

Order Fulfillment Optimization in Automated Warehouses

by

Jiwoo Christian Suh

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Mechanical and Mechatronics Engineering

Waterloo, Ontario, Canada, 2025

© Jiwoo Christian Suh 2025

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

In warehousing, order batching is one of the most popular strategies for optimizing order fulfillment as it groups similar orders into the same batch to optimize picking. The order similarity can be determined based on item locations, availability, and order compositions. The objectives include minimizing travel time, maximizing the number of picked items, and maximizing simultaneous multi-order processing. In this thesis, we study the order fulfillment problem in automated warehouses and propose an order fulfillment heuristic method that to minimize the number of required pick-up sequences to fulfill given order lists by integrating various independent order fulfillment techniques. Three independent algorithms are modified and integrated: (1) FP-Growth-based Association Rule Mining, (2) Order Batching using Similarities Between Orders, and 3) A Hybrid of Public and Personal Item Storage. The resulting heuristic approach is capable of finding optimal solutions when compared to exact results based on Integer Programming. Additionally, a custom-built Python simulation platform is created and run to prove the scalability of the devised algorithm. The Python simulation platform has been further developed into an ROS- and Gazebo-communicable simulation platform for more visualized and intuitive simulation results. Based on the simulation results involving 2000 orders and 1000 items, the algorithm reduced the total number of required pick-up sequences by approximately 50% in comparison to traditional First-Come, First-Served approach.

Acknowledgments

First and foremost, I would like to thank Professor Amir Khajepour, who provided me with valuable guidance and advice throughout my project and offered encouragement when challenges arose. Before and even during my project, I was unsure what truly interested me and what I wanted to do after graduation. However, Professor Khajepour helped me realize my true passion for engineering and the importance of that realization. I am deeply grateful for the clarity and inspiration he provided.

I would also like to thank Professor Samir Elhedhli, who guided me in management engineering and optimization techniques. With his support, I established a solid foundation in optimization problem-solving.

Additionally, I would like to thank the members of the MVS Lab for giving me the privilege of learning from co-workers throughout my time in the lab. In particular, I would like to thank Chen Sun, Yukon Lu, Ladan Khoshnevisan, Jiaming Zhong, Frank Yang, and Changye Ma for their mentorship and collaboration on the Autonomous Warehouse Optimization project.

Finally, I thank my family and friends for their unwavering encouragement and support.

Dedication

My God, family, and friends, thank you for always being by my side and inspiring me to do what I love.

Table of Contents

Author’s Declaration	ii
Abstract	iii
Acknowledgments	iii
Dedication	v
List of Figures	viii
List of Tables	x
Chapter 1. Introduction	1
1.1 Motivation	1
1.2 Objectives and Contributions	2
1.3 Outline	4
Chapter 2. Literature Review	5
2.1 Introduction	5
2.2 Various Order Fulfillment Optimization Approaches	5
2.2.1 Decentralized and Dynamic Slotting Item Storages	5
2.2.2 Zone Picking	7
2.2.3 Real-Time Control (Algorithms)	8
2.2.4 Real-Time Control (Digital Twin)	9
2.2.5 Automated Storage and Retrieval Systems (AS/RS)	11
2.2.6. Order Batching	11
2.3 Exact and Heuristic Methods for Order Fulfillment Optimization	13
2.4.3 Association Rule Mining with Apriori Algorithm and FP-Growth	17
a) Apriori Algorithm-based Association Rule Mining (AA-ARM)	17
b) FP-Growth-based Association Rule Mining (FP-ARM)	18
2.5 Summary	18

Chapter 3. Order Fulfillment Optimization Algorithm	20
3.1 Warehouse Environment.....	20
3.2 Order Fulfillment Optimization Algorithm.....	23
3.2.1 FP-Growth-based Association Rule Mining.....	24
3.2.2 Order Shuffling.....	32
3.2.3 Order Fulfillment Optimization with Buffer Bins	33
3.2.4 Extension to Multi-Operator Scenario	48
Chapter 4. Simulation Results & Analysis.....	57
4.1 Policies Evaluation.....	57
4.2 Impact of Change in Parameters.....	61
4.3 Simulations in ROS & Gazebo	65
Chapter 5. Conclusion and Future Work.....	72
5.1 Conclusion	72
5.2 Future Work	73
Reference	76

List of Figures

Figure 1. (a) The layout of an exemplary warehouse, (b) the layout of the exemplary warehouse with three picking zones. [16]	8
Figure 2. The physical-to-virtual and virtual-to-physical twinning process [26].....	10
Figure 3. Entire Warehouse Layout.....	22
Figure 4. Item Storage Shelf with a cable robot and pick-up & return conveyors.....	23
Figure 5. Order Fulfillment Optimization Algorithm.....	24
Figure 6. Constructed Example FP-Tree	28
Figure 7. Operator Station Layout.....	34
Figure 8. Algorithm for Optimal Pick-Up Sequences	35
Figure 9. Warehouse Example for Policy 1 & Policy 2	36
Figure 10. Algorithm of Policy 1: Order-Based Buffer Area.....	37
Figure 11. Algorithm of Policy 2: Item-Based Buffer Area.....	46
Figure 12. Multi-Operator Scenario Warehouse Example	49
Figure 13. Algorithm of Multi-Operator Scenario	51
Figure 14. Tournament Selection, 2-Point Crossover, and Mutation in GA	53
Figure 15. Demonstration of Performance for Each Proposed Policy	57
Figure 16. Simulation Results for 2000 Orders & 1000 Items with Uniform Item Distribution	59
Figure 17. Simulation Results for 2000 Orders & 1000 Items with Biased Item Distribution	60
Figure 18. Impact of Change in Beta on Policy 1 and Policy 2	62
Figure 19. Impact of Change in Alpha on Policy 1 and Policy 2.....	63
Figure 20. Impact of Change in Number of Buffer Bins on Policy 1 and Policy 2	64
Figure 21. Impact of Change in Beta on Policy 1 and Policy 2	65
Figure 22. RQT Graph from Gazebo and ROS Simulation	69
Figure 23. Fine-Tuning of Angular Velocity using Camera	70
Figure 24. a) AGV Travel without Camera, b) AGV Travel with Camera.....	70

Figure 25. ROS & Gazebo Simulations for Warehouse Operations a) Initial State, b) Middle State 1, c) AGVs in Buffer Zone, d) Middle State 2 71

List of Tables

Table 1. Example Order-Item List.....	25
Table 2. Item-Order List	26
Table 3. Reorganized Item-Order List	27
Table 4. Conditional Pattern Bases	28
Table 5. Conditional FP-Trees	29
Table 6. Similarities between Every Order-Pair	30
Table 7. Order List for Policy 1 & Policy 2 Examples	40
Table 8. Shuffled Order List for Policy 1 & Policy 2 Examples	40
Table 9. Order List after Item Searching for the First Three Shelves.....	42
Table 10. Total Pick-Ups Required to Fulfill the Order List with Policy 1	43
Table 11. Max. Number in One Buffer Bin for Each Item	47
Table 12. Total Pick-Ups Required to Fulfill the Order List with Policy 2.....	48
Table 13. Order List for Multi-Operator Scenario Example.....	49
Table 14. Order Clusters with Max. Internal Similarities.....	54
Table 15. Shuffled Order List for Multi-Operator Scenario Example	54
Table 16. Total Pick-Ups Required to Fulfill the Order List with Policy 1 in Multi-Operator Scenario	55

Chapter 1

Introduction

1.1 Motivation

Order fulfillment optimization is a critical area of focus that provides diverse benefits to various aspects of warehouse operations. These benefits include not only increased customer satisfaction but also reduced management costs, enhanced productivity, higher flexibility, and responsiveness [1]. Therefore, tremendous research is being conducted to optimize order fulfillment and maximize warehouse throughput. This section introduces three significant challenges in effective order fulfillment optimization.

First, identifying appropriate objectives for optimization models is a challenging task. Current order fulfillment optimization research often focuses on minimizing item retrieval time, Automated Guided Vehicle (AGV) travel time, or order cycle time without considering what ultimately determines warehouse throughput within a specific warehouse layout [2]. Effective order fulfillment optimization, therefore, requires the deliberate establishment of optimization objectives that align with the unique characteristics of the provided warehouse layout.

The second challenge is the lack of scalability of the traditional similarity-based order batching for large-scale applications. In practice, orders in the order list are often grouped into batches based on item location, order priority, or operator workload balance to enhance order fulfillment efficiency [3]. One of the most common and intuitive methods for order fulfillment is identifying similarities between orders [4]. The Apriori Algorithm is a commonly used technique for calculating such similarities. It is widely used in data mining thanks to its strong foundation in Association Rule Mining and relatively simple implementation [5]. However, the traditional Apriori Algorithm has two main drawbacks: (1) It requires scanning the entire database iteratively, and (2) It must generate a vast number of candidate-sets [6]. Therefore, the Apriori Algorithm-based Association Rule Mining is computationally inefficient in large-scale instances dealing with

thousands of orders and items [7]. Notably, the original research that introduced the application of the Apriori Algorithm to order fulfillment optimization conducted simulations involving up to only 300 orders, which are relatively small compared to the scope of this thesis. Furthermore, mathematical models for similarity-based order batching depend highly on their optimization objectives. The second challenge requires more efficient computational methods and the development of corresponding mathematical models with well-defined objective functions and constraints.

The final challenge is the physical separation between zones, which necessitates numerous interactions and results in unnecessary working time. Items are retrieved and transported from item storage to sorting operators to fulfill orders in the list. In most warehouses, the storage zones for items and the sorting operator zones are designated separately [8]. A significant challenge arises from this separation of the zones, which are often located far from each other. This spatial separation increases the working time of AGVs, causing operators to remain idle while waiting for deliveries to arrive.

Although many solutions and studies address these challenges individually, the information is highly scattered and lacks integration. This fragmentation makes it difficult to combine multiple solutions to tackle more than one challenge simultaneously. Even if dividing a large warehouse operational process into several stages to address each challenge independently may yield optimal solutions for individual challenges, it complicates the discovery of globally optimal solutions for overall warehouse throughput because each stage's optimal solution is obtained without considering its impact on other stages.

1.2 Objectives and Contributions

The primary objective of this thesis is to develop an advanced order fulfillment strategy to minimize the number of pick-up sequences required to fulfill the entire order list. This is achieved by seamlessly integrating various independent optimization strategies to obtain optimal harmony and efficiency in warehouse operations. Several existing approaches, such as order batching, association rule mining, and meta-heuristic algorithms, are considered. Additionally, mathematical models of objectives and constraints in a provided warehouse environment are formulated and

solved using integer programming in small instances to prove that the results obtained by the devised heuristic strategy are indeed optimal.

The second objective is to reduce the computational load required to solve order batching problems. Well-defined order batching models are crucial for optimizing order fulfillment by enhancing efficiency and reducing operational complexities. Many traditional order batching approaches, based on Jaccard Similarity, Cosine Similarity, and the Apriori Algorithm for association rule mining, are unsuitable for handling large-scale instances. In this study, the Apriori Algorithm is replaced with a modified FP-Growth Algorithm tailored to solve the order batching problem more efficiently.

The third objective is to analyze the impact of parameter changes on warehouse throughput. Since every warehouse has a unique layout, workload, and environment, it is essential to identify optimal parameter values tailored to each warehouse's specific conditions. The devised order fulfillment strategy is simulated in large-scale instances under various parameter settings. The simulation results are analyzed to understand how each parameter, such as AGV capacity and the maximum number of orders each operator can handle simultaneously, affects warehouse throughput.

This thesis contributes to the field by integrating and refining a set of independent order fulfillment techniques, as well as their adaptation to large-scale instances. A custom simulation platform, created using Python libraries, is utilized to simulate and demonstrate the performance of the devised algorithm under various conditions. Moreover, the impact of parameter changes is examined to gain a deeper understanding of how the devised strategy influences warehouse throughput. This analysis can also identify optimal parameter values under various warehouse environments, such as the capacity of AGVs and the size of operator stations. Finally, linear mathematical models are created to solve the order batching and order fulfillment problems. These models demonstrate the algorithm's ability to find optimal solutions. The findings of this thesis facilitate the future development of mathematical models for searching global optima without relying on computationally expensive and complex nonlinear optimization solvers.

1.3 Outline

Chapter 2 provides essential background information about this thesis's order batching and fulfillment techniques. The literature reviewed in this chapter encompasses various order fulfillment and order batching approaches, including association rule mining for data mining, specifically Apriori-based and FP-Growth-based methods. Additionally, a comparison is made between exact and heuristic methods to solve order fulfillment and order batching problems. This chapter focuses on various strategies developed to address order batching and fulfillment challenges to date. Additionally, it examines current advancements and future directions in research aimed at optimizing order fulfillment.

Chapter 3 describes the warehouse considered in this thesis. Then, two order fulfillment approaches and their corresponding mathematical models, developed through the integration of various traditional methods, are introduced to address the objectives outlined in Chapter 1. The two approaches are distinguished by the role of buffer bins, which is explained in greater detail in Chapter 3. Initially, the approaches are devised for single-operator scenarios and later expanded to multi-operator scenarios for more realistic applications.

Chapter 4 demonstrates the proposed approaches' capability to find optimal solutions by comparing their results with those of mathematical models solved using integer programming. The methods are tested under various environments in a custom-built simulation platform to evaluate their ability to address challenges in order fulfillment optimization, such as computational costs in large-scale instances. Furthermore, the chapter analyzes the impact of changes in various parameters on warehouse throughput.

Chapter 5 concludes this thesis with a summary of its contents and an outline of potential future research directions.

Chapter 2

Literature Review

2.1 Introduction

This chapter reviews the literature on various order fulfillment optimization approaches, focusing on order batching strategies. Additionally, several techniques for identifying similarities between orders, including the Similarity Coefficient, Jaccard, and Cosine Similarity, and Association Rule Mining using the Apriori Algorithm and the FP-growth algorithm, are examined in detail. Then, exact, heuristic, and meta-heuristic approaches to solve order batching and order fulfillment problems are studied and compared. Finally, the chapter reviews the development of meta-heuristic methods to overcome the weaknesses of the original approaches.

2.2 Order Fulfillment Optimization Approaches

The selection of appropriate order fulfillment optimization techniques heavily depends on the specific warehouse environment, including operational methods, warehouse layouts, objectives, conditions, and constraints. Understanding various optimization techniques when devising new order fulfillment strategies tailored to specific warehouse scenarios is beneficial as this enhances efficiency. Minimizing the cost function can be a key objective when processing multiple customer orders. The cost function can be defined as travel distance, elapsed time, or completion time [11]. Conversely, when the problem is defined as a maximization problem, the cost function can be customer satisfaction, system performance, or overall throughput. This section reviews various cost functions and corresponding optimization techniques.

2.2.1 Decentralized and Dynamic Slotting

In many warehouses, item storage zones and sorting zones are designated separately. This separation results in unnecessary item retrieval and transportation time, decreasing operational efficiency. To address this issue, numerous optimization studies have explored more efficient arrangements of item storage to facilitate easier access for sorting operators to the required items. One approach is to extend the public item storage conceptually to sorting operators, enabling them

to have personal access to needed items with shorter item retrieval times. For example, decentralized item storage refers to distributing public item storage strategically across various locations instead of centralizing inventory in a single, large storage zone [12]. This decentralized item storage offers advantages, including increased flexibility, reduced transportation costs, faster item dispatch, and higher delivery frequency. However, it may require more expensive management costs and an initial investment to determine the optimal positions for each storage location and the optimal placement of items. Therefore, a reasonable balance between convenience and tolerable costs must be deeply considered. Felix et al. developed a mathematical model for the Scattered Storage Assignment (SSA) to evaluate the effectiveness of decentralized item storage. They compared its impact on warehouse throughput with traditional centralized storage [13]. In their study, Stock Keeping Units (SKUs), unique identifiers assigned to products to track their inventory in a warehouse, were intentionally distributed across the entire warehouse to allow pickers to access required items more efficiently [14]. The new warehouse layout significantly reduced travel distances and time for the pickers. The findings indicated that SSA saved item-picking time and proved particularly effective in Business-to-Customer (B2C) warehouses characterized by small and frequent orders. However, it also increased the complexity of SKU replenishment, as workers were required to distribute SKUs across multiple storage locations. Furthermore, manual picking without a Warehouse Management System (WMS) in an SSA environment was found to be nearly infeasible. In conclusion, while SSA generally outperforms traditional warehouse layouts by reducing picker travel distances and demonstrating particular suitability for B2C warehouses, it necessitates frequent replenishment and the implementation of a Warehouse Management System (WMS) to maintain operational efficiency.

Extensive research has also been undertaken on the dynamic allocation of SKU locations in item storage. Kofler et al. suggested an “Affinity-Based Slotting (ABS)” policy to enhance operational convenience while minimizing management costs [14]. ABS is an item allocation strategy that locates frequently co-ordered SKUs close to each other within the item storage. It builds on two traditional dynamic slotting methods with a focus on item correlations: (1) random slotting – allocating SKUs in random available locations, and (2) turnover-based slotting – allocating fast-moving items near the pickers or I/O points. Experimental results showed that turnover-based (COI) and affinity-based (PF-PA) slotting notably reduced the total travel time of the pickers and improved overall warehouse throughput.

2.2.2 Zone Picking

Zone picking refers to a task allocation strategy assigning workers to specific zones within a warehouse, where they pick items only from their designated areas [15]. A simple example of zone picking is illustrated in Figure 1. This approach effectively reduces the pickers' travel time. It improves item-picking efficiency, as the zones are designated considering various factors, including item locations, tasks, and the pickers' routes. Ho et al. proposed a “zone picking system,” a framework designed to minimize total order-picking travel distance (TTD) and enhance labor efficiency [16]. Their framework is as follows:

1. Allocations of items to different zones
 - Calculate similarity coefficients, which measure relationships between items based on their co-occurrence in orders.
 - Group items into different zones through clustering algorithms.

2. Application of two storage location policies to the zones
 - Within-Aisle Policy: High-demand items are stored closer to the I/O points within each aisle.
 - Across-Aisle Policy: Items with high demand are stored near the front of the warehouse and distributed across multiple aisles.

3. Order batching and route planning
 - Calculate optimal paths with minimum travel distance for item-picking.

Experimental results demonstrated that the redesigned warehouse reduced TTD by over 50% compared to the original setup.

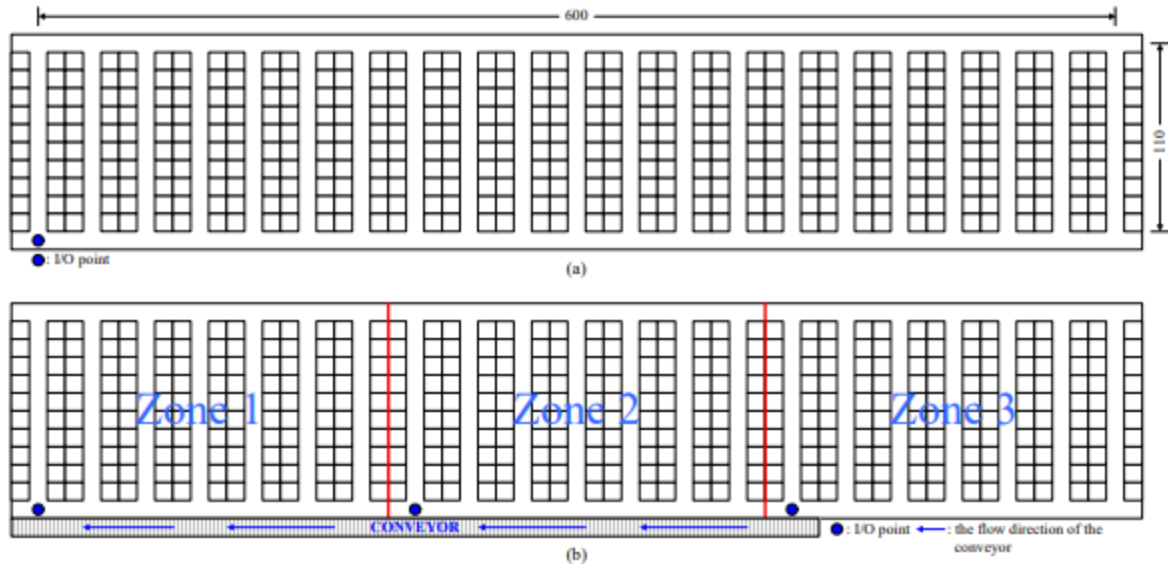


Figure 1. (a) The layout of an exemplary warehouse, (b) the layout of the exemplary warehouse with three picking zones [16]

2.2.3 Real-Time Control (Algorithms)

Warehouses are highly dynamic environments where cooperation among operators is critical under constantly changing conditions. Consequently, real-time applications are crucial for optimizing effective order fulfillment. Due to the dynamic nature of the features and the need for quick decision-making with incomplete information about future orders and inventory replenishment, the initial task assignments are often myopic and require revision as operations progress [17].

To address these challenges, iterative optimization during operations has been studied. Xu et al. developed two heuristic algorithms: **Order Swap** and **SKU Exchange** [18].

- **Order Swap:** Minimization of split shipments by reassigning individual orders
 - For instance, if an order requiring a book and a DVD is split into Warehouses A and B, and Warehouse C becomes available later to fulfill both items, the heuristic reassigns the order to Warehouse C to eliminate the split shipments.
- **SKU Exchange:** Transportation issues resolved by reallocating inventory to minimize shipments.

- For instance, overstocked warehouses can reassign items to understocked ones to balance supplies and demands across the network for better operational efficiency.

Levin et al. introduced a real-time control policy called the “Max-Pressure Control Policy,” which utilizes a metric called “pressure” to iteratively re-prioritize order fulfillment tasks during operations [19]. The pressure is calculated based on the backlogs at each node. Nodes with high pressure indicate many unfulfilled orders and vice versa. A mathematical model describing this policy dynamically reassigns order fulfillment tasks in response to the pressure values iteratively observed in real time.

2.2.4 Real-Time Control and Digital Twin

In addition to the development of algorithms, advanced technologies such as digital twin systems are increasingly adopted for real-time applications. Digital twins create a precise virtual replica of physical systems, enabling simulations to predict future outcomes and implement corresponding actions in the real world [20]. The outcomes in the real world are fed back to the virtual model, allowing for bi-directional data exchange that facilitates real-time monitoring, simulations, and optimization. The simplified operational process of the digital twins is illustrated in Figure 2.

