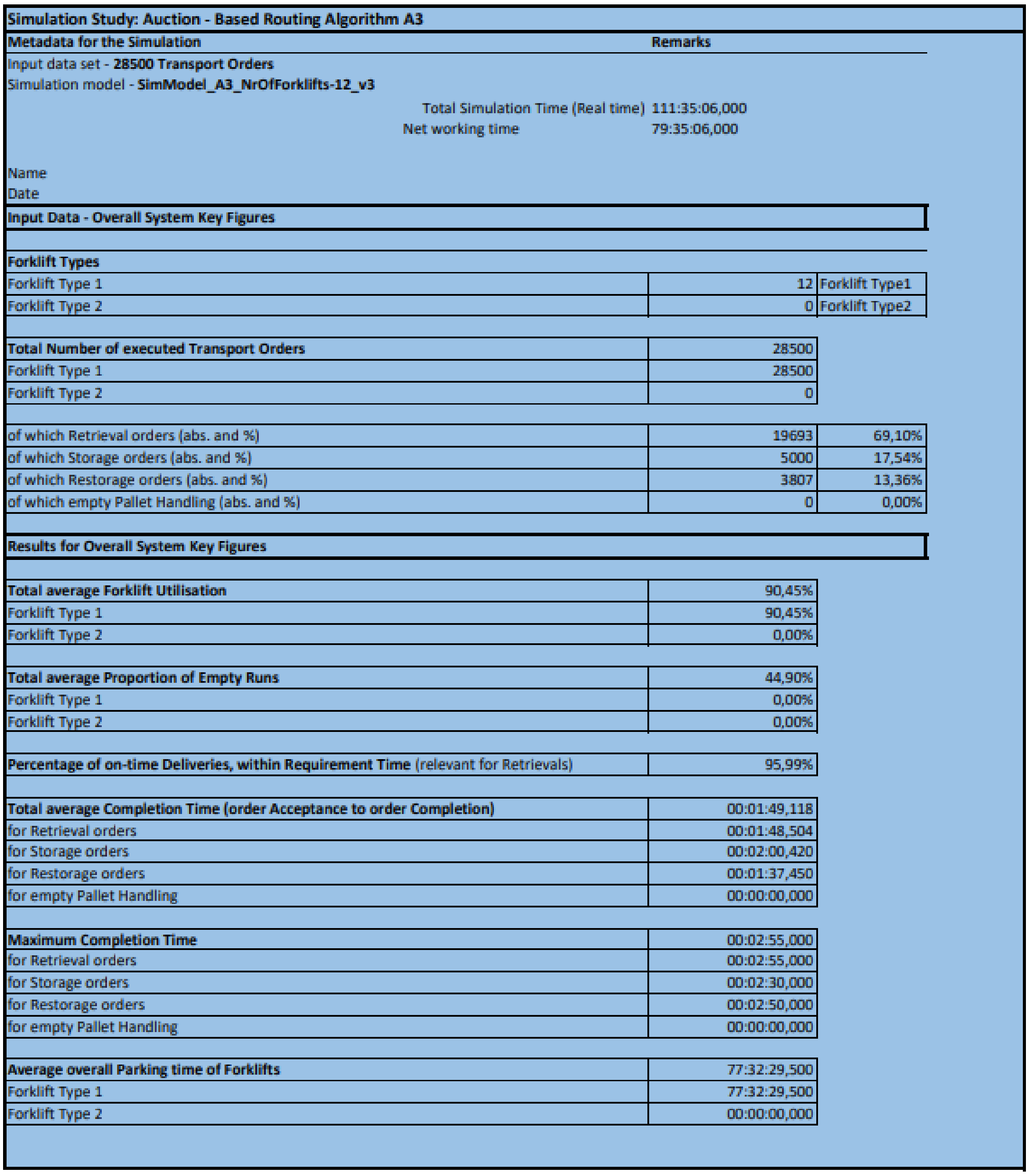
