Eindhoven University of Technology

MASTER

Comparing order-picking solutions for a warehouse

Nelemans, M.

Award date:
2021

Link to publication

Disclaimer
This document contains a student thesis (bachelor's or master's), as authored by a student at Eindhoven University of Technology. Student theses are made available in the TU/e repository upon obtaining the required degree. The grade received is not published on the document as presented in the repository. The required complexity or quality of research of student theses may vary by program, and the required minimum study period may vary in duration.

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain

Comparing order-picking solutions for a warehouse

Graduation project at Vanderlande

Eindhoven University of Technology

M.Nelemans

October 2020 - July 2021

Supervisor TU/e: M.A.A Boon

Supervisor Vanderlande: P. Thoonen

Abstract

Due to labour scarcity, aiming for a more automated warehouse can be beneficial for warehouse owners. Automated systems have become more flexible, and require a shorter build up or break down time. This project is carried out at Vanderlande, a global market leader in logistic solutions, and we investigate and compare multiple types of automated warehouses systems. Specifically, we compare Goods-to-Person order picking systems where each product carrier is brought to a workstation only using one non-stationary autonomous vehicle. First, we create a basic mathematical queuing model to get initial insights in a specific part of the order picking process. Then, to obtain detailed results for a more realistic model, we develop a simulation to measure the performance of an order picking strategy. This model is used to determine the best order picking strategy (out of the discussed strategies) for a warehouse. The simulation and its results can be used as a groundwork for an even more complex and realistic model to even more accurately determine the best order picking strategy for a warehouse, and possibly to function as a tool to decide which order picking systems to further develop.

Table of contents

Contents

1 Introduction

Automated storage and retrieval systems are used in many different types of warehouses after they were introduced in the 1950s, with advantages such as increased reliability and saving costs in labour and floor space. A clear disadvantage is the high investment costs, but flexible systems allow for smaller companies to use small systems with a much lower price [1]. Systems like this can be a solution to the order picking process in a warehouse. Order picking is described as one of the most labour-intensive and costliest processes in a warehouse, with an estimated 55% of the total warehouse costs [2]. Therefore, a poor order picking process can lead to unnecessary high costs, which impacts the whole supply chain negatively. Therefore, having an order picking process suitable for the type of warehouse is of great interest.

Vanderlande is a global market leader in logistic solutions, and is known for their process automation at airports, parcels and warehouses. In 2017, Vanderlande was acquired by Toyota Industries Cooperation [3], which is specialized in material handling and logistical equipment; mainly forklift trucks, and the most recent years autonomous vehicles. Recently, these autonomous, self-driving vehicles are growing in development as technology improves [4, 5] (for example vehicles on the road or vehicles used in logistics), and industries can develop a strategic advantage by using these vehicles effectively [6]. However, it is of interest to determine what type of vehicle to develop, and to decide what types of vehicles create the best opportunities for warehouses.

The recent rise in e-commerce is developing the market quickly and requires fast delivery times, flexible return processes, processing orders of small size and offering a large assortment of products [7, 8]. Due to labour scarcity, many warehouses require high levels of automation. These should also be flexible and fast-implemented in order to fulfill the continuously changing needs of the customers. Therefore, it is of interest to develop an autonomous vehicle that suits the needs of the e-commerce segment. E-commerce is common in the GMF (general food and merchandise) segment of Vanderlande, and therefore the main focus will be on GMF warehouses.

Order picking systems can typically be divided in two categories: Person-to-Goods (PtG) and Goods-to-Person (GtP). In this research we will focus on GtP systems, where the goods are brought to pickers, since it is suitable for a fast e-commerce warehouse and reduces labour [9]. Although combinations of GtP and PtG systems exist [10], we will mainly focus on the Goods-to-picker systems, since this category has many interesting possibilities that can be investigated, and by limiting the research scope we can discuss GtP systems in more detail.

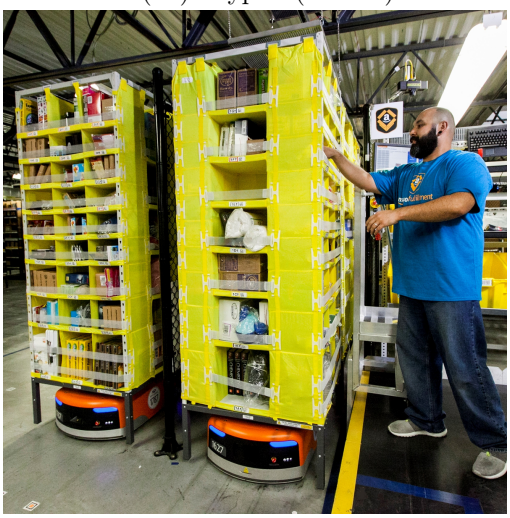
There are many order picking systems that can be classified as a GtP system. To limit the research scope, we will focus on GtP systems where one or multiple product carriers are brought to the workstation by using only one non-stationary autonomous vehicle, in most cases this will be a shuttle. This means no separate cranes, lifts, or conveyors can be used to transport one or multiple product carriers. A small exception is made; product carriers may be transported by a conveyor in front of the workstation, but the conveyor may not do any sorting, and product carriers may not be moved to a different workstation using conveyors. We will now give a few examples of vehicles that meet these requirements. The first example is a Skypod shuttle from Exotec [11] (Figure 1b) , which can travel over floor space and

climb into racks. The second example is a vehicle from Geek+ (Figure 1c), which can carry multiple product carriers at once. This vehicle can not climb in racks, but has a platform which functions as a lift. Therefore, this system cannot use the full height of a warehouse to its advantage, but has a higher carrying capacity. The last example is Kiva from Amazon Robotics [12] (Figure 1d), which can bring an entire rack with different types of products to a picker. An advantage of this system is that it is very flexible (a warehouse owner can easily buy or hire a few more vehicles/racks to increase capacity) and if the storage method is efficient, one can use the fact that there are multiple types of products in a rack to its advantage. Although all these vehicles bring product carriers to a picker, all have certain advantages and disadvantages, and therefore a vehicle might be suitable for one warehouse, but not for the other. Therefore, it is of interest to compare these vehicles, and create a model that can compare the vehicles in a quantitative manner to determine which vehicle is the best choice for a warehouse.

(1a) Adapto (Vanderlande)

(1b) Skypod (Exotec)

(1c) C200M (Geek+)



(1d) Kiva (Amazon Robotics)

Figure 1: Different types of vehicles used in warehouses

Although much literature can be found that gives a performance measure of a system [13], it proved to be challenging to find literature that compares these specific types of vehicles. A few examples are *Ekren and Heragu* [14], who compare a traditional crane-based retrieval system with a tier-captive system (vehicles such as Adapto (Figure 1a) that can only travel and move on one specific height-tier of the storage layout), that uses lifts for vertical vehicle

movement. *Küçükyaşar* et al [15] compare a tier-captive system (which again uses lifts) with a tier-to-tier system (where vehicles can move between tiers, and therefore have vertical movement), and furthermore *Bozer and Aldarondo* [16] compare a kiva system with a miniload system (product carriers are retrieved by a storage/retrieval (S/R) machine, and brought to a workstation by conveyor loops). However, none of these examples compare multiple types of systems like we are investigating. We attempted to search for more literature using terms like *"shuttle warehouse compare"*, *"AGV warehouse comparison"*, *"rack captive comparison"*, *"AVS/RS compare"* and *"compare performance warehouse system"*, but could not find any literature that compares multiple systems like we are interested in.

Given that we could not find any literature that compares the order picking systems of interest, the goal of this research is to create a model to compare Goods-to-Person systems in a quantitative manner. The first step is to determine the exact order picking strategies we are going to compare. The next step is to determine the performance of each system, and to quantify this we will use key performance indicators (KPI's). Once defined, we will create a model of each GtP system, and compute the KPI's. Then, once we have a model for each GtP system, we can select a certain warehouse and determine the most suitable GtP system for this warehouse. To summarize, the main research question is:

- *For a new warehouse, with all its given information, decide the best order picking strategy (out of the discussed strategies) for this warehouse.*

The report will have the following structure. First, Chapter 2 will give an introduction to warehousing, and explain some important processes for this research. We also further specify the exact research goal, namely which types of order picking strategies we are going to compare. To answer the research question, we started by creating a mathematical queuing model, which is discussed in Chapter 3. However, to create a more complex model, it turned out we needed a simulation model, which is explained in detail in Chapter 4. The results of this simulation are displayed and interpreted in Chapter 5. Finally, we give a conclusion of the research in Chapter 6, and discuss the weaknesses and possible improvements. This last chapter also includes the recommendations for Vanderlande, who proposed this research topic.

2 Introduction to warehousing

A warehouse is a commercial building that stores goods. Driven by a competitive market and lack of labor, many companies want to improve the efficiency and lower the (future) costs of their warehouse, and therefore invest in automation. There are many solutions to automate the warehouse processes, such as conveyors, automated storage and retrieval systems, and many more. In this report, the focus is the comparison of various automated storage and retrieval systems. Therefore, the warehouse process is simplified as shown in Figure 2.

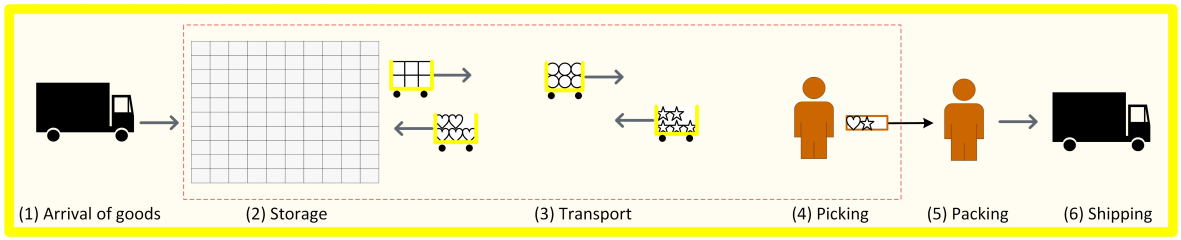


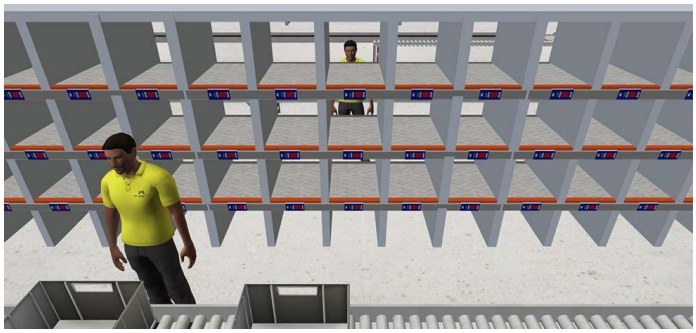
Figure 2: Processes in a warehouse

(1) Arrival of goods: In this first step, goods arrive at the warehouse, possibly delivered by other parties (either new goods coming from suppliers, or returned goods coming back from customers). In most warehouses, products that arrive will first go through a quality check to ensure the right products in the right amounts are delivered. Then, a storage location will be assigned according to a storage technique; for example creating chaos (putting products in random locations) or storing goods closer to the workstations depending on how commonly it is ordered. Then, goods are transported to the assigned location. All aspects of this process, except the storage technique which is relevant for process (2) as well, are beyond the scope of this research.

(2) Storage: Goods are stored in certain racks or shelves, varying between high, stationary racks that utilize much building height, or flexible racks that can be moved by vehicles. There are many distinct products, and a distinct product with all its attributes is called a stock keeping unit (SKU). In this warehouse process, it is of interest to know where all products are located (which depends on the storage technique described in (1)), since this determines the time to retrieve a (set of) product(s). Furthermore, we assume that all products are stored in "product carriers", where a product carrier is a tote, box, (plastic) bag, shelf, or any other type of carrier that can carry any number of products.

(3) Transport: To fulfill an order, product carriers will be retrieved from storage and brought to a workstation (process (4)). In this research, we will look at Goods-to-person solutions only, which means the retrieval of the product carriers will be done by automated vehicles. These vehicles bring the product carrier(s) to the workstation, and this means operators do not have to walk long distances to retrieve products.

(3a) Workstation



(3b) Put wall

Figure 3: Different type of workstations

(4) Picking: Picking stations are ergonomically designed stations where product carriers arrive and orders can be consolidated. Consolidation means that products of an order are combined and placed in an order carrier (again a carton (box), envelope, (plastic) bag or any other suitable packaging material that can contain all products of an order), making this order carrier a coherent whole: exactly one order. There can be multiple picking stations (which we will refer to as workstations) in a warehouse, and every active workstation has an operator. An operator is a trained person who can consolidate orders. Goods for orders will be delivered at the workstation in product carriers. In the workstation, a product carrier is most often automatically identified and then presented to an operator. An operator can then pick up a product from the product carrier, and a screen (for example) will show the instructions that indicate where the product needs to go. The operator will move to the order carrier and place the item inside. When all items are inside, the order is completed ("consolidated") and the order carrier is moved to the next stage, which is packing. Then, an operator can pick up a new product and new instructions are indicated on the screen. Using this method, all items are placed in the correct boxes, and all orders are properly consolidated.

(5) Packing: After the operator at the workstation consolidates an order, this order carrier containing all products of an order is brought to the packing process. During this process, the order carrier is weighted, packed, and a label is printed to label the order carrier. The new trend is to automate this process, but a manual process, where a second operator does the final preparations before shipping, is still quite common. In Figure 2 a manual process is visualized, but in this research we do not specify how the packing process is dealt with. All processes in a warehouse that are *after* the consolidation of orders at a workstation, including this process, are beyond the scope of this project.

(6) Shipping: The final process is shipping. After packing, order carriers are moved to a shipping area. Some warehouses have a temporary storage where carriers are stored until the required truck or delivery vehicle arrives. Another important process is to sort the order carriers properly, for example to collect multiple carriers on a pallet or container, or to bring order carriers to the right delivery vehicle.

This means that for the order picking process, the storage of goods, retrieval of the goods and the picking stations (workstations) are processes that are investigated in this research (marked in red in Figure 2). We will now turn our attention to different approaches for all of

these three process categories.

2.1 Order picking categories

One of the goals is to compare multiple approaches to order picking, and in this section, we establish which approaches of each category are going to be compared. Figure 2 illustrates the categories of interest in this research, which are storage, retrieval and picking. Therefore, to compare order picking approaches, we will create variations of each of the categories. Table 1 gives a comprehensive overview of which variations of each category could be compared.

The first category is the storage of products, and a meaningful variation on this process is to vary what a vehicle can carry. Either the vehicle can carry a single product carrier (like Skypod), the vehicle can carry multiple product carriers (like the C200M vehicle from Geek+) or an entire rack (like Kiva). While introducing the vehicles it was mentioned that a different vehicle can require a different storage type (for example the storage height), and in this way we take the storage type and retrieval process into account.

The second category is order picking, and to compare multiple variations, single order picking and batch picking can be compared. These two processes vary in the way orders are retrieved and in what manner product carriers are brought to a workstation. Section 2.1.1 further elaborates what these concepts mean.

The last category is the workstation, and to compare different variations of this concept, we will vary the number of orders that can be handled at once at a workstation. We take three realistic versions of an $1 : n$ station, which means a product can be moved from one location (the product carrier) to one of the n pick-to locations (the order carriers). The first $1 : n$ station will be a put wall (Figure 3b), which can accept around 50 orders at a time which means $n \approx 50$, the second $1 : n$ station is a medium-sized workstation like "PICK@EASE" (Figure 3a), which can only accept a few orders at a time ($n \approx 5$), but will be notably faster, and the last station is a $1 : 1$ station ($n = 1$), where the operator can move a product from only one product carrier to only one possible order carrier.

We will now elaborate on the three order picking categories and their possible variations separately.

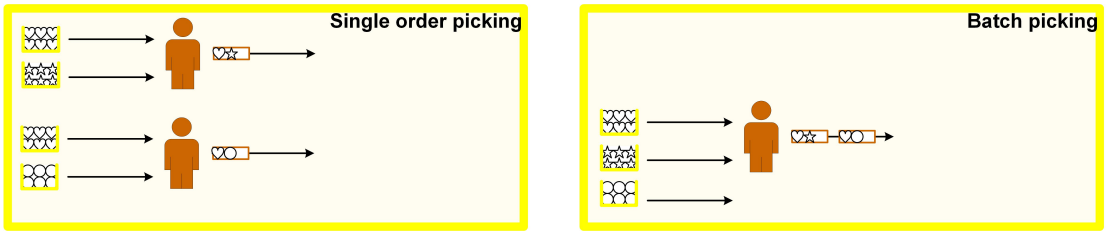
Parallel picking method	**Vehicle carrying capacity**	**Workstation capacity**
• Single order picking, or • Batch picking	• Carries 1 product carrier, or • Carries multiple product carriers, or • Carries a rack	• 1-to-1 station, or • Medium workstation, or • Put wall

Table 1: Possible configurations for order picking strategies

2.1.1 Parallel picking method

The simplest parallel picking method is single order picking, where each item is used for exactly one order. Although simple, this method can certainly be improved, since it is likely

that an item is required in multiple orders. Batch picking is such an improved method. Batch picking, as the other parallel picking method, is a strategy to gain more efficiency in the product retrieval process. This is done by grouping the orders into groups (called "batches"), such that the orders in one batch have many products in common. If a certain item (product) is included in multiple orders, a single retrieved product carrier with this item, can fulfill (part of) multiple orders. Figures 4a and 4b show a small example of this. With single order picking (Figure 4a), there are four vehicle movements, since for each order, all product carriers need to be retrieved separately. However, for batch picking (Figure 4b), there are only three vehicle movements because in this case, the product carrier with the heart-shaped product only needs to be retrieved once. The factor that indicates how many orders an item can fulfill, is called "synergy". A batch-picking algorithm with a synergy factor of X, would mean that in that batch, each item (product) can be used for X orders on average. Generally, a high synergy factor is preferred, since less (time-expensive) vehicle movements are required to achieve a certain number of completed orders per time unit.



(4a) Single order picking: 4 vehicle movements

(4b) Batch picking: 3 vehicle movements

Figure 4: Vehicle movement in different parallel picking methods

Yang et al [17] describe order batch picking problems, and discuss algorithms on how to solve them. The authors mention that so far, only few literature discussed optimization problems when there are multiple storage locations, which is the scenario we are investigating, and the authors aim to close this gap. In section 3.2 of the article, a mathematical model is proposed that can be extended to optimize the batch-picking strategy: batches can be chosen such that product carriers in a batch are on an optimal route, and are positioned in close proximity to one another. The batching of orders to minimize the total travel time (which is widely used in Person-to-goods systems) can be described as the vehicle routing problem, which is a generalization of the travelling salesman problem (TSP). *Gademann & Velde* [18] prove the problem is NP-hard when a batch contains more than one order. However, creating and investigating an optimal batch picking algorithm is outside the scope of this project, and therefore we will use a relatively intuitive batch picking method. This method is further explained in section 4.2.3, where the simulation model is discussed in more detail.

2.1.2 Vehicle carrying capacity

The vehicles we are investigating can be divided in three categories; it carries either a single product carrier, multiple product carriers, or a rack. A well-known example of a vehicle that can carry a rack is Kiva from Amazon robotics (Figure 1d). This rack has multiple different products inside, and therefore many different products can be delivered to an operator at once. The introduction also shows an example of a vehicle that can carry multiple product carriers, in this case from Geek+ (Figure 1c). This vehicle has a shelf that can move vertically

and pick up product carriers, and a few storage shelves to carry products during transport. These three systems work very differently, as many item types in a Kiva rack are not actually required at a workstation, while the other two types of vehicles can more specifically choose what item types to retrieve.