Since 2017, digital twins have expanded their scope from academic research to practical applications in various fields, including industrial operations, healthcare, and urban planning [21]. For example, they are used in traffic control to predict vehicle speeds and traffic flow in real time [22]. In automated warehouses, numerous operational challenges arise from the complex interactions among various entities, including item-retrieving robots, AGVs, and conveyor robots. The utilization of digital twins can mitigate problems such as path planning, vehicle conflicts, and congestion by enabling the timely prediction of warehouse conditions. Wu et al. proposed an AGVs Multi-Objective Dynamic Scheduling (AMODS) method through the implementation of digital twin technology [23]. AMODS dynamically allocates tasks to AGVs in real time. It determines the optimal path, number of AGVs, and their speeds in varying scenarios by predicting the future warehouse status based on historical data.

Digital twins offer more substantial global optimization capabilities than algorithm-based real-time control through monitoring and predictive analytics [24]. Once input values are entered

into the digital twins, they can predict corresponding real-world outcomes through simulations that leverage various sensors and models. This predictive capability enables users to make more informed and confident decisions by being aware of potential risks and implementing proper early responses. However, implementing digital twins is significantly more complicated than implementing algorithm-based real-time control [25]. They require large amounts of data for accurate predictions and are less efficient in resource utilization than algorithm-based approaches. Furthermore, their computational speed, which is critical in real-time control, might be slower than that of the algorithm-based approach, and they are highly data-dependent and vulnerable to the availability and quality of the data. Therefore, it is essential to consider both the quality and quantity of data available in the warehouse when selecting a real-time control approach, such as between algorithm-based methods and digital twins.

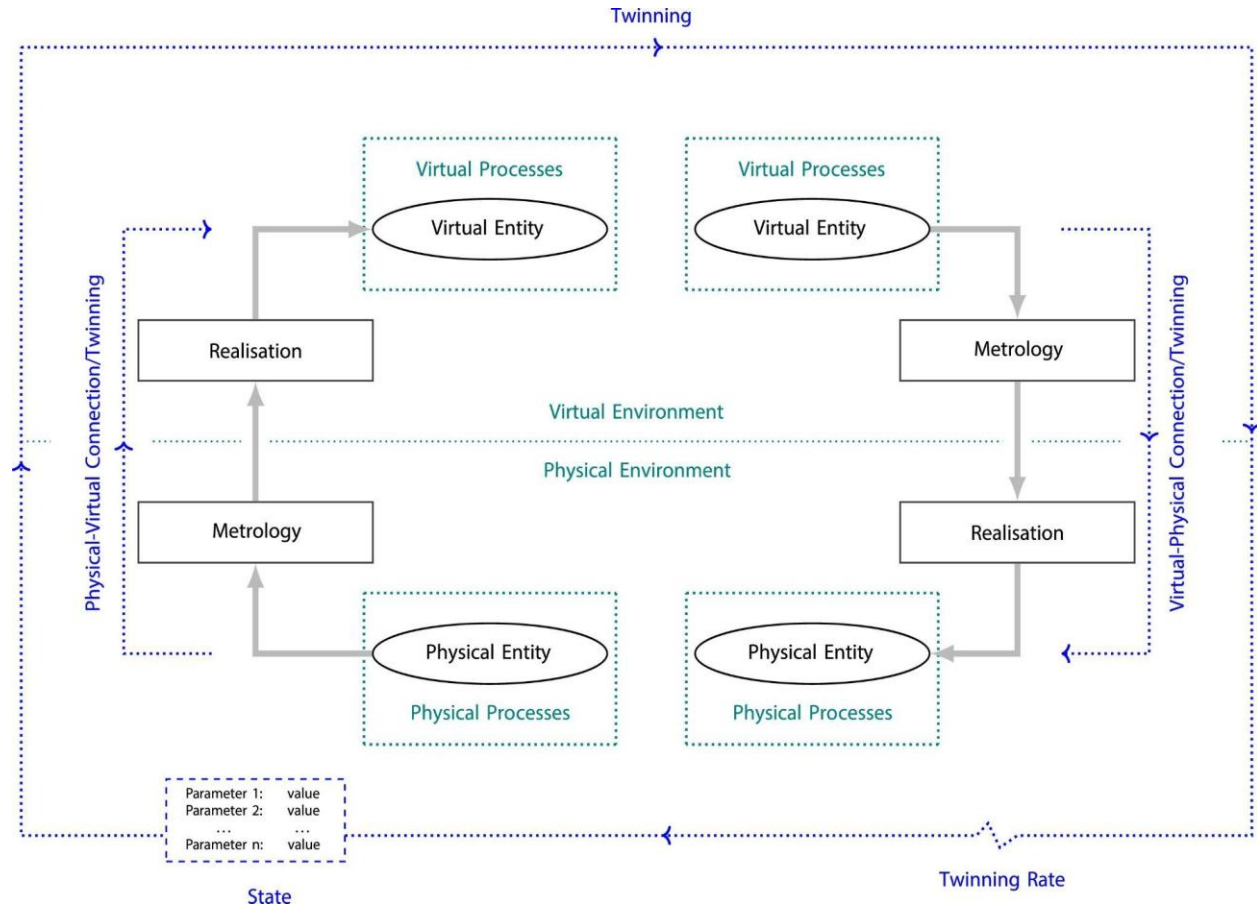


Figure 2. The physical-to-virtual and virtual-to-physical twinning process [26]

2.2.5 Automated Storage and Retrieval Systems (AS/RS)

Automated Storage and Retrieval Systems (AS/RS) represent advanced, high-tech solutions for optimizing order fulfillment. These systems prevent human errors and reduce operational inefficiencies and costs by utilizing robotic mechanisms to automate the storage and retrieval of items in the warehouse [27]. Many modern warehouses are increasingly implementing AS/RS systems to enhance productivity by streamlining inventory management, improving space utilization, and increasing order processing speed.

Zhou et al. proposed a Permutation-Combination Heuristic (PCH) that can minimize both total order fulfillment time and energy consumption simultaneously [28]. Their approach converts dual-objective optimization problems to single-objective problems through weight methods. Furthermore, they integrated features of the Genetic Algorithm (GA) with other heuristic techniques for enhanced efficiency and robustness. According to the experimental results, the AS/RS with the PCH algorithm consistently outperformed the traditional warehouse with standalone GA. Primarily, it was evaluated as a feasible solution to enhance warehouse efficiency by achieving notably reduced order fulfillment times and improved energy efficiency.

In another study, Shan et al. proposed a queueing network model designed for multi-line order management in multi-tote storage and retrieval systems that utilize Autonomous Mobile Robots (AMRs) [29]. This model can be a valuable tool to help warehouse operators make informed decisions about optimized resource allocation and budgeting during the pre-deployment stage. Furthermore, it helps predict potential bottlenecks and assess the system performance under various operational scenarios. Integrating warehouse management and queueing theory to enhance decision-making processes ensures that AMR-based systems operate with maximum efficiency and reliability.

2.2.6. Order Batching

Among the various approaches for optimizing order fulfillment, order batching is one of the most effective methods for enhancing efficiency. Order batching problems (OBPs) can be described as the challenge of finding an optimal batching combination that includes multiple customer orders, either to maximize or minimize internal costs [11] [30]. This technique allows pickers to process

items for multiple orders simultaneously. For instance, cluster picking is one of the most widely known order batching techniques. As its name suggests, it groups items close to each other in the item storage into a batch and assigns that batch to pickers [31]. By retrieving items in the same batch, total travel distance and pick-up time can be significantly reduced. Similar to order fulfillment problems (OFPs), heuristic or meta-heuristic methods are generally preferred over exact methods to solve these problems for reasonable computational time and solution quality [33].

Cano et al. proposed a Grouped Genetic Algorithm (GGA) developed to solve the Order Batching and Sequencing Problem with Multiple Pickers (OBSPMP) and minimize the makespan [34]. GGA shares many similarities with conventional genetic algorithms (GAs). Still, some of its operational stages, such as selection, crossover, and mutation, have been adjusted to satisfy the specific needs of order batching and sequencing. Unlike traditional GAs, whose genes represent individual orders, GGA uses a group-oriented chromosome representation, where genes represent a batch containing multiple orders. Additionally, its crossover operation exchanges batch-related information while preventing the generation of infeasible solutions. Grouping orders with GGA reduced the total completion time for a given list by 18.3%, demonstrating its ability to solve order batching problems. Similarly, Koch et al. compared GGA with other traditional approaches to reduce picker travel distance and order picking time, demonstrating the superiority of GGA [35].

Huang et al. aimed to minimize the total order processing time while uniformly distributing workloads across storage zones by combining a bi-objective genetic algorithm for OBP with Pseudo-Boolean Optimization (PBO) for sequencing order batches [36]. Their optimization process was divided into two stages: (1) order batching and (2) sequencing those created batches. According to the numerical experimental results, the workload differences among batches have been significantly reduced, and the total processing time has vastly decreased compared to the unoptimized results. Even though dividing optimization into two separate stages brings high computational efficiency and more manageable implementations, it has a few potential weaknesses. There could be the possibility of getting stuck in local optima and a risk of weakening capabilities of finding global optimal solutions due to sequential dependency [37].

2.3 Exact and Heuristic Methods for Order Fulfillment Optimization

Heuristic or meta-heuristic methods are often preferred over exact mathematical methods, particularly for large-scale instances, reasonable and practical computational time, and solution quality to fulfill optimization. [38].

Aboelfotoh et al. conducted two separate experiments aimed at minimizing the total distance traveled by pickers while assigning customer orders to batches under capacity constraints. In the first experiment, they employed an exact method of Mixed-Integer Programming (MIP) for small-scale instances, whereas in the second experiment, they utilized a heuristic method developed by DHL Supply Chain for large-scale instances [39]. The experimental results demonstrated that the MIP and heuristic methods outperformed the traditional First-Come-First-Served (FCFS) batching regarding travel distance. Notably, the heuristic method achieved even more advanced solution quality and computational efficiency in large-scale instances within a significantly shorter time frame than the exact process. Similarly, Hwang et al. developed an order batching algorithm based on cluster analysis to minimize the total distance traveled by pickers [40]. Their computational experiments compared the proposed heuristic method with traditional exact batching methods such as the seed and Cluster-Weighted (CW) algorithms. The results revealed that the heuristic method outperformed the conventional exact methods in both solution quality and computational efficiency.

Although heuristic methods often perform better than exact methods for solving OBPs and OFPs, these heuristic methods also have weaknesses. For example, while Genetic Algorithms (GAs) have a strong global search capability, their local search capability is relatively unsteady [41]. There have been numerous attempts to integrate various heuristic methods to address the weaknesses of individual methods while preserving their strengths. Mousavi et al. combined Genetic Algorithms (GAs) with Particle Swarm Optimization (PSO) to find optimal schedules for Automated Guided Vehicles (AGVs) in warehouses [42]. GA offers strong global search capabilities but lacks local search refinement, whereas PSO excels at local search but has weaker global search capabilities. The hybrid GA-PSO method enhanced performance by integrating GA's global exploration with PSO's local convergence capabilities. Experimental results showed that the hybrid approach

outperformed standalone GA and PSO, resulting in shorter makespans and more efficient AGV utilization.

There have also been many efforts to enhance the capabilities of GAs themselves. Neri et al. developed a Memetic Algorithm (MA), which is an improved version of the traditional GA [43]. The Memetic Algorithm has an additional step, “memes,” to the original GA, which performs a local search after the conventional selection, crossover, and mutation steps. This local search integrates heuristic and problem-specific optimizations for refined individual solutions. Memetic Algorithms generally converge faster, but at the same time, they effectively avoid getting stuck at a local optimum due to premature convergence compared to the original GA. However, it incurs higher computational costs due to the additional local search stage. For further improvement of local search ability of Memetic Algorithms, Chawla et al. proposed a hybrid approach of MA and PSO called Modified Memetic Particle Swarm Optimization (MMPSO) [44]. The primary objective of MMPSO is to leverage the strong global search capabilities of MA and the local search capabilities of PSO to compensate for their weaknesses. Experimental results demonstrated that MMPSO produced more robust AGV scheduling than the traditional GA, MA, and PSO. However, further research is needed to reduce computational costs and address more complex tasks in dynamic warehouse environments.

2.4 Various Order Similarity Calculations

2.4.1 Similarity Coefficient using Cube per Order Index (COI)

The similarity coefficient is a strongly problem-oriented similarity index designed to evaluate the correlation between orders in order-picking systems [45]. It aims to group frequently co-ordered items into the same clusters to minimize travel distance and enhance picking efficiency [46]. The similarity coefficient measures the correlation between item pairs based on “belonging frequency information” and “cube per order index (COI),” which respectively indicate the frequency of items appearing together in orders and a metric to guide storage location decisions [47]. Bindi et al. calculated the similarity coefficient $S_{i,j}$ by the following formula:

$$S_{i,j} = \left(\frac{a + \frac{1}{4}(b + c)}{a + b + c} \right) \times \left(\frac{\min(COI_i, COI_j)}{\max(COI_i, COI_j)} \right) \quad (1)$$

where a, b, and c are several orders containing items i and j respectively, item i but not j, and item j but not i. COI_i and COI_j represent the ratio of an item's total movement to its average stock level.

They simulated the proposed similarity coefficient and various strategies in a 12000 m² warehouse with multiple configurations, and the devised similarity index reduced the total travel distance by up to 13% compared to a random plan. However, the proposed similarity coefficient is specifically designed to reduce the travel distance of pickers through COI factors. In other words, the correlations of order pairs obtained through the similarity coefficient account for item composition and spatial separation in storage. Therefore, it is crucial to carefully consider the intended optimization purpose when using the similarity between order pairs.

2.4.2 Jaccard Similarity & Cosine Similarity

Jaccard Similarity measures the similarity between two sets by comparing their intersection to the size of their union [48]. It calculates the number of elements in common between two sets compared to the total number of elements across both sets using the following equation [49]:

$$J_{A,B} = \frac{|A \cap B|}{|A \cup B|} \quad (2)$$

where A and B are two sets, $|A \cap B|$ represents the number of elements in both sets, and $|A \cup B|$ represents the number of elements across both sets.

Although centered on web news clustering, Kumar et al. discuss the use of Jaccard Similarity in grouping similar items based on the similarities found by Jaccard Similarity [50]. They mined high-utility webpage sets using Jaccard Similarity to identify frequently co-visited webpage sets. Jaccard Similarity was used to identify meaningful patterns in the dataset and discard irrelevant ones that do not satisfy the thresholds. They determined that appropriate Jaccard Similarity Threshold values result in noise reduction and a fair irrelevant pattern filter, thereby increasing computational efficiency and pattern quality. The methodologies can be adjusted to warehouse order clustering, where orders are treated as webpages and items are treated as content within those webpages.

Cosine Similarity is another method to measure the similarity between two datasets. Unlike Jaccard Similarity, Cosine Similarity treats the datasets as vectors and finds the similarity based on the angle between the vectors [51]. Its formula to find similarity is as follows:

$$S_{A,B} = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} \quad (3)$$

where θ is the angle between the vectors, A and B are vectors of datasets, and $\|A\|$ represents the L2 norm or magnitude of the vector.

Muflikhah et al. combined Cosine Similarity with Latent Semantic Indexing (LSI), Singular Value Decomposition (SVD), and Principal Component Analysis (PCA) for document clustering [52]. The combined techniques of LSI, SVD, and PCA preprocess the document data by reducing dimensionality, removing noise, and highlighting meaningful patterns to enhance the performance of Cosine Similarity. They conducted experiments on 20 news groups and evaluated the results on both external and internal qualities. They observed that the combined cosine similarity outperformed the Euclidean distance-based clustering, thanks to its sensitivity to orientation rather than magnitude.

Both Jaccard Similarity and Cosine Similarity are frequently found in item-clustering tasks. Each method has its strengths and weaknesses. Jaccard Similarity focuses on the intersection and union of sets, treating data as sets. It ignores the frequency of items appearing, and only the presence of the items matters. Jaccard Similarity is relatively simple, based on set theory, and excels at binary comparisons [53]. Moreover, it resists frequency variations, resulting in strong noise tolerance. However, it shows poor performance in sparse data as the union grows. On the other hand, Cosine Similarity focuses on the angle between vectors as it treats data as vectors. It considers the frequency of the items appearing through vector weights. As it focuses on orientation rather than magnitude, it is more suitable for extended dataset analysis and clustering and widely adopted finding similarity and clustering [54]. However, since it is insensitive to magnitude, datasets with different lengths but similar item distribution may be incorrectly considered similar. Furthermore, it is computationally more expensive, and the result is less intuitive than that of the Jaccard Similarity. Therefore, a careful analysis of dataset characteristics is required when choosing a similarity calculation method between Jaccard Similarity and Cosine Similarity.

2.4.3 Association Rule Mining with Apriori Algorithm and FP-Growth

a) Apriori Algorithm-based Association Rule Mining (AA-ARM)

Apriori Algorithm-Based Association Rule Mining (AA-ARM) is one of the most popular methods in data mining for extracting frequent item sets and deriving association rules from transactional datasets [55]. Frequent itemsets refer to groups of commonly co-ordered items according to an order list. Association rules are implications of the form $A \rightarrow B$, indicating how the presence of item A affects the probability of item B appearing [56]. The steps of the Apriori Algorithm are as follows:

1. Generate Candidate Item Sets

- Start with single items (1-itemsets).

2. Count Support

- For each candidate, count its occurrences in the database.
- Remove candidates that do not meet the minimum support threshold.

3. Generate Frequent Item Sets

- Expand itemsets (e.g., 2-itemsets, 3-itemsets) and repeat the process until no more frequent itemsets are observed.

4. Generate Association Rules

- Use frequent itemsets to derive rules in the form $A \rightarrow B$.
- Evaluate the rules using metrics such as:
 - **Confidence:** Measures how frequently item B appears when item A is present.
 - **Lift:** Indicates the strength of the rule compared to random occurrences.

The Apriori Algorithm has the strengths of relatively intuitive implementations and practical capabilities for discovering patterns based on frequently occurring itemsets. However, it has a significant limitation in that it needs to scan the entire database iteratively to count the occurrence of each candidate itemset. As the dataset size increases, the computational load of the algorithm grows exponentially. Therefore, this approach is more suitable for small-scale instances. For example, Chen et al., who applied the Apriori Algorithm to identify similarities between order

pairs, simulated their approach with only between 50 and 300 orders, which are relatively small compared to the scope of this study [57].

b) FP-Growth-based Association Rule Mining (FP-ARM)

To address the limitations of the Apriori Algorithm in large-scale instances, the FP-Growth algorithm can be adopted as an alternative approach for discovering frequent itemsets and deriving association rules. Although the implementation of the FP-Growth is more complicated than that of the Apriori Algorithm, it can provide identical results with significantly reduced computational load [7].

FP-Growth constructs a Frequent Pattern Tree (FP-Tree), a compact and hierarchical data structure that stores relationships between itemsets. Unlike the Apriori Algorithm, which requires many database scans, FP-growth requires only two scans, regardless of the database size. [58]:

1. **First Scan:** To construct the FP-Tree.
2. **Second Scan:** To derive association rules from the constructed FP-Tree.

Thanks to this computational efficiency, the FP-ARM can serve as a powerful tool in large-scale instances to overcome the scalability issues of the Apriori Algorithm. In this thesis, the FP-ARM is adjusted to calculate similarities between every order pair in large-scale cases, and the detailed methodology is discussed further in Chapter 3.

2.5 Summary

In this chapter, various order fulfillment optimization approaches, including order batching, were reviewed. The following key insights were gained:

- **Various Order Fulfillment Strategies:**
 - Decentralization and Dynamic Slotting of Item Storages
 - Zone Picking
 - Real-Time Control
 - Automated Storage and Retrieval Systems (AS/RS)
 - Wave Picking (Order Batching)
- **Various Order Batching Criteria:**

- Item Locations
- Workload Balance
- Similarities Between Orders
- **Apriori Algorithm-Based Association Rule Mining (AA-ARM):**
 - A widely used method to identify similarities between orders.
 - High computational load due to the large number of database scans required to calculate frequencies for each item set.
- **FP-Growth-Based Association Rule Mining (FP-ARM):**
 - Requires only two database scans, regardless of data size.
 - More suitable for large-scale instances compared to the Apriori Algorithm.

According to the highlights, it is evident that each order fulfillment approach addresses a specific objective for certain stages in warehouse operations. Individual strategies, such as order batching, dynamic item allocation, and decentralized storage, can provide optimal solutions for specific aspects of warehouse operations. However, combining them, devised independently without considering their interdependence, may not guarantee global optimality in warehouse throughput. As a result, it is necessary to develop and integrate the required strategies seamlessly and intelligently to discover global optimal solutions for the entire warehouse operations.

Chapter 3

Order Fulfillment Optimization Algorithm

This chapter outlines the warehouse layout, structure, and operational process described in this thesis, along with a brief recap of the research problems and objectives introduced in Chapter 1. The assumptions, constraints, and conditions made while solving the research problems are also defined. Then, detailed mathematical models describing the warehouse operations are developed to find optimal solutions in small instances. Next, a heuristic order fulfillment optimization algorithm is devised by modifying and integrating various independently existing approaches. Its order fulfillment results are compared to the ones obtained by the mathematical models to demonstrate its capability of finding optimal solutions.

3.1 Warehouse Environment

The entire warehouse considered in this thesis is illustrated in Figure 3. First, the item storages shown in Figure 4 are composed of four compositions. The black rectangles represent compartments to store item totes containing large quantities of the specific item types. The blue square represents a cable robot that retrieves the item totes. The robot moves across the cables, quickly retrieving the totes regardless of location, without needing external equipment such as ladders. The purple and brown rectangles represent pick-up and return conveyors, respectively. The cable robots retrieve item totes in need from the corresponding compartments and place them in a pick-up tote waiting for the item totes on the pick-up conveyor. Once the pick-up tote is prepared, AGVs (indicated by red rectangles in Figure 3) deliver the pick-up tote from the pick-up conveyor to the operator stations. AGVs are represented in Figure 3 as red and blue rectangles on the black lines connecting the nodes. The red AGVs are responsible for transporting items from the shelves to the operator stations in the middle of the warehouse, and the blue AGVs handle the subsequent transportation of items from the operator stations to the packaging station. Human operators receive items from the AGVs at the operator stations and sort the required quantities for each order. The operator stations are represented as yellow and green rectangles in the middle in Figure 3. Green indicates the operator is present at the station, and yellow means the operator is absent. The various operator station statuses will demonstrate the algorithm's flexibility in a dynamic environment during the simulation. Once the operators receive the required items from

the pick-up tote, they return the tote containing the remaining items to the AGVs. After visiting every operator and waiting for the items in the pick-up tote, the AGVs transport the pick-up tote to the return conveyor, allowing the cable robot to return the item totes to the compartments from which they were retrieved. While the remaining items are transported back to the item storages, the sorted items are stored in order bins, where each bin is assigned to a single order. The operator stations and order bins are described in more detail in section 3.2.3. Once the orders assigned to the order bins are fulfilled, another AGV (in blue in Figure 3) transports the order bins to the packaging station, where the items in each order bin are packaged and shipped to the end-customers.