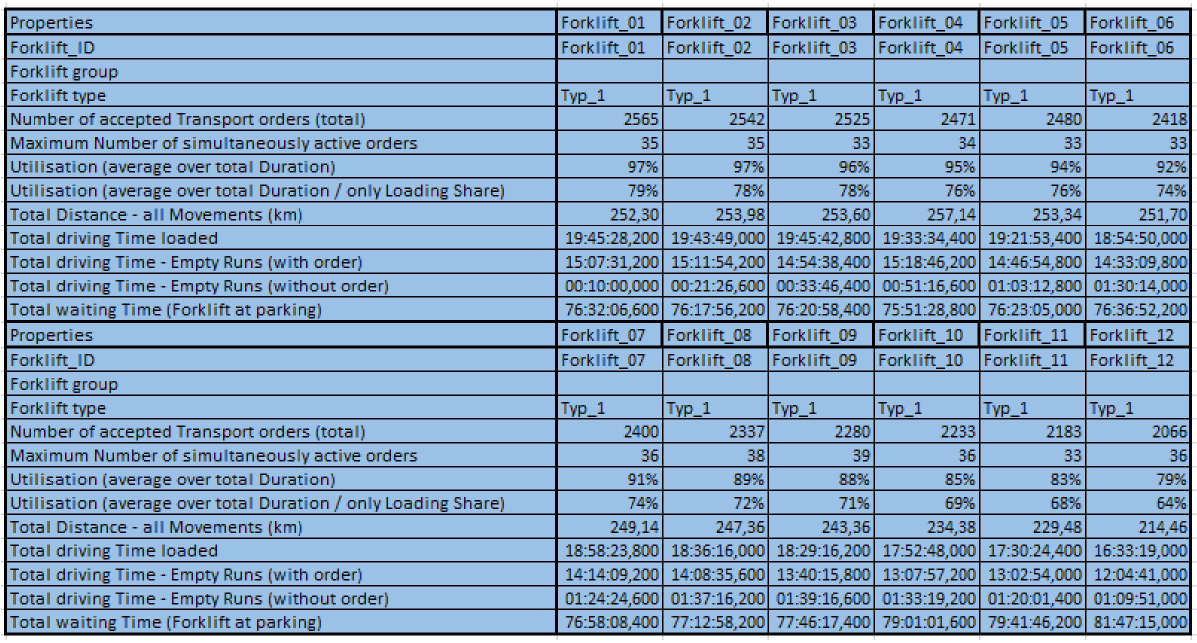