(5a) InVia robot pickers

(5b) Skypod (climbing in rack)

Figure 5: Vehicles with different picking mechanisms

It is of importance to investigate how the vehicle deals with product carriers on different heights. If the vehicle can utilize little to no vertical height, a solution would be to find an alternative way of using the warehouse height, or to use more "floor space" and use a building larger in length or width, which can be more expensive. There are a lot of ways a vehicle can work with vertical movement. An example is a mechanism that can extend, that is used by, for example, InVia robot pickers (Figure 5a). Another mechanism is to climb in custom racks, which is for example used by Skypod, as shown in the introduction (Figure 5b). In this research, a generalization will be used that parametrizes the time for a vehicle to move vertically and retrieve a product carrier.

2.1.3 Workstation capacity

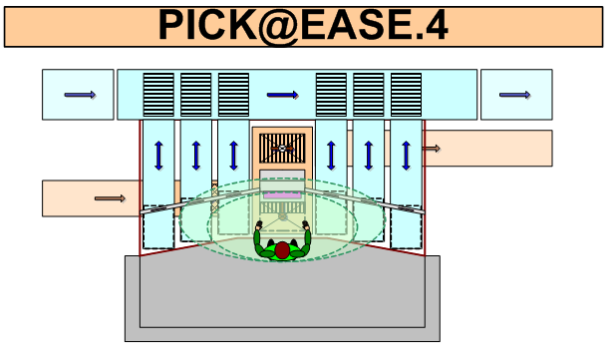
At a workstation, an operator moves one or multiple products from a product carrier or a rack to an order carrier or compartment in a put wall. There are multiple types of $1 : n$ workstations, and in this research we will look at three realistic options for n. The first one is a station with a put wall ($n \approx 50$). This put wall can have a varying number of compartments (locations where items for different orders can be placed), and a put wall can be accompanied with technology to efficiently tell the operator where to put a certain item. The most common system will detect which item an operator is holding, and it turns on a light at the compartment(s) where that specific item has to be placed. Compared to the other workstations of interest, at this workstation the operator has the most work since the walking and reaching distance is the highest, and the operator has to do part of the sorting; putting the items in the correct compartment.

The second station is a 1-to-1 station ($n = 1$), where the number of pick-to locations for order carriers is one. This means the operator is handed one product carrier, and can place a product in only one possible order carrier. In this process, the operator never has to check in which order carrier the product needs to go, and never has to walk. Therefore, the operator

can reach a high operational productivity. However, while an order is active, only product carriers belonging to one order can be delivered to the operator. Therefore, vehicles have to do the sorting process by delivering product carriers to the correct workstation.

The last workstation is a medium-sized one, inspired by the PICK@EASE 4 workstation by Vanderlande (Figure 6a). This station has a capacity of $n \approx 5$ order carriers, meaning that the operator places an item from the product carrier into one of the five order carriers. Although the operator does not have to walk and move a lot, which saves a lot of time, the operator still has to process the information that the screen shows, and cannot instantly put an item in an order carrier, which is possible with the 1-to-1 station. This means the operator will most likely work somewhat slower at this workstation than the 1-to-1 station, but is most likely faster than the put wall. Similar to the put wall, operators at this workstation have to do part of the sorting process, by putting the product in the correct order carrier.

(6a) PICK@EASE 4 Demonstrator



(6b) PICK@EASE 4 process diagram

Figure 6: The Pick@Ease 4 workstation

2.1.4 Combinations of interest

Now that the three categories and its variations are further elaborated, we want to define what combinations of variations are interesting to investigate. Table 1 shows 3 categories with relatively 2, 3, and 2 different variations, which means there are in total 12 unique combinations of settings. However, some of these combinations are not really interesting or do not make much sense to investigate, and therefore, in this section we will discuss what combinations will be investigated, and which will not.

First of all, when a vehicle carries a rack, a logical combination would be to combine it with batch picking and a put wall. A vehicle retrieving a rack while single order picking will be highly inefficient, because the strength of getting a rack is to get multiple items (of multiple orders) to the workstation at once. With single order picking, it is possible to complete an order in a relatively small time frame, since all products arrive around the same time. Therefore it makes sense to combine this approach with a small workstation. Therefore, we combine single order picking with the 1-to-1 station or a medium-sized workstation. With batch picking however, we expect many items of many different orders to arrive at approxi-

mately the same time. If the workstation would be small, it would lack capacity to deal with all the product carriers of different orders that arrive, and most likely give an increase in vehicle waiting time. A put wall however is designed to deal with this issue, and therefore we will combine batch picking with a put wall. Table 2 illustrates which combinations we will investigate, and by peer-review, we confirmed that these combinations are indeed the most interesting to compare. We will refer to each combination as a configuration.

To compare the combinations fairly, we need a way to measure how well a certain combination performs. Therefore, we will now describe some Key Performance Indicators (KPIs), which can be used to measure how well a combination performs.

nr	Parallel picking method	Vehicle carrying capacity	Workstation capacity
1	Single order picking	Carries 1 product carrier	1-to-1 station
2	Single order picking	Carries 1 product carrier	Medium workstation
3	Single order picking	Carries multiple product carriers	1-to-1 station
4	Single order picking	Carries multiple product carriers	Medium workstation
5	Batch picking	Carries 1 product carrier	Put wall
6	Batch picking	Carries multiple product carriers	Put wall
7	Batch picking	Carries 1 rack	Put wall

Table 2: Combinations to compare for order picking approaches

2.2 Key performance indicators

To measure the performance of a system in a quantitative manner, some performance measurements are required. In literature [19, 20], two key performance indicators (KPIs) are very common. The first KPI is the throughput [21, 22], which is the number of items (products) the system can output in a fixed time frame, which is related to the vehicle and operator cycle time. The second indicator is the utilization of lifts (in a tier-captive system) and vehicles, and because tier-captive systems are beyond the research scope, only the utilization of vehicles is one of the performance indicators. This measurement is equivalent with the vehicle idle time. *Ekren* et al [14] mention both of these KPIs, but include the cost as well. This key performance indicator is very important, since most companies want to keep the cost as low as possible. *Roy* et al [23] mention the throughput and utilization as well, but also measure how many items are in the buffer, which can be any type of place or location where goods are temporarily stored in the system. Some of the models in this research will use a buffer, and therefore this KPI will be taken into account. The order lead time (for an order, the time between entering and leaving the system) is another measure that is of importance [24], but is less commonly found in literature. Additionally, it turned out this KPI was hard to model and implement, and because of time constraints we decided to not use this KPI to measure the performance of a system. Finally, *Ten Homel & Thorsten* [25] discuss many other KPIs that might be interesting for a warehouse, but most of these, such as *readiness to deliver* and *warehouse fill degree*, are not relevant if we want to compare certain systems. Therefore, we will not create unnecessary complexity and only use the KPIs mentioned above. In short, these KPIs are:

- Throughput; how many items are finished per fixed time frame.

- The mean queue length of all queues; this indicates where vehicles are located or possibly waiting. This gives insight into how many product carriers are placed in the buffer (conveyor in front of the workstation) and how many vehicles are idle.
- How many vehicles and workstations are used (this is moderately equivalent with the cost)
- Order synergy; this indicates how efficient the batching-algorithm works, and how much synergy can be reached when batch-picking with a certain combination of parameters. A low synergy is not preferable, but might be necessary for completing (many) urgent orders in time.

3 Mathematical model

In this chapter, a mathematical queuing model will be used to analyze a simplified scenario in a warehouse. After describing the scenario, we will describe how the mathematical queuing model will be structured and the assumptions that will be used. For this queuing model, we are interested in finding the mean waiting time for an order in a queue. We will approximate this mean waiting time using a M/G/c queue, however for this approximation we need to find the mean and variance of the service time, which we will compute as well.

When a customer places an order, this order will be placed on a list of all orders currently in progress. A fixed set of vehicles will retrieve the orders on the list, prioritizing orders with a higher urgency. Every order has a certain urgency; some orders need to be completed within one day, while other orders have less strict time requirements. The vehicles retrieve orders independently, meaning that when a vehicle finishes retrieving all items for an order, it will request the next order on the list and retrieve all items of this order without assistance of other vehicles. The vehicle type in this scenario can carry only one product carrier at once. A vehicle starts at a certain "origin" point, and then moves to the storage, picks up one product carrier, and proceeds to the workstation to drop off the product carrier. Then, it moves back to the origin point to either retrieve the next items of the order, or wait for a new order to be assigned. This process is also visualised in Figure 7. An operator at the workstation will place the items from the product carrier to an order carrier. When the operator has placed all requested items in the order carrier, the order can be sent to the packing process.

This scenario can be modelled using an open queuing model where an arriving customer (in this case an order) is served by two different servers. The first servers are the vehicles, which are c parallel servers where c is the number of vehicles in the system, and suppose the occupation rate [26] is smaller than one. An order is assigned to a vehicle, and the vehicle service time represents the time a vehicle takes to retrieve all product carriers of an order. When the vehicle has retrieved all product carriers, the order arrives to the next servers which are the workstations, which are w parallel servers where w is the number of workstations. At a workstation, an operator consolidates an order, and this time is the service time of this second server. When the order is consolidated, the order leaves the system. To model this scenario, we assume that we can use an exponential inter arrival time with mean $1/\lambda$ for arriving orders. The service time for the first server (the time for a vehicle to retrieve all items of an order) has mean $1/\mu$, but its distribution is unknown. To prevent the model from being too complex, we also assume we have an infinite number of workstations. This means a customer (order) never has to wait in front of the workstation, and can always be served by this 2nd server. With this assumption, the focus is on the performance of the vehicles. The final queuing model that will be analyzed is visualized in Figure 8.

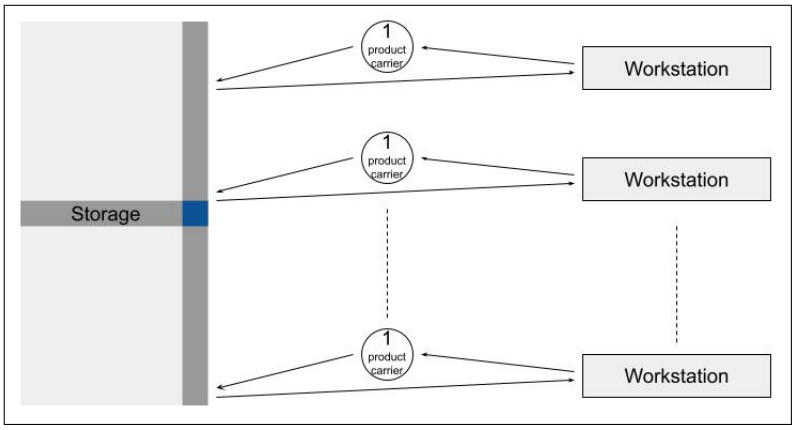


Figure 7: Overview of picking process

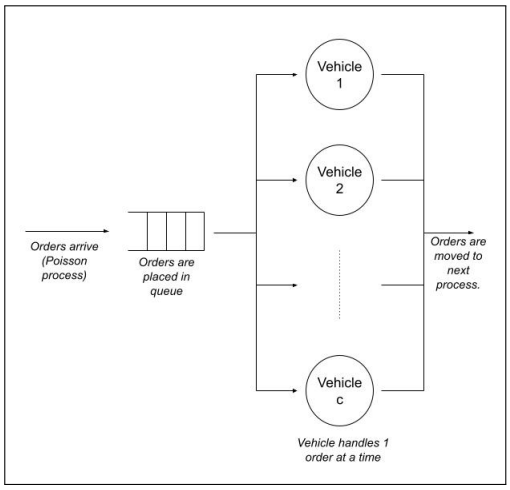


Figure 8: Queuing model

3.1 Design choices and assumptions

While creating a mathematical model, some assumptions have to be made to not make the model too complex. In this section, we will describe all the assumptions to ensure tractability of the model, and further explain some other design choices like the layout of the warehouse and the number of items in an order.

As mentioned before, the first assumption is that we have an infinite number of workstations. When all items of an order are delivered to any workstation, the order is instantly consolidated, and therefore finished. This means the process at the workstation is not taken into account, and neither are all other processes outside the order picking process of the vehicles. This means the processing time at the workstation is also not taken into account, as this is not relevant since there is an infinite number of workstations, and therefore no waiting time for these queues. We also assume there are no "errors" during the process; the vehicle never retrieves a wrong product, breaks and/or malfunctions, and all orders are consolidated correctly. It is important to note that in reality such errors occur and should be taken into account.

Now we move on to assumptions about the order picking process of the vehicle. When a vehicle is initiated to retrieve an item, we assume that this item is always available, even if another vehicle is picking this item as well. Although a specific product type is possibly found multiple times in storage (5 times for a fast mover, 3 for a medium mover and once for a slow mover), product carriers will never be empty and therefore vehicles will always prioritize the closest product carrier possible. The product carriers are stored according to the chaotic storage method, which is a suitable storage method for complex e-commerce warehouses, since it improves chances to find a product near the origin, or near another type of product. The process of bringing product carriers back from the workstation to storage is not taken into account. Vehicles will travel at a constant speed, and the acceleration or deceleration is not taken into account. vehicles can always move in the most efficient path possible, and do not have to consider other vehicles' location; all vehicle movement is modelled independently of other vehicles' location and movement. Charging of vehicles is not considered in this model.

Furthermore, we assume that the arrival rate of orders does not change throughout the day. Additionally, we assume that the arrival of orders can be modeled using a Poisson process with rate λ. All orders have the same priority to be picked, and we do not take the weight and size of the items into account, we only assume that all items can be transported by a vehicle in the same manner and speed. We assume that the geometric distribution can be used to model the number of items in an order. When comparing this to findings in literature [16], this distribution appears to be sufficiently suitable for this model. This is also visualized in Figure 9. Picking a specific item in an order is done independently of all other items in this order, where every item has a fixed probability to be chosen (higher for fast movers, for example). To determine the exact likelihood an order contains a certain product type, we used a rule of thumb, which is related to the Pareto curve [27]. This rule states that 80% of all orders cover 20% of the SKUs, 15% of all orders cover 30% of the SKUs, and the last 5% of the orders cover the remaining 50% of all SKUs.

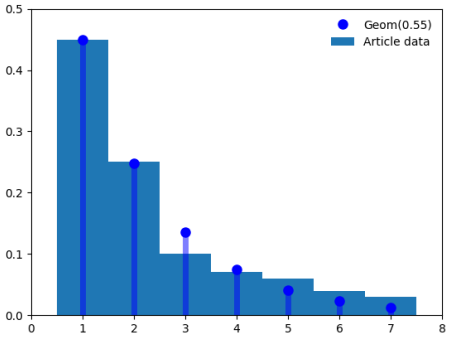
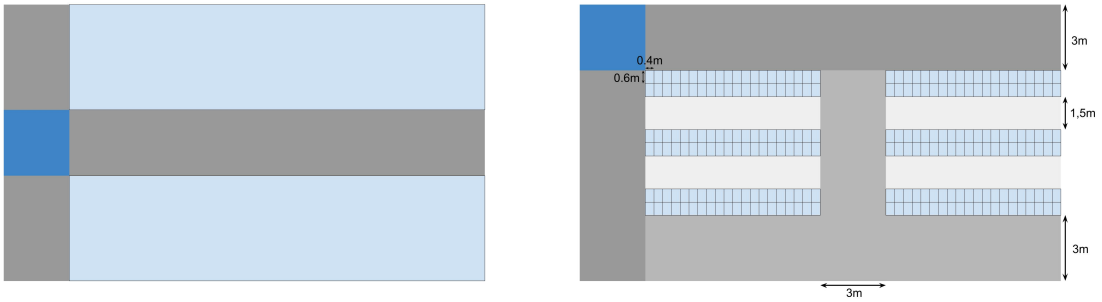


Figure 9: Comparing the chosen distribution with literature

The final design choice is about the layout of the warehouse. For this model, we choose one specific warehouse to imitate. The total setup is visualized in Figure 10a, where the light blue square is shown in detail in Figure 10b. The product carriers are assumed to have size 600cm by 400cm, which is the standard product carrier size of Vanderlande. In the middle left is a dark blue square, and this is the origin point and simultaneously the point where all workstations are located. In this warehouse, there are three horizontal and two vertical dark gray lanes, these are broad lanes of width $3m$, which is floor space where vehicles can travel to the target destination in storage. The broad lanes demarcate four smaller areas (blocks). In one block, there are two horizontal smaller lanes of width $1,5m$. These are the smaller roads that vehicles can use to travel to the target destination. In one block are 60 product carriers in horizontal direction and six product carriers (two next to every lane, plus two next to the broad lanes on both edges) in the vertical direction. This means there are 4 blocks of 360 product carriers, giving a total of 1440 spaces on the floor to place product carriers. Product carriers can be placed 20 high, therefore this gives $20 \cdot 1440 = 28800$ product carrier locations. We assume that a fast mover can be found in 5 different locations, a medium mover 3 times and a slow mover only once. Therefore, with 10.000 SKU, at least $10.000(0.2 \cdot 5 + 0.3 \cdot 3 + 0.5 \cdot 1) = 24000$ product carrier locations are required, meaning that this warehouse layout (with 28800 product carrier locations) satisfies this requirement.

Based on data given by Vanderlande, we used an average speed of $2.2m/s$, which can be

used to compute the mean and variance of the time to travel from the origin, to a storage location, to the workstation and back to the origin. Using the described layout, the mean is $12.20896s$ and the variance is $66.417s$. *Sui* et al [20] describe a more complex method to model the retrieval speed, however, for this project we decided to keep the model rather simple and use the average speed. The time to pick up an item from storage and drop off an item at a workstation is set on $8s$, which is again deduced from data given by Vanderlande. The variance is set to $1s$, however no data was used to support this claim.



(10a) Layout of a segment

(10b) Overview of the total layout

Figure 10: Overview of the layout of a warehouse

3.2 Service time

Now that the scenario and its assumptions are defined, the next step is to compute the mean and variance of the service time. The (random) variables and parameters used in the computation can be found in Table 3. The goal is to compute the mean and variance of the total time to handle an entire order, which is Z. We have $Z = Y_1 + ... + Y_I$, where I is the total number of items (in this case 10.000) and Y_i denotes the total time a vehicle spends to retrieve all products of type i. This means that if product type j is not included in an order, $Y_j = 0$ for this order. Note that Y_i are dependent variables. R_i is the total travelling time of a vehicle to retrieve all products of type i, where R_i is independent and identically distributed (i.i.d.) for all $i \in I$. W_i is the time for a vehicle to pick up and drop off all items of type i, and this distribution is i.i.d. for each i. Moreover, we assume that the random variable G represents the total number of items in an order, and that G is geometrically distributed with parameter g. The tuple $(G_1, ..., G_I)$ denotes the number of times each product is selected in an order, where $G_i \in \mathbb{N}$ and $\sum_i Gi = G$, which means G_i are dependent. There are possibly multiple items in an order, each item is selected using the categorical distribution, where product type i has probability p_i to be selected. These probabilities are fixed and are determined based on the Pareto curve. In this case, this means p_i is $\frac{0.8}{0.2 \cdot 10000}$ for $0 \leq i \leq 2000$, it is $\frac{0.15}{0.3 \cdot 10000}$ for $2000 \leq i \leq 5000$ and finally $\frac{0.05}{0.5 \cdot 10000}$ for $5000 \leq i \leq 10000$.

Abbreviation	Description
I	Total number of items available
p_i	Probability that item i is picked, when choosing 1 particular item for an order.
G	Total number of items in an order (random variable)
g	Let G be geometrically distributed with parameter g, where $0 < g < 1$.
$(G_1, ..., G_I)$	Counts number of times item i is picked when choosing an entire order.
$(Y_1, ..., Y_I)$	Total time spent to retrieve products of type i when handling an entire order.
Z	Total time spent to handle an entire order.
R_i	Driving time of vehicle to retrieve a product of type i.
W_i	Time for vehicle to pick up and drop off an item of type i.