The AGVs are restricted to moving along designated black lines on the ground, each with a fixed direction that the AGVs must follow. The AGVs are not permitted to travel against the assigned direction. As illustrated in Figure 3, the direction of each line is indicated by arrows. According to the topology of the AGV paths, any broken AGVs or unexpected obstacles on one of the lines may make specific routes inaccessible. In such cases, the red dotted lines, typically not used, are activated, allowing the AGVs to move around obstacles without severe delays.

The entire warehouse can be divided into two major parts. The first part involves the item storage shelves, where items are stored and retrieved by the operator and then sorted at the operator stations according to the order list. The second part involves the operator stations connected to the packaging stations where the sorted items are packaged and shipped to the end customers. This study focuses on optimizing the warehouse's first part, specifically the order fulfillment process, from retrieving items from storage shelves to sorting them at the operator stations. Optimization of the second part will be addressed in more detail in *Chapter 5: Conclusion & Future Work*.

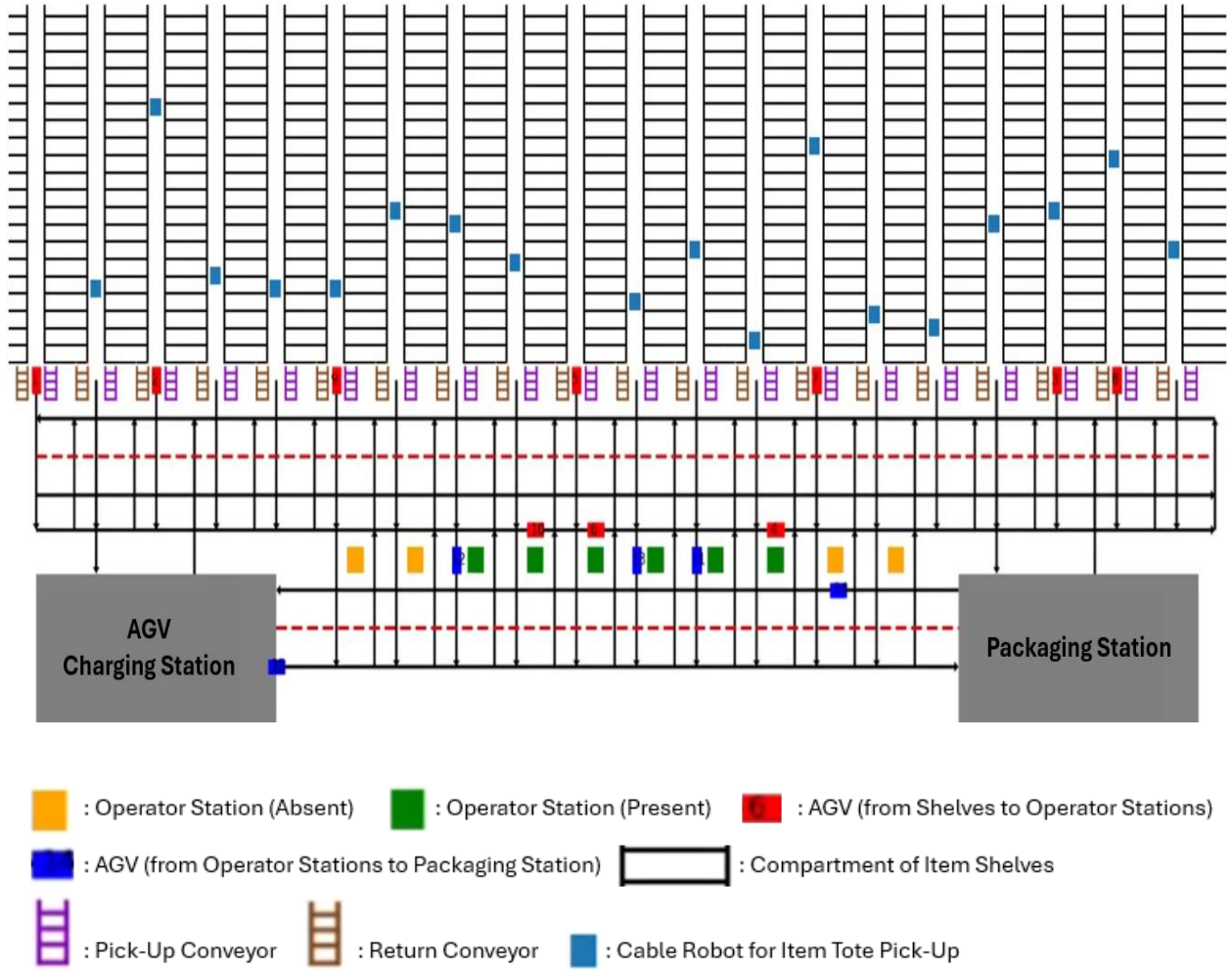


Figure 3. Entire Warehouse Layout

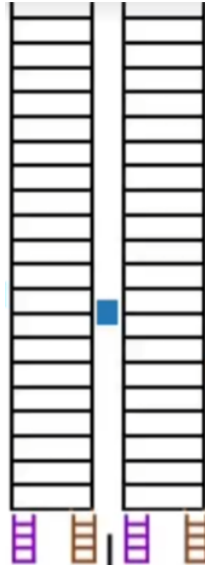


Figure 4. Item Storage Shelf with a cable robot and pick-up & return conveyors

3.2 Order Fulfillment Optimization Algorithm

Three independent order fulfillment optimization (OFO) approaches reviewed in Chapter 2 are modified and integrated into a new OFO algorithm: (1) FP-Growth-based Association Rule Mining, (2) Order Batching, and (3) A Hybrid of Public and Personal Item Storages. The algorithm is depicted in Figure 5. Policy 1 and Policy 2 refer to the devised approaches, which will be discussed in greater detail as the chapter progresses.

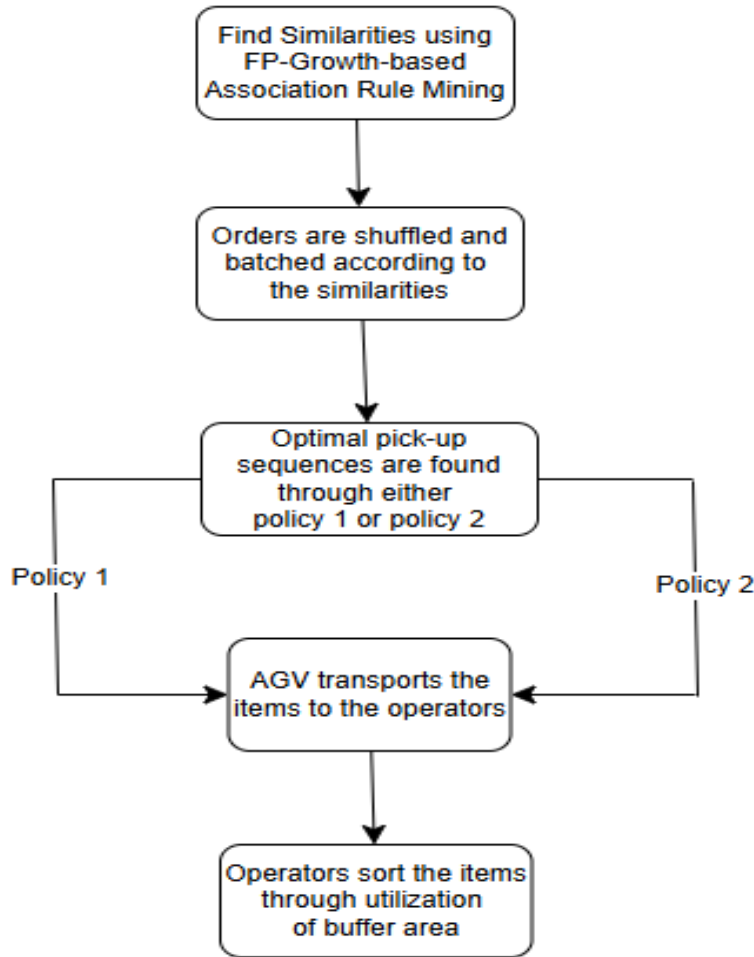


Figure 5. Order Fulfillment Optimization Algorithm

3.2.1 FP-Growth-based Association Rule Mining

As discussed in *Chapter 2: Literature Review*, the Apriori and FP-Growth algorithms find frequent item sets in datasets. However, they differ considerably in methodology, efficiency, and scalability. This study uses FP-Growth-based Association Rule Mining to identify similarities between every order pair in large-scale instances, leveraging its superior scalability compared to widely adopted Apriori-based approaches. The FP-Growth algorithm is outlined below:

- 1) Convert the “Order-Item List” to an “Item-Order List.”
- 2) Count the frequencies of each order appearing in the Item-Order List.

- 3) Reorganize the orders in the Item-Order List in descending order of frequency.
- 4) Construct an FP-Tree.
- 5) Extract the “Conditional Pattern Base.”
- 6) Build a “Conditional FP-Tree.”
- 7) Calculate Similarities as follows:

$$S_{i,j} = 100 * \frac{freq(i,j)}{N} \quad (4)$$

, where $S_{i,j}$ represents the similarity between orders i and j , $freq(i,j)$ represents the frequency of (order i , order j) pairs, and N represents the sum of 2-Order Set frequencies.

The first four steps are to construct an FP-Tree, and the last three steps are to mine the constructed FP-Tree. A simple example order list with six orders and nine items is illustrated in Table 1 for easier understanding. First, convert the “Order-Item List” (Table 1) into an “Item-Order List” (Table 2) where each item indicates the orders in which it appears. Then, count the frequencies of each order appearing in the Item-Order List. The frequencies extracted from Table 2 are as follows: {O1: 6, O2: 5, O3: 4, O4: 1, O5: 5, O6: 5}. Consequently, the priority is {O1, O2, O5, O6, O3, O4}. Next, reorganize the Item-Order List according to the assigned priority, as demonstrated in Table 3. Each item now lists its associated orders in the order of their priority.

Table 1. Example Order-Item List

Order	Item
O1	{A, C, D, E, G, H}

O2	{B, C, F, H, I}
O3	{A, B, D, E}
O4	{C}
O5	{D, E, G, H, I}
O6	{A, C, E, F, I}

Table 2. Item-Order List

Items	Orders
A	O1, O3, O6
B	O2, O3
C	O1, O2, O4, O6
D	O1, O3, O5
E	O1, O3, O5, O6
F	O2, O6
G	O1, O5

H	O1, O2, O5
I	O2, O5, O6

Table 3. Reorganized Item-Order List

Items	Reorganized Order-Set
A	O1, O6, O3
B	O2, O3
C	O1, O2, O6, O4
D	O1, O5, O3
E	O1, O5, O6, O3
F	O2, O6
G	O1, O5
H	O1, O2, O5
I	O2, O5, O6

The FP-Tree is constructed with hierarchical arrangements based on the frequencies and priorities of the orders extracted from the Reorganized Item-Order List, as shown in Figure 6. Each node of

the FP-Tree is represented in the form of “Order Number: Frequency of the Order in This Position.” For example, the node labeled “O2: 3” is branched from “Null” which indicates the O2 appears three times in the first position across the sequences “B: {O2, O3},” “F: {O2, O6},” and “I: {O2, O5, O6}.” Every node connected to the “Null” represents the starting point of the new paths. Additionally, multiple nodes may indicate the same order in different positions, depending on the contextual placement within the dataset. For example, the node “O2: 2” is connected to the node “O1: 6”, representing occurrences of O2 after the appearance of O1 in the sequence, unlike “O2: 3” which is connected to “Null.” This characteristic of the FP-Tree demonstrates that the algorithm considers the frequencies of the orders appearing and the relational sequences of each order within the dataset.

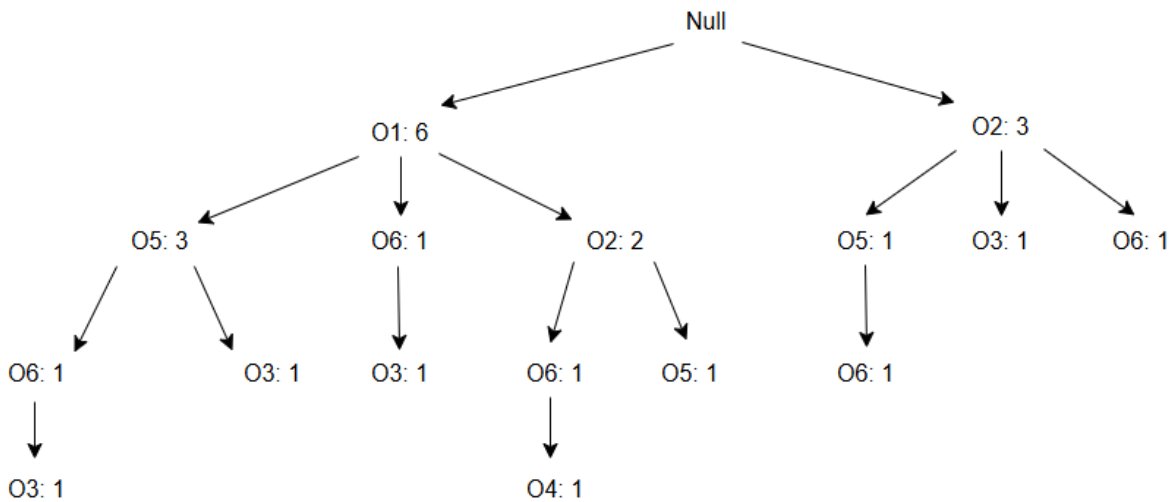


Figure 6. Constructed Example FP-Tree

After the FP-Tree is completed, Conditional Pattern Bases are extracted from the orders with the lowest priority. All paths in the FP-Tree that lead to these orders under the extraction must be identified during this process. Each path is recorded with its respective frequency, as shown in Table 4. This process ensures that every pattern of the lowest-priority orders within the datasets is captured and forms a strong foundation for the subsequent steps in the FP-Growth algorithm.

Table 4. Conditional Pattern Bases

Orders	Conditional Pattern Bases
O4	{O1, O2, O6, O4: 1}
O3	{O1, O5, O6, O3: 1}, {O1, O5, O3: 1}, {O1, O6, O3: 1}, {O2, O3: 1}
O6	{O1, O5, O6: 1}, {O1, O6: 1}, {O1, O2, O6: 1}, {O2, O6: 1}, {O2, O5, O6: 1}
O5	{O1, O5: 3}, {O1, O2, O5: 1}, {O2, O5: 1}
O2	{O1, O2: 2}
O1	N/A

Next, count the frequency of each order's appearance in the Conditional Pattern Bases for each target order. This step aggregates the occurrences of orders across all extracted paths and associates them with the respective target orders, as shown in Table 5. This frequency information is crucial for calculating the similarities between order pairs in subsequent algorithm steps.

Table 5. Conditional FP-Trees

Orders	Conditional FP-Tree

O4	{O1: 1}, {O2: 1}, {O6: 1}
O3	{O1: 3}, {O5: 2}, {O6: 2}, {O2: 1}
O6	{O1: 3}, {O5: 2}, {O2: 3}
O5	{O1: 4}, {O2: 2}
O2	{O1: 2}
O1	N/A

Finally, it can calculate similarities for every 2-order set with equation (4). The achieved similarities for each order pair are shown in Table 6. This step evaluates the quantitative relationship between orders based on their co-occurrences, providing the basis for subsequent batching and optimization processes.

Table 6. Similarities between Every Order-Pair

2-Order Sets	Frequency	Similarity
(O1, O2)	2	7.7%
(O1, O3)	3	11.5%
(O1, O4)	1	3.8%
(O1, O5)	4	15.4%

(O1, O6)	3	11.5%
(O2, O3)	1	3.8%
(O2, O4)	1	3.8%
(O2, O5)	2	7.7%
(O2, O6)	3	11.5%
(O3, O4)	0	0%
(O3, O5)	2	7.7%
(O3, O6)	2	7.7%
(O4, O5)	0	0%
(O4, O6)	1	3.8%
(O5, O6)	2	7.7%
Total	N = 27	100%

The frequencies and similarities of 2-order sets extracted by the Apriori Algorithm are identical to those achieved by the FP-Growth algorithm. However, the methodologies of the two algorithms show significant differences in computational efficiency. The Apriori Algorithm requires iterative scanning of the Item-Order list to count the frequencies of each 2-order set appearing. For instance,

the earlier example has fifteen 2-order sets, so the algorithm must scan the entire Item-Order list fifteen times. In contrast, FP-Growth requires only two scans: one for constructing the FP-Tree and another for mining the constructed tree. As the dataset size increases, the computational load for the Apriori algorithm grows exponentially. Consequently, it is unsuitable for large-scale instances that handle thousands of orders and items. On the other hand, FP-Growth can maintain excellent efficiency through just two scans, regardless of the dataset size. The FP-Growth algorithm described in this chapter will be tested to identify similarities in large-scale simulations, demonstrating its efficiency and superiority over the Apriori Algorithm presented in *Chapter 4: Simulation Results & Analysis*.

3.2.2 Order Shuffling

According to the order fulfillment optimization algorithm in Figure 5, after identifying similarities between orders, they are shuffled based on the obtained similarities. The order list is rearranged to maximize total internal similarity in every sliding window of Alpha and Beta orders. Alpha and Beta are parameters with positive integer values mainly used in the devised algorithm, which will be introduced in the next section. For a given window size of Alpha + Beta, every consecutive window-sized subset of the permutation is considered. For a permutation p_i , these windows are:

$$W_k = \{p_{i_k}, p_{i_{k+1}}, \dots, p_{i_{k+\alpha+\beta-1}}\} \text{ for } k = 1, 2, \dots, N - \alpha - \beta + 1 \quad (5)$$

, where N = total number of orders. The objective is to maximize the sum of the internal similarities over all windows. The internal similarity of a window W_k can be computed as the sum of $S_{i,j}$ for all pairs (i, j) in that window. Let O and P be the sets of orders and positions, respectively, where $O = P = \{0, \dots, N - 1\}$. Then, an optimal rearranged order list to satisfy the objective can be obtained by solving the following mathematical model:

$$\max \sum_{p=1}^{N-(\alpha+\beta)+1} \sum_{i=0}^{\alpha+\beta-1} \sum_{j=i+1}^{\alpha+\beta-1} \sum_{o_1=0}^{N-1} \sum_{o_2=0}^{N-1} S_{o_1, o_2} * x_{p+i, o_1} \times x_{p+j, o_2} \quad (6)$$

$$\text{s. t.} \quad \sum_{o \in O} x_{p, o} = 1 \quad \forall p \in P \quad (7)$$

$$\sum_{p \in P} x_{p, o} = 1 \quad \forall o \in O, \quad (8)$$

$$x_{p,o} \in \{0,1\}, p \in P, o \in O.$$

where $S_{i,j}$ represents the similarity between *Order i* and *Order j*, and $x_{p,o}$ is a binary variable that indicates whether *Order o* is placed in *Position p* in the permutation. (6) is an objective function to maximize the sum of internal similarities in sliding windows. (7) and (8) are constraints, respectively, ensuring that each *Position p* has exactly one order, and each *Order o* is assigned to precisely one position in the permutation.

This mathematical model ensures that each order in the shuffled order list is highly similar to other nearby orders in the sequence and to the preceding and following orders. As a result, it helps the AGVs efficiently retrieve items for multiple orders through a single pickup travel, as many orders near the currently processing orders share everyday items.

3.2.3 Order Fulfillment Optimization with Buffer Bins

Order fulfillment optimization in this study leverages the effect of similarity-based order shuffling by introducing buffer bins as personal item storage for each operator. The devised order fulfillment method can significantly reduce pick-up cycles by eliminating redundant item retrievals and AGV travels. The layout of the operator stations where operators sort the retrieved items is illustrated in Figure 7. Each operator station is composed of three key areas:

- **Pick-Up (Donor) Bins Area:** This is where the AGV stays while the operator retrieves items from the pick-up tote.
- **Order Bins Area:** This is where items for orders with the highest priority in the order list are sorted and prepared for processing.
- **Buffer Bins Area:** This area temporarily stores items for orders not yet assigned to the order bins. The buffer area is an extension of the warehouse shelves and is crucial in optimizing order fulfillment in this study.

As explained later, each buffer bin can store only one type of item or order, and the maximum capacity for each item per buffer bin is known in advance. The central controller determines the number of items to be picked and the specific bins to place them in and displays them on a screen

for the operator to follow. This system ensures the operator follows the instructions, minimizing human error risk.

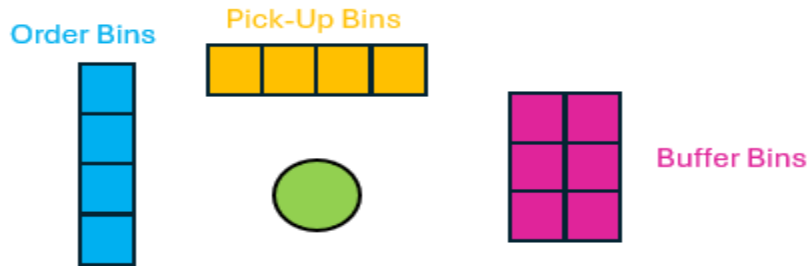


Figure 7. Operator Station Layout

The algorithm to find optimal pick-up sequences for order fulfillment optimization is depicted in Figure 8. First, orders in the order list are shuffled to maximize the internal similarities in every sliding window of $(\text{Alpha} + \text{Beta})$ orders as described in the previous section. Then, the first Alpha and Beta orders are assigned to order bins and buffer bins, respectively. The current orders assigned to the order bins and buffer bins are transferred to the central controller along with the updated order list. Then, the central controller determines which item and order should be retrieved and processed for optimal order fulfillment through either Policy 1 or Policy 2. The two policies represent different approaches to utilizing the buffer bins. They are differentiated by the allocation strategies for the buffer bins as follows:

- **Policy 1:** Order-Based Buffer Bins
- **Policy 2:** Item-Based Buffer Bins

To facilitate easier understanding, these two approaches are explained through mathematical models and algorithm flowcharts, accompanied by a simple example under a single-operator scenario. Then, they are extended to a multi-operator scenario for more realistic applications.