Press to perform all individual steps
Total Simulation Time:
111:35:06,000
Actual Work Time
79:35:06,000
Reset Tool
Export Reports
Number of Forklifts Type 1
12
Number of Forklifts Type 2
0
1. Input Simulation times and Forklift quantity
2. Import Input Data
3. Import Output Data
4. Clean Output Data
5. Sort Output Data by Forklifts
6. Sort Forklift Tables and input their Process steps
7. Sort Forklift Tasks by Process steps
8. Calculate Delta Times
9. Create Results Tables
10. Overall Results by individual Forklifts
11. Overall Results by Forklift Table
12. Determine the Total Table
13. Compile Overall Results
14. Import Forklift route data
15. Create individual Forklift Report
16. Create General Report

Figure C- 1: Evaluation Tool – Interface.



Simulation Study: Auction - Based Routing Algorithm A3		
Metadata for the Simulation	**Remarks**	
Input data set - **28500 Transport Orders**		
Simulation model - **SimModel_A3_NrOfForklifts-12_v3**		
Total Simulation Time (Real time)	111:35:06,000	
Net working time	79:35:06,000	
Name		
Date		
Input Data - Overall System Key Figures		
Forklift Types		
Forklift Type 1	12	Forklift Type1
Forklift Type 2	0	Forklift Type2
Total Number of executed Transport Orders	28500	
Forklift Type 1	28500	
Forklift Type 2	0	
of which Retrieval orders (abs. and %)	19693	69,10%
of which Storage orders (abs. and %)	5000	17,54%
of which Restorage orders (abs. and %)	3807	13,36%
of which empty Pallet Handling (abs. and %)	0	0,00%
Results for Overall System Key Figures		
Total average Forklift Utilisation	90,45%	
Forklift Type 1	90,45%	
Forklift Type 2	0,00%	
Total average Proportion of Empty Runs	44,90%	
Forklift Type 1	0,00%	
Forklift Type 2	0,00%	
Percentage of on-time Deliveries, within Requirement Time (relevant for Retrievals)	95,99%	
Total average Completion Time (order Acceptance to order Completion)	00:01:49,118	
for Retrieval orders	00:01:48,504	
for Storage orders	00:02:00,420	
for Restorage orders	00:01:37,450	
for empty Pallet Handling	00:00:00,000	
Maximum Completion Time	00:02:55,000	
for Retrieval orders	00:02:55,000	
for Storage orders	00:02:30,000	
for Restorage orders	00:02:50,000	
for empty Pallet Handling	00:00:00,000	
Average overall Parking time of Forklifts	77:32:29,500	
Forklift Type 1	77:32:29,500	
Forklift Type 2	00:00:00,000	

Figure C- 2: Evaluation Tool – Overall Report.



Properties	Forklift_01	Forklift_02	Forklift_03	Forklift_04	Forklift_05	Forklift_06	Forklift_07	Forklift_08	Forklift_09	Forklift_10	Forklift_11	Forklift_12
Forklift_ID	Forklift_01	Forklift_02	Forklift_03	Forklift_04	Forklift_05	Forklift_06	Forklift_07	Forklift_08	Forklift_09	Forklift_10	Forklift_11	Forklift_12
Forklift group												
Forklift type	Typ_1	Typ_1	Typ_1	Typ_1	Typ_1	Typ_1	Typ_1	Typ_1	Typ_1	Typ_1	Typ_1	Typ_1
Number of accepted Transport orders (total)	2565	2542	2525	2471	2480	2418	2400	2337	2280	2233	2183	2066
Maximum Number of simultaneously active orders	35	35	33	34	33	33	36	38	39	36	33	36
Utilisation (average over total Duration)	97%	97%	96%	95%	94%	92%	91%	89%	88%	85%	83%	79%
Utilisation (average over total Duration / only Loading Share)	79%	78%	78%	76%	76%	74%	74%	72%	71%	69%	68%	64%
Total Distance - all Movements (km)	252,30	253,98	253,60	257,14	253,34	251,70	249,14	247,36	243,36	234,38	229,48	214,46
Total driving Time loaded	19:45:28,200	19:43:49,000	19:45:42,800	19:33:34,400	19:21:53,400	18:54:50,000	18:58:23,800	18:36:16,000	18:29:16,200	17:52:48,000	17:30:24,400	16:33:19,000
Total driving Time - Empty Runs (with order)	15:07:31,200	15:11:54,200	14:54:38,400	15:18:46,200	14:46:54,800	14:33:09,800	14:14:09,200	14:08:35,600	13:40:15,800	13:07:57,200	13:02:54,000	12:04:41,000
Total driving Time - Empty Runs (without order)	00:10:00,000	00:21:26,600	00:33:46,400	00:51:16,600	01:03:12,800	01:30:14,000	01:24:24,600	01:37:16,200	01:39:16,600	01:33:19,200	01:20:01,400	01:09:51,000
Total waiting Time (Forklift at parking)	76:32:06,600	76:17:56,200	76:20:58,400	75:51:28,800	76:23:05,000	76:36:52,200	76:58:08,400	77:12:58,200	77:46:17,400	79:01:01,600	79:41:46,200	81:47:15,000

Figure C- 3: Evaluation Tool – Individual Forklifts Report (Note – This report is displayed in two parts due to space requirements).