Table 3: (random) variables and parameters used in the mathematical model

We already have $Z = Y_1 + ... + Y_I$, and now we are looking for a closed expression for Y_i. The total time spent on all products of type i consists of two parts: vehicle travel time, and picking up and dropping off the items. We have R_j if product type j is included in the order, and zero otherwise. Therefore, this part can be expressed as $R_i \mathbb{1}_{\{G_i>0\}}$ for each item i. Then we have a total time to pick up and drop off an item (W_i), which are actions for each item in the order; if $G_i = 2$ for i, then the total time spent on picking up and dropping off is $2 \cdot W_i$. Therefore, this part can be denoted as $\sum_{j=1}^{G_i}(W_i)_j$ for each i. In total, we have $Y_i = R_i \mathbb{1}_{\{G_i>0\}} + \sum_{j=1}^{G_i}(W_i)_j$. When computing the expectation of Z, we use Wald's identity, which states:

Theorem 3.1. *(Wald's identity)*
Let $(X_N)_{n\in\mathbb{N}}$ be a sequence of real-valued, independent and identically distributed random variables and let N be a non-negative integer-value random variable that is independent of the sequence (X_N). Suppose that N and the X_n have finite expectations. Then

$$\mathbb{E}[X_1 + ... + X_N] = \mathbb{E}[N]\mathbb{E}[X_1]$$

In our case we have real-valued, i.i.d. sequence $((W_i)_j)$, and G_i a non-negative integer-valued random variable. $(W_i)_j$ does not depend on G_i for each i, and we assume the expectation of $(W_i)_j$ is finite. We know the expectation of G_i is finite since it is bounded by the expectation of G, a geometrically distributed random variable with parameter $g < 1$, which means the expectation of G (which is $\frac{1}{1-g}$) is finite. This means we can apply Wald's identity on $\sum_{j=1}^{G_i}(W_i)_j$, giving:

$$\begin{aligned}
\mathbb{E}[Z] &= \mathbb{E}[Y_1 + ... + Y_I] \\
&= \sum_{i=1}^{I} \mathbb{E}[R_i \mathbb{1}_{\{G_i>0\}} + \sum_{j=1}^{G_i}(W_i)_j] \\
&= \sum_{i=1}^{I} (\mathbb{E}[R_i]\mathbb{P}(G_i > 0) + \mathbb{E}[G_i]\mathbb{E}[W_i]) \\
&\overset{(*)}{=} \sum_{i=1}^{I} (\mathbb{E}[R_i]\frac{p_i}{1-g(1-p_i)} + \mathbb{E}[W_i]\frac{p_i}{1-g}).
\end{aligned}$$

A detailed elaboration on the expectation (Lemma A.1) and probability (Lemma A.2.1) used at $(*)$ can be found in the appendix. By definition we have $\mathrm{Var}(Z) = \mathbb{E}[Z^2] - \mathbb{E}[Z]^2$. The term $\mathbb{E}[Z]$ is already known and therefore the main focus lies on $\mathbb{E}[Z^2]$.

$$\begin{aligned}
\mathbb{E}[Z^2] &= \mathbb{E}[\Big(\sum_{i=1}^{I}\Big(R_i\mathbb{1}_{\{G_i>0\}} + \sum_{j=1}^{G_i}(W_i)_j\Big)\Big)\Big(\sum_{k=1}^{I}\Big(R_k\mathbb{1}_{\{G_k>0\}} + \sum_{m=1}^{G_k}(W_k)_m\Big)\Big)] \\
&= \mathbb{E}[\Big(\sum_{i=1}^{I}R_i\mathbb{1}_{\{G_i>0\}}\Big)\Big(\sum_{k=1}^{I}R_k\mathbb{1}_{\{G_k>0\}}\Big) + 2\Big(\sum_{i=1}^{I}R_i\mathbb{1}_{\{G_i>0\}}\Big)\Big(\sum_{k=1}^{I}\sum_{j=1}^{G_k}(W_i)_j\Big) \\
&\quad + \Big(\sum_{i=1}^{I}\sum_{j=1}^{G_i}(W_i)_j\Big)\Big(\sum_{k=1}^{I}\sum_{m=1}^{G_k}(W_k)_m\Big)].
\end{aligned}$$

The expectation of Z^2 consists of three different cross terms, and we will elaborate them separately. To expand the first term, notice that R_i and G_j are independent for all i, j, and move the expectation inside the sums and use that the expectation of an indicator can be written as a probability. Then, split the cases $i = k$ and $i \neq k$.

$$\begin{aligned}
\mathbb{E}[\Big(\sum_{i=1}^{I}R_i\mathbb{1}_{\{G_i>0\}}\Big)\Big(\sum_{k=1}^{I}R_k\mathbb{1}_{\{G_k>0\}}\Big)] &= \mathbb{E}[\sum_{i=1}^{I}\sum_{k=1}^{I}R_iR_k\mathbb{1}_{\{G_i>0\}}\mathbb{1}_{\{G_k>0\}}] \\
&= \sum_{i=1}^{I}\sum_{k=1}^{I}\mathbb{E}[R_iR_k]\mathbb{P}(G_i>0, G_k>0) \\
&= \sum_{i=1}^{I}\mathbb{E}[R_i^2]\mathbb{P}(G_i>0) + \sum_{i=1}^{I}\sum_{k=1,k\neq i}^{I}\mathbb{E}[R_i]\mathbb{E}[R_k]\mathbb{P}(G_i>0, G_k>0).
\end{aligned}$$

The second cross term requires a larger computation. We start off similarly to the first cross term, where we split the product of expectations of independent random variables. Then, we use the law of total probability on the probabilities $\mathbb{P}(G_i > 0)$ and $\mathbb{P}(G_i = 0)$, where in the latter case the sum $\sum_{j=1}^{G_k}(W_k)_j$ is zero, and therefore that whole term is zero.

$$\begin{aligned}
&2\mathbb{E}[\Big(\sum_{i=1}^{I}R_i\mathbb{1}_{\{G_i>0\}}\Big)\Big(\sum_{k=1}^{I}\sum_{j=1}^{G_k}(W_k)_j\Big)] \\
&= 2\sum_{i=1}^{I}\sum_{k=1}^{I}\mathbb{E}[\sum_{j=1}^{G_k}(W_k)_jR_i\mathbb{1}_{\{G_i>0\}}] \\
&= 2\sum_{i=1}^{I}\sum_{k=1}^{I}\mathbb{E}[R_i](\mathbb{E}[\sum_{j=1}^{G_k}(W_k)_j|G_i>0]\mathbb{P}(G_i>0) + 0)
\end{aligned}$$

Now, we condition on the property that $G = m$, where $\mathbb{P}(G = 0) = 0$ so we start at $m = 1$.

$$= 2\sum_{i=1}^{I}\sum_{k=1}^{I}\mathbb{E}[R_i]\sum_{m=1}^{\infty}\mathbb{E}[\sum_{j=1}^{G_k}(W_k)_j|G_i > 0, G = m]\mathbb{P}(G = m|G_i > 0)\mathbb{P}(G_i > 0)$$

$$= 2\sum_{i=1}^{I}\sum_{k=1}^{I}\mathbb{E}[R_i]\sum_{m=1}^{\infty}\mathbb{E}[\sum_{j=1}^{G_k}(W_k)_j|G_i > 0, G = m]\mathbb{P}(G_i > 0|G = m)\mathbb{P}(G = m)$$

Now, condition on the property that $G_k = g_k$.

$$= 2\sum_{i=1}^{I}\sum_{k=1}^{I}\mathbb{E}[R_i]\sum_{m=1}^{\infty}\sum_{g_k=0}^{\infty}\mathbb{E}[\sum_{j=1}^{g_k}(W_k)_j|G_i > 0, G = m, G_k = g_k]$$
$$\cdot\,\mathbb{P}(G_k = g_k|G_i > 0, G = m)\mathbb{P}(G_i > 0|G = m)\mathbb{P}(G = m)$$

$$= 2\sum_{i=1}^{I}\sum_{k=1}^{I}\mathbb{E}[R_i]\sum_{m=1}^{\infty}\sum_{g_k=0}^{\infty}g_k\mathbb{E}[W_k]\mathbb{P}(G_k = g_k|G_i > 0, G = m)\mathbb{P}(G_i > 0|G = m)\mathbb{P}(G = m)$$

The next step is to split the cases $i = k$ and $i \neq k$

$$= 2\sum_{k=1}^{I}\mathbb{E}[W_k]\mathbb{E}[R_k]\sum_{m=1}^{\infty}\mathbb{P}(G_k > 0|G = m)\mathbb{P}(G = m)\sum_{g_k=0}^{\infty}g_k\mathbb{P}(G_k = g_k|G_k > 0, G = m)$$
$$+ 2\sum_{i=1}^{I}\sum_{k=1:k\neq i}^{I}\mathbb{E}[R_i]\mathbb{E}[W_k]\sum_{m=1}^{\infty}\mathbb{P}(G_i > 0|G = m)\mathbb{P}(G = m)\sum_{g_k=0}^{\infty}g_k\mathbb{P}(G_k = g_k|G_i > 0, G = m).$$

Now we focus on expanding the term within the sum of g_k. In both cases (conditioned on $G_k > 0$ or $G_i > 0$), we use the rule of total probability, $\mathbb{P}(G_a = g_a|G = m) = \mathbb{P}(G_a = g_a|G_b > 0, G = m)\mathbb{P}(G_b > 0|G = m) + \mathbb{P}(G_a = g_a|G_b = 0, G = m)\mathbb{P}(G_b = 0|G = m)$, for respectively $a = k, b = k$ and $a = k, b = i$. Then, we use that the probability $\mathbb{P}(G_k = g_k|G_k = 0, G = m)$ is only nonzero for $g_k = 0$, and because of the g_k in front of the probability, this term is always zero. For the final step, the definition of the expectation of a binomially distributed random variable can be used, since $(G_k|G = m)$ is binomially distributed with m trials and success probability p_k.

$$\sum_{g_k=0}^{\infty}g_k\mathbb{P}(G_k = g_k|G_k > 0, G = m)$$
$$= \sum_{g_k=0}^{\infty}\frac{g_k\mathbb{P}(G_k = g_k|G = m) - g_k\mathbb{P}(G_k = g_k|G_k = 0, G = m)\mathbb{P}(G_k = 0|G = m)}{\mathbb{P}(G_k > 0|G = m)}$$
$$= \frac{1}{\mathbb{P}(G_k > 0|G = m)}\sum_{g_k=1}^{\infty}g_k\mathbb{P}(G_k = g_k|G = m)$$
$$= \frac{mp_k}{\mathbb{P}(G_k > 0|G = m)}.$$

The second term proves to be a bigger challenge, because of the probability $\mathbb{P}(G_k = g_k, G_i = 0|G = m)$. This probability requires that G_k should be g_k, which means we have a

term $p_k^{g_k}$. Simultaneously, the other $m-g_k$ times, we should not draw item k (with probability p_k), but also not draw item i (with probability p_i). This means that the other $m-g_k$ times, we have to draw something with probability $1-p_k-p_i$. These two type of draws can be arranged in $\binom{m}{g_k}$ ways, which means in total this probability is $\binom{m}{g_k}p_k^{g_k}(1-p_k-p_i)^{m-g_k}$.

$$
\begin{aligned}
&\sum_{g_k=0}^{\infty} g_k \mathbb{P}(G_k = g_k | G_i > 0, G = m) \\
&= \sum_{g_k=0}^{\infty} g_k \Big(\frac{\mathbb{P}(G_k = g_k | G = m) - \mathbb{P}(G_k = g_k | G_i = 0, G = m)\mathbb{P}(G_i = 0 | G = m)}{\mathbb{P}(G_i > 0 | G = m)} \\
&= \frac{1}{\mathbb{P}(G_i > 0 | G = m)} \Big(\sum_{g_k=0}^{\infty} g_k \mathbb{P}(G_k = g_k | G = m) - \mathbb{P}(G_k = g_k, G_i = 0 | G = m) \Big) \\
&= \frac{1}{\mathbb{P}(G_i > 0 | G = m)} \Big(mp_k - \sum_{g_k=0}^{\infty} g_k \binom{m}{g_k} p_k^{g_k} (1-p_k-p_i)^{m-g_k} \Big) \\
&= \frac{1}{\mathbb{P}(G_i > 0 | G = m)} \Big(mp_k - (1-p_k-p_i)^m \sum_{g_k=1}^{\infty} g_k \binom{m}{g_k} \frac{p_k}{(1-p_k-p_i)}^{g_k} \Big).
\end{aligned}
$$

In the next steps, we will use the definition of the binomial coefficient, which is $\binom{m}{g_k} = \frac{m!}{(m-g_k)!g_k!}$, which we use to rewrite the binomial coefficient. Then, we use the definition $(1+x)^n = \sum_{k=1}^{\infty}\binom{n}{k}x^k$ on the sum over g_k.

$$
\begin{aligned}
&\sum_{g_k=0}^{\infty} g_k \mathbb{P}(G_k = g_k | G_i > 0, G = m) \\
&= \frac{1}{\mathbb{P}(G_i > 0 | G = m)} \Big(mp_k - (1-p_k-p_i)^m \sum_{g_k=1}^{\infty} \frac{m(m-1)!}{((g_k-1)!((m-1)-(g_k-1))!} \frac{p_k}{(1-p_k-p_i)}^{g_k} \Big) \\
&= \frac{1}{\mathbb{P}(G_i > 0 | G = m)} \Big(mp_k - (1-p_k-p_i)^{m-1} mp_k \sum_{g_k=1}^{\infty} \binom{m-1}{g_k-1} \frac{p_k}{(1-p_k-p_i)}^{g_k-1} \Big) \\
&= \frac{mp_k}{\mathbb{P}(G_i > 0 | G = m)} \Big(1 - (1-p_k-p_i)^{m-1} \sum_{g_k=0}^{\infty} \binom{m-1}{g_k} \frac{p_k}{(1-p_k-p_i)}^{g_k} \Big) \\
&= \frac{mp_k}{\mathbb{P}(G_i > 0 | G = m)} \Big(1 - (1-p_k-p_i)^{m-1} (\frac{1-p_i}{1-p_k-p_i})^{m-1} \Big) \\
&= \frac{mp_k}{\mathbb{P}(G_i > 0 | G = m)} \Big(1 - (1-p_i)^{m-1} \Big).
\end{aligned}
$$

These two findings will be substituted back in the original equation, where at $(*)$ we use the theorem that if $\sum_{n=0}^{\infty} a_n$ exists, and $\sum_{n=0}^{\infty} b_n$ exists, then the sum $\sum_{n=0}^{\infty}(a_n+b_n)$ exists, and this sum is equal to $\sum_{n=0}^{\infty} a_n + \sum_{n=0}^{\infty} b_n$. Furthermore, we use that $|g| < 1$ and since $(1-p_i) < 1$ we also have $|g(1-p_i)| < 1$.

$$
\begin{aligned}
&2\sum_{k=1}^{I}\mathbb{E}[W_k]\mathbb{E}[R_k]\sum_{m=1}^{\infty}\mathbb{P}(G_k>0|G=m)\mathbb{P}(G=m)\sum_{g_k=0}^{\infty}g_k\mathbb{P}(G_k=g_k|G_k>0,G=m)\\
&\quad+2\sum_{i=1}^{I}\sum_{k=1:k\neq i}^{I}\mathbb{E}[R_i]\mathbb{E}[W_k]\sum_{m=1}^{\infty}\mathbb{P}(G_i>0|G=m)\mathbb{P}(G=m)\sum_{g_k=0}^{\infty}g_k\mathbb{P}(G_k=g_k|G_i>0,G=m)\\
&=2\sum_{k=1}^{I}\mathbb{E}[W_k]\mathbb{E}[R_k]\sum_{m=1}^{\infty}\frac{\mathbb{P}(G_k>0|G=m)}{\mathbb{P}(G_k>0|G=m)}\mathbb{P}(G=m)mp_k\\
&\quad+2\sum_{i=1}^{I}\sum_{k=1:k\neq i}^{I}\mathbb{E}[R_i]\mathbb{E}[W_k]\sum_{m=1}^{\infty}\frac{\mathbb{P}(G_i>0|G=m)}{\mathbb{P}(G_i>0|G=m)}\mathbb{P}(G=m)mp_k\Big(1-(1-p_i)^{m-1}\Big)\\
&=2\sum_{k=1}^{I}\mathbb{E}[W_k]\mathbb{E}[R_k]\frac{(1-g)p_k}{g}\sum_{m=0}^{\infty}g^m m\\
&\quad+2\sum_{i=1}^{I}\sum_{k=1:k\neq i}^{I}\mathbb{E}[R_i]\mathbb{E}[W_k]\frac{(1-g)p_k}{g}\sum_{m=1}^{\infty}g^m m\Big(1-\frac{(1-p_i)^m}{1-p_i}\Big)\\
&\overset{(*)}{=}2\sum_{k=1}^{I}\mathbb{E}[W_k]\mathbb{E}[R_k]\frac{p_k}{(1-g)}\\
&\quad+2\sum_{i=1}^{I}\sum_{k=1:k\neq i}^{I}\mathbb{E}[R_i]\mathbb{E}[W_k]\frac{(1-g)p_k}{g}\Big(\frac{g}{(1-g)^2}-\frac{g(1-p_i)}{(1-p_i)(1-g(1-p_i))^2}\Big)\\
&=2\sum_{k=1}^{I}\mathbb{E}[W_k]\mathbb{E}[R_k]\frac{p_k}{(1-g)}\\
&\quad+2\sum_{i=1}^{I}\sum_{k=1:k\neq i}^{I}\mathbb{E}[R_i]\mathbb{E}[W_k]\Big(\frac{p_k}{(1-g)}-\frac{p_k(1-g)}{(1-g(1-p_i))^2}\Big).
\end{aligned}
$$

When starting on the last cross term, we first use the same technique seen before, which is in this case conditioning on $(G_i=g_i,G_k=g_k)$.

$$
\begin{aligned}
&\mathbb{E}[\Big(\sum_{i=1}^{I}\sum_{j=1}^{G_i}(W_i)_j\Big)\Big(\sum_{k=1}^{I}\sum_{m=1}^{G_k}(W_k)_m\Big)]\\
&=\sum_{i=1}^{I}\sum_{k=1}^{I}\mathbb{E}[\Big(\sum_{j=1}^{G_i}(W_i)_j\Big)\Big(\sum_{m=1}^{G_k}(W_k)_m\Big)]\\
&=\sum_{i=1}^{I}\sum_{k=1}^{I}\mathbb{E}[\sum_{j=1}^{G_i}\sum_{m=1}^{G_k}(W_i)_j(W_k)_m]
\end{aligned}
$$

$$= \sum_{i=1}^{I}\sum_{k=1}^{I}\sum_{g_i=0}^{\infty}\sum_{g_k=0}^{\infty}\mathbb{E}[\sum_{j=1}^{G_i}\sum_{m=1}^{G_k}(W_i)_j(W_k)_m|G_i=g_i,G_k=g_k]\mathbb{P}(G_i=g_i,G_k=g_k)$$
$$= \sum_{i=1}^{I}\sum_{k=1}^{I}\sum_{g_i=0}^{\infty}\sum_{g_k=0}^{\infty}\mathbb{E}[\sum_{j=1}^{g_i}\sum_{m=1}^{g_k}(W_i)_j(W_k)_m]\mathbb{P}(G_i=g_i,G_k=g_k).$$