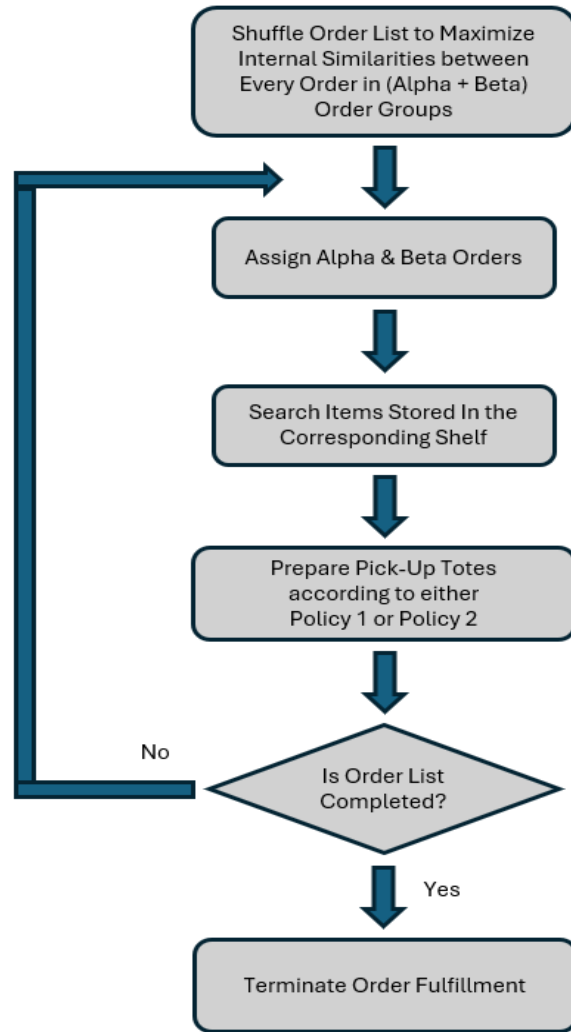


Figure 8. Algorithm for Optimal Pick-Up Sequences

Before describing the policies, an example warehouse environment will be introduced to help illustrate each policy. As shown in Figure 9, the system comprises four shelves, cable robots, three AGVs, twenty different types of items, and one operator. Each shelf contains five kinds of items, and the operator has four order bins, four pick-up bins, and six buffer bins. The maximum capacity of the AGVs is assumed to equal the number of pick-up bins. In other words, the AGVs can transport a maximum of four item totes in one trip. The primary goal is to complete an order list with the fewest number of pick-up sequences as possible.

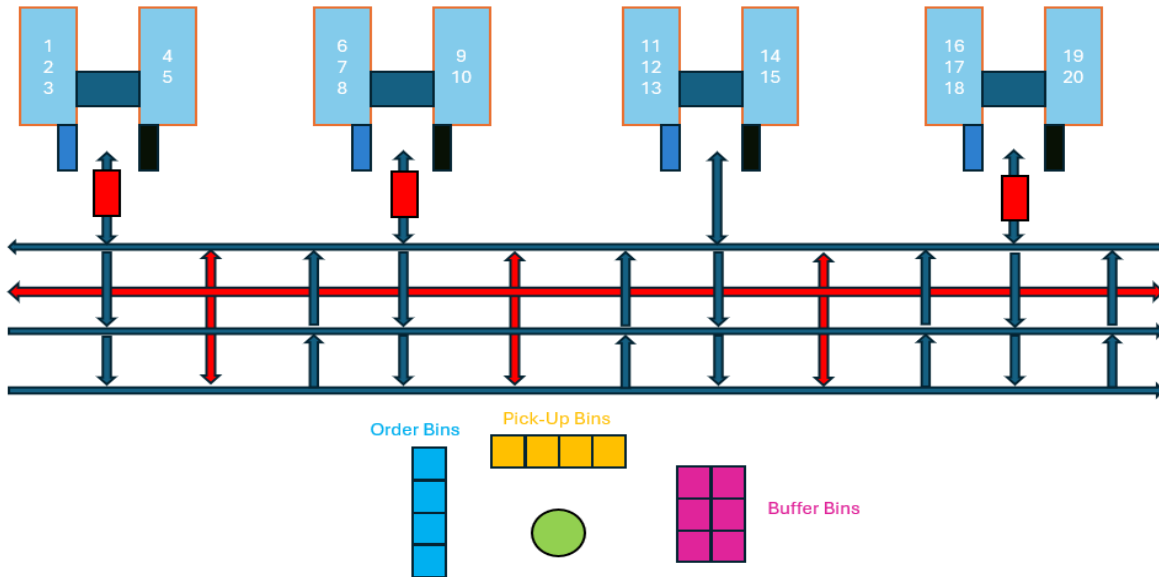


Figure 9. Warehouse Example for Policy 1 & Policy 2

1) Policy 1: Order-Based Buffer Area

In Policy 1, buffer bin allocation is order-centric, meaning each buffer bin is assigned to a specific order. In other words, each buffer bin stores items for one particular order. The orders in the order list are classified as Alpha, Beta, and Gamma orders as follows:

- **Alpha Orders:** These are the highest-priority orders in the order list and are assigned to the order bins.
- **Beta Orders:** The items required for Alpha orders are also stored in the buffer bins. This strategy allows operators to simultaneously fulfill multiple Alpha and Beta Orders with a single item retrieval.
- **Gamma Orders:** The remaining orders in the list have not yet been classified as Alpha or Beta. Their priorities can change dynamically in response to various situations. This will be explained as the example progresses.

In Figure 10, the algorithm for Policy 1 is illustrated as a flowchart, describing how the system determines the priorities of the orders and allocates them to the buffer bins based on their classification. This order-centric buffer bin allocation manner can first address the Alpha orders

with the highest priority while simultaneously handling the Beta orders by storing any everyday items picked for the Alpha orders. Gamma orders are considered under specific conditions, which will be clarified through an upcoming example.

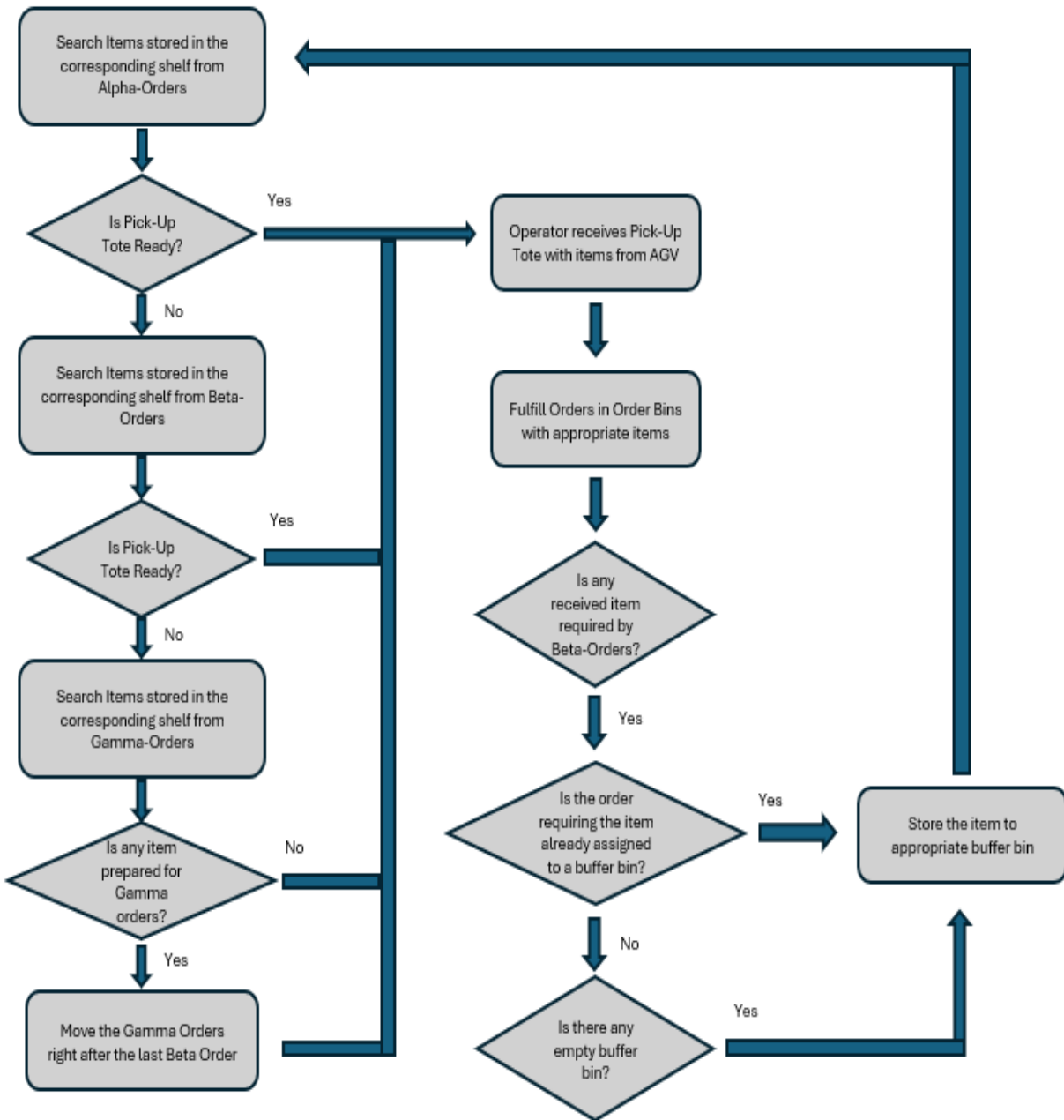


Figure 10. Algorithm of Policy 1: Order-Based Buffer Area

A mathematical model of Policy 1 is created first to prove that Policy 1 can find an optimal solution. Sets, notations, parameters, and decision variables are defined as follows:

- $O =$ A set of the entire orders in the order list, indexed by o
- $I =$ A set of the entire items in the warehouse, indexed by i
- $S =$ A set of shelves in the warehouse, indexed by s
- $x_{o,i} = \begin{cases} 1, & \text{if item } i \text{ for order } o \text{ has been picked up} \\ 0, & \text{otherwise} \end{cases}$
- $x_i = \begin{cases} 1, & \text{if item } i \text{ has been picked up} \\ 0, & \text{otherwise} \end{cases}$
- $r_{o,i} = \begin{cases} 1, & \text{if item } i \text{ is required by order } o \\ 0, & \text{otherwise} \end{cases}$
- $C_{max} =$ Maximum capacity of one pick-up sequence
- $w_o = \begin{cases} w_\alpha, & \text{if order } o \text{ is assigned to } \alpha \text{ orders} \\ w_\beta, & \text{if order } o \text{ is assigned to } \beta \text{ orders, } (w_\gamma \leq w_\beta \leq w_\alpha) \\ w_\gamma, & \text{otherwise} \end{cases}$
- $m_{i,s} = \begin{cases} 1, & \text{if item } i \text{ is available from the current shelf } s \\ 0, & \text{otherwise} \end{cases}$
- $y_o^B = \begin{cases} 1, & \text{if order } o \text{ is assigned to buffer bins} \\ 0, & \text{otherwise} \end{cases}$
- $z_o^\beta = \begin{cases} 1, & \text{if order } o \text{ is a Beta order} \\ 0, & \text{otherwise} \end{cases}$

The first three sets, O , I , and S , represent sets of orders, items, and shelves, respectively. $x_{o,i}$ and x_i are decision variables to be optimized through the calculation. The parameters, $r_{o,i}$, C_{max} , w_o , $m_{i,s}$, y_o^B , and B_{max} , will be determined based on the warehouse's status, such as the list of orders and buffer bins.

Here are the objective function and constraints for Policy 1:

$$\max \quad \sum_{o \in O} \sum_{i \in I} w_o * x_{o,i} \quad (9)$$

$$\text{s. t.} \quad x_{o,i} \leq x_i \quad \forall o \in O, i \in I \quad (10)$$

$$\sum_{i \in I} x_i \leq C_{max} \quad (11)$$

$$x_i \leq m_{i,s} \quad \forall i \in I, s \in S \quad (12)$$

$$x_{o,i} \leq r_{o,i} \quad \forall o \in O, i \in I \quad (13)$$

$$\sum_{o \in O} y_o^B \leq B_{max} \quad (14)$$

$$z_o^\beta - \left(1 - \sum_{i \in I} x_{o,i} \right) \leq y_o^B \quad (15)$$

$$x_{o,i} \in \{0,1\}, o \in O, i \in I$$

$$x_i \in \{0,1\}, i \in I$$

$$y_o^B \in \{0,1\}, o \in O$$

, where (9) represents maximizing the degree of order processing for a given list (Maximizing the weighted sum of picked items, prioritizing based on Alpha, Beta, and Gamma). (10) ensures that *Order o* can be fulfilled only if *Item i* has been picked up. (11) prevents the total number of item totes in one pick-up sequence from exceeding the maximum capacity of the pick-up sequence. (12) ensures each item in the pick-up sequence is from the current shelf. (13) ensures that items are prepared only if the items are required by orders to be processed. (14) ensures that the total number of orders assigned to buffer bins does not exceed the total number of buffer bins. Finally, (15) ensures that Beta orders are assigned to buffer bins if their items are picked up. This mathematical model will be run and compared through simulation to demonstrate that Policy 1 can achieve optimal results that maximize the degree of order processing efficiency.

Policy 1 will be explained through a simple example. The example is composed of a total of 10 orders and 20 different types of items. As described in Figure 9, each operator has six buffer bins in the buffer area. Table 7 presents an example order list, where:

- The left column represents the order number.
- The right column represents the items required for that order in the format:

[Item Number: Required Quantity]

Table 7. Order List for Policy 1 & Policy 2 Examples

Order	Item
0	[6: 2], [1: 2]
1	[3: 2], [2: 1]
2	[13: 3], [15: 2], [5: 2], [20: 2], [17: 3]
3	[6: 1], [20: 2], [7: 3], [17: 2], [14: 2]
4	[19: 2], [20: 1], [18: 1], [13: 3], [3: 1]
5	[5: 2], [4: 1]
6	[16: 1], [8: 2], [20: 2]
7	[20: 1], [14: 1], [18: 1], [11: 2], [3: 1]
8	[12: 1], [7: 2], [1: 1]
9	[13: 1], [8: 1], [1: 1], [20: 3]

First, the similarities between the orders and order list shuffling are calculated based on the obtained similarities, as described in sections 3.2.1 and 3.2.2. The shuffled example order list is presented in Table 8.

Table 8. Shuffled Order List for Policy 1 & Policy 2 Examples

Order	Item
0	[6: 2], [1: 2]
3	[6: 1], [20: 2], [7: 3], [17: 2], [14: 2]
8	[12: 1], [7: 2], [1: 1]
9	[13: 1], [8: 1], [1: 1], [20: 3]
2	[13: 3], [15: 2], [5: 2], [20: 2], [17: 3]

4	[19: 2], [20: 1], [18: 1], [13: 3], [3: 1]
7	[20: 1], [14: 1], [18: 1], [11: 2], [3: 1]
1	[3: 2], [2: 1]
6	[16: 1], [8: 2], [20: 2]
5	[5: 2], [4: 1], [10: 1]

Once the order list shuffling is completed, the order fulfillment process begins. The algorithm follows the flowchart illustrated in Figure 10. First, the top four (= Alpha) orders are assigned to the order bins. Then, the following four orders (Beta) are assigned to the buffer bins. The order bins and the buffer bins are assigned as follows:

- **Order Bins:** [0, 3, 8, 9] (marked in red in Table 8)
- **Buffer Bins:** [2, 4, 7, 1] (marked in blue in Table 8)

The central controller then determines which items should be retrieved by the cable robots. Since Shelf 1 contains items 1 to 5, as shown in Figure 6, the cable robot retrieves a tote for Item 1 from the Alpha orders for Order 0. According to the policy, Orders 8 and 9, which also require Item 1, can be simultaneously processed. No other Alpha orders need items from Shelf 1. However, as per the constraint, the number of prepared item totes must match the maximum capacity of the AGV, so the algorithm proceeds to the Beta orders.

Order 2 from the Beta orders requires Item 5. Additionally, Orders 4 and 7 require Item 3, while Orders 1 and 3 require Item 2. To satisfy these requirements, the pick-up tote is prepared with the following four item totes (Item 1, Item 5, Item 3, and Item 2) reaching its maximum capacity.

Then, the item retrieval sequencing moves to Shelf 2, where items 6 to 10 are stored, following a similar procedure. First, the cable robot retrieves the Item 6 tote for Order 0 and simultaneously processes Order 6. Next, an Item 7 tote is retrieved for Order 3 and Order 8, and an Item 8 tote is retrieved for Order 9. Every item tote available on Shelf 2 for the Alpha and Beta orders has been retrieved. However, the number of retrieved item totes does not match the maximum capacity of AGVs. Therefore, the item search proceeds to the Gamma orders.

Among the Gamma orders, Order 5 requires Item 10. Therefore, an Item 10 tote is retrieved as the final contents of the pick-up tote. Since Order 5 belongs to the Gamma orders, its priority moves to the same level as the Beta orders and is assigned to one of the available buffer bins. To represent the changed priority explicitly, Order 5 is moved to follow the last Beta order in the order list immediately. By doing so, Order 5 will be the first Ex-Gamma order to be processed while fulfilling the Alpha and Beta orders. This allows Order 5 to exit the buffer bins quickly, freeing up space to ensure an efficient flow of items through the buffer bins.

After completing similar operations for Shelf 3, the order list has been fulfilled, as shown in Table 9. It is demonstrated that the Alpha orders in the order bins and Beta and Gamma orders in the buffer bins have been chiefly processed. This indicates that the buffer bins enable the operators to extend the number of simultaneously processed orders by retrieving one item. Finally, the operator can complete the Alpha orders in the order bins once they receive items from Shelf 4. For the Alpha orders, the cable robots prepare totes containing Items 20 and 17 for Order 3. As noted earlier, this also satisfies the requirement for Item 20 in Order 9, fulfilling all Alpha orders.

Table 9. Order List after Item Searching for the First Three Shelves

Order	Item
0	[6: 2] , [1: 2]
3	[6: 1] , [20: 2], [7: 3] , [17: 2], [14: 2]
8	[12: 1] , [7: 2] , [1: 1]
9	[13: 1] , [8: 1] , [1: 1] , [20: 3]
2	[13: 3] , [15: 2] , [5: 2] , [20: 2], [17: 3]
4	[19: 2], [20: 1], [18: 1], [13: 3] , [3: 1]
7	[20: 1], [14: 1] , [18: 1], [11: 2], [3: 1]
1	[3: 2] , [2: 1]
5	[5: 2], [4: 1], [10: 1]
6	[16: 1], [8: 2], [20: 2]

Following this, the central controller assigns the next Alpha and Beta orders. The updated Alpha and Beta orders are as follows:

- Alpha Orders: [2, 4, 7, 1]
- Beta Orders: [5, 6]

The Item 10 tote for Order 5 is already stored in the buffer bin. Therefore, to satisfy the maximum capacity of AGVs, the central controller proceeds with additional item retrieval calculations based on the new Alpha and Beta orders. Since Items 20 and 17 have already been prepared, they can process Order 2, Order 4, and Order 7 from the Alpha orders and Order 6 from the Beta orders. Then, totes for Items 19 and 18 are retrieved for Orders 4 and 7.

The retrieval process continues until the entire order list is fulfilled. As shown in Table 10, eight pick-up cycles were required to complete all 10 orders.

Table 10. Total Pick-Ups Required to Fulfill the Order List with Policy 1

Pick-Up Number	Item Totes in Pick-Up Tote
1	[1, 5, 3, 2]
2	[6, 7, 8, 10]
3	[14, 12, 13, 15]
4	[20, 17, 19, 18]
5	[5, 4]
6	[8]
7	[11]
8	[16]

2) Policy 2: Item-Based Buffer Area

Policy 2 assigns each buffer bin to a specific item type. The algorithm for Policy 2 is illustrated in Figure 11. To describe Policy 2 with a mathematical model, the following sets, notations, parameters, and decision variables are necessary:

- $O = A$ set of the entire orders in the order list, indexed by o
- $I = A$ set of the entire items in the warehouse, indexed by i
- $S = A$ set of shelves in the warehouse, indexed by s
- $x_{o,i} = \begin{cases} 1, & \text{if item } i \text{ for order } o \text{ has been picked up} \\ 0, & \text{otherwise} \end{cases}$
- $x_i = \begin{cases} 1, & \text{if item } i \text{ has been picked up} \\ 0, & \text{otherwise} \end{cases}$
- $r_{o,i} = \begin{cases} 1, & \text{if item } i \text{ is required by order } o \\ 0, & \text{otherwise} \end{cases}$
- $z_i^B = \begin{cases} 1, & \text{if item } i \text{ is assigned to an empty buffer bin} \\ 0, & \text{otherwise} \end{cases}$
- $C_{max} = \text{Maximum capacity of one pick-up sequence}$
- $w_o = \begin{cases} w_\alpha, & \text{if order } o \text{ is assigned to } \alpha \text{ orders} \\ w_\beta, & \text{if order } o \text{ is assigned to } \beta \text{ orders, } (w_\gamma \leq w_\beta \leq w_\alpha) \\ w_\gamma, & \text{otherwise} \end{cases}$
- $m_{i,s} = \begin{cases} 1, & \text{if item } i \text{ is available from the current aisle } s \\ 0, & \text{otherwise} \end{cases}$
- $y_i^B = \begin{cases} 1, & \text{if item } i \text{ has been already assigned to buffer bins} \\ 0, & \text{otherwise} \end{cases}$
- $y_o^\alpha = \begin{cases} 1, & \text{if order } o \text{ is assigned to } \alpha \text{ orders} \\ 0, & \text{otherwise} \end{cases}$
- $B_{max} = \text{Total number of buffer bins}$
- $Max_i^B = \text{Maximum capacity for item } i \text{ per buffer bin}$
- $Q_i^B = \text{Current quantity of item } i \text{ in buffer bins}$
- $Q_{o,i} = \text{Required quantity of item } i \text{ for order } o$

The first three sets, O, I, and S, represent sets of orders, items, and shelves, respectively. $x_{o,i}$, x_i , and z_i^B are decision variables to be optimized through calculation. The rest are parameters to be determined based on the warehouse's given status, such as the list of orders and buffer bins.