Notice that $(W_i)_j$ and $(W_k)_m$ are independent if $i \neq k$ or $j \neq m$. We split the sums into cases where $(W_i)_j$ and $(W_k)_m$ are dependent or independent. Since $(W_k)_j$ is i.i.d for all j, we can say that $(W_k)_j$ and $(W_k)_m$ are independent if $j \neq m$, and if we use the Law of the unconscious statistician at $(*)$, we get:

$$= \sum_{i=1}^{I}\sum_{k=1,k\neq i}^{I}\sum_{g_i=0}^{\infty}\sum_{g_k=0}^{\infty}\mathbb{E}[\sum_{j=1}^{g_i}\sum_{m=1}^{g_k}(W_i)_j(W_k)_m]\mathbb{P}(G_i=g_i,G_k=g_k)$$
$$+\sum_{k=1}^{I}\sum_{g_k=0}^{\infty}\mathbb{E}[\sum_{j=1}^{g_k}\sum_{m=1}^{g_k}(W_k)_j(W_k)_m]\mathbb{P}(G_k=g_k)$$
$$= \sum_{i=1}^{I}\sum_{k=1,k\neq i}^{I}\sum_{g_i=0}^{\infty}\sum_{g_k=0}^{\infty}\mathbb{E}[\sum_{j=1}^{g_i}\sum_{m=1}^{g_k}(W_i)_j(W_k)_m]\mathbb{P}(G_i=g_i,G_k=g_k)$$
$$+\sum_{k=1}^{I}\sum_{g_k=0}^{\infty}\mathbb{E}[\sum_{j=1}^{g_k}\sum_{m=1,m\neq j}^{g_k}(W_k)_j(W_k)_m]\mathbb{P}(G_k=g_k)$$
$$+\sum_{k=1}^{I}\sum_{g_k=0}^{\infty}\mathbb{E}[\sum_{m=1}^{g_k}(W_k)_m(W_k)_m]\mathbb{P}(G_k=g_k)$$
$$= \sum_{i=1}^{I}\sum_{k=1,k\neq i}^{I}\sum_{g_i=0}^{\infty}\sum_{g_k=0}^{\infty}\sum_{j=1}^{g_i}\sum_{m=1}^{g_k}\mathbb{E}[W_i]^2\mathbb{P}(G_i=g_i,G_k=g_k)$$
$$+\sum_{k=1}^{I}\sum_{g_k=0}^{\infty}\mathbb{E}[W_k]^2((g_k)^2-g_k)\mathbb{P}(G_k=g_k)$$
$$+\sum_{k=1}^{I}\sum_{g_k=0}^{\infty}\sum_{m=1}^{g_k}\mathbb{E}[W_k^2]\mathbb{P}(G_k=g_k)$$
$$\overset{(*)}{=} \sum_{i=1}^{I}\sum_{k=1,k\neq i}^{I}\mathbb{E}[W_i]^2\mathbb{E}[G_iG_k]$$
$$+\sum_{k=1}^{I}\mathbb{E}[W_k]^2\Big(\mathbb{E}[G_k^2]-\mathbb{E}[G_k]\Big)$$
$$+\sum_{k=1}^{I}\mathbb{E}[W_k^2]\mathbb{E}[G_k].$$

All that remains is to combine the three cross terms to give a closed form for the variance.

$$
\begin{aligned}
Var(Z) = &\sum_{i=1}^{I} \mathbb{E}[R_i^2]\mathbb{P}(G_i > 0) + \sum_{i=1}^{I}\sum_{k=1,k\neq i}^{I} \mathbb{E}[R_i]\mathbb{E}[R_k]\mathbb{P}(G_i > 0, G_k > 0) \\
&+ 2\sum_{k=1}^{I} \mathbb{E}[W_k]\mathbb{E}[R_k]\frac{p_k}{(1-g)} \\
&+ 2\sum_{i=1}^{I}\sum_{k=1:k\neq i}^{I} \mathbb{E}[R_i]\mathbb{E}[W_k]\Big(\frac{p_k}{(1-g)} - \frac{p_k(1-g)}{(1-g(1-p_i))^2}\Big) \\
&+ \sum_{i=1}^{I}\sum_{k=1,k\neq i}^{I} \mathbb{E}[W_i]^2\mathbb{E}[G_iG_k] \\
&+ \sum_{k=1}^{I} \mathbb{E}[W_k]^2\Big(\mathbb{E}[G_k^2] - \mathbb{E}[G_k]\Big) \\
&+ \sum_{k=1}^{I} \mathbb{E}[W_k^2]\mathbb{E}[G_k] \\
&- \mathbb{E}[Z]^2.
\end{aligned}
$$

To make sure no errors are made in the process, we are interested in verifying these results. This will be done by composing the Laplace Stieltjes Transformation (LST) of Z. This transformation can be used to find the first and second moment, which give the expectation and variance, which should be the same as computed above.

$$
\begin{aligned}
\mathbb{E}[e^{-sZ}] &= \mathbb{E}[e^{-s(\sum_{i=1}^{I} Y_i)}] \\
&= \mathbb{E}[e^{-s\sum_{i=1}^{I}(R_i\mathbb{1}_{\{G_i>0\}} + \sum_{j=1}^{G_i}(W_i)_j)}] \\
&= \sum_{k=1}^{\infty} \mathbb{E}[e^{-s\sum_{i=1}^{I}(R_i\mathbb{1}_{\{G_i>0\}} + \sum_{j=1}^{G_i}(W_i)_j)}|G=k]\mathbb{P}(G=k) \\
&= \sum_{k=1}^{\infty}\sum_{g_1=0}^{\infty}\ldots\sum_{g_I=0}^{\infty} \mathbb{E}[e^{-s\sum_{i=1}^{I}(R_i\mathbb{1}_{\{G_i>0\}} + \sum_{j=1}^{G_i}(W_i)_j)}|G=k, G_1=g_1, ..G_I=g_I]\mathbb{P}(G=k) \\
&\quad \cdot \mathbb{P}(G_1=g_1, ..G_I=g_I|G=k) \\
&= \sum_{k=1}^{\infty}\sum_{g_1=0}^{\infty}\ldots\sum_{g_I=0}^{\infty} \mathbb{E}[e^{-s\sum_{i=1}^{I}(R_i\mathbb{1}_{\{g_i>0\}} + \sum_{j=1}^{g_i}(W_i)_j)}|G=k]\mathbb{P}(G=k)\mathbb{P}(G_1=g_1, ..G_I=g_I|G=k).
\end{aligned}
$$

Now, we use the fact that R_i and W_j are independent, $R_1, ..., R_I$ are independent, and $(W_i)_1, ..., (W_i)_I$ are independent. Because g_i is now a constant value, we can move it out

of the expectation.

$$
\begin{aligned}
&\mathbb{E}[e^{-sZ}] \\
&= \sum_{k=1}^{\infty}\sum_{g_1=0}^{\infty}...\sum_{g_I=0}^{\infty}\mathbb{E}[e^{-s\sum_{i=1}^{I}R_i}]^{\mathbb{1}_{\{g_i>0\}}}\mathbb{E}[e^{-s\sum_{i=1}^{I}(W_i)_j)}]^{g_k}\mathbb{P}(G=k)\mathbb{P}(G_1=g_1,..G_I=g_I|G=k) \\
&= \sum_{k=1}^{\infty}\sum_{g_1=0}^{\infty}...\sum_{g_I=0}^{\infty}\prod_{i=1}^{I}\Big(\mathbb{E}[e^{-sR_i}]^{\mathbb{1}_{\{g_i>0\}}}\mathbb{E}[e^{-sW_i}]^{g_k}\Big)\mathbb{P}(G=k)\mathbb{P}(G_1=g_1,..G_I=g_I|G=k).
\end{aligned}
$$

The probability $\mathbb{P}(G_1=g_1,..G_I=g_I|G=k)$ is only nonzero when the sum of all g_i is exactly k. Therefore, we can write:

$$
\begin{aligned}
&= \sum_{k=1}^{\infty}\sum_{g_1+..+g_I=k}\prod_{i=1}^{I}\Big(\mathbb{E}[e^{-sR_i}]^{\mathbb{1}_{\{g_i>0\}}}\mathbb{E}[e^{-sW_i}]^{g_k}\Big)\mathbb{P}(G=k)\mathbb{P}(G_1=g_1,..G_I=g_I|G=k) \\
&= \sum_{k=1}^{\infty}\mathbb{P}(G=k)\mathbb{E}[e^{-sW_1}]^k\sum_{g_1+..+g_I=k}\prod_{i=1}^{I}\Big(\mathbb{E}[e^{-sR_i}]^{\mathbb{1}_{\{g_i>0\}}}\Big)\mathbb{P}(G_1=g_1,..G_I=g_I|G=k) \\
&= \sum_{k=1}^{\infty}\mathbb{P}(G=k)\mathbb{E}[e^{-sW_1}]^k\sum_{g_1+..+g_I=k}\prod_{i=1}^{I}\Big(\mathbb{E}[e^{-sR_i}]^{\mathbb{1}_{\{g_i>0\}}}\Big)\frac{k!}{g_1!...g_I!}p_1^{g_1}...p_I^{g_I} \\
&= \sum_{k=1}^{\infty}k!(1-g)g^{k-1}\mathbb{E}[e^{-sW_1}]^k\sum_{g_1+..+g_I=k}\prod_{i=1}^{I}\Big(\mathbb{E}[e^{-sR_i}]^{\mathbb{1}_{\{g_i>0\}}}\Big)\frac{p_1^{g_1}...p_I^{g_I}}{g_1!...g_I!}.
\end{aligned}
$$

The LST of Z has been implemented in Mathematica. It was doable to run this for $I \leq 6$, but for $I = 10$ the program was not finished after 31 hours, and because of time constraints we decided to compare and verify the LST and hand-computations for $I = 6$. A warehouse layout is used where the travel distance has mean $\mathbb{E}[R] = 12.20896$ and variance $\text{Var}(R) = 66.417$. A pickup time with mean $\mathbb{E}[W] = 8$ and variance $\text{Var}(W) = 1$ is used. Furthermore, we have 6 SKUs, and all 6 product types have probability 1/6 to be chosen, so $p_i = 1/6 \ \forall i$.

Term	**Hand-computations (6 SKUs)**	**LST (6 SKUs)**
$\mathbb{E}[Z]$	40.317	40.2920
$\mathbb{E}[Z^2]$	2394.75	2389.17
Var(Z)	769.26	765.816
Run time	$\approx$ 0.00 sec	$\approx$ 70 min

Table 4: Mean and variance using 2 different methods

The results are stated in Table 4. For the LST we bounded k (the number of products in an order) by 15, to give a faster approximation of the mean and variance. The CDF of the geometric distribution is $\mathbb{P}(G \geq x) = (1-p)^x$, which means with probability $(1-0.55)^{16} \approx$

$2.8 \cdot 10^{-6}$ there are 16 or more items in an order, and this scenario is not taken into account in the LST. Although this probability is small, this means the LST is somewhat shifted. If there are 16 or more items, the travel time is expected to be very large, and therefore we expect the mean and variance computed at the LST will appear to be lower than they actually are. This is exactly the behaviour that can be seen in Table 4. Both the mean and variance of the LST are slightly lower, which is as we expected, but the values seem to be close enough to give confidence in the correctness of the computations. The most noticeable aspect is the computation time: the hand-computations were instantly done, while the LST took over an hour to generate. Therefore, the hand-computations can be used to compute the mean and variance for larger cases, while the LST will only keep its purpose as verification in a small case.

We also aim to verify for a higher (and more realistic) number of SKUs, in this case 10.000. Computing the LST is difficult, and computing this for 10.000 SKUs would take too much time. Therefore, we create a basic simulation, shown in Algorithm 1, where we choose some distributions for R and W to verify the mean and variance of the service time. Then, the results of the hand-computation and the basic simulation are compared, these results are visualised in Table 5. The mean and variance are both extremely close, and this verifies that the mathematical computations are likely to be correct.

Algorithm 1 Verifying hand-computations mathematical model

procedure INITIALISE PARAMETERS AND DISTRIBUTIONS
    Distribution R: Gamma(2.24429, scale=5.44002)
    Distribution W: Gamma(64, scale=1/8)
    Distribution G: Geometric(1-0.55)
    pi = $\frac{1}{2.500}$ or $\frac{1}{20.000}$ or $\frac{1}{100.000}$
procedure RUNTRIALS
    for Trial **do**
        draw g from G
        Gi = Multinomial(g, pi)
        total travel time: draw |nonzero(Gi)| values from R, and sum them
        total pickup time: draw g values from W, and sum them
        Total service time is total travel time + total pickup time
procedure RESULTS
    Compute mean and variance of total service time of trials.

Term	**Hand-computations (10K SKUs)**	**Simulation (10K SKUs)**
$\mathbb{E}[Z]$	44.898	44.8968
Var(Z)	1257.617	1258.4966

Table 5: Mean and variance using 2 different methods

3.3 M/G/c queue approximation

The final queuing model, also shown in Figure 8, has the following properties. First, we assumed orders arrive according to a Poisson process. Then, we assumed to have c identical parallel servers, which are in this case the vehicles. Recall that we assume (in this chapter) that vehicles retrieve orders independently and do not block each other, meaning vehicles would not affect each other's performance. Additionally, all deployed vehicles are built identical (and therefore have identical performance, such as speed) and orders are assigned randomly to vehicles, which leads to the assumption that all vehicles have an identically distributed service time. We already mentioned the service times are considered to be identically distributed. Additionally, it is unlikely that the service times are exponentially distributed, considering their mean and variance. This is also a realistic assumption, since we assume the vehicle has a minimum travel distance, and therefore a minimum service time. Therefore, we consider a queuing model that allows for generally distributed service times. Taking all of this into account, it follows that this queuing model can be described by a M/G/c queue, which requires Markovian arrivals, service times with a general distribution, and c servers.

Unfortunately, an exact analysis for an M/G/c queue with generally distributed service times is not available. Therefore, we will use an approximation to find the mean waiting time, $\mathbb{E}[W]$, using a known general formula [28];

$$\mathbb{E}[W] = \Pi_w(M/M/c)\frac{\mathbb{E}[Z_{res}]}{c(1-\rho)}. \tag{1}$$

Where $\rho = \frac{\lambda}{c\mu}$ and we have inter arrival times with mean $\frac{1}{\lambda}$ and service times with mean $\frac{1}{\mu}$. Additionally, we can compute the residual service time $\mathbb{E}[Z_{res}]$, which is;

$$\mathbb{E}[Z_{res}] = \frac{1+v^2}{2\mu}.$$

Where v is the coefficient of variation; $v = \frac{\sigma(Z)}{\mathbb{E}[Z]}$, and Z is the service time, and therefore $\mathbb{E}[Z]$ the expected service time and $\sigma^2(Z)$ the variance of the service time. Furthermore, we need to compute $\Pi_w(M/M/c)$, the probability of waiting in an M/M/c queue. This quantity, also referred to as the *delay probability*, can be derived using equilibrium probabilities and PASTA [26], and is defined as follows;

$$\Pi_w = \frac{c\rho}{c!}\Big((1-\rho)\sum_{n=0}^{c-1}\frac{(c\rho)^n}{n!} + \frac{(c\rho)^c}{c!}\Big)^{-1}.$$

The closed-form formula (1) can now be used to find the mean waiting time when varying the number of vehicles (c) in the system. The first step is to compute the mean and variance of the travel time using the hand-computations, these results are shown in Table 5. In this scenario, we use $I = 10.000$ and p_i according to the Pareto curve. All other parameters are the same as in the previous section. $\mathbb{E}[Z_{res}]$ is determined for 12-30 vehicles, where 12 is the minimum such that $\rho = \frac{\lambda}{c\mu} < 1$. The approximated mean waiting time is visualised in Figure 11.

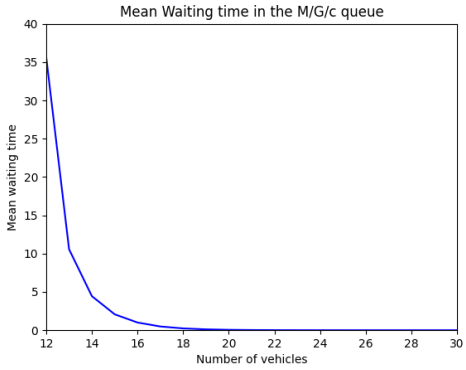


Figure 11: Mean waiting time in the queuing model

When the number of vehicles is minimal, the mean waiting time is more than 30 seconds. However, as the number of vehicles increases, the mean waiting time first decreases rapidly, and after a while, it only decreases slowly. Therefore, by using the elbow rule, a good recommendation would be to deploy 15 vehicles.

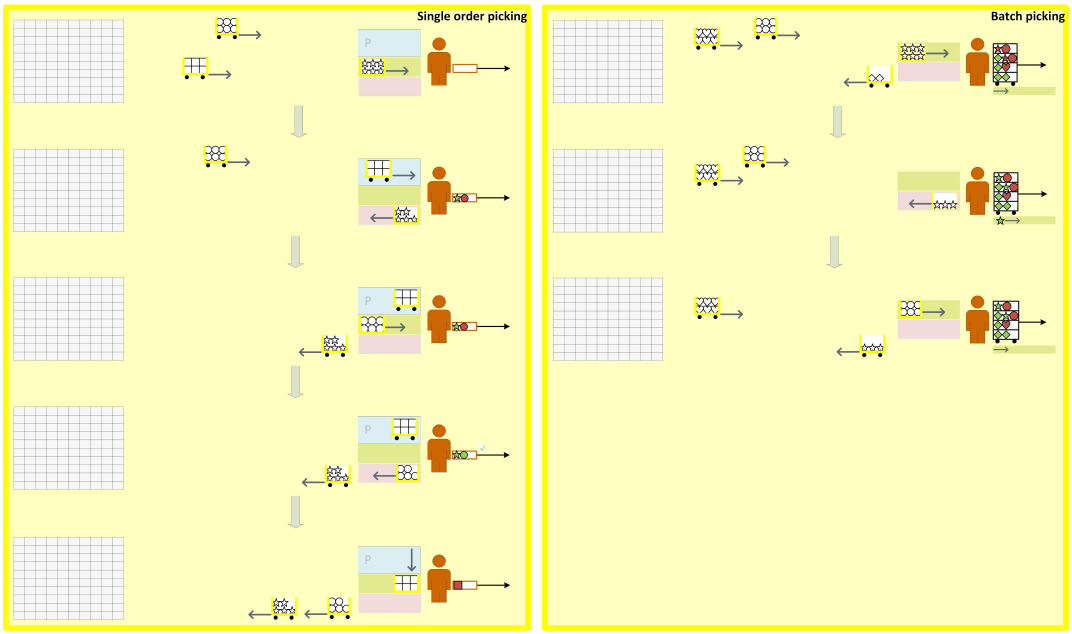
Although these results are interesting, we acknowledge that many assumptions are used, which means the scenario is not very realistic. One of the most crucial assumptions is the infinite number of workstations, and adding a second server to the model that is highly dependent on the first server will make the model more complex. Additionally, we aim for vehicles to work together instead of handling orders independently, and want them to carry multiple product carriers. All these factors make it extremely challenging to find and develop a mathematical model and therefore we decided to create a simulation.

4 Simulation model

Making the mathematical queuing model more complex is very challenging, and therefore we will create a simulation to gain more realistic insights in the performance of each configuration. The goal of this section is to describe the model behind the simulation, and to give a deeper insight in how the simulation works. To do this, we will start by giving an introduction to the model, which explains the process and the different scenarios we can encounter and what actions are taken. Then, we will describe the simulation and go in further detail on the structure and behaviour of this simulation to ensure tractability of the model.