With additional information and a new assignment approach to the buffer bins, Policy 2 comes up with the following mathematical model:

$$\max \quad \sum_{o \in O} \sum_{i \in I} w_o * x_{o,i} \quad (16)$$

$$\text{s. t.} \quad x_{o,i} \leq x_i \quad \forall o \in O, i \in I \quad (17)$$

$$\sum_{i \in I} x_i \leq C_{max} \quad (18)$$

$$x_i \leq m_{i,s} \quad \forall i \in I, s \in S \quad (19)$$

$$x_{o,i} \leq r_{o,i} \quad \forall o \in O, i \in I \quad (20)$$

$$\sum_{i \in I} (y_i^B + z_i^B) \leq B_{max} \quad (21)$$

$$x_{o,i} * (1 - y_o^\alpha) \leq z_i^B \leq x_i \quad \forall i \in I, o \in O \quad (22)$$

$$\sum_{o \in O} Q_{o,i} * x_{o,i} * (1 - y_o^\alpha) \leq Max_i^B \quad \forall i \in I, o \in O \quad (23)$$

$$x_{o,i} \in \{0,1\}, o \in O, i \in I$$

$$x_i \in \{0,1\}, i \in I$$

$$y_i^B \in \{0,1\}, i \in I$$

where (16) is an objective function to maximize the degree of order processing for a given order list. (17), (18), (19), and (20) are identical to those described for (10), (11), (12), and (13). (21), (22), and (23) constrain the utilization of the buffer bins. (21) ensures that the number of unique item types in the buffer bins cannot exceed the total number of buffer bins. (22) prevents the storage of *Item i* in the buffer bins for alpha orders assigned to the order bins. Finally, (23) ensures that the number of *Item i* in buffer bins does not exceed the maximum capacity of the buffer bins.

To explain how Policy 2 operates, the same example used for Policy 1, Order-Based Buffer Area, is revisited. An additional information table, Table 11, specifies the maximum number of items per buffer bin for each item type.

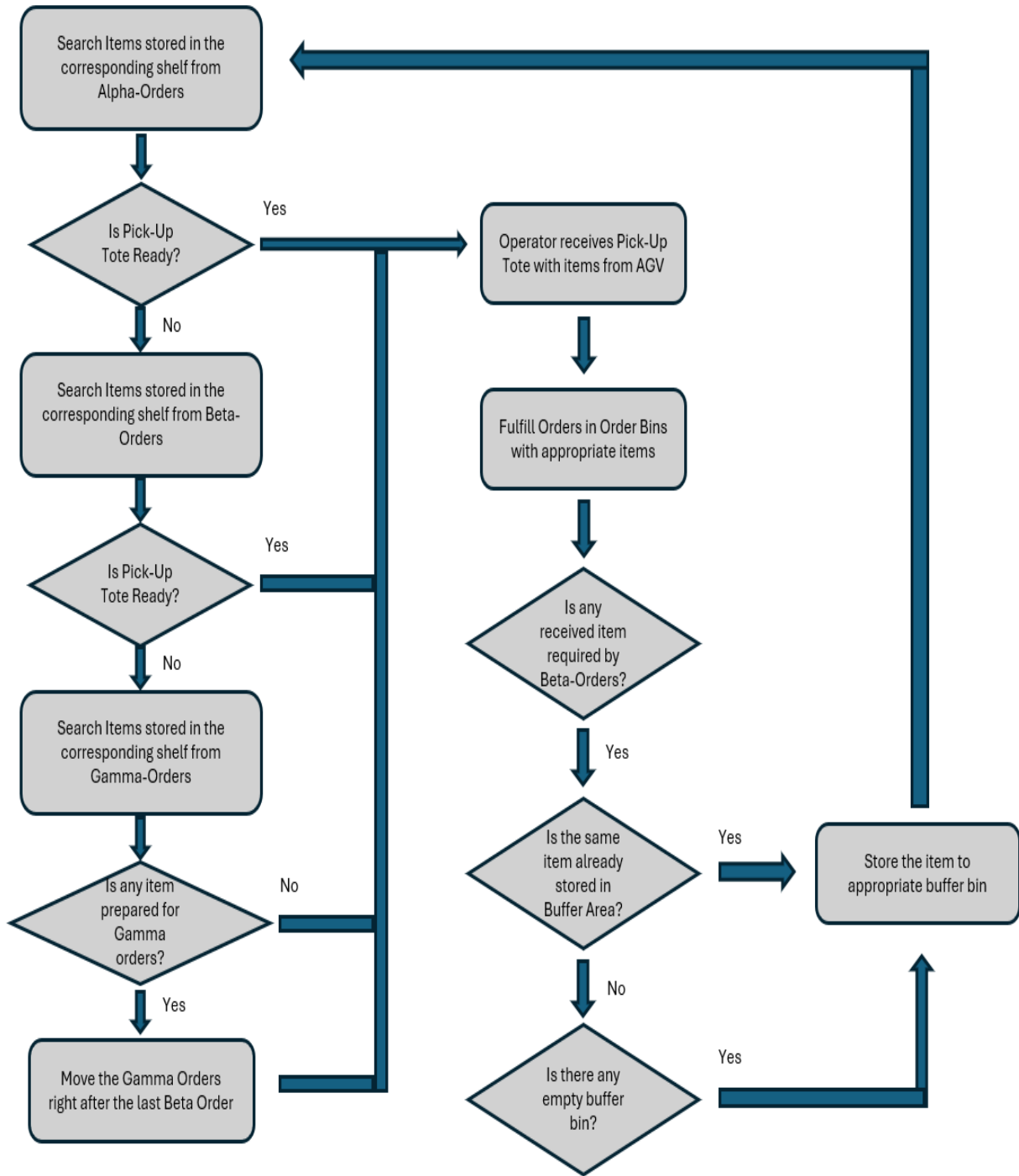


Figure 11. Algorithm of Policy 2: Item-Based Buffer Area

Table 11. Max. Number in One Buffer Bin for Each Item

Item	Max. Number in One Buffer Bin	Item	Max. Number in One Buffer Bin
1	3	11	3
2	1	12	3
3	3	13	3
4	1	14	3
5	3	15	3
6	3	16	5
7	2	17	3
8	1	18	2
9	1	19	2
10	4	20	3

Similar to Policy 1, the similarity between all orders is calculated using FP-Growth-based association rule mining. Then, the order list is shuffled so that the total internal similarity of (Alpha + Beta) order sliding windows reaches the maximum. As in the previous example, Alpha is 4 (the number of order bins), and Beta is predefined as 4. After shuffling, a new order list is obtained, as shown in Table 8.

The order fulfillment process then begins. As in Policy 1, the top four Alpha orders are assigned to the order bins. However, the Beta orders have not been transferred to the buffer bins. For the Alpha orders, the cable robot retrieves an Item 1 tote for Orders 0, 8, and 9. Since no additional items can be retrieved from Shelf 1 for the Alpha orders, we proceed to the Beta orders.

Order 2 from the Beta orders requires Item 5, and according to Table 11, a buffer bin can store up to three items of Item 5. Since all buffer bins are empty, the operator receiving the items stores two Item 5s in Buffer Bin 1. Similarly, two Item 3s for Orders 4 and 7, respectively, and one Item 2 for Order 2 are stored in each empty buffer bin. However, the buffer bin assigned to Item 3 already contains two items, and according to Table 11, the maximum number of Item 3s per buffer

bin is three. Therefore, even though Order 1 from the Beta orders requires two Item 3s, its request is ignored due to insufficient space in the Item 3 buffer bin.

The retrieval process continues until the entire order list is fulfilled. Table 12 shows eight pick-up cycles were needed to complete all 10 orders.

Table 12. Total Pick-Ups Required to Fulfill the Order List with Policy 2

Pick-Up Number	Item Totes in Pick-Up Tote
1	[1, 5, 3, 2]
2	[6, 7, 8, 10]
3	[14, 12, 13, 15]
4	[20, 17, 19, 18]
5	[3, 5, 4]
6	[8]
7	[13, 14, 11]
8	[16]

3.2.4 Extension to Multi-Operator Scenario

The previous algorithms for Policy 1 and Policy 2 were devised under a single-operator scenario. However, most warehouses are operated through the cooperation of multiple operators. Therefore, extending the earlier approaches to the multi-operator scenario is essential for more practical real-world applications. In Figure 12, an additional operator has been added, unlike the previous example. Also, the new order list in Table 13 contains 30 orders. Other settings, such as the maximum number of items per buffer bin, warehouse environment, and the definitions of Alpha and Beta, remain unchanged from the previous examples. The algorithm for the multi-operator scenario is depicted in Figure 13.

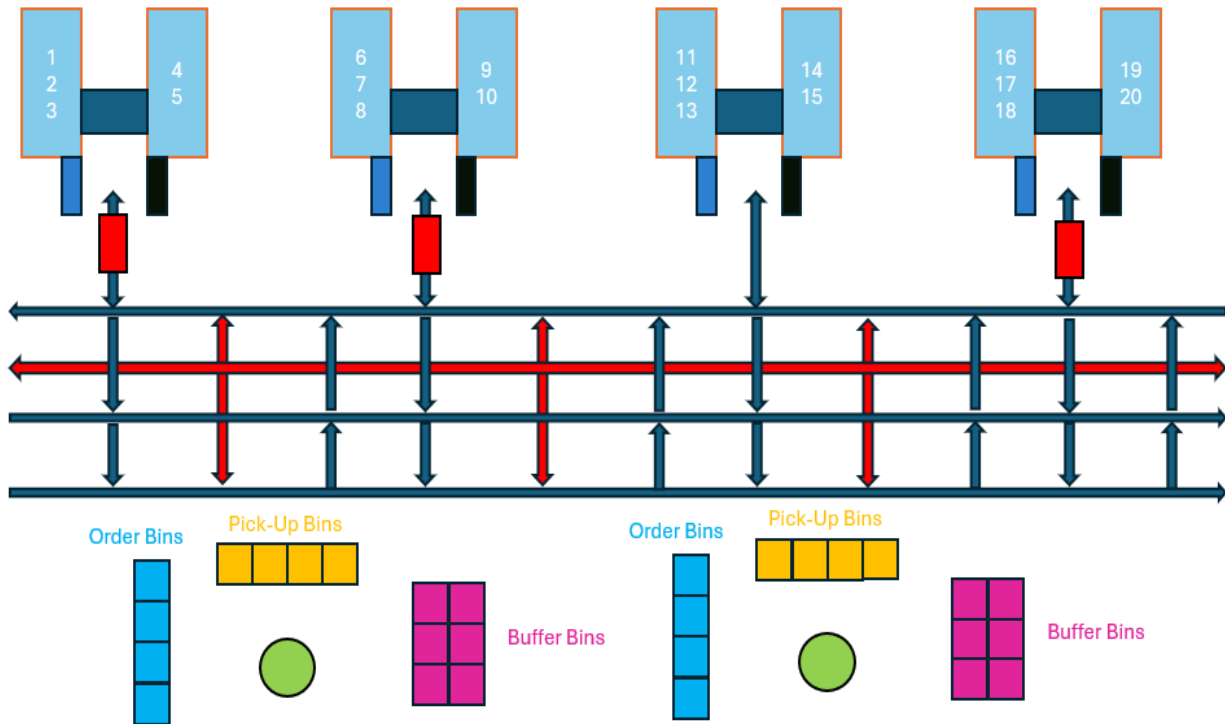


Figure 12. Multi-Operator Scenario Warehouse Example

Table 13. Order List for Multi-Operator Scenario Example

Order	Item	Order	Item
0	[10: 1], [7: 1]	15	[7: 3]
1	[2: 3], [18: 2], [10: 2], [11: 2]	16	[10: 1], [18: 1]
2	[1: 1], [15: 2]	17	[11: 4]
3	[10:1], [1:3]	18	[4: 2], [18: 3]
4	[16: 3], [17: 3]	19	[7: 1], [10: 1]
5	[19: 3]	20	[13: 1]
6	[17: 1]	21	[15: 2], [20: 2]
7	[10: 2], [20: 3]	22	[2: 2]
8	[10: 1], [13: 2], [17: 2]	23	[11: 2], [4: 2]

9	[17: 1], [15: 1]	24	[10: 1], [12: 3], [17: 2], [9: 1]
10	[10: 3]	25	[17: 1], [20: 2], [2: 2]
11	[10: 1], [15: 2]	26	[15: 2], [10: 2]
12	[2: 1], [20: 3], [19: 1]	27	[3: 1], [10: 2], [5: 3], [17: 3]
13	[14: 1], [20: 2]	28	[5: 2], [8: 3], [14: 2]
14	[19: 2], [15: 1]	29	[2: 1]

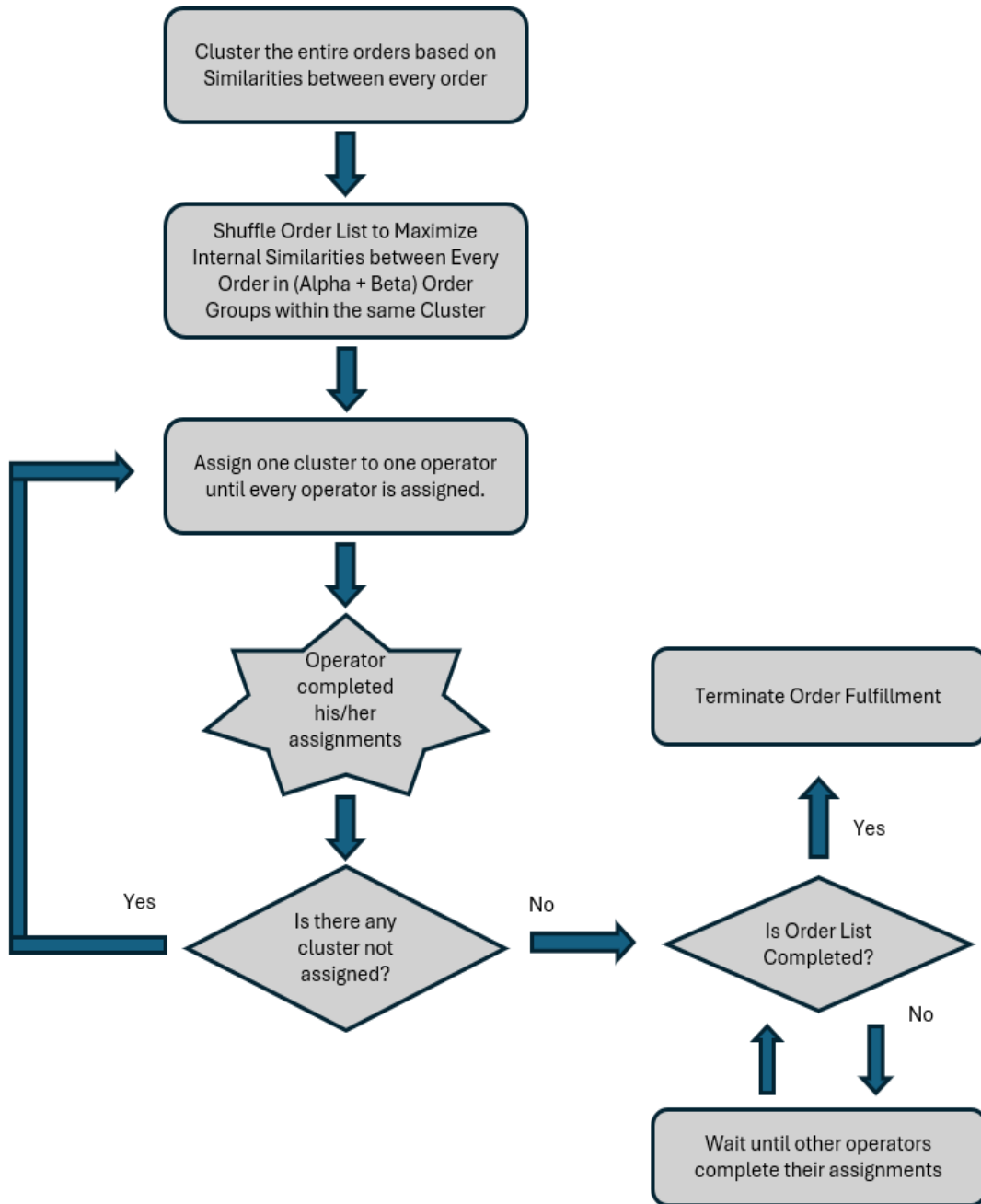


Figure 13. Algorithm of Multi-Operator Scenario

As in the single-operator scenario, the similarity between all orders is calculated using FP-Growth-based association rule mining. Then, order clustering is performed using a genetic algorithm (GA) to maximize the internal similarity of orders within each cluster. The number of clusters does not necessarily have to match the number of operators, as initially unassigned clusters can be allocated

later once operators have completed their previously assigned clusters. Order clustering facilitates efficient order sharing among operators by grouping orders with high similarities into distinct clusters. Notations, parameters, variables, and mathematical models for the order clustering are as follows:

- $M \subseteq O =$ A set of orders chosen as cluster medians $= \{1, \dots, K\}$
- $S_{i,j} =$ Similarity between order i and order j
- $x_{i,j} = \begin{cases} 1, & \text{if order } i \text{ and order } j \text{ are in the same cluster} \\ 0, & \text{otherwise} \end{cases}$
- $y_o = \begin{cases} 1, & \text{if order } o \text{ is a cluster median} \\ 0, & \text{otherwise} \end{cases}$
- $K =$ Number of clusters
- $N =$ Number of orders

$$\max \quad \sum_{i \in O} \sum_{j \in O} S_{i,j} * x_{i,j} \quad (24)$$

$$\text{s. t.} \quad \sum_{j \in O} x_{ij} = 1 \quad \forall i \in M \quad (25)$$

$$x_{i,j} \leq y_j \quad \forall i \in O, j \in M \quad (26)$$

$$x_{i,j} \in \{0,1\}, i, j \in O$$

Where (24) is an objective function to maximize internal similarity between every order in the same cluster. (25) and (26) ensure that each order is assigned to one cluster. An optimal clustering result is obtained using the Genetic Algorithm (GA) for real-world applications with large instances.

When solving the model with GA, each potential solution (or individual) lists the total number of orders, with each element indicating the assigned cluster for each order. An initial population is created by randomly generating these individuals, and parents for the next generation are selected through tournament selection in each generation. The tournament selection method compares and

selects better individuals for the next generation by randomly picking k individuals from the population and comparing their fitness values. The individual with the highest fitness among those k competitors wins the tournament and is selected as a parent. After the tournament selection, the selected parents create new offspring through a two-point crossover, swapping all the elements (i.e., assigned clusters) between randomly chosen crossover points. Then, the offsprings experience mutation, where one gene (i.e., one order) in each individual is randomly selected and assigned to a different cluster. It helps maintain diversity and avoid convergence to suboptimal solutions as the generations proceed. After the manipulation of the individuals, their fitness values are recalculated. Individuals with the highest intra-cluster similarities are likelier to persist in the next generation. This iterative process continues until a predefined number of generations is reached. Brief descriptions of the tournament selection, 2-point crossover, and mutation are depicted in Figure 14.

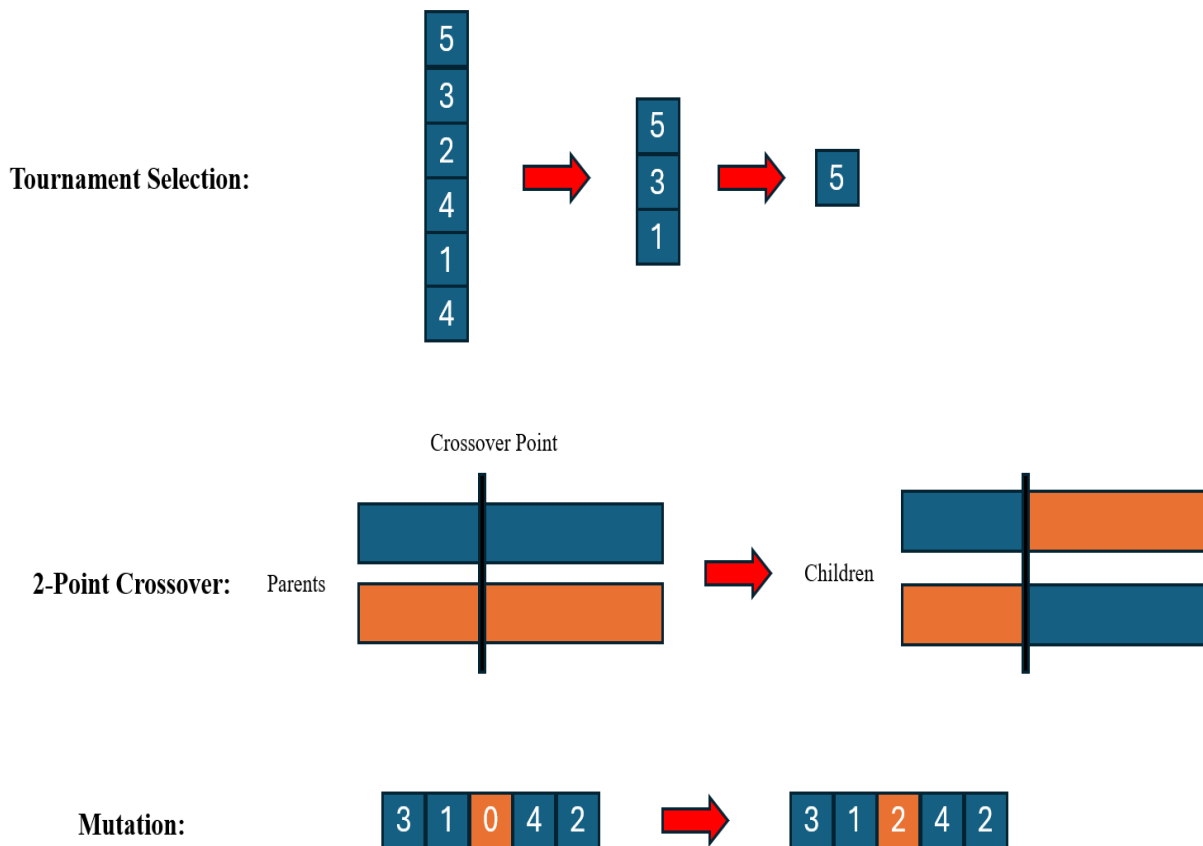


Figure 14. Tournament Selection, 2-Point Crossover, and Mutation in GA

In this example, the number of clusters is set to 3. The orders are clustered by solving the mathematical model using GA as explained. The obtained clusters are shown in Table 14. Then, the order list is rearranged by the clusters, and, like the single-operator scenario, the orders in each cluster are shuffled so that each group of Alpha and Beta orders achieves maximum internal similarity, resulting in the shuffled order list shown in Table 15.