4.1 Introduction to the simulation model

We will start by giving a more general explanation to the model, where we omit some details, however all details are described in the next few sections, which include a more elaborate description of the simulation. We start by describing the route and flow of the product carriers, this is visualized by a process flow diagram. In this model, we have two types of process diagrams; one for configurations (1-4) shown in Figure 12a, which represents single order picking with a 1-to-1 station or a medium workstation. The other process diagram is for configurations (5-7) shown in Figure 12b, which represents batch picking with a put wall.



(12a) Process for single order picking (12b) Process for batch picking

Figure 12: Process flow for two types of configurations

Single order picking With single order picking, every product carrier will be used to supply one item (product) of one order. This means that if one type of item is required twice in an order, we assume that this product carrier needs to be retrieved twice. In a GMF (general merchandise and fashion) warehouse, it is very likely that an item is only required at most once per order, and therefore this assumption is considered to be realistic. Initially, no orders are assigned to the workstations, and instead all vehicles are initiated to retrieve

product carriers of pending orders, prioritising orders with higher urgency. When a vehicle arrives at the area of workstations, there are multiple possible scenarios. We will now describe the different scenarios and what actions are taken.

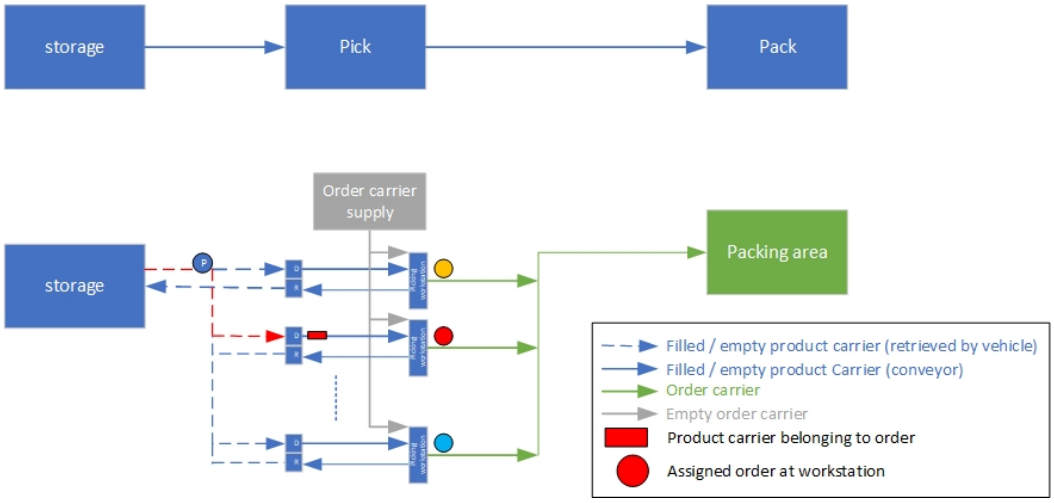


Figure 13: Order of product carrier is already assigned

Scenario 1: Order of product carrier is already assigned

The first scenario is shown in Figure 13. In this case, the arriving product carrier belongs to the red order, and the red order is already assigned to the second station. This means the vehicle does not have to wait in the parking area, represented by the blue circle with the letter "P". The vehicle directly moves to the second workstation, and drops off the product carrier at the conveyor ("D" for deposit), and the conveyor transports the product carrier to the operator at the second workstation. Notice that we assume that product carriers are not required to arrive in a specific sequence within an order. Then, the vehicle will move to the discard conveyor ("R" for return). Product carriers that are used by operators and have to be moved back to storage (because the operator does not need it any more) are located on this conveyor. The vehicle takes a discarded product carrier (if there are any) and travels back to storage. The vehicle travels to the new product carrier it needs to retrieve, and places the returned product carrier near this location, retaining chaotic storage and keeping the travel time low.

This scenario is also encountered in the third step of Figure 12a, where a vehicle with circle-shaped items arrives, and the order that requires this circle-shaped item is already assigned (in the box with orange border on the right). Therefore, this vehicle can drop the product carrier on the deposit conveyor (green rectangle in the middle), and will return to storage with a product carrier from the discard conveyor, which is represented by the red rectangle.

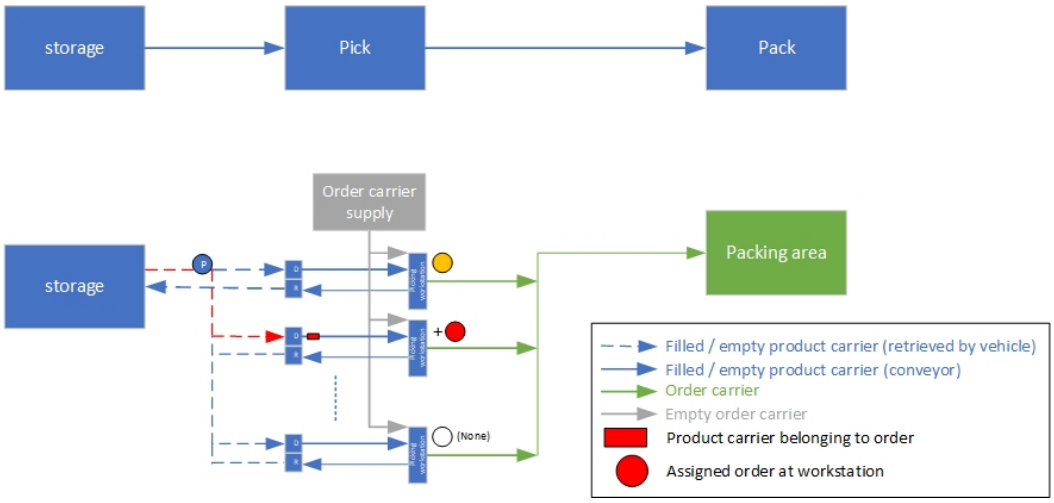


Figure 14: Order is assigned at new workstation

Scenario 2: Order of product carrier is not assigned, but there is room for new assignment

In the second scenario, shown in Figure 14, the order which the product carrier belongs to is not assigned to any workstation, but there is a workstation that does not have its maximum number of assigned orders. In this case, the order belonging to this product carrier is assigned to a workstation, which in this figure represents the + in front of the red order. Then, like previous scenario, the vehicle will drop its product carrier at this workstation, and attempt to take a product carrier from the discard conveyor to return to storage.

This scenario is also encountered in the first step of Figure 12a where the order, that belongs to the product carrier with the star-shaped item, is assigned to the workstation.

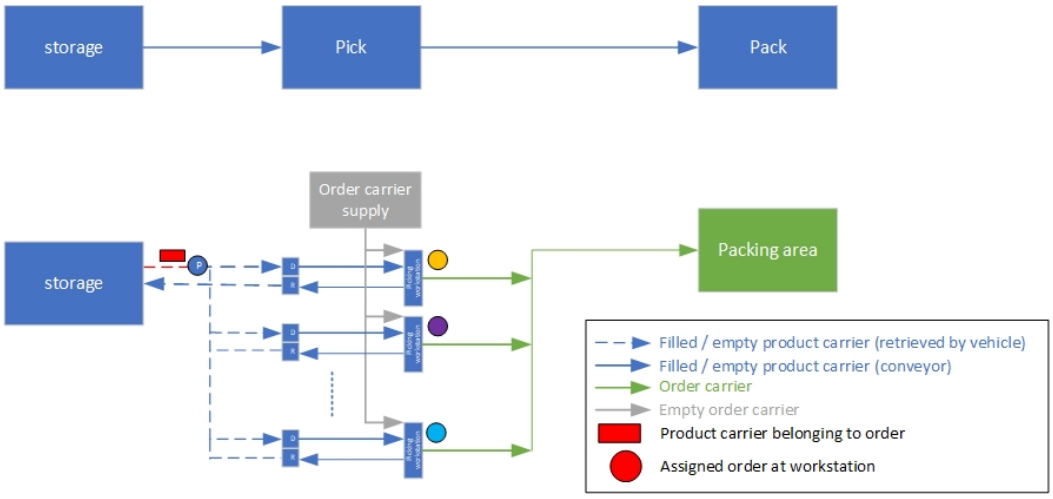


Figure 15: Vehicle has to wait for order completion

Scenario 3: Order of product carrier can not be assigned

The last scenario (Figure 15) shows that a product carrier belonging to the red order arrives, but the red order is not yet assigned to any station. Furthermore, all workstations already have a different order assigned (yellow, purple, blue), meaning that the red order can

not be assigned. In this case, the vehicle will have to wait. The vehicle waits at a designated location until one of the assigned orders is completed, meaning there is room for a new order to be assigned; possibly the red order. When the red order is finally assigned, the vehicle moves to this workstation, drops off the product carrier and returns to storage (with a discarded product carrier) as usual.

This scenario is also visualised in the second and fourth step of Figure 12a. The second step shows that the order with a square is not assigned, and that the vehicle will wait in a designated area (blue rectangle above the deposit conveyor). In the fourth step, an order is completed, and the order of the square is assigned, and the product carrier will be dropped off at this station.

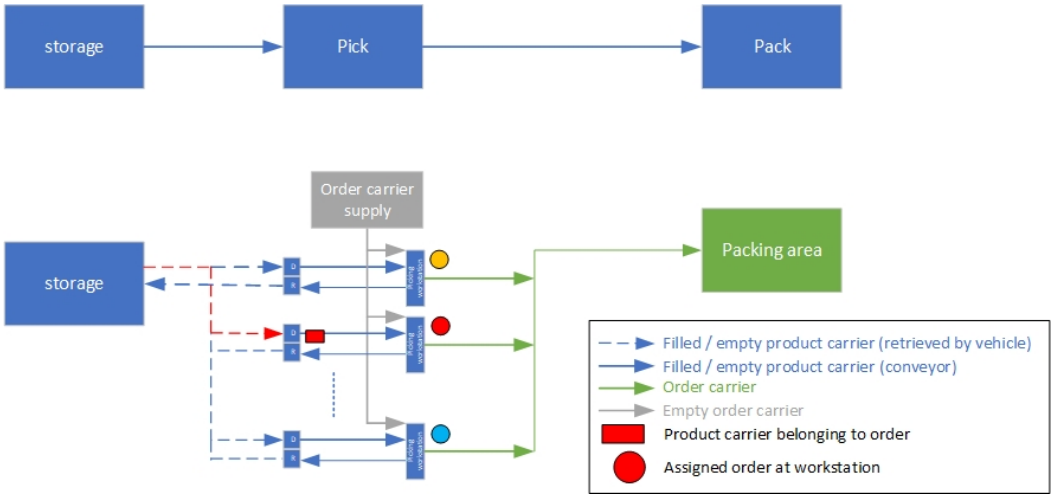


Figure 16: Product carrier can always go to workstation

Batch picking With batch picking, we have a put wall with a certain number of pick-to locations. A batch is created for this put wall, and all orders of this batch are assigned to this workstation. All multi-item orders (orders with at least two items) will be placed in the put wall, and the single-item orders will be placed in a single-order order carrier (by the operator) and brought directly to the packing process, for example by using a conveyor. This last aspect is visualised as the green conveyor below the put wall in Figure 12b. All vehicles are requested to retrieve product carriers required at the workstations. When a vehicle with a product carrier arrives at the area of workstations, it will travel to the workstation where the product carrier is requested, which can be seen in Figure 16. The vehicle will drop off the product carrier, and it will possibly pick up a product carrier from the discard conveyor to place back in storage while retrieving a new product carrier. When all orders from a put wall are fulfilled, the put wall will be moved away and a new, empty put wall will be placed, and a new batch will be created for this put wall. The main difference between the two processes is that at single order picking, all vehicles are always initiated to retrieve product carriers, and are never idly waiting at the origin point. However, it is possible a vehicle has to wait in front of the workstation if its order is not (and can not be) assigned to any workstation. At the batch picking scenario, it is possible vehicles have to wait at the origin point (when all product carriers still required by all put walls are either on a vehicle or on the deposit conveyor already), but vehicles never have to wait in front of the workstation, because the

order of the product carrier is already assigned to a workstation. Other waiting times could occur if vehicles have to charge or are blocking each other, but both these aspects are not taken into account in this research.

Now, we will describe some extensions of the scenarios we just described. One of these extensions is what happens if the vehicle carries multiple product carriers or a rack, and we will also further discuss the discarded product carriers.

Vehicle carrying multiple product carriers With single order picking, if a vehicle carrying multiple product carriers arrives at the workstation area, the following steps will be taken, which are similar to the three scenarios we just described. First, we check for each product carrier if the order belonging to it has already been assigned to a workstation. If this is the case, we drop off the product carriers at the proper workstation. Even though we aim to retrieve product carriers for as few different orders as possible, it is still possible the vehicle needs to drop off product carriers at multiple different stations, meaning in reality this would give some travel time. However, we do not take this travel time into account in the model. If the vehicle is not empty, we check if orders of the vehicle's product carriers can be assigned to a workstation. We assign as many orders as possible, and drop off the product carriers accordingly. Then, if all product carriers are dropped off, the vehicle returns to storage. If the vehicle could not drop off all product carriers, the vehicle waits in a designated area with the remaining product carriers. The vehicle will wait for one or multiple order completions to assign the orders belonging to its remaining product carriers. When all product carriers have been dropped off, the vehicle can finally return to storage for a new retrieval action.
For batch picking, when a vehicle arrives, it can instantly start dropping off all its product carriers at the proper workstation. We also aim for a vehicle to retrieve product carriers of few orders, and additional travel time is not taken into account in the model.

Vehicle carrying a rack When a vehicle carries a rack, it is interesting to see what happens with the conveyor in front of the workstation; it would be strange to move an entire rack on a conveyor. Therefore, in this configuration, we deploy an additional vehicle (A) that functions as a replacement of this conveyor. When a vehicle with a rack arrives, the rack is placed in the deposit area, and the vehicle that carried this rack returns to storage. Then, whenever the operator needs a rack, vehicle A takes this rack from the deposit area and places it in front of the operator. The operator can now take items from the rack and place them in the put wall. When the operator is done with the rack, vehicle A takes the rack, and puts it in the discard area (to be returned to storage). Then, it attempts to take a new rack from the deposit area to move this rack to the operator. This process takes time in reality, but to avoid complexity of the simulation, we do not take this time into account.

Discard of product carriers In both processes, we mentioned that a vehicle attempts to take a product carrier to place back in storage when retrieving a new product carrier. For this process, we assume that the discard conveyor will never be too full, and will not give any sort of waiting time in the model. With these assumptions, this conveyor does not have to be taken into account.

4.2 Detailed explanation of the simulation

Now we generally described the model of the simulation, we will describe the simulation in more detail. We will start by describing the structure of the simulation. Then, we elaborate on the different scenarios the simulation can encounter and how these are handled. Then, we describe how the simulation is initiated and what main simulations are ran, and how we verified the simulation.

Structure of the simulation We are using a Discrete-event simulation (DES), where we will use a Future-Event set (FES). This is a set that contains events that the simulation will have to handle in the future. Every event has certain information that indicates how the simulation should handle the event; in this case one piece of information is the event type. To model the described warehouse process, we use three types of events;

- **Order arrival** (by a customer).
- **Vehicle arrival**, which means a vehicle with one or multiple product carriers or a rack arrives at the workstation.
- **Item consolidation**; this name might sound confusing, but we simply mean the moment where an operator finished moving one item from a product carrier to an order carrier.

These events are strongly connected to each other. For example, if a new order arrives ("order arrival"), and there is a vehicle without work, we want this vehicle to retrieve a product carrier for this order. We can compute when this vehicle will arrive at the workstation; and at that time, there will be an "vehicle arrival" event. Therefore, we can already add this "vehicle arrival" event in the FES, accompanied with the computed time this event will occur. Each type of event has to be handled differently, and in the next three sections, we will describe in detail what actions have to be taken for each type of event.

During the simulation, we will use multiple queues to keep track of all the product carriers, vehicles, conveyors and workstations. A compact overview of all queues can be found in Table 6. We also keep track of other statistics such as the number of items an operator picked, and the synergy of batches. It is important to note that we also apply a warm-up time to the simulation, since we are more interested in the behaviour when all components are properly running. Therefore, we only keep track of statistics when the warm-up time has passed. Furthermore, we also use some parameters, for example the vehicle and operator speed. All parameters are indicated in the appendix in Table 25, and have been subjected to peer-review to verify that they were realistic for the purposes of this research.

Name →	Configurations	Description of queue
Qs0	1-4	List of product carriers that need to be retrieved.
	5-6	Multiple lists; each contains product carriers that need to be retrieved for a workstation.
	7	Multiple lists; each contains item types (SKUs) that need to be retrieved for a workstation.
Qs1	1-6	Multiple lists; each contains product carrier(s) a vehicle is retrieving.
	7	Multiple lists; each contains the item types (SKUs) a vehicle is retrieving in its rack, that are also required for workstations.
Qs2	1-4	Multiple lists; each contains product carrier(s) a vehicle is holding while waiting.
preBatch	5-7	List of orders that need to be batch picked.
WsX	1-7	For a workstation X, what product carriers are on the conveyor in front of the workstation.
WResX	1-7	For a workstation X, what orders are currently assigned to this station.

Table 6: Queues used in the simulation to keep track of all entities.

4.2.1 Order arrival

When the FES encounters an order arrival event, it first generates the number of items in this order (using the geometric distribution with parameter g) and then the specific items in this order (using the multinomial distribution with parameters p_i and the number of items in the order). With single order picking, the order is directly added to Qs0, and we check if there is any vehicle without work. If such a vehicle is found and there are enough product carriers to retrieve (greater or equal to the carrying capacity of the vehicle), we take the first y product carriers from Qs0, where y is the carrying capacity of the vehicle. These are the product carriers that the vehicle will retrieve. A list of these product carrier(s) is added to Qs1. Then, we take a random travel distance (this will be described in Section 4.2.2), and with the mean speed and pickup/dropoff time we can compute the time this vehicle will arrive at the workstation. At this time, a new vehicle arrival event is scheduled.
In the batch picking scenario, the order is added to *preBatch*. Then, we check if any station has no assigned orders, although in reality this is extremely unlikely to occur because this would mean that earlier in the simulation, there were not enough orders to create a batch. However, if this still occurs, a new batch is attempted to be created, which is described in detail in Section 4.2.3.
Lastly, for both simulations, we increase the total number of orders (to assign the proper order number to a new order) and then add a new order arrival to the FES. Since we have a distribution for the inter arrival time (in this case exponential with intensity $\frac{6}{5}$), we draw an inter arrival time from this distribution. The new order arrival of the FES will have the current time, plus the time drawn from the exponential distribution.

4.2.2 Travel distances

To determine the travel distance, we will create an example of a layout of a warehouse, which we will use for the simulation. This layout varies based on some parameters, such as the width of small or large lanes, and on the number of SKUs we are investigating. Most parameters are fixed in all simulation runs, and these can be found in Table 7. Some other parameters, found in Table 8, will vary between runs, for example because we want to observe the performance when varying these parameters. These parameters will also influence the layout, for example if the number of SKUs is high, the layout will be larger. The general layout coincides with the layout visualized in Figure 10a, and a block is similar to the block shown in Figure 10b, except the number of small lanes in one block can differ, and the number of blocks in the vertical direction can differ for each configuration of interest.