Table 14. Order Clusters with Max. Internal Similarities

Cluster Number	Orders in Each Cluster
1	0, 1, 3, 7, 8, 10, 11, 16, 19, 24, 26, 27
2	4, 6, 15, 17, 18, 20, 23, 28
3	2, 5, 9, 12, 13, 14, 21, 22, 25, 29

Table 15. Shuffled Order List for Multi-Operator Scenario Example

Order	Item	Order	Item
0	[10: 1], [7: 1]	17	[7: 3]
19	[2: 3], [18: 2], [10: 2], [11: 2]	23	[10: 1], [18: 1]
1	[1: 1], [15: 2]	18	[11: 4]
16	[10:1], [1:3]	20	[4: 2], [18: 3]
3	[16: 3], [17: 3]	28	[7: 1], [10: 1]
7	[19: 3]	2	[13: 1]
10	[17: 1]	9	[15: 2], [20: 2]
11	[10: 2], [20: 3]	14	[2: 2]
26	[10: 1], [13: 2], [17: 2]	21	[11: 2], [4: 2]
8	[17: 1], [15: 1]	12	[10: 1], [12: 3], [17: 2], [9: 1]
24	[10: 3]	25	[17: 1], [20: 2], [2: 2]
27	[10: 1], [15: 2]	13	[15: 2], [10: 2]

4	[2: 1], [20: 3], [19: 1]	5	[3: 1], [10: 2], [5: 3], [17: 3]
6	[14: 1], [20: 2]	22	[5: 2], [8: 3], [14: 2]
15	[19: 2], [15: 1]	29	[2: 1]

Afterward, each cluster is assigned to an operator. In this example, since the number of clusters is greater than the number of operators, Cluster 1 is assigned to Operator 1, Cluster 2 is assigned to Operator 2, and Cluster 3 is initially left unassigned. The first operator completing their orders is then assigned to Cluster 3. Once assignments are made, the operators proceed with order fulfillment using either Policy 1 or Policy 2, whichever suits their assigned order list.

The pick-up totes are prepared based on the primary operator's order list in the multi-operator scenario. If the AGV transporting those totes contains any items other sub-operators require, it visits their stations. The role of the leading operator rotates to prevent item retrieval from being concentrated on a single operator and ensure smooth and balanced item circulation among the operators.

In this example, item preparation for Shelf 1 begins with Operator 1 as the primary operator, utilizing Policy 1: Order-Based Buffer Bins. According to Policy 1 and the orders in Cluster 1, totes for Items 2, 1, 3, and 5 are retrieved. Among the Alpha and Beta orders in Cluster 2 (assigned to Operator 2), Orders 23, 18, 20, and 28 are allocated to each buffer bin, and two Item 5s are stored for 28, respectively. After item retrieval, instead of moving on to the next shelf, the retrieval process repeats for the same shelf but with the following operator as the leading operator.

Operator 2 needs two Item 4s for each Order 23 and 18 in the Beta orders. Therefore, four Item 4s are retrieved and stored in the buffer bins. After several iterations, Operator 2 completes the order list for Cluster 2 before Operator 1 and begins order fulfillment for Cluster 3. Ultimately, 10 pick-up cycles are required to fulfill the 30 orders with two operators, as shown in Table 16.

Table 16. Total Pick-Ups Required to Fulfill the Order List with Policy 1 in Multi-Operator Scenario

Pick-Up Number	Item Totes in Pick-Up Tote
----------------	----------------------------

1	[2, 1, 3, 5]
2	[4]
3	[10, 7, 9, 8]
4	[11, 15, 12, 13]
5	[14, 15]
6	[18, 20, 17, 16]
7	[17, 19, 20]
8	[10]
9	[15, 14]
10	[1, 2]

Chapter 4

Simulation Results & Analysis

4.1 Policy Evaluation

In this chapter, the policies introduced in the previous chapter are tested using a custom-built Python simulation platform, and the results are analyzed and compared. The example with 10 orders from Chapter 3 is initially processed using the proposed policies and corresponding mathematical models to demonstrate their capability in finding optimal solutions. Figure 15 shows that the policies and mathematical models successfully determined the expected optimal pick-up sequences, validating their effectiveness in achieving the desired outcomes.

<pre>Pick-Ups (Order-based Buffer Bins - Policy 1): [[1, 5, 3, 2], [6, 7, 8, 10], [14, 12, 13, 15], [20, 17, 19, 18], [5, 4], [8], [11], [16]]</pre>	<pre>Pick-Ups (Order-based Buffer Bins - Mathematical) [[1, 2, 3, 5], [6, 7, 8, 10], [12, 13, 14, 15], [17, 18, 19, 20], [4, 5], [8], [11], [16]]</pre>
<pre>Pick-Ups (Item-based Buffer Bins - Policy 2): [[1, 5, 3, 2], [6, 7, 8, 10], [14, 12, 13, 15], [20, 17, 19, 18], [3, 5, 4], [8], [13, 14, 11], [16]]</pre>	<pre>Pick-Ups (Item-based Buffer Bins - Mathematical) [[1, 2, 3, 5], [6, 7, 8, 10], [12, 13, 14, 15], [17, 18, 19, 20], [3, 4, 5], [8], [11, 13, 14], [16]]</pre>

Figure 15. Demonstration of Performance for Each Proposed Policy

Then, the simulation is run under a large instance scenario of 2000 Orders and 1000 Items. Through the large-scale simulations, the results of Policy 1 (Order-Based Buffer Area), Policy 2 (Item-Based Buffer Area), and Manual Top-Down (with FCFS & no policies) will be compared. Additionally, the impact of changes in parameter values for Beta and Alpha, the number of Buffer

Bins, and the maximum capacity of the pick-up tote on operational performance is examined. The evaluation will be based on three metrics:

1. **Consolidation Time:** The time taken to fulfill the orders in the order bins. A shorter consolidation time indicates smoother order circulation in the order bins. Consolidation time will be measured as both the maximum consolidation time and the average consolidation time.
2. **Total Fulfillment (Estimation) Time:** The time to fulfill the entire order list.
3. **Total Number of Pick-Up Sequences:** The number of pick-up sequences required to fulfill the entire order list.

To measure the time, time parameters are defined as follows:

- It takes **5 seconds** for the operator to move an item **from the (Donor) AGV to the order bins.**
- It takes **5 seconds** to move an item **from the (Donor) AGV to the buffer bins.**
- It takes **5 seconds** to move an item **from the buffer bins to the order bins.**
- There is a **10-second** transition time for the operator to **switch from one AGV to the next.**
- There is a **10-second** transition time for the operator to switch **from old fulfilled order bins to the new empty order bins.**

By comparing Policy 1 and Policy 2 under different item distribution conditions, it is possible to observe whether the trends in the order list impact the selection of the more appropriate policy based on item distribution in an actual warehouse setting.

First, the orders and items were set to 2000 and 1000, respectively. The simulations were independently run in both uniform and biased item distribution scenarios. The simulation results for Policy 1, Policy 2, and the Top-Down (No Policy) order fulfillment approaches are illustrated in Figure 16 and Figure 17. According to the results, the trends in item distribution did not noticeably affect the total number of pick-up cycles or total fulfillment time for both Policy 1 and

Policy 2. It is assumed to be due to the presence of Gamma Orders. In most cases, the buffer bins are already assigned to specific orders or items because the model aims to maximize the capacity of each pick-up tote. Therefore, if there is any vacancy in a pick-up tote, it considers items for the Gamma orders. As we increased the number of orders and items, there were more candidates for the buffer bins, implying that the buffer bins are likely to be full most of the time. Consequently, the model's performance is not significantly affected by the trends of items in the order list. In conclusion, the performance of Policy 1 and Policy 2 does not depend on the trends of the item distribution.

```
Total Number of Pick-Up Cycles (Manual Top-Down Method): 2343
Total Estimation Time (Manual Top-Down Method): 46425
Max. Consolidation Time (Manual Top-Down Method): 155
Avg. Consolidation Time (Manual Top-Down Method): 92.81

Total Number of Pick-Up Cycles (Policy 1: Order-Based Buffer Area): 1099
Total Estimation Time (Policy 1: Order-Based Buffer Area): 43965
Max. Consolidation Time (Policy 1: Order-Based Buffer Area): 210
Avg. Consolidation Time (Policy 1: Order-Based Buffer Area): 87.89

Total Number of Pick-Up Cycles (Policy 2: Item-Based Buffer Area): 1208
Total Estimation time (Policy 2: Item-Based Buffer Area): 40065
Max. Consolidation Time (Policy 2: Item-Based Buffer Area): 145
Avg. Consolidation Time (Policy 2: Item-Based Buffer Area): 80.11
```

Figure 16. Simulation Results for 2000 Orders & 1000 Items with Uniform Item Distribution

```
Total Number of Pick-Up Cycles (Manual Top-Down Method): 2370
Total Estimation Time (Manual Top-Down Method): 46695
Max. Consolidation Time (Manual Top-Down Method): 145
Avg. Consolidation Time (Manual Top-Down Method): 93.35

Total Number of Pick-Up Cycles (Policy 1: Order-Based Buffer Area): 1085
Total Estimation Time (Policy 1: Order-Based Buffer Area): 43825
Max. Consolidation Time (Policy 1: Order-Based Buffer Area): 255
Avg. Consolidation Time (Policy 1: Order-Based Buffer Area): 87.61

Total Number of Pick-Up Cycles (Policy 2: Item-Based Buffer Area): 1169
Total Estimation time (Policy 2: Item-Based Buffer Area): 39685
Max. Consolidation Time (Policy 2: Item-Based Buffer Area): 160
Avg. Consolidation Time (Policy 2: Item-Based Buffer Area): 79.35
```

Figure 17. Simulation Results for 2000 Orders & 1000 Items with Biased Item Distribution

Policies 1 and 2 generally reduced the required pick-up cycles by approximately 1,285 cycles (54.2%) and 1,201 cycles (50.6%), respectively, compared to the Manual Top-Down Method. Furthermore, both policies significantly reduced the total estimated time for fulfillment compared to the Top-Down scenario, with Policy 2 proving even more effective than Policy 1. In some cases, it was observed that the total fulfillment time increases as the interaction between order bins and buffer bins becomes more active. In Figures 16 and 17, the reason why Policy 2 has noticeably shorter fulfillment time despite a more significant number of pick-up cycles than Policy 1 is because it requires fewer interactive actions between order bins and buffer bins for the operators, implying that Policy 2 used the buffer bins with more optimal degree (not excessively) than Policy 1. Another reason for this observation is that the simulation was conducted assuming that all external factors are always optimally prepared. For example, in the simulation, it is believed that whenever the operators require items, they have already been retrieved and delivered by the AGVs. However, in real-world warehouse operations, items cannot always be instantly prepared upon demand due to various factors, such as item unavailability, stock shortages, AGV congestion, and other logistical challenges. Therefore, more accurate and precise approaches are necessary to calculate the estimated time in simulations for a perfect comparison between Policy 1 and Policy 2 concerning total fulfillment time.

Policy 2 results in a distinctly smaller maximum consolidation time than Policy 1, meaning that with Policy 1, orders in order bins sometimes get stuck and remain relatively longer without fulfillment. This result is easily understandable because, in Policy 1, one buffer bin can contain items for only one order, as each buffer bin is assigned to a specific order. In contrast, Policy 2 allows one buffer bin to store items for multiple orders, as each buffer bin is assigned to a particular item type. Accordingly, the average consolidation time is lower with Policy 2 than with Policy 1.

4.2 Impact of Change in Parameters

Beyond comparing the policies, it is also essential to understand the impact of parameter changes on overall performance. Although Policy 2 demonstrated the best performance regarding total fulfillment time, the effects of parameter changes are still examined for both policies. The following parameters will be observed and analyzed: (1) Alpha, (2) Beta, (3) Number of Buffer Bins, and (4) Maximum Capacity of Pick-Up Totes.

Figure 18 illustrates the impact of changes in Beta on Total Estimation (Fulfillment) Time, Maximum Consolidation Time, Average Consolidation Time, and Number of Pick-Up Cycles. The figure shows that the trends of Total Estimation (Fulfillment) Time and Average Consolidation Time are identical. Increased Beta represents a more significant number of candidates for the buffer bins. In Policy 1, each buffer bin is assigned to one order. Therefore, if Beta is greater than or equal to the number of buffer bins (8 in the simulations), the change in Beta does not noticeably affect the performance of Policy 1. On the other hand, in Policy 2, each buffer bin is assigned to one item, meaning that unlike Policy 1, each buffer bin can intervene processing for more than one order at a time. Consequently, as Beta increases, the number of orders each buffer bin can process also increases, resulting in a decrease in Total Estimation (Fulfillment) Time.

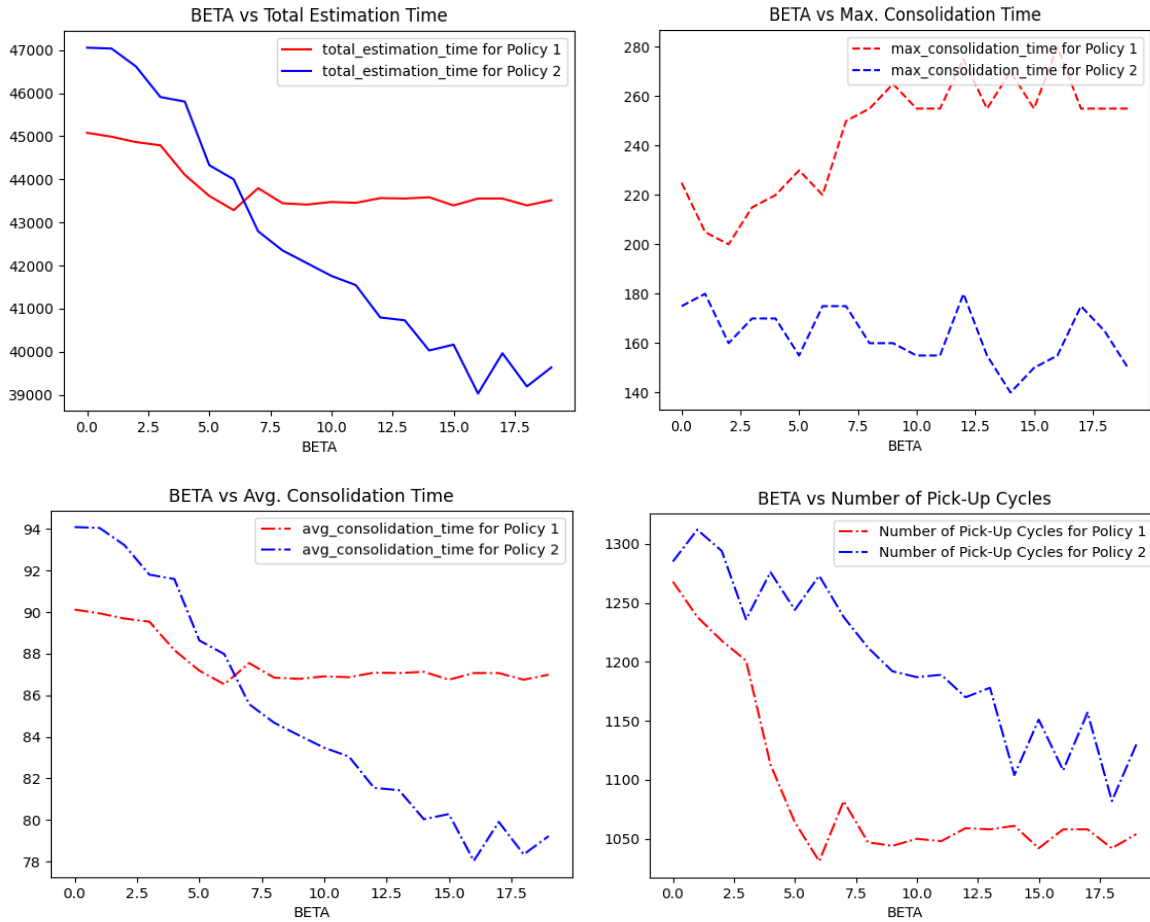


Figure 18. Impact of Change in Beta on Policy 1 and Policy 2

In Figure 19, the impact of change in Alpha is shown. An increase in Alpha indicates a higher number of order bins. Intuitively, as we process more orders in the order bins, the total number of orders each operator can process simultaneously increases and relieves congestion in the buffer bins. As a result, it reduces both the total estimation (fulfillment) time and the number of pick-up cycles for Policy 1 and Policy 2. According to the operational process of the given warehouse, orders in the order bins can be transferred to the packaging station once every order in the bins has been fulfilled. Since the value of Alpha is equal to the number of order bins, the number of orders in the order bins increases as the Alpha increases, resulting in higher maximum and average consolidation times.

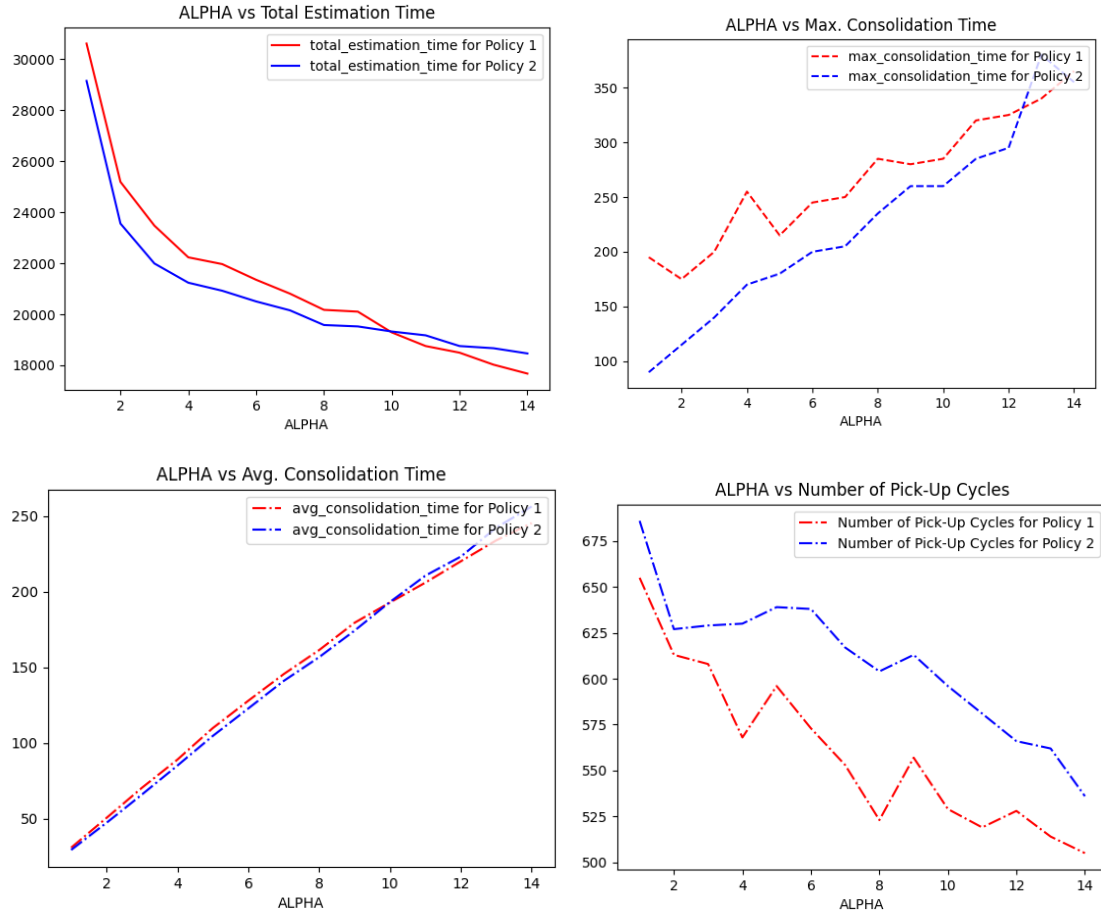


Figure 19. Impact of Change in Alpha on Policy 1 and Policy 2

Figure 20 illustrates the impact of change in the Number of Buffer Bins. As the Number of Buffer Bins increases, the Number of Pick-Up Cycles significantly decreases, allowing for simultaneous processing of more orders. However, the total estimation (fulfillment) time is minimized with no buffer bins and maximized with four buffer bins. This is because the presence of the buffer bins adds one more responsibility to the operators, requiring them to move, store, and retrieve items from the buffer bins. As mentioned in the previous section, in this simulation, the total estimation (fulfillment) time considers solely the operation time of the operators, ignoring any external factors such as AGV travel time, item retrieval time, pick-up tote preparation time, etc. In real-world applications, not only the operation time of the operators but also external factors must contribute to the overall throughput of the warehouse. The objective function in the mathematical model from Chapter 3 was to maximize the order-processing degree in each pickup sequence, thereby reducing the total number of pickup cycles required to complete an order list, rather than minimizing the

total estimation (fulfillment) time for the operators. According to the simulation results, Policy 1 and Policy 2 yield a decent total estimation (fulfillment) time from 8 buffer bins. However, it is still more significant than the scenario with no buffer bins. Furthermore, both resulted in a significantly reduced total number of pick-up cycles. Therefore, in real-world applications, finding an optimal number of buffer bins with the best trade-off between the operation time of the operators and the total number of pick-up cycles should be crucial.