Abbreviation	**Value**	**Description**
StockR(m)	StockR(fast mover) = 0.2 StockR(medium mover) = 0.3 StockR(slow mover) = 0.5	For a certain type of mover, what ratio of the stock consists of this type.
occ(m)	occ(fast mover) = 5 occ(medium mover) = 3 occ(slow mover) = 1	For a certain type of mover, how often can this type be found in storage.
SDepth	1	Storage depth
LLwid	3	Width of a large lane
SLwid	1.5	Width of a small lane
YComparts	2	Number of blocks (horizontally).
totesPerYCompart	60	Number of product carriers in one block (horizontally).

Table 7: Fixed parameters used to determine layout

Abbreviation	**Value**	**Description**
nrSKU	10.000 or 100.000	Number of different products
StackHeight	20 (carries 1 product carrier) 8 (carries 5 product carriers)	How many product carriers can be stacked in one 0.6×0.4m floor location.
StackHeight	20 (carries rack)	How many product carriers are located in one 1×1m (one pod) floor location.

Table 8: Parameters that vary and influence the layout

To determine the number of small lanes and vertical blocks, we first determine how many storage locations we need, which is equal to:

$$\#\text{ carrier locations} = \frac{\sum_m \lceil StockR(m) \cdot nrSKU \rceil \cdot occ(m)}{StackHeight}$$

Where m indicates the type of mover. The total number of vertical blocks in one of the halves (either below or above the blue square in Figure 10a) is called $Xcomparts$, and these

blocks are divided by large lanes of $3m$. The layout is mirrored in a horizontal line spanning from the middle of the blue square. $Xcomparts$ is determined on a rough estimate, namely the number of floor spaces required, divided by 7500. The number of horizontal (small) lanes per block is minimized to satisfy the required number of product carrier locations as follows:

$$\#\text{ of small lanes} = \left\lceil \frac{\#\text{ carrier locations} \cdot StackHeight}{4 \cdot SDepth \cdot Xcomparts \cdot YComparts \cdot totesPerYCompart} - 1 \right\rceil$$

When the number of small lanes is defined, the layout of the warehouse is fixed. Since we run multiple configurations, the layout will vary between the configurations. The number of vertical blocks and the number of small lanes for each layout can be found in Table 24. When a vehicle carries a rack, the product carrier size switches from $0.6 \times 0.4m$ to $1 \times 1m$. In this case, the width of the lanes are also multiplied accordingly, meaning that the large lanes have width 5 and small lanes width 2.5.

Coordinates and their distances Now, we will use coordinates to define the exact location the vehicle has to travel to, to start picking a product carrier. To start picking a product carrier, the vehicle is located in the middle of the width of this carrier. The distance between the product carrier and the vehicle depends on the width of a small lane; a vehicle is always positioned half the width of a small lane from a product carrier. This is also visualized in Figure 17: for the yellow product carrier, the vehicle (yellow dot) is located exactly in the middle of the product carrier width, and is located in the middle of the small lane. For the green product carrier, the distance between the product carrier and vehicle (green dot) is also exactly half the width of a small lane: $0.75m$. We assume that from this location, the vehicle starts the pickup/dropoff process, for example by climbing into a rack. Furthermore, the orange dot represents the origin point, where all the workstations are located.

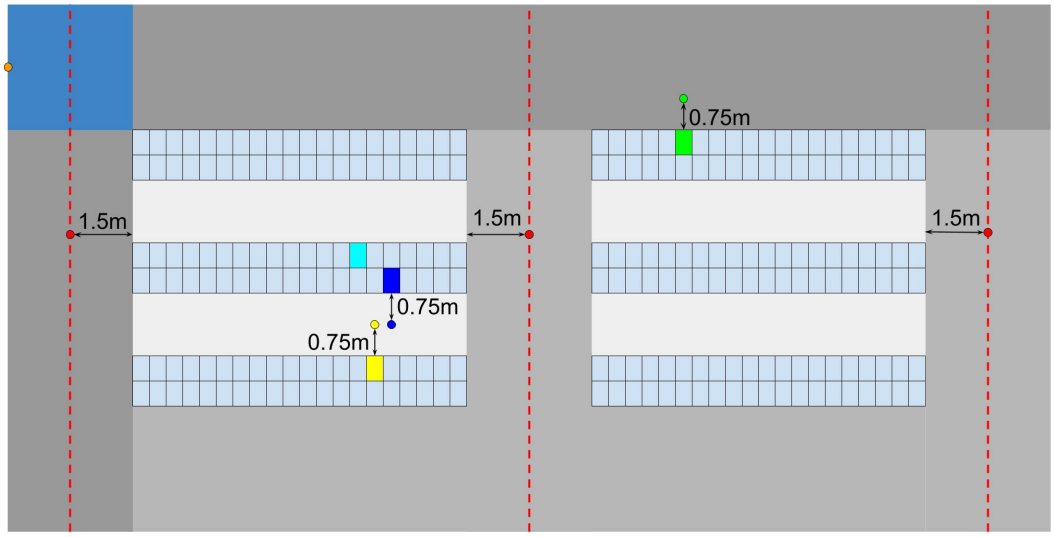


Figure 17: Distances in the layout of a warehouse

The next step is to determine the travel distance between two product carriers in a warehouse. The travel distance between two locations in this 2-dimensional grid (the floor space)

is not simply computed by taking the 2-norm between two points, since a vehicle can only drive in the given lanes. Furthermore, when a vehicle is in a small lane and needs to travel to a small lane on a different height (for example, from dark blue to light blue), the vehicle first has to travel to the nearest large lane (in this case to the right). To deal with this issue, we define three (since the number of vertical large lanes is three) lines, for simplicity located in the middle of a large lane. We assume that if a vehicle has to travel to a small lane on another level, it has to move to any of the three red dotted lines first, where we always choose the red line that gives the smallest travel distance.

If a set of product carriers is given, we want to generate an appropriate travel distance, which depends on the types of movers of the product carriers. It requires a lot of memory and computing power to store where every product carrier is located and to compute the travel distance for every retrieval request. Therefore, we will create a pre-computed dictionary, which we can use to find a proper travel distance for a set of product carriers. We assume these travel distances are a proper approximation of the true travel distances. A detailed explanation on how this pre-computed dictionary is created, is described in Section B.0.1.

4.2.3 Creating a new batch

When creating the morning list (this will be described later) or, as mentioned before, when there are no assigned orders in a put wall, an attempt will be made to create a new batch. In this section, we will explain how this process works in the simulation. First, we check if the number of orders in *preBatch* is larger than the size of the putwall, and if this is the case, we will create the batch. The *preBatch* object is a heapq, and we sort on the priority of arriving orders. The priority of an order is simply the arrival time plus a constant, but this property could potentially be used to prioritize certain orders in the batch picking algorithm. To create the batch, we take the first order of *preBatch* and add it to the batch. Then, we loop over all order lines (product types) in this order. We take the first z elements from *preBatch*, where z is the horizon (meaning we can only use the z most urgent orders to create a batch), and check if any order has order lines that overlap with the order lines in the batch. All orders that have overlap, are added into the batch, until the number of multi-item orders in the batch has reached its maximum. When this point is reached, only single-item orders with overlap will be added to the batch. This is to avoid exceeding the physical size (number of compartments) of the put wall, and as mentioned before, single-item orders are placed in a separate single-order carrier, meaning the physical size of the put wall is not exceeded. When the loop over all order lines is done, we repeat the loop for all order lines of all orders we just added to the batch. When these loops are finished, we check if the number of multi-item orders in the batch is on its maximum. If this is the case, the batch is done and we return the batch. If this is not the case, we add the first order from *preBatch* that has not been selected yet to the batch, and repeat the process. This is done until the maximum number of multi-item orders in the batch is reached. Then, the orders will be added to the assigned orders of this station. Every unique order line is added to a list once, since if an order line is required in multiple orders, the product carrier only needs to be retrieved once. This list is then added to Qs0. A compact overview of the batch picking algorithm can be found in Algorithm 2.

Algorithm 2 Creating a new batch

```
1: procedure makeNewBatch
2:     station = next station where # assigned orders is zero
3:     if station exists and length preBatch is larger than putwall size then
4:         batch = computeBatch()
5:         add batch to assigned orders of station
6:         add batch to Qs0: addProductsToQs0()
7:     else
8:         return: no batch can be made.
9: procedure computeBatch
10:    Create list BatchOrders and BatchProducts, MIObatchsize = 0, prodIndex = 0
11:    while MIObatchsize < putwall size do
12:        try: neworder = pop next order from preBatch
13:        add neworder to BatchOrders
14:        for o doorderline in products of neworder
15:            if orderline not in BatchProducts then
16:                add orderline to BatchProducts
17:        if Number of products in order is larger than 1 then
18:            MIObatchsize += 1
19:        while prodIndex < length of BatchProducts do
20:            indexPicklist = 0
21:            picklist = first (z-length(BatchProducts)) orders of preBatch
22:            while indexPicklist < min(z-length(BatchProducts),length(preBatch) do
23:                if BatchProducts(prodIndex) in picklist(indexPicklist) then
24:                    if Batch is not full, or order is a single-item order then
25:                        Add order to BatchOrders
26:                        for products in this order do
27:                            If not in BatchProducts, add to BatchProducts.
28:                        If an multi-item order; MIObatchsize += 1
29:                        Remove order from preBatch and heapify this queue.
30:                        picklist = first (z-length(BatchProducts)) orders of preBatch
31:                    else
32:                        indexPicklist += 1
33:                else
34:                    indexPicklist += 1
35:            prodIndex += 1
36:    return BatchOrders
```

4.2.4 Vehicle arrival

The second type of event is a vehicle arrival, where we have a list of product carriers that have been retrieved by a vehicle. This list is first removed from Qs1; meaning that the vehicle is done retrieving the product carrier(s). The next part of the simulation works differently for single order picking or batch picking.

Single order picking For single order picking, we start by looping over the product carrier(s) that the vehicle retrieved. For every product carrier, we check if the order belonging to this product carrier is already assigned to any station, and if so, we select the corresponding station. If the order is not assigned to any station, we check if there is any station where the number of assigned orders is not equal to its maximum. However, we want to spread the load equally to all workstations, to avoid one workstation having much work while another workstation is idle. The following will be done to solve this; every time we encounter this state (a new order will be assigned to a workstation), we alternate the workstation where the order will be assigned to. So for example, if we have 3 workstations, the first order is assigned to station 1, the second order at station 2, the third order at station 3 and the fourth order again at station 1. If a workstation has been chosen this way, the product carrier is moved on the conveyor of this workstation and the product carrier is added to WsX, where X is the station we just selected. When an order is assigned to a station, we check if the operator at station X is idle. If this is the case, we schedule an item consolidation. We have a fixed operator time, and therefore, the time of this new event is the current time plus this fixed parameter. We add this new event in the FES, accompanied with the product and the order it belongs to. However, it is possible that not all product carriers can be moved on a station. All product carriers that are not moved on a station are added to a list. When all product carriers have been looped over, we check if this list has a positive length, and if this is the case, the list is added to Qs2. This means that the vehicle is waiting in front of the workstation, holding the product carriers that are in this list. If the list does not have positive length, the vehicle can collect new product carriers. This means we check if Qs0 has enough product carriers (larger or equal to the carrying capacity of the vehicle), and if so, we take the first y product carriers (where y is the vehicle carrying capacity) from Qs0, and add this list of product carriers to Qs1, and schedule a new vehicle arrival event.

Assumptions for single order picking Concerning this process, we have two assumptions. The first assumption is that dropping off a product carrier, travelling to a different workstation, and moving the product carrier to an operator is instant, meaning that if the vehicle arrival is at time t, the product carrier is placed on WsX at time t and an operator can also reach the product carrier (and start consolidating an order) at time t. In reality, both of these processes take some time, meaning this assumption is unrealistic. For the second assumption, we note that in advanced systems, it is possible to detect if the last product carrier of an order is placed on a conveyor. If this is detected, a new order could already be assigned and new product carriers could be dropped off at the conveyor as well. This option is not taken into account in the model, but if it were, it would have been a more realistic assumption that an operator can instantly reach a new product carrier after completing an order at the workstation. Therefore, the two assumptions weakly compensate for each other, meaning that the assumptions used in the simulation are moderately realistic.

Batch picking For batch picking, we are sure that a vehicle can drop off all product carriers, and therefore we do not have to take waiting vehicles into account. Therefore, we can loop over all product carriers retrieved by the vehicle, and directly add them on the workstation, which is again instant. A product carrier is only required at one specific workstation, which is a property of this product carrier. Therefore, we can easily add the product carrier to the corresponding WsX queue, and then we check if the operator at station X was idle. If

this is the case, we schedule a consolidation. We take the current product carrier, and find the first order of all orders that the product needs to supply. Then, we add a new event to the FES, where an item of the current product carrier is moved to the order carrier of the selected order. The time of this event is the current time plus a fixed time to pick one item, and we also add the product exchange time if this is the first item the operator takes from the product carrier. The exchange time is the time it takes for an operator to push away the old product carrier and to receive a new product carrier. If the vehicle carries a rack, this exchange time only includes the time to read the instructions and find the correct SKU in the rack. When all product carriers are dropped off, the vehicle will be available to retrieve new product carriers. If the vehicle does not carry a rack, and the total number of product carriers in Qs0 is larger or equal than the vehicle carrying capacity, the first items (1 or 5) of a specific index of Qs0 (which rotates between all indices every time new product carrier(s) are retrieved, to spread the load of the workstations) will be selected for the vehicle to retrieve. A new vehicle arrival event is scheduled with this time plus a total travel time (which includes a randomly drawn travel distance), and the product carriers the vehicle is retrieving are accompanied as well. If the vehicle carries a rack and Qs0 has any item, we will find the product carriers that are most optimal to retrieve (the method to find this, is described below). This procedure returns the total time and the type of items that will be retrieved. These item types can be removed from Qs0, and added in a list to Qs1, and then a vehicle arrival event is scheduled. Notice that if there are not enough product carriers in Qs0, we take no further action.

Finding optimal rack to retrieve It proved to be very challenging to create a general formula or distribution to find a suitable set of SKUs retrieved by a randomly chosen rack without saving where all SKUs are located. One of the main reasons for this, is that at first, many racks can be found with many SKUs from Qs0, however, once these racks are retrieved, the remaining SKUs have less racks in common, and therefore the number of product carriers retrieved from one rack is possibly lower. This number will vary even more when some put walls are finished (while other put walls are not), and new orders (and therefore SKUs) are assigned (by creating a new batch). Therefore, to avoid complicating the simulation by trying to model this behaviour, we chose to create a specific rack layout. This means that for each instance of the simulation, we define what SKUs can be found together in one rack, and this will stay fixed during this instance. We have chosen to do this, because when batch picking, when deciding what items the vehicle will retrieve, this will be optimized to retrieve as many items (in one go) as possible (this will be explained in detail later). To make the layout, we first create a list of all product numbers, and we add them multiple times according to its occurrence. So product number 1 is added five times (because it represents a fast mover), while the last product number is only added once (because it represents a slow mover). Then, given how many different product carriers can be found in a rack (in this case 75, based on peer review), the required number of racks is determined. This is equal to the length of the list of product numbers divided by 75, and then rounded up. The total number of empty spots is equal to the total number of racks times 75, minus the length of the list of product numbers. Now, we have a list of all product numbers and some empty spots. We perform a random permutation on this list, to create a random ordering of products to racks. The first 75 SKUs are in rack 1, the next 75 are in rack 2, etcetera. This means we have the assumption that when assigning products to racks, duplicate SKUs in one rack are not avoided, and based

on peer review, this is a realistic scenario.

4.2.5 Item consolidation

The last type of event in the FES is an item consolidation, which means an operator finished moving one item from a product carrier (or rack) to an order carrier. We take the list that states which items this order is demanding, and we remove the current item from this list (once, since there could be duplicates), and we also increase the total number of items finished by one. Then, we check if this order is completed. If this is the case, we remove this order from the assigned orders of this station. The next steps are different for the two simulations.

Single order picking For single order picking, if an order is completed, we want to know if any vehicles were waiting; since there is now a free spot to reserve an order at the workstation. If this is the case, we can check which product carriers the waiting vehicles are holding to find a new order to reserve. Although it is possible to find the most optimal order to reserve, we simply take the first product carrier from the first vehicle in Qs2, which is the longest waiting vehicle, to avoid complexity of the simulation. The order belonging to this product carrier is assigned, and the first product carrier of this vehicle is moved on the station. Then, for all vehicles in Qs2, we check if it is holding a product carrier that is required for this newly accepted order. If this is the case, the product carrier is moved on the conveyor of the workstation. If a product carrier moved on the station is the last product carrier the vehicle was holding, we check if there are enough products in Qs0, and if this is the case, the vehicle will be initiated to retrieve new product carrier(s). In this case, a new vehicle arrival event will be scheduled as well. Now we are done with Qs2, the next step is to remove the current product carrier the operator is holding from the workstation (thus from WsX), since this product carrier can only be used for one order. Then, we check if there is a new product carrier on the conveyor, and if this is the case, a new item consolidation with this product carrier is added to the FES. This process uses the assumption that products are instantly reachable by the operator once the vehicle drops off an item, and the dropoff time is not taken into account.

Batch picking For batch picking, if an order is completed, we remove the current order from the list of orders the current product carrier needs to supply, and then we check if this was the last order on the list. If this is the case, we remove the product carrier from the workstation queue (WsX). Then, we check if there are any product carriers left on the conveyor, and if this is the case, we schedule a new item consolidation with this first product carrier in the queue (WsX). If this was not the last order from the list of orders, we are sure that this product carrier can directly supply a different order (since all orders of this product carrier are already assigned), and therefore we schedule a new consolation with this item, now with the next order the product carrier needs to supply. Furthermore, we check if all orders of the putwall are completed, by checking if the number of assigned orders at this station is zero. If this is the case, we attempt to create a new batch, which is done as described in section 4.2.3. Now the workstation has to wait idly for 40 seconds, however, the vehicles can start retrieving product carriers during these 40 seconds. This means we instantly create a batch and add it to Qs0 as mentioned before. Product carriers can be retrieved and dropped off at the conveyor at the workstation, but the operator can only start picking items after the 40 seconds are over.

4.3 Initialisation the simulation

We have described the behaviour of the main simulation, but before starting some runs, we have to initialize the simulation. One of these steps is to initialize all the queues and parameters that are described or referred to in Section 4.2. Then, we assume that during the night, many orders have arrived, and are ready to be processed. These orders will be placed in a *morning list*. We will assume that this morning list has so many orders, that the system will never run out of work. In this way, it is easier to compare the configurations, and the runs will be less based on luck; we can rule out that a certain configuration performed better because more orders arrived, or orders had relatively many items. With this assumption, we can compare the configuration while the throughput is not limited by the arriving orders. How this morning list is created and how some events are created with this list is described in Section 4.3.1.

During a simulated day, the total number of deployed vehicles and workstations is fixed. Each combination of number of vehicles and number of workstations can give a different throughput. Therefore, if a required throughput is given, it is of interest to find what combinations of number of vehicles/workstations can reach this throughput. Additionally, we aim to minimize both numbers to avoid overcapacity, which leads to unnecessary costs. In Section 4.3.2, we will describe the theoretical minimum for the number of vehicles and workstations separately, such that the output of the vehicles or workstations is minimally the throughput. However, in reality, the workstations and vehicles have interaction and sometimes have to wait for each other, meaning with the theoretical minimum, the required throughput can not always be reached. To solve this, we will run a small portion of the main simulation to test if a certain number of vehicles/workstations can reach the required throughput together. This testing is done with so-called "test runs", and these test runs are described in Section 4.3.3.