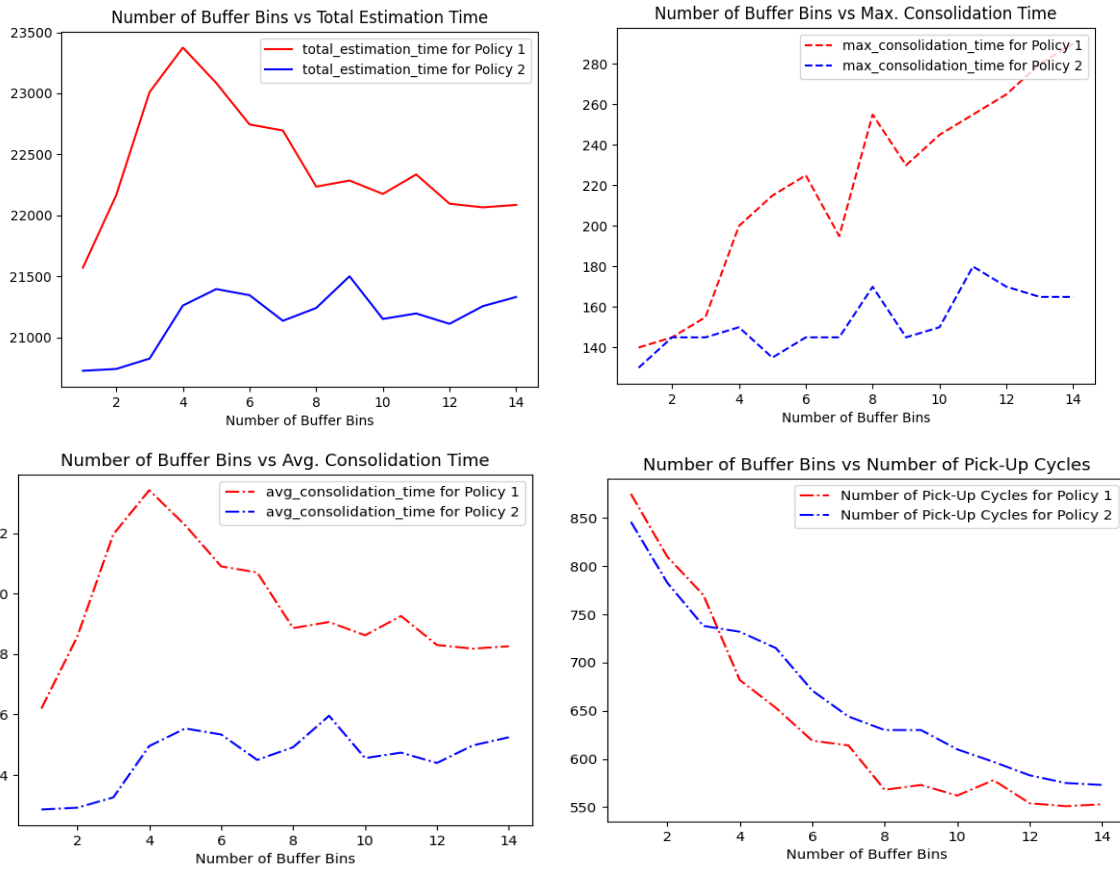


Figure 20. Impact of Change in Number of Buffer Bins on Policy 1 and Policy 2

Finally, the effect of change in maximum capacity of each pick-up sequence is illustrated in Figure 21. The increased maximum capacity of each pick-up sequence means that more types of item totes can be retrieved and processed in each pick-up sequence. The larger capacity of the pick-up sequence improved overall performance in every aspect.

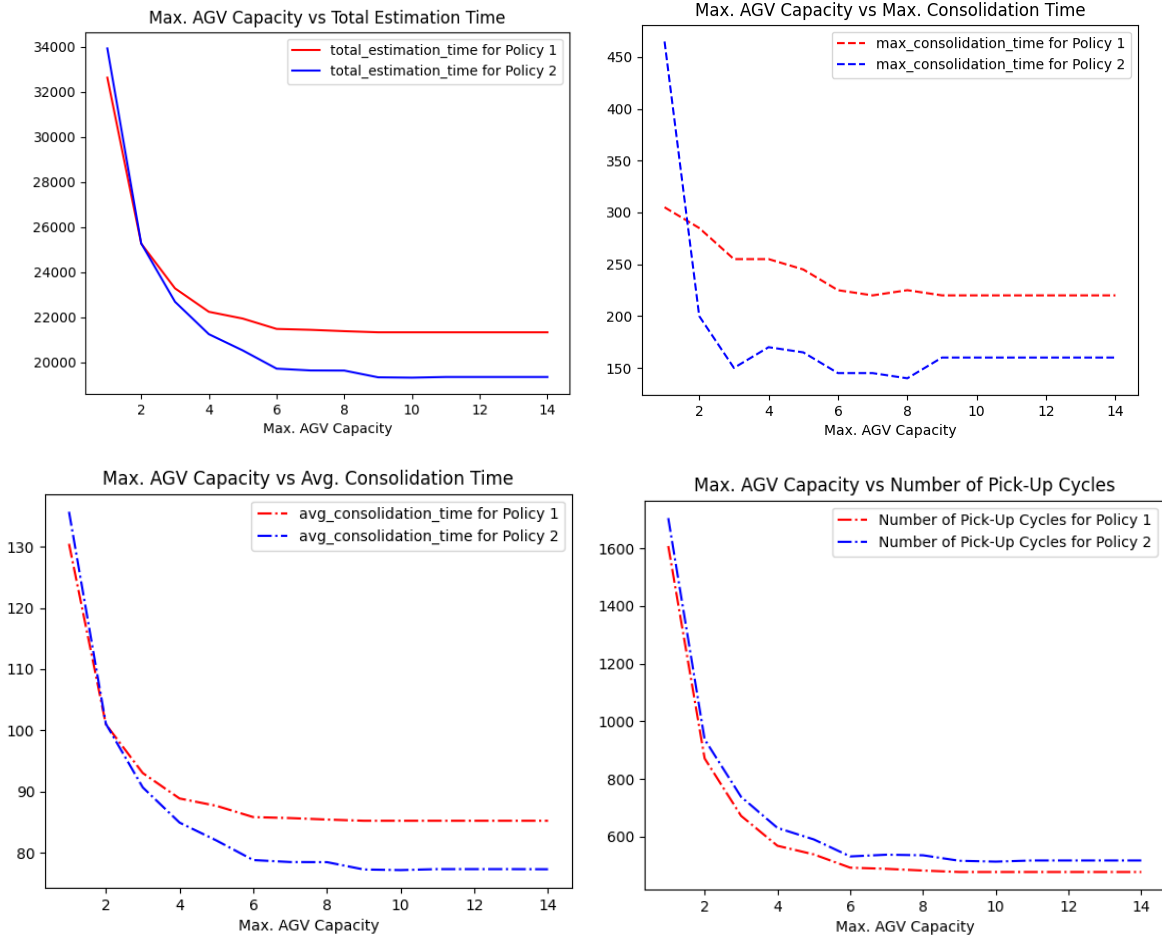


Figure 21. Impact of Change in Beta on Policy 1 and Policy 2

4.3 Simulations in ROS & Gazebo

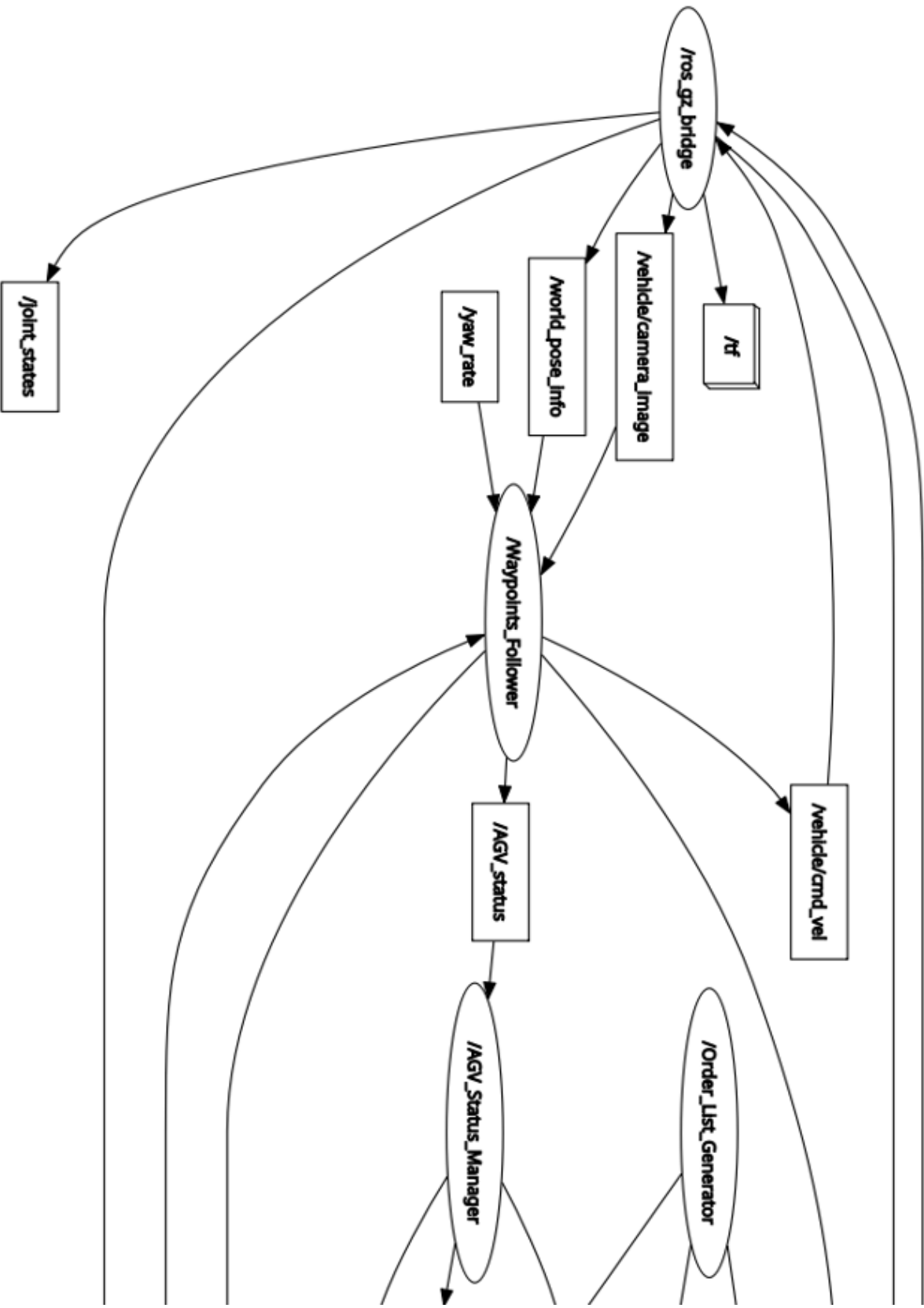
An advanced simulation environment has been created to represent and demonstrate warehouse operations with the devised order fulfillment optimization using ROS2-Jazzy and Gazebo Harmonic. In this virtual world, the entire warehouse operational process is illustrated, ranging from order fulfillment, the main topic of this thesis, to order packaging, which is briefly discussed in Chapter 3.1: Warehouse Environment. The efficiency and feasibility of the devised optimization strategies can be evaluated more visually and intuitively in a controlled yet realistic manner through this simulation. Like an actual warehouse, this virtual warehouse is operated through extensive communication among various systems. Most systems are reasonably complete to avoid misunderstandings, but communication among them has not yet been fully implemented. Although

not yet finalized, the following explanation aims to illustrate clearly how this simulation platform operates and the interactions that occur behind the scenes.

The interrelationship among the systems is shown in Figure 22. The graph, known as an “RQT Graph,” is generated by a built-in ROS function to represent the communication occurring behind the simulations visually. The circles indicate “nodes,” which are functions either waiting for or publishing a message to other functions. Generally, each node is designed for one specific task, such as path planning, line tracking, and order sequencing. The messages interchanged among the nodes are called “topics” and represented in rectangles in the RQT Graph. Most nodes subscribe to topics under specific names and data types they are waiting for and do not initiate the tasks until the subscribed topics are received. Finally, the arrows indicate the direction of the communication flow. Arrows extending from a node to a topic represent the node publishing the topic, while arrows pointing from a topic to a node represent the node subscribing to the topic.

The simulation runs as follows: first, the `/Order_List_Generator` node randomly generates an order list. The user can manually adjust the total number of orders and available items in the warehouse. Additionally, the item distribution within the order list can be modified to make the simulations more realistic by reflecting variations in item popularity within the warehouse. The Order List Generator publishes the generated order list and is subscribed to by the `/Order_Fulfillment_Optimizer`. The `/Order_Fulfillment_Optimizer` is responsible for order clustering, pickup sequencing, and AGV allocation to optimize order fulfillment. The devised order fulfillment policies in this thesis are implemented in this node to calculate optimal pick-up sequences. This node also subscribes to the `/updated_AGV_status` topic, which was initially set to “all AGVs are available.” Then, the `/Order_Fulfillment_Optimizer` publishes the `/vehicle/travel_info` topic to the `/Path_Planner` node. The `/vehicle/travel_info` contains each vehicle's starting and destination positions, as determined by the calculated optimal pick-up sequences and received AGV availability. The `/Path_Planner` calculates the shortest path using the A* Algorithm. The computed path is received by the `/Waypoints_Follower` node under the `/vehicle/waypoints` topic. The `/Waypoints_Follower` subscribes not only to `/vehicle/waypoints` from ROS but also to various other data sources, including the AGV's current position and odometry measurements, images from the camera attached to the AGV, linear and angular velocities, and interactions with other objects, such as collisions detected in Gazebo. With the

received information, the `/Waypoints_Follower` performs two tasks. First, it calculates the appropriate linear and angular velocity on a larger scale using a PID controller based on the distance from the vehicle's current position to the next waypoint. Second, it fine-tunes the AGV's angular velocity on a smaller scale using the camera images to ensure accurate path following. The node crops the received images to the region of interest (ROI). It aligns the camera's center with the detected AGV path by adjusting the angular velocity, as illustrated in Figure 23. The comparison between with and without the camera is conducted in a test world and shown in Figure 24. Finally, the `/Waypoints_Follower` confirms whether the AGV has arrived at its final destination and updates the status of that specific AGV. Then, the status of the AGV is published and then subscribed to by the `/AGV_Status_Manager`, which updates and publishes the complete AGV status list. The Order Fulfillment Optimizer again subscribes to the updated AGV status, and the steps are repeated until the entire order list is completed. Most of the topics published by ROS nodes are subscribed by `/ros_gz_bridge`. This bridge enables communication between ROS and Gazebo, which operate on separate platforms, allowing for more realistic simulations and active interactions. The remaining nodes and topics are used to generate and run the simulations, which are beyond the scope of this thesis.



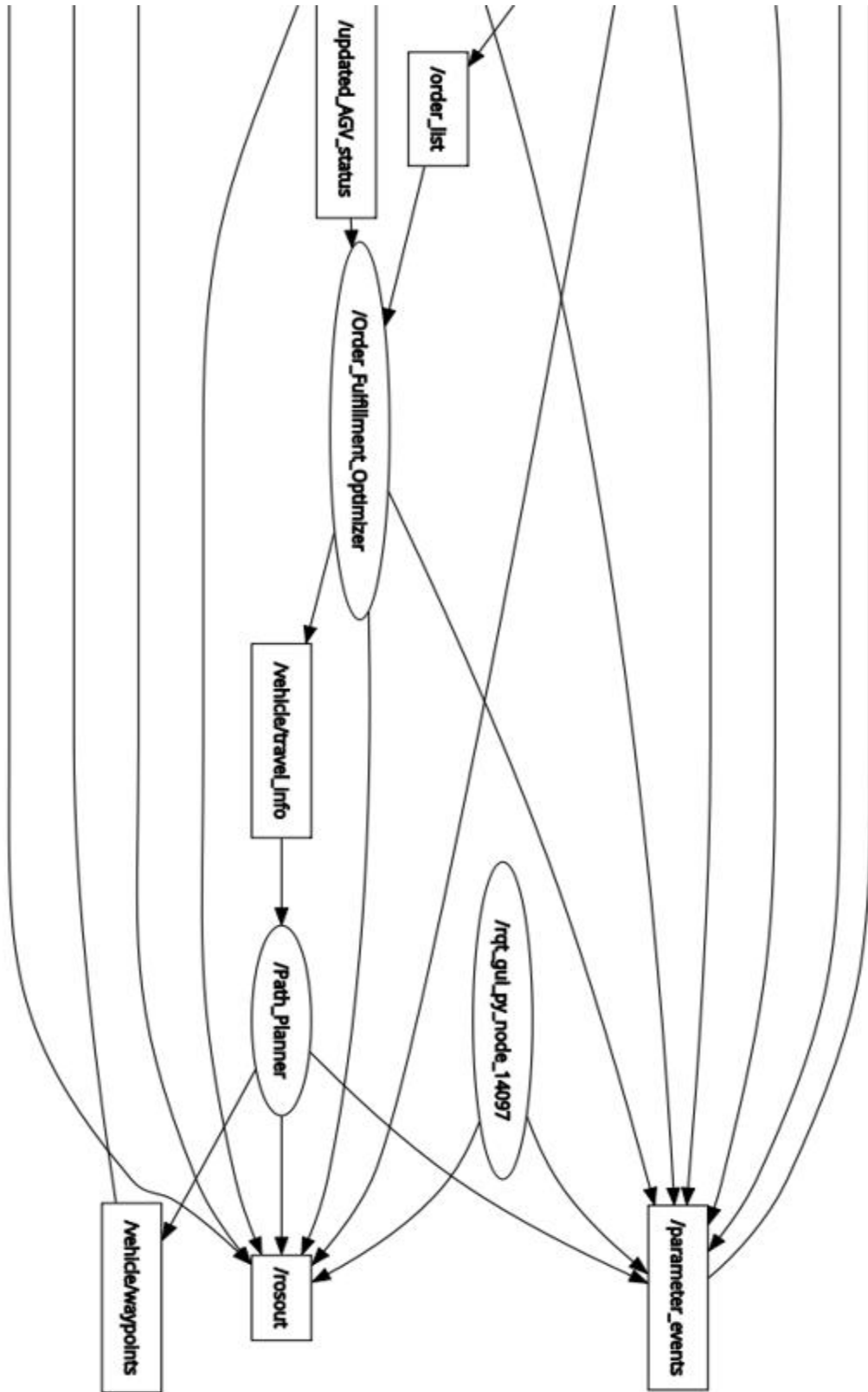


Figure 22. RQT Graph from Gazebo and ROS Simulation

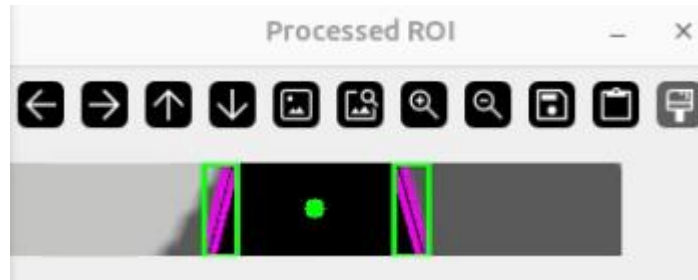


Figure 23. Fine-Tuning of Angular Velocity using a Camera

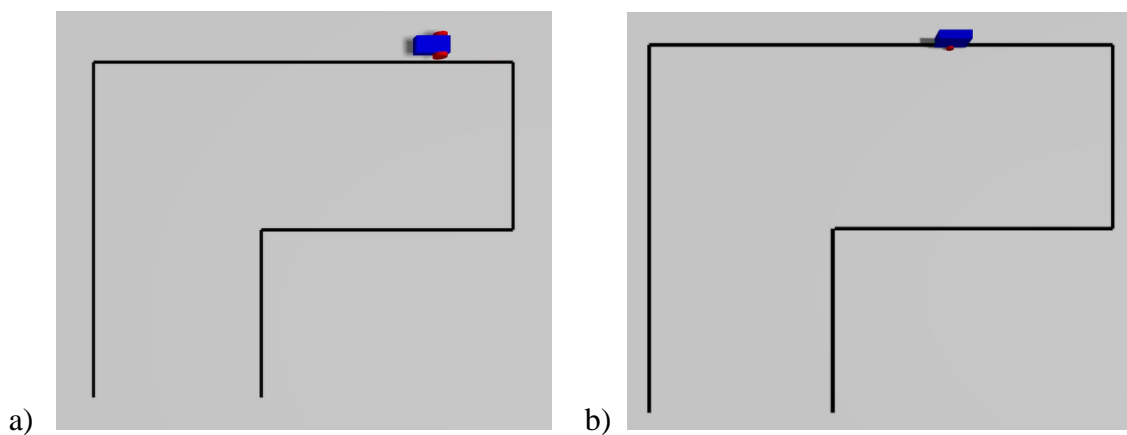


Figure 24. a) AGV Travel without Camera, b) AGV Travel with Camera

Figure 25 illustrates a simulation of the multi-operator scenario example discussed in Chapter 3. The green and yellow stations in the middle and bottom represent operator and packaging stations, respectively. In this simulation, three AGVs transport items from the shelves to the operator stations, while two AGVs transport the sorted items from the operator stations to the packaging stations. The two AGVs wait for the order bins to be fulfilled in the Order AGV buffer zones next to the operator stations.

Once the simulation is initialized, the `/Order_List_Generator` generates the order list in Table 13. `/Order_Fulfillment_Optimizer` subscribes to the generated order list with the initial `/updated_AGV_status` of "all AGVs available." The `/Order_Fulfillment_Optimizer` utilizes Order Clustering through a Genetic Algorithm and Policy 2: Item-Based Buffer Area to group orders, assign them to operators, calculate optimal pickup sequences, and allocate available AGVs. The obtained pick-up sequences are as shown in Table 16. Next, the `/vehicle/travel_info`, which

contains the starting destination positions for each AGV, is sent to the /Path Planner. In this simulation, AGVs 1, 2, and 3 are assigned to Pick-Up sequences 1, 2, and 3, starting from Shelves 1, 2, and 3, as illustrated in Figure 25(a). The AGVs navigate along the designated waypoints via the shortest path, following the AGV lines on the floor using their onboard cameras. As shown in Figures 25(b) and (c), the AGVs arrive at the operator stations successfully. While one AGV occupies an operator station, other AGVs wait in the Donor AGV buffer zone, ensuring a seamless workflow and minimizing idle time so operators can proceed to the next order fulfillment stage. Once the alpha orders in the order bins are fulfilled, AGVs 4 and 5, which have been waiting in the Order AGV buffer zone, transport the sorted items to the corresponding packaging stations, as shown in Figure 25 (d).

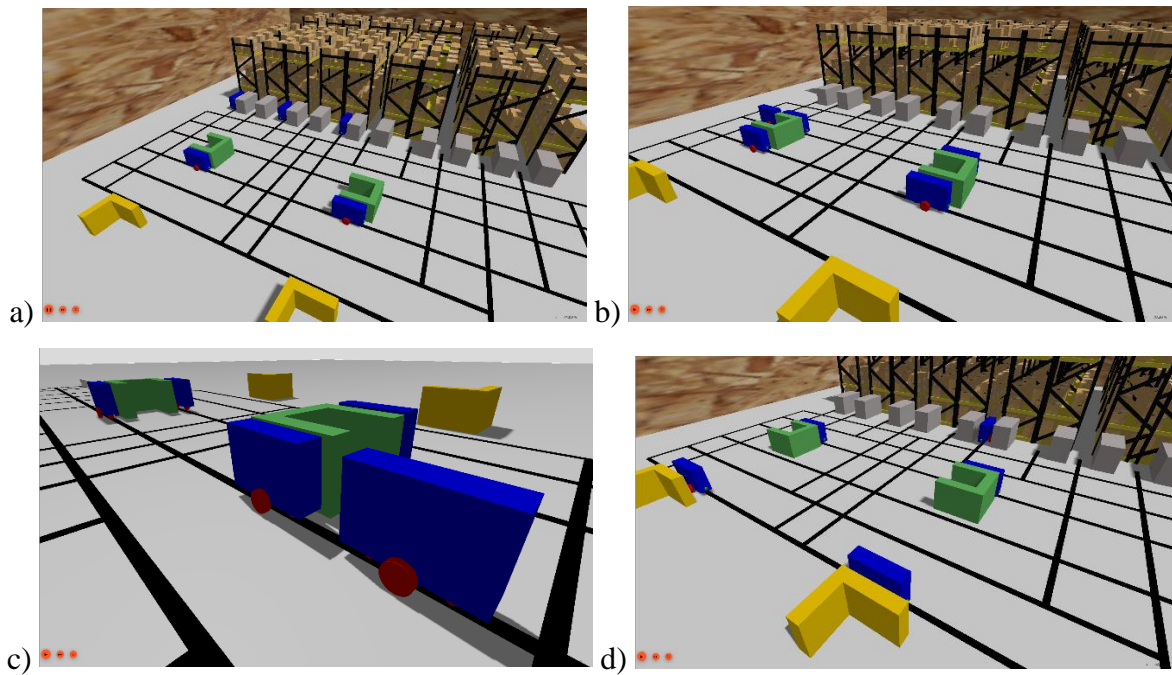


Figure 25. ROS & Gazebo Simulations for Warehouse Operations
a) Initial State, b) Middle State 1, c) AGVs in Buffer Zone, d) Middle State 2

Although the simulation platform operates smoothly for simple scenarios, it is still being actively developed to enhance its robustness, scalability, and stability. The RQT Graph in Figure 22 illustrates the communication behind the simulations and their execution. However, further development is necessary to enhance node communication, ensuring smoother operation. In Chapter 5, "Conclusion and Future Work," future enhancements will be discussed in more detail."

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this study, order fulfillment optimization in an autonomous warehouse has been enhanced by integrating and devising various approaches, including a hybrid of public and private storage, association rule mining, and order batching. Notably, the improved approach enables the method to be applied to large-scale instances, emphasizing the active utilization of buffer bins that serve as personal storage for each operator's items.

Unlike many other studies, FP-Growth-based Association Rule Mining was introduced as an alternative to the Apriori-based approach, as it is more suitable for handling large datasets, such as warehouse order lists. The algorithm was simulated under 2000-order and 1000-item scenarios, demonstrating its capability for efficient similarity calculation with a reasonable computational cost.

Then, two policies were developed for a single-operator scenario:

- **Policy 1: Order-Based Buffer Area**
- **Policy 2: Item-Based Buffer Area**

Both policies share some standard features. First, both policies shuffle the order list based on the calculated similarities. This ensures that orders with similar item compositions are grouped closely, increasing the likelihood of handling multiple orders simultaneously with a single item retrieval.

Both policies were extended to a multi-operator scenario for more helpful applications in real-world warehouses. For efficient order fulfillment by each operator, order clustering is performed using a genetic algorithm to maximize internal similarity between orders within the same cluster. Each cluster is assigned to an operator until the entire order list is fulfilled.

Following the introduction of the proposed order fulfillment methods, a performance comparison was conducted between Policy 1 and Policy 2. Additionally, observations and analysis of the

impact of changes in parameter values for Beta, Alpha, the Number of Buffer Bins, and the Maximum Capacity of Pick-up Totes Are Presented under simulations involving 2000 orders and 1000 items. Policy 1 and Policy 2 demonstrated their efficiency compared to Manual Top-Down fulfillment. In most cases, Policy 2 outperformed Policy 1 regarding total fulfillment time, while Policy 1 resulted in fewer pick-up sequences. Additionally, it was observed that increased Beta, Alpha, and maximum capacity of pick-up totes led to improved warehouse throughput as expected. However, it was realized that a higher number of buffer bins does not always guarantee better performance because of excessive interaction with the buffer bins, which adds extra operational time.

5.2 Future Work

In this thesis, simulations were conducted assuming that AGVs had already delivered all items required by each operator. However, according to the warehouse description in Chapter 3, this assumption is not feasible in a real-world warehouse, even without AGV traffic, congestion, obstacles, or other external factors, due to the distributed nature of the requirements across the operator stations. In actual warehouse operations, multiple operators may request the same type of item simultaneously. If some AGV is transporting the corresponding item totes, that AGV must visit each operator sequentially. This means that while one operator receives the items, the others must wait. As a result of this operational process, inefficient AGV allocation can lead to operator idle time, negatively impacting overall warehouse throughput. Therefore, developing a smart strategy for AGV allocation is crucial—one that optimally assigns AGVs to transport the appropriate items and determines the most efficient sequence of operator station visits to minimize total operator idle time.

The proposed order fulfillment model has room for further development. The current model is solved every time step (or pick-up sequence) to find optimal item retrieval, $x_{o,i}$ for all orders and items in a provided warehouse status. Once optimal $x_{o,i}$ for one time step is achieved, the warehouse status, including items in the order bins and buffer bins, orders assigned to the order bins, and the order list, is updated correspondingly to the retrieved items. Then, the model is solved again with the updated warehouse status to determine the optimal item retrieval for the next step. This process repeats until the entire order list is fulfilled. This approach may not yield global

optimal solutions because it does not consider the impact of the currently retrieved items on later item retrieval calculations. Therefore, developing the model to account for the interdependence among the retrieved items in each time step would be beneficial. In the developed model in that sense, an objective function would be as follows:

$$\min \sum_{t \in T} z_t \quad (27)$$

, where z_t is a decision variable whether any items have been picked up in each time step t (or pick-up sequence t). The equation (27) minimizes the number of pick-up sequences through the entire time steps. The new objective function would also require new constraints. One would be allocating new orders to the order and buffer bins. The most promising approach is to assign orders with the highest fulfillment rate among the Alpha and Beta orders to the order bins and buffer bins, respectively. Furthermore, the decision variable $x_{o,i}$ should be changed to $x_{t,o,i}$ to obtain time-sensitivity. Although the developed model will require a more considerable computational cost due to its higher complexity, it will be more effective at finding the global optimality of the problems, as it solves the problem as a whole rather than breaking it down into individual time steps.

The ROS and Gazebo simulation platform, described in Chapter 4, is currently being developed, with significant progress already made. Each node (or function) performs as expected and successfully utilizes data from other tasks in small-scale instances. The next step involves fully integrating communication through ROS and Gazebo, as illustrated in Figure 22. This can be achieved by converting standalone Python scripts for each function into ROS- and Gazebo-compatible nodes. Additionally, refining controller parameters, such as PID gains and maximum and minimum linear and angular velocities, will further enhance the precision and accuracy of AGV movements, contributing to a more robust and efficient system.

The warehouse is a highly dynamic workplace. Therefore, real-time control is essential for optimization in this frequently changing environment. Digital twin technology enables real-time warehouse monitoring and control of warehouse activities. As reviewed in Chapter 2, a digital twin creates a virtual replica of a physical world, allowing the simulations to predict future outcomes and implement appropriate actions in the real world [44]. The outcomes from the

physical world are then fed into the virtual model, creating a bidirectional data exchange that supports real-time monitoring, simulation, and optimization [20]. Consequently, a digital twin would help main controllers analyze the current situation, such as the locations of AGVs, and predict future scenarios, such as potential AGV traffic and workload for each operator. For example, based on the current positions, allocated tasks, starting positions, and destinations of each AGV, a digital twin can predict potential congestion points and regenerate paths for some AGVs to prevent bottlenecks. The digital twin is widely applicable for AGVs and item retrieval, task allocation, and operator stations. Strong adaptability to real-time control will significantly improve the overall warehouse throughput.

As mentioned in *Chapter 3*, the order fulfillment optimization discussed in this thesis focuses on the first half of the warehouse process, from item retrieval at the item storage to item sorting at the operator stations. The primary goal was to minimize the number of pick-up sequences required to process the orders in the order list. After completing the optimization for the first half of the warehouse, a new optimization method for the second half—encompassing operations from the operator stations to the packaging stations needs to be studied. The primary goal of the second-half optimization is to minimize the total time required to transport the sorted items from the operator stations to the packaging stations. An appropriate allocation strategy for the AGVs will be key to achieving the goal. Similar to what has been done in this thesis, mathematical models will be created to describe the second half of the warehouse, initially to obtain optimal solutions in small instances. Then, heuristic optimization strategies with improved computational efficiency will be developed based on the obtained solutions to achieve optimal solutions in significant cases.

Reference

- [1] K. Lumsden, F. Dallari, and R. Ruggeri, "Improving the efficiency of the hub and spoke system for the SKF European distribution network," *International Journal of Physical Distribution & Logistics Management*, vol. 29, no. 1, pp. 50-66, 1999.
- [2] M. Masae, C. H. Glock, and E.H. Grosse, "Order Picker routing in warehouses: A Systematic Literature Review," *International Journal of Production Economics*, vol. 224, p. 107564, 2020.
- [3] E. BOZ, "The order batching problem: A state-of-the-art review," *Sigma Journal of Engineering and Natural Sciences - Sigma Muhendislik ve Fen Bilimleri Dergisi*, 2022.
- [4] M. CHEN and H. WU, "An association-based clustering approach to order batching considering customer demand patterns," *Omega*, vol. 33, no. 4, pp. 333-343, 2005.
- [5] X. Yuan, "An improved apriori algorithm for Mining Association Rules," *AIP Conference Proceedings*, 2017.
- [6] S. Harikumar and D. U. Dilipkumar, "Apriori algorithm for association rule mining in high dimensional data," *2016 International Conference on Data Science and Engineering (ICDSE)*, pp. 1-6, 2016.
- [7] M. Patil and T. Patil, "Apriori algorithm against FP Growth Algorithm: A comparative study of data mining algorithms," *SSRN Electronic Journal*, 2022.
- [8] T. van Gils, K. Ramaekers, K. Braekers, B. Depaire, and A. Caris, "Increasing order picking efficiency by integrating storage, batching, zone picking, and routing policy decisions," *International Journal of Production Economics*, vol. 197, pp. 243-261, 2018.
- [9] X. Wu and Z. Chen, "Fulfillment scheduling for buy-online-pickup-in-store orders," *Production and Operations Management*, vol. 31, no. 7, pp. 2982-3003, 2022.

- [10] D. S. Hochba, "Approximation algorithms for NP-Hard problems," *ACM SIGACT News*, vol. 28, no. 2, pp. 40-52, 1997.
- [11] Ç. Cergibozan and A. S. Tasan, "Order batching operations: An overview of classification, solution techniques, and future research," *Journal of Intelligent Manufacturing*, vol. 30, no. 1, pp. 335-349, 2016.
- [12] T. Hopstack, "Decentralization of warehouses in the e-commerce age (pros, cons, and types)," Hopstack, 11 August 2022. [Online]. Available: <https://www.hopstack.io/blog/decentralization-of-warehouses>.
- [13] Felix Weidinger, Nils Boysen, "Scattered Storage: How to Distribute Stock Keeping Units All Around a Mixed-Shelves Warehouse," *INFORMS*, vol. 52, no. 6, pp. 1412-1427, 2017.
- [14] M. Kofler, A. Beham, S. Wagner, and M. Affenzeller, "Affinity based slotting in warehouses with dynamic order patterns," *Topics in Intelligent Engineering and Informatics*, pp. 123-143, 2014.
- [15] J. J. B. III, *Warehouse and Distribution Science*, Georgia: Supply Chain and Logistics Institute, School of Industrial and Systems Engineering, Georgia Institute of Technology, 2008.
- [16] Y.-C. Ho and C.-F. Liu, "A design methodology for converting a regular warehouse into a zone-picking warehouse," *Journal of the Chinese Institute of Industrial Engineers*, vol. 22, no. 4, pp. 332-345, 2005.
- [17] Z. Gao, Z. Ling, V. Gupta, and L. Xin, "Real-time omnichannel fulfillment optimization," *SSRN Electronic Journal*, 2022.
- [18] P. J. Xu, R. Allgor, and S. C. Graves, "Benefits of reevaluating real-time order fulfillment decisions," *Manufacturing & Service Operations Management*, vol. 11, no. 2, pp. 340-355, 2009.
- [19] M. Levin, "A real-time control policy to achieve maximum throughput of an online order fulfillment Network," *Transportation Science*, vol. 58, no. 2, pp. 434-453, 2024.
- [20] M. Liu, S. Fang, H. Dong, and C. Xu, "Review of digital twin about concepts, technologies, and Industrial Applications," *Journal of Manufacturing Systems*, vol. 58, pp. 346-361, 2021.

- [21] B. Nedić, "Gartner's top strategic technology trends," *Proceedings on Engineering Sciences*, vol. 1, no. 2, pp. 433-442, 2019.
- [22] C. H. e. al., "Digital twin-assisted real-time traffic data prediction method for 5G-enabled internet of vehicles," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 4, pp. 2811-2819, 2022.
- [23] W. X. W. L. L. C. a. C. W. S. Wu, "Dynamic scheduling and optimization of AGV in factory logistics systems based on Digital Twin," *Applied Sciences*, vol. 13, no. 3, p. 1762, 2023.
- [24] A. ur Rehman, M. U. Ahmed, and S. Begum, "Cognitive Digital Twin in manufacturing: A heuristic optimization approach," *IFIP Advances in Information and Communication Technology*, pp. 441-453, 2023.
- [25] M. E. Kuhl, R. Bhisti, S. S. Bhattathiri, and M. P. Li, "Warehouse Digital Twin: Simulation Modeling and Analysis Techniques," *2022 Winter Simulation Conference (WSC)*, 2022.
- [26] D. Jones, C. Snider, A. Nassehi, J. Yon, and B. Hicks, "Characterising the Digital Twin: A Systematic Literature Review," *CIRP Journal of Manufacturing Science and Technology*, vol. 29, pp. 36-52, 2020.
- [27] K. J. Roodbergen and I. F. A. Vis, "A survey of literature on automated storage and retrieval systems," *European Journal of Operational Research*, vol. 194, no. 2, pp. 343-362, 2009.
- [28] H. Zhou, G. Chen, Y. Lu, X. Cheng, and H. Xin, "A permutation-combination heuristics for crane-based automated storage and retrieval systems considering order fulfillment time and energy consumption," *Mathematical Biosciences and Engineering*, vol. 21, no. 1, pp. 116-143, 2023.
- [29] X. Shan, Y. Jin, P. Li, and K. Kondo, "Modeling and analysis of multi-line orders in multi-tote storage and Retrieval Autonomous Mobile Robot Systems," *2024 IEEE 20th International Conference on Automation Science and Engineering (CASE)*, pp. 1088-1095, 2024.
- [30] J. Gu. M. Goetschalckx, and L. F. McGinnis, "Research on warehouse operation: A comprehensive review," *European Journal of Operational Research*, vol. 177, no. 1, pp. 1-21, 2007.

- [31] M. Mirzaei, N. Zaerpour, and R. de Koster, "The impact of integrated cluster-based storage allocation on parts-to-picker warehouse performance," *Transportation Research Part E: Logistics and Transportation Review*, vol. 146, p. 102207, 2021.
- [32] N. Gademann and S. Velde, "Order batching to minimize total travel time in a parallel-aisle warehouse," *IIE Transactions*, vol. 37, no. 1, pp. 63-75, 2005.
- [33] J. Žerovnik, "Heuristics for NP-hard optimization problems - simpler is better!?", *Logistics & Sustainable Transport*, vol. 6, no. 1, pp. 1-10, 2015.
- [34] J. A. Cano, P. Cortes, E. A. Campo, A. A. Correa-Espinal, "Solving the order batching and sequencing problem with multiple pickers: A grouped genetic algorithm," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 11, no. 3, p. 2516, 2021.
- [35] S. Koch and G. Wascher, "A grouping genetic algorithm for the order batching problem in distribution warehouses," *Journal of Business Economics*, vol. 86, no. 1-2, pp. 131-153, 2016.
- [36] M. Huang, Q. Guo, J. Liu, and X. Huang, "Mixed Model Assembly Line scheduling approach to order picking problem in online supermarkets," *Sustainability*, vol. 10, no. 11, p. 3931, 2018.
- [37] T.-L. Chen, C.-Y. Cheng, Y.-Y. Chen, and L.-K. Chan, "An efficient hybrid algorithm for integrated order batching, sequencing and routing problem," *International Journal of Production Economics*, vol. 159, pp. 158-167, 2015.
- [38] C. Pan, "A comparative study of order batching algorithms," *Omega*, vol. 23, no. 6, pp. 691-700, 1995.
- [39] A. Aboelfotoh, M. Singh, and G. Suer, "Order batching optimization for warehouses with cluster-picking," *Procedia Manufacturing*, vol. 39, pp. 1464-1473, 2019.
- [40] H. Hwang and D. G. Kim, "Order-batching heuristics based on cluster analysis in a low-level picker-to-part warehousing system," *International Journal of Production Research*, vol. 43, no. 17, pp. 3657-3670, 2005.

- [41] S. Katoch, S. S. Chauhan, and V. Kumar, "A review on Genetic Algorithm: Past, present, and future," *Multimedia Tools and Applications*, vol. 80, no. 5, pp. 8091-8126, 2020.
- [42] M. Mousavi, H. J. Yap, S. N. Musa, F. Tahriri, and S. Z. Md Dawal, "Multi-objective AGV scheduling in an FMS using a hybrid of genetic algorithm and particle swarm optimization," *PLOS ONE*, vol. 12, no. 3, 2017.
- [43] F. Neri and C. Cotta, "Memetic algorithms and Memetic Computing Optimization: A literature review," *Swarm and Evolutionary Computation*, vol. 2, pp. 1-14, 2012.
- [44] V. K. Chawla, A. K. Chanda, and S. Angra, "Scheduling of multi load agvs in FMS by modified memetic particle swarm optimization algorithm," *Journal of Project Management*, pp. 39-54, 2018.
- [45] Donald A. Jackson, Keith M. Somers, and Harold H. Harvey, "Similarity Coefficients: Measures of Co-Occurrence and Association or Simply Measures of Occurrence?," *The American Naturalist*, vol. 133, no. 3, pp. 436-453, 1989.
- [46] J. C. Gower, "A General Coefficient of Similarity and Some of Its Properties," *Biometrics*, vol. 27, no. 4, pp. 857-871, 1971.
- [47] F. Bindi, R. Manzini, A. Pareschi, and A. Regattieri, "Similarity Coefficients and Clustering Techniques for the Correlated Assignment Problem in Warehousing Systems," *19th International Conference on Production Research*, 2007.
- [48] Jasmine Irani, Nitin Pise, and Madhura Phatak, "Clustering Techniques and the Similarity Measures used in Clustering: A Survey," *International Journal of Computer Applications*, vol. 134, no. 7, 2016.
- [49] Suphakit Niwattanakul, Jatsada Singthongchai, Ekkachai Naenudorn, and Supachanun Wanapu, "Using of Jaccard Coefficient for Keywords Similarity," *IMECS*, vol. 1, 2013.
- [50] Vinod Kumar and Ramjeevan Singh Thakur, "Jaccard Similarity based Mining for High Utility Webpage Sets from Weblog Database," *International Journal of Intelligent Engineering & Systems*, vol. 10, no. 6, 2017.

- [51] A. Prakash, "Understanding Cosine Similarity: A key concept in data science," Medium, 21 Sep 2023. [Online]. Available: <https://medium.com/@arjunprakash027/understanding-cosine-similarity-a-key-concept-in-data-science-72a0fcc57599>.
- [52] Lailili Muflikhah and Baharum Baharudin, "Document Clustering using Concept Space and Cosine Similarity Measurement," *2009 International Conference on Computer Technology and Development*, 2009.
- [53] Vikas Thada and Vivek Jaglan, "Comparison of Jaccard, Dice, Cosine Similarity Coefficient to Find Best Fitness Value for Web Retrieved Documents using Genetic Algorithm," *International Journal of Innovations in Engineering and Technology (IJJET)*, vol. 2, no. 4, 2013.
- [54] L. Zathrotun, "Comparison Jaccard similarity, Cosine similarity, and Combined both of the data clustering with shared nearest neighbor method," *Computer Engineering and Applications*, vol. 5, no. 1, 2016.
- [55] M. H. Santoso, "Application of association rule method using apriori algorithm to find sales patterns case study of Indomaret Tanjung Anom," *Brilliance: Research of Artificial Intelligence*, vol. 1, no. 2, pp. 54-66, 2021.
- [56] J. Dongre, G. L. Prajapati, and S. V. Tokekar, "The role of Apriori Algorithm for finding the Association rules in Data Mining," *2014 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT)*, pp. 657-660, 2014.
- [57] M.-C.Chen, C.-L. Huang, K.-Y. Chen, and H.-P. Wu, "Aggregation of orders in distribution centers using data mining," *Expert Systems with Applications*, vol. 28, no. 3, pp. 453-460, 2005.
- [58] Utkarsh, "FP growth algorithm in Data Mining," Scaler Topics, 11 June 2023. [Online]. Available: <https://www.scaler.com/topics/data-mining-tutorial/fp-growth-in-data-mining>.
- [59] A. Bloomenthal, "SKU: What it is and how it works," Investopedia, 05 June 2024. [Online]. Available: <https://www.investopedia.com/terms/s/stock-keeping-unit-sku.asp>.