4.3.1 Morning list

Before a simulated day, we create a morning list with orders that have been ordered overnight. As mentioned before, we assume that the morning list is so large, that the system will never run out of work. This means that the length of the list depends on the required throughput; if the throughput is higher, we expect to finish more orders and therefore need a longer morning list. With a required throughput of 2000 items/h, and a mean of 2.4 items per order, this means we expect ≈ 833 orders to finish each hour. If we were to run a simulation of 8.5 hours (how we deduced this number is described in Section 4.3.4), the expected number of orders finished is $8.5 * 2000/2.4 \approx 7083$ orders. However, we want the system to never run out of work, even if the system has a slightly higher throughput than 2000, and additionally, with batch picking, we want to have enough orders to create a batch with high synergy (preferably, the length of the list of orders in *preBatch* is larger than the horizon). Therefore, we round this 7083 to 10.000 orders, and we assume that these are enough orders to never run out of work. Similarly, for a different required throughput or time frame, we can compute the length of the morning list, and in all cases we will round up to prevent running out of orders. The total length of morning lists that we used in the simulation can be found in Table 9. Some of the numbers might appear lower than expected, however there are still orders arriving throughout the day ($3600/(6/5) = 3000$ orders per hour, on average), and with these orders taken into account, the morning list is assumed to still be of sufficient length.

Required throughput	Time frame	Length of morning list
2000	1 hour	1.000 orders.
2000	8.5 hours	10.000 orders.
10.000	1 hour	10.000 orders.
10.000	8.5 hours	35.000 orders.

Table 9: Length of the morning list in different scenarios.

We will now describe what actions are taken concerning the morning list, which is again different for single order picking and batch picking.

Single order picking We start by discussing single order picking. For each order in the morning list, we draw the number of items and the types of items. Then, for each item, the product carrier(s) that need to be retrieved for this item is added to Qs0. Whenever a product carrier is added to Qs0, we check whether any vehicle is idle. If there is any idle vehicle and Qs0 has equal or more product carriers than the carrying capacity of the vehicle, the vehicle will be requested to retrieve the first y product carriers of Qs0, where y is the carrying capacity. Knowing which product carrier(s) the vehicle will retrieve, we can draw a random travel distance and compute when the vehicle will arrive at the workstation. At this computed time, a vehicle arrival event is scheduled and added in the FES.

Batch picking For batch picking, this works somewhat differently. For each order we draw the number of items and the type of items, and we add all orders to *preBatch*, which contains all orders that are not batch picked yet. Only after all orders of the morning list are added to *preBatch*, we will start creating batches. For each workstation, we create a batch as described in Section 4.2.3. A batch contains a specific set of orders, which will first be removed from *preBatch*. A list of all product carriers that need to be retrieved for this batch is added to Qs0, meaning the vehicles can retrieve them. We check for idle vehicles (until there are no more idle vehicles) and if there are, we will determine what product carriers this vehicle will retrieve. How we select what product carriers a vehicle retrieves, has been described in Section 4.2.4.

4.3.2 Minimum number of vehicles/workstations

Later in the test runs we will find the practical minimum of vehicles and workstation to reach the required throughput, however, to give an appropriate starting value to the test runs, we first compute the theoretical minimum. One of the benefits of knowing this theoretical minimum, is that the test runs will be completed faster, since the theoretical (input) value will most likely be close to the practical minimum.

Minimum number of vehicles For vehicles, we are interested in knowing the "cycle time", which is the time one vehicle takes to travel along all storage locations, pick up (at the workstation) and drop off (in storage) discarded product carriers and pick up (in storage) and drop off (at the workstation) all requested product carriers. We devise the mean travel distance from the pre-computed dictionary as described in Section 4.2.2, and use the mean speed and fixed total pickup/dropoff to compute the mean cycle time. Note that the vehicle

speed is constant in this model, we assume that vehicles move independently (never blocking each other's way) and acceleration and deceleration is not taken into account.
The final aspect taken into account is that a vehicle possibly carries multiple product carriers, or even a rack, which means more work is done for each cycle. This gives a minimum number of vehicles for single order picking;

$$\min \#\text{ vehicles} = \left\lceil \frac{\text{required throughput per hour}}{\#\text{ different items to utilize} \cdot 3600/\text{cycle time}} \right\rceil$$

With batch picking, the number of vehicles can be even lower since one product carrier at the workstation can potentially be used to fulfill multiple orders. Therefore, with batch picking, the minimum number of vehicles can be divided by the synergy factor.

Minimum number of workstations To determine the minimum number of workstations, we start by looking at the time a workstation takes to move one item from a product carrier to an order carrier. For single order picking, this time consists of waiting until a new product carrier appears, waiting until a new order carrier appears (if applicable), and picking the item (moving the item from product carrier to order carrier). The number of items in an order is geometrically distributed with g, and therefore the mean number of items in an order is $\frac{1}{1-g}$. This means that for a random item, with probability $1-g$ the order is completed and the operator needs to wait for both the order carrier and product carrier to appear (which can be done in parallel). With probability g, the order is not completed, but since an item is only used for one order, only the product carrier needs to be exchanged. This gives the theoretical minimum number of workstations;

$$\min \#\text{ workstations} = \left\lceil \frac{\text{picktime} + 1 - g \cdot \max(\text{order ex.t.}, \text{product ex.t.}) + g \cdot \text{product ex.t.}}{3600/(\text{throughput per hour})} \right\rceil$$

When batch picking, we assume that for every utilized item in the product carrier, we have a certain pick time, and we have an exchange time for every product carrier. However, multiple items can be used from each product carrier, depending on the synergy. With synergy s, we have probability $\frac{1}{s}$ that a random item is the first item taken from the product carrier, meaning the product carrier needs to be exchanged. There is an even smaller probability that a random item completes the last order of the put wall. In this case, the put wall needs to be exchanged (full put wall is moved away, and a new, empty put wall is placed) which takes e seconds. If a putwall is expected to require i items, then the minimum number of workstations is;

$$\min \#\text{ workstations} = \left\lceil \frac{\text{Pick time} + \frac{1}{s} \cdot \text{Exchange time} + \frac{1}{i} \cdot e}{3600/(\text{required throughput per hour})} \right\rceil$$

4.3.3 Test runs

This theoretical number of workstations/vehicles can be used to compute the practical minimum, which will be described in this section and can also be found in Algorithm 3. An important assumption in this process is that we assume that a low number of workstations

is of much higher priority than a lower number of vehicles, since the goal is to automate the warehouse as much as possible. We start by taking the minimum workstations, and the number of vehicles as the minimum number of vehicles plus ten. With these two numbers, we run four (equally distributed) test runs. A test run works exactly the same as the main simulation, the only difference is the time frame; we only run a small portion of the day, specifically, one hour with a warm-up time of thirty minutes. For each test run, we determine the achieved throughput. If for all four runs the achieved throughput is higher than the required throughput, we consider the check passed, otherwise the check is considered to have failed. We keep increasing the number of vehicles by 10, until one of the checks is passed. If none of the checks passed after ten tries, we expect that an increase in the number of workstations can not be avoided. Therefore, the number of workstations will be increased by one, the number of vehicles is set at the theoretical minimum plus ten, and the process is repeated. At one point, a check will pass. In this case, we keep lowering the number of vehicles by one, until a check fails. The last check that passed determines what number of vehicles and workstations we will use for the main simulation.

Algorithm 3 Testing the simulation for required throughput

```
1:  procedure Initialization
2:      (...)
3:      Define minimum number of vehicles (minV) and workstations (minW).
4:      Set workstations on minW, vehicles on minV.
5:      while Throughput check is not met do
6:          Vehicles += 10
7:          if Vehicles > minV + 100 then
8:              Workstations += 1
9:              Vehicles = minV
10:         TestRun()
11:         while Throughput check is met do
12:             Vehicles -= 1
13:             TestRun()
14:             if Throughput check is not met then
15:                 Return current workstations and current vehicles +1.
16:     (...)
17: procedure TestRun
18:     Set required throughput
19:     Runs is 4, startup time 1800, finish time is 3600
20:     run main simulation until time 3600, save # items finished after time 1800.
21:     Check if for all runs, (3600 · # items finished)/ (finish time - startup time) ≥ through-
    put
22:     Return if throughput is met
```

4.3.4 Running and verifying the main simulation

When running the main simulation, we start by taking the first event in the FES and select the time of this event, which we call t. Then, we run a while loop for $t < T$, where t represents the current time and T is the maximum time of the simulation, which is in this case 30600

seconds (eight hour and thirty minutes). We chose this number because a regular work-day is eight hours, and furthermore we have a warm-up time of thirty minutes, which we deemed to be enough for the purposes of this research. Whenever an event of the FES has a time larger than T, the simulation instance will end. With the simulation ready and all parameters defined, we can run the simulation to analyze and compare the behaviour and performance of different configurations. For every configuration and unique set of parameters, we have 100 independent runs of the main simulation. From these runs, we can extract the key performance indicators of interest, and in the next chapter, we will discuss the performance and results for all configurations.

We also verified the simulation, to make sure we did not make any coding mistakes, which could give unintended system behaviour. To verify the simulation, we printed queues and other statistics during the run, for example to verify if the product carriers indeed go to the right queue. We also printed statistics to verify if the simulation applies all parameters correctly. Furthermore, results and behaviour have been subjected to peer review to verify if it behaved as intended.

5 Results of simulation model

In this chapter, the results of the simulation as described in chapter 4 are assessed and discussed. We also aim to find an answer to the research question, namely what the best order picking strategy is (out of the 7 (Table 2) discussed strategies) for a new warehouse. In the simulation model we described some warehouse layouts (that depend on the vehicle type and number of SKUs) that were used to run the simulation, and for these warehouses we aim to find the best order picking strategy.

5.1 Vehicle speed

Instead of showing the general results separately, we will look at the effect of the vehicle speed on the performance as well. In this section, a speed multiplier of one gives the most general results and these can be found in Figure 18 and Tables 10, 11 and 12. With the number of runs we did, we reached a high precision, which means that for the vehicle and workstation idle time, we can say with 95% certainty that the mean from the runs deviates at most 0.001 from the true mean of the simulation model. We did not compute a confidence interval of the throughput, since we do not require a very precise value. The number of workstations, computed by the test runs (Section 4.3.3), turned out to remain constant per configuration, which is respectively 3, 4, 3, 4, 5, 5 and 5 workstations per configuration.

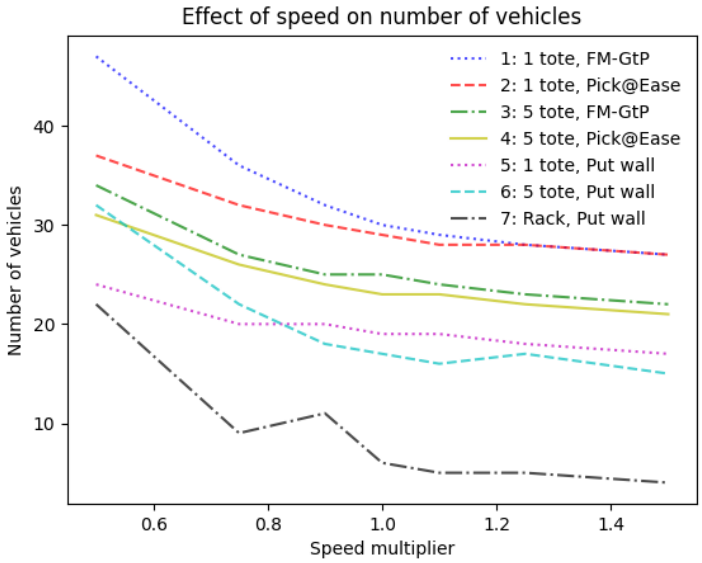


Figure 18: Effect of vehicle speed on the required number of vehicles

We start by investigating the effect of the vehicle speed on the total number of required vehicles in a certain configuration. Figure 18 shows that for single order picking, the total number of vehicles for configurations 1-2 (a vehicle carrying 1 product carrier) seems to be grouped, similar to configurations 3-4 (a vehicle carrying five product carriers). This makes a lot of sense, since the warehouse layout is the same for configuration 1-2, and configuration 3-4. Furthermore, the travel distances, mean vehicle speed and pickup times are also the same for configuration 1-2, and for 3-4. The only difference between configuration 1-2 or 3-4 is what

the process is at the workstation. Table 10 shows the vehicle idle time, and from this can be derived that we are confident that the idle time of a vehicle while using a medium workstation (configuration 2 and 4) is nearly zero. Practically, this means the vehicle never has to wait in front of the workstation. The vehicle idle time for a small workstation (configuration 1 and 3) is small, but not zero. This means there is some idle time because the vehicle has to wait in front of the workstations; which makes sense because only one order can be accepted at a time, and there is a small chance that all workstations are full and waiting for a specific product carrier. This behaviour and the results from Table 10 explain why configurations that use a small workstation require more vehicles than configurations using a medium workstation.

Speed multiplier → Configuration ↓	· **0.5**	·**0.75**	·**0.9**	·**1**	·**1.1**	·**1.25**	·**1.5**
1	0.244	0.140	0.095	0.074	0.062	0.050	0.038
2	0.000	0.000	0.000	0.000	0.000	0.000	0.000
3	0.082	0.054	0.046	0.046	0.043	0.039	0.036
4	0.000	0.000	0.000	0.000	0.000	0.000	0.000
5	0.027	0.011	0.054	0.030	0.052	0.033	0.019
6	0.503	0.370	0.247	0.208	0.178	0.285	0.175
7	0.447	0.189	0.122	0.092	0.050	0.124	0.105

Table 10: Ratio of idle vehicles during the day

Speed multiplier → Configuration ↓	· **0.5**	·**0.75**	·**0.9**	·**1**	·**1.1**	·**1.25**	·**1.5**
1	2020	2057	2037	2013	2023	2042	2074
2	2104	2126	2111	2102	2082	2148	2155
3	2112	2042	2027	2094	2074	2064	2068
4	2098	2078	2040	2020	2076	2054	2047
5	2479	2459	2499	2496	2502	2500	2491
6	2258	2265	2271	2279	2279	2293	2295
7	2148	2134	2130	2131	2130	2144	2147

Table 11: Achieved throughput (items/h) during the day.

Furthermore, we want to point out that for Table 10, the number of vehicles is not constant per row, since we want to determine the minimum number of vehicles for each speed multiplier (and configuration). If the number of vehicles would be constant, the idle time would eventually increase if the speed increases (and the number of workstations is constant), but as mentioned before, the number of vehicles is not constant.

Now we take a look at batch picking. The number of vehicles is clearly lower compared to single order picking, which can easily be explained; this is because with batch picking, we have synergy between orders. This means less vehicle movements are necessary, which means to meet the required throughput, less vehicles can be deployed. Configurations 5-7 all use the same workstation (a put wall), so we will investigate the difference between the vehicles

and possible effect on workstation performance. To explain the difference, we can look at the mean cycle time. The mean travel distance can be computed using the dictionary we created (Section 4.2.2), which is $30.04m$ for a vehicle carrying one product carrier, $97.48m$ for a vehicle carrying five product carriers, and $112m$ for a vehicle carrying a rack. The number of product carriers a vehicle brings is fixed for a vehicle that carries one or five product carriers, however this number is variable (depending on the rack layout, batches and the orders) if a vehicle carries a rack. For demonstration, we ran a small simulation which gave a mean travel distance of $112m$ and a mean of 3.3 utilized product carriers per rack. This gives cycle times;

- 1 product carrier: $36 + \frac{30.04}{2.2} = 49.65s$ per cycle
- 5 product carriers: $144 + \frac{97.48}{1.6} = 204.925s$ per cycle
- Rack: $12 + \frac{112}{1.3} = 98.15s$ per cycle

A vehicle possibly carries multiple product carriers, and therefore the cycle time can be translated to the mean time it takes to retrieve 1 product carrier, which is:

- 1 product carrier: $\frac{49.65}{1} = 49.65s$
- 5 product carriers: $\frac{204.925}{5} = 40.985s$
- Rack: $\frac{98.15}{3.3} = 29.74s$

If we, for example, look at the default speed (multiplier 1), then the number of vehicles for batch picking is 19, 17 and 6 for respectively configurations 5, 6 and 7. To find the difference between the configurations, We can compute $\frac{19}{49.65} \cdot 40.985 = 15.68$ which is somewhat close to 17 vehicles. This outcome does not perfectly correspond with the results. The expected reason is the idle time of the vehicles, shown in Table 10. Especially at configuration 6, there is a significant ($\approx 20\%$ at default speed) idle time. We have two possible explanations for the rise in idle time, explained in the next two paragraphs.

Unretrieved product carriers First, the workstations all release batches of at least 50 orders, and all vehicles are immediately initiated to retrieve all required product carriers. In the simulation, we assumed that vehicles alternate for which put wall it retrieves product carriers. Therefore, it is possible that if all put walls require a similar number of product carriers, all workstations are "full" at the same time. This means that all product carriers are either on a vehicle or on a conveyor in front of a workstation. The vehicles have to wait until one of the workstations finishes its batch, and only then a new batch is released. During this waiting time, some vehicles can be idle. However, in the case a vehicle carries five product carriers, we assumed that this vehicle can not carry less than five product carriers. Therefore, it is possible that less than five product carriers remain to be retrieved, but will not be retrieved until more product carriers are released through new batches, this scenario is also visualised in Figure 19. In this specific situation, we have to wait until all product carriers are retrieved for workstations that do not require some of the unretrieved product carriers, then we have to wait until this workstation is finished so a new batch is released. Although we are not sure how often this situation occurs, we expect that it occurs and therefore result in vehicle idle time (vehicles have to wait longer for this specific workstation to finish) and workstation idle time (even though a workstation could be done, new product carriers will only start arriving after one specific workstation is done, resulting in idle time).

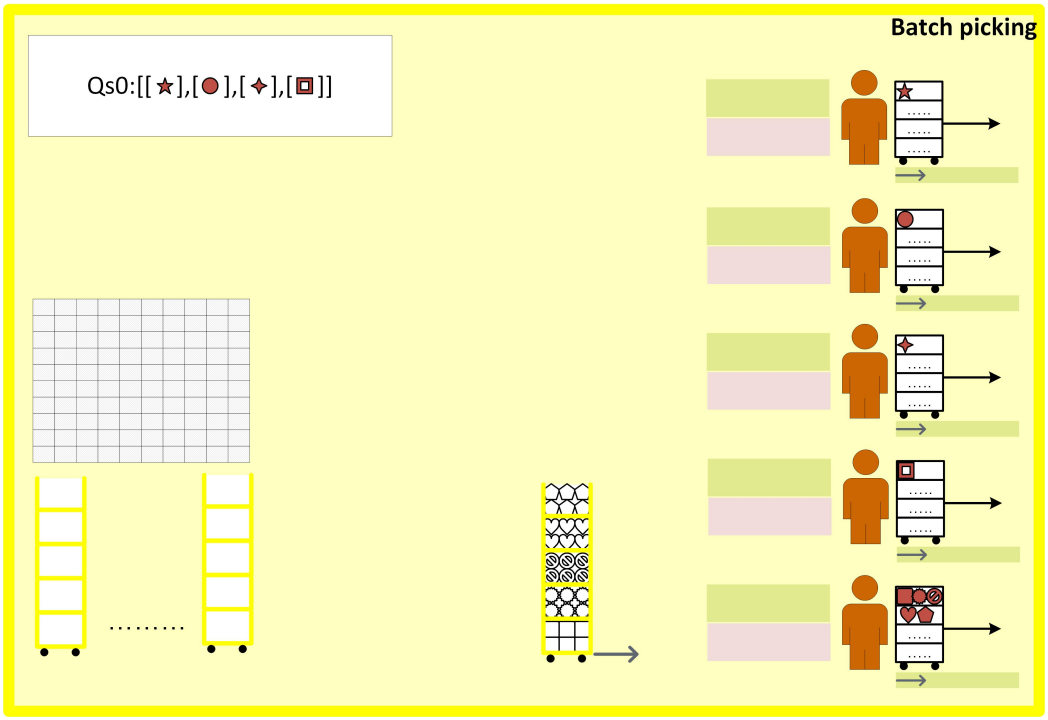


Figure 19: Worst-case scenario if some product carriers remain unretrieved.

"Test runs" The second reason is related to the "test runs" we do before the main simulation. These tests have a relatively small time frame, and additionally, we do multiple testing, which means it is possible we unnecessarily reject a certain number of vehicles for being too small, while actually this number would have been sufficient. Therefore, we expect that it is likely that in quite some configurations, the actual number of vehicles could be lower. If we deploy too many vehicles while the number of workstations is at its minimum (and therefore the bottleneck of the system), then it is likely that vehicles are idle, simply because the workstations can not keep up with the work the vehicles are offering, resulting in vehicle idle time. These "test runs" are also an explanation as to why the number of vehicles is in some cases (visually very clearly for configurations 1 and 6) higher than expected when the speed of a vehicle is lower. For example, in configuration 1, the mean travel time is $\frac{30.037}{2.2}s$. If the speed is halved, the mean travel time is now $\frac{30.037}{0.5 \cdot 2.2} = 2(\frac{30.037}{2.2})$, meaning the time is doubled. However, the interesting part is that the variance $\text{Var}(w) = \frac{v}{2.2}$ is now $\frac{v}{0.5 \cdot 2.2} = \text{Var}(2w) = 4\text{Var}(w)$, which means the variance is four times higher. With such a high variance, it is likely that a test run gives highly varying travel times, and with the short time span, this certainly could influence the throughput. One can imagine that if a specific vehicle has a high travel time, the workstation that requires this product carrier has to wait relatively long, resulting in workstation idle time (and therefore vehicle idle time). This effect is even more significant if the workstation has to wait for only one product carrier (small workstation) instead of maximally five (medium workstation), which would explain why the number of vehicles (with low speed) is so high for configuration 1, and why the vehicle idle time is relatively high.
It is important to notice that in reality, increasing the vehicle speed can only be done to a certain degree. Furthermore, increasing the speed could possibly give other issues, like lots

of traffic at busy locations, for example close to the workstations. These two factors are not taken into account in the figures and tables, but are important to remember while finding a suitable vehicle.

5.2 Vehicle pickup/dropoff time

Similar to varying the vehicle speed (Section 5.1), we will now vary the vehicle pickup/dropoff time. The results are very similar to the results of Section 5.1, and therefore we will not go into much detail. The results can still be found in Figure 20 and Tables 27, 28 and 29. With the test runs, we found that the number of deployed workstations is constant per configuration, and is per configuration respectively 3, 4, 3, 4, 5, 5 and 5 workstations.

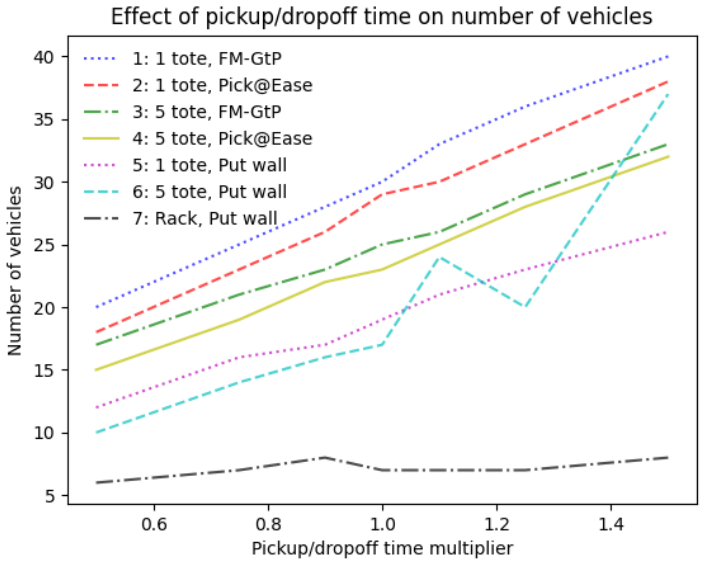


Figure 20: Effect of vehicle pickup/dropoff time on the required number of vehicles

The ratio between the pickup/dropoff time and travel time for a vehicle carrying 1 product carrier is $\frac{36}{30/2.2} \approx 2.647$ and for a vehicle carrying 5 product carriers $\frac{144}{97/1.6} \approx 2.375$. This means that if the speed is halved, the ratio is expected to be approximately $2.647/2 \approx 1.32$ which means with 20 vehicles at half speed, the estimated number of vehicles at default speed is $20 \cdot 1.32 \approx 26.47$. In this case we have 30 vehicles, but for this estimation we do not take variance, vehicle idle time and unlucky test runs into account, therefore we find this close enough. The ratio between the pickup/dropoff time and travel time also explains why the slope between configurations 1-2 and 3-4 is somewhat different, but still very similar; a vehicle carrying 1 product carrier has a slightly higher ratio of vehicles (between two different speeds) meaning we indeed expect a slightly steeper slope.
A different aspect of the figure that strikes attention is the strange behaviour of configuration 6; the number of vehicles seems to be off for speed multipliers 1.1 and 1.5. This is indeed the case, Table 28 shows that there is no additional workstation idle time (but a very low idle time, meaning the workstation is the bottleneck of the system). A quick look in Table 27 (shown in blue) shows that there is however a significant increase in the vehicle idle time.

We expect that this behaviour occurs because of unlucky test runs, which also explains why the number of vehicles and vehicle idle time seems to be stable when not including speed multipliers 1.1 and 1.5.

5.3 Workstation idle time

We will now turn our attention to the workstation idle time for varying vehicle speed. Since varying the vehicle pickup/dropoff time gave similar results, we will only look at varying vehicle speed, and the results can be found in Table 12. As mentioned before, if we have a low vehicle speed, some of this idle time can be induced by a workstation waiting for a vehicle that is very slow. Furthermore, some workstation idle time can also be induced if a vehicle carrying five product carriers does not retrieve the final product carriers, and therefore workstations have to wait for each other. However, most of the workstation idle time is for a very different reason, which is rounding. We will illustrate this effect using configurations 1-4 for single order picking. The minimum number of workstations for a throughput of 2000 items per hour is:

- Small workstation: $\frac{2000}{3600/(3+0.55+0.45)} = 2.22$
- Medium workstation: $\frac{2000}{3600/(4.7+0.55+0.45)} = 3.1667$

In reality, we can not use 2.2 workstations and therefore we round to 3 workstations. This is percentually a significant increase; 35% for a small workstation and 26% for a medium workstation. Initially, one could argue that the throughput will scale accordingly, but this is not true. Because both the number of workstations and vehicles is minimized such that a throughput of 2000 items per hour is achieved, the vehicles will only give work close to 2000 items per hour. This means that even though the workstations could achieve more items per hour, the vehicles are the bottleneck of the system. The number of vehicles also has to be rounded, but the number of vehicles is relatively high compared to the number of workstations, meaning that the percentage increase because of rounding is much lower. We can compute the expected idle time using the expected throughput of the workstation and the realized throughput, which can be found in Table 11. The expected idle time using default speed and using a vehicle that carries one product carrier is:

- Small workstation: $1 - (2013/(3600/4))/3 = 0.2544$
- Medium workstation: $1 - (2102/(3600/5.7))/4 = 0.1680$

This expected idle time is exactly within the confidence interval of the idle time given in Table 12, shown in blue, which means in this case (default speed and carry capacity 1) rounding is most likely the only contribution to the workstation idle time.

When batch picking, we also have some idle time because of rounding, but a different reason for idle time is the moment where a full put wall is moved away, and a new put wall is brought to the operator. During this time, we do not consider the workstation to be idle, since we consider idle time to be a redundant time, while switching a put wall is a necessary process. As mentioned before, if all product carriers have been retrieved, it is possible some (or all) vehicles are idle. When a workstation finishes the batch, a new batch is released. The time to renew a put wall is $40s$, however, the mean travel time is higher than $40s$. Therefore, it is possible no vehicles have arrived after the $40s$. However, with a mean cycle

time of approximately 50, 198 and 98 seconds (default speed) for configurations 5, 6 and 7 respectively, and multiple vehicles retrieving product carriers, we do not expect workstation idle time for configurations 5 and 7, which coincides with the results of Table 12. However, the mean cycle time is very high for configuration 6, and although renewing a put wall after all vehicles were idle does not occur often, we still expect that part of the workstation idle time originates from this behaviour.

Speed multiplier → Configuration ↓	· **0.5**	·**0.75**	·**0.9**	·**1**	·**1.1**	·**1.25**	·**1.5**
1	0.252	0.238	0.246	0.254	0.251	0.244	0.232
2	0.167	0.159	0.165	0.168	0.176	0.150	0.147
3	0.218	0.244	0.249	0.224	0.232	0.236	0.234
4	0.169	0.177	0.192	0.200	0.178	0.187	0.189
5	0.026	0.029	0.012	0.014	0.011	0.012	0.016
6	0.110	0.106	0.104	0.101	0.101	0.095	0.094
7	0.013	0.020	0.022	0.021	0.022	0.016	0.014

Table 12: Ratio of idle workstations during the day

5.4 Increasing the number of SKUs

To compare the given configurations, we want to investigate the influence of warehouse parameters on the performance. Therefore, we increased the number of SKUs from 10.000 to 100.000. This increase had a heavy impact on simulation performance, and because of time constraints, we only have results for two numbers of different SKUs. The results can be found in Tables 13, 14, 15, 16 and 17.

Number of workstations. First, it is important to notice that when increasing the number of SKUs, the number of workstations remains the same, except for configuration 6 and 7, where the number is increased by one. An increase of SKUs does not affect workstation speed or performance, which is the reason why the number of workstations remains constant in most cases. If the number of SKUs increases, we need more vehicles (we have a larger warehouse and therefore increased travel distances). When a new batch releases, only a limited number of vehicles can retrieve product carriers for this batch simultaneously. This means that after adding a certain number of vehicles, adding more vehicles has little effect on the throughput, since there are not enough put wall compartments to keep all vehicles occupied. This exact behaviour explains that for configuration 6 and 7, we need an additional workstation that gives more work to the vehicles, meaning we can reach the required throughput.

Increase in vehicles. We turn our attention to the difference in deployed vehicles for an increase in SKU. From the table we derive that we need more vehicles, which makes sense because we have more items and a larger warehouse. Recall the assumption that we do not take traffic into account, meaning that there is no penalty for "too many" vehicles which would in reality block each other. For a vehicle carrying one product carrier, the mean travel distance increases from $30.037m$ to $71.3726m$. Taking the pickup/dropoff time into account, we expect that the number of vehicles could be multiplied by $\frac{36+71.3726/2.2}{36+30.037/2.2} \approx 1.378$. This

means that for configuration 1 we would expect 30 to go to 41 and for configuration 2 we expect 29 to go to 40. Table 13 shows that this is true for configuration 2, which explains why the vehicle idle time is extremely similar. However, this does not seem to be true for configuration 1. We expect that the variance of the travel distance increases, which means in some cases the workstations have to wait unexpectedly long for a certain product carrier. This gives more workstation idle time than expected, meaning the throughput is suddenly not met. To compensate for this, we add more vehicles that offer more work to the workstations, lowering the workstation idle time. It appears that in this case, we need 6 more vehicles to make sure we reach the throughput. Similar behaviour, where configuration 4 is as expected and configuration 3 is hard to explain, is seen for the differences in configurations 3 and 4, where the mean travel distance goes from $97.478m$ to $289.1889m$, therefore we will not go into this in detail.

Finally, we have configurations 5-7, where for configuration 5, we would expect 19 to increase to 26 vehicles. Again, more vehicles are used, and one reason to explain this number of vehicles is by using the workstation idle time. Like before, the idle time of the workstation does not increase, but this time the idle time is very small, where part of the idle time comes from the time it takes for new product carriers to arrive after a put wall has been renewed. This means that the workstations are the bottleneck of the system, and because we expect that the variance of the travel distance increases as well, this could lead to an unlucky test run. With this unexpectedly high number of vehicles, the throughput is also higher, which is behaviour we expected to see. The second reason is a lower synergy for 100k SKU, explained in Section 5.6. If we have a lower synergy, we need more vehicle movements to offer the workstation enough items, and therefore we need more vehicles.

# SKUs → Configuration ↓	**10.000**	**100.000**
1	30	47
2	29	40
3	25	46
4	23	37
5	19	35
6	17	48
7	7	39

Table 13: Number of vehicles used to achieve throughput of 2000 items per hour.

# SKUs → Configuration ↓	**10.000**	**100.000**
1	3	3
2	4	4
3	3	3
4	4	4
5	5	5
6	5	6
7	5	6

Table 14: Number of workstations used to achieve throughput of 2000 items per hour.

# SKUs → Configuration ↓	**10.000**	**100.000**
1	0.074	0.182
2	0.000	0.000
3	0.046	0.188
4	0.000	0.001
5	0.030	0.054
6	0.208	0.319
7	0.092	0.215

Table 15: Ratio of idle vehicles during the day

# SKUs → Configuration ↓	**10.000**	**100.000**
1	0.254	0.252
2	0.168	0.167
3	0.224	0.233
4	0.200	0.189
5	0.014	0.015
6	0.101	0.134
7	0.021	0.024

Table 16: Ratio of idle workstations during the day

# SKUs → Configuration ↓	**10.000**	**100.000**
1	2013	2021
2	2102	2104
3	2094	2071
4	2020	2049
5	2496	2257
6	2278	2377
7	2131	2210

Table 17: Achieved throughput (items/h) during the day.

5.5 Throughput of 10.000 items/h

We now aim to investigate the number of vehicles and workstations when we increase the system throughput. We increased the required throughput from 2000 to 10.000 items per hour, and the results can be found in Tables 18, 19, 21, 22 and 20.

The most interesting part we notice is that the workstation idle time (Table 22) is very low, especially compared to its counterpart where the required throughput is 2000 (Table 16). The results of this table show that the workstation idle times from 2000 items/h throughput indeed originate from the rounding of the number of workstations. With a required throughput of 10.000 items/h, the rounding of workstations is a smaller percentage increase, which means the number of deployed workstations is much closer to 10.000 items/h throughput, giving few idle time to be able to reach the required throughput.

There are some other interesting things to see here, for example the number of vehicles compared to a throughput of 2000 items/h. In some cases, the number of vehicles is approximately five times higher, which is what we expect. Only in a few cases, this is not true. For example in configurations 1 and 6, the number of vehicles is much higher. The effects of this can directly be seen in Table 21, there is a relatively (compared to 2000 items/h) high vehicle idle time. We expect that the explanation of the additional vehicles is similar as described in Section 5.1; where it can be either unretrieved product carriers or an unlucky test run (where the throughput can not go up because the workstations is the bottleneck of the system).

On the other hand, the number of vehicles is lower than expected for configuration 2. This

can be explained by rounding of the expected throughput; a number of vehicles is chosen so the throughput is met. One can see that with 2000 throughput, the achieved throughput with a minimum of vehicles is relatively higher (2102 instead of 2000) than the achieved throughput requiring 10.000 items/h (10.072 instead of 10.000). This means that because both cases barely have workstation idle time, in the 10.000 case we can simply choose relatively less vehicles because we can get closer to the minimum throughput.

The number of workstations can be explained using the computed minimum number of workstations. For example, for a small workstation, the minimum is 2.2222 workstations. The throughput is five times higher, and therefore the new minimum is 11.1111 workstations, which is rounded to 12. This would give an expected idle time of $1 - 2.22222 \cdot 5/12 \approx 0.07$, which is extremely similar to the workstation idle time found in Table 16.

# SKUs → Configuration ↓	**10.000**
1	167
2	139
3	120
4	114
5	92
6	120
7	15

Table 18: Number of vehicles used to achieve throughput of 10.000 items per hour.

# SKUs → Configuration ↓	**10.000**
1	12
2	16
3	12
4	16
5	20
6	22
7	24

Table 19: Number of workstations used to achieve throughput of 10.000 items per hour.

# SKUs → Configuration ↓	**10.000**
1	10023
2	10072
3	10086
4	10010
5	9915
6	9931
7	9951

Table 20: Achieved throughput (items/h) while requiring 10.000 items/h.

# SKUs → Configuration ↓	**10.000**
1	0.172
2	0.001
3	0.043
4	0.000
5	0.165
6	0.521
7	0.008

Table 21: Ratio of idle vehicles during the day (requiring 10.000 items/h)

# SKUs → Configuration ↓	**10.000**
1	0.072
2	0.003
3	0.066
4	0.009
5	0.005
6	0.095
7	0.025

Table 22: Ratio of idle workstations during the day (requiring 10.000 items/h)

5.6 Synergy

The synergy of a few cases (either 10.000 or 100.000 SKUs) is shown in Table 23, where we can say with 95% certainty that the mean from the runs deviates less than 0.01 from the true mean of the simulation. We are not interested in finding a good or optimal batching algorithm, but we still want to mention the synergy since it shows the difference in expected vehicle movement compared to single order picking. Furthermore, we like to add that there is a difference in synergy between a configuration using 10.000 and 100.000 SKUs, this is mainly because there are more SKUs to choose from when creating an order. Therefore, while creating a batch, we have a lower probability of finding a certain SKU multiple times. While running the test runs, it occurred a few times that there were less orders in *preBatch* than the horizon, meaning this could give a slight decrease in synergy for the test runs. However, this only happened rarely, and therefore we expect that the impact on the results is neglectable.

# SKUs → Configuration ↓	**10.000**	**100.000**
5	1.88	1.31
6	1.89	1.31
7	1.91	1.31

Table 23: Achieved synergy with the current batch picking algorithm.

5.7 Best order picking strategy

After doing a lot of analyses on the simulation and its results, the final question is; what is the best order picking strategy? We first want to state that it is very hard to draw conclusions from the analysis; we used many (unrealistic) assumptions. We rather view the work on the simulation and its results as groundwork for a more accurate and complex model. A more realistic simulation can be developed and its results can be used to more accurately determine the best order picking strategy for a warehouse, and potentially even decide what type(s) of system(s) to further develop.

Suitability of the simulation model If the expected throughput of a warehouse is low, we see that using few workstations gives a lot of (unwanted) workstation idle time, because of rounding. The current simulation model is more suitable for warehouses that have a higher expected throughput, therefore we recommend that this model and simulation is mainly used for warehouses with a higher throughput. If the number of SKUs increases, the number of vehicles increases as well, however in reality this might give issues with traffic. Therefore, we expect that having up to 100.000 SKUs will be fine, but we recommend to not utilize the (results of the) simulation for even more SKUs, since it will most likely give issues in a practical sense.

Best type of workstation It appears that batch picking in combination with a put wall performs better in terms of vehicles than single order picking with a small or medium workstation, this is most clearly seen in Tables 18 and 22. In Table 18, one can see that the number of vehicles is lower for configurations 5 and 6 than 1-4. However, if we do the same analysis for the number of workstations (Table 19), we can see that the number of workstations is higher for configurations 5-6. In general we aim for as much automation as possible, and if this is the goal, then a small workstation would be preferred. However, this station might be more expensive to build and maintain compared to a put wall, and furthermore this workstation could give unwanted idle time. On the other hand, part of the idle time is not an issue since an operator needs to take small breaks during the day, and will be even less of an issue if the pipeline (what orders are requested to be retrieved, and when can a product carrier go on a conveyor) is modelled more optimally. With these two reasons taken into account, a small workstation is the preferred workstation.

Best type of vehicle With the parameters we chose, a vehicle carrying 5 product carriers performs somewhat better than a vehicle carrying 1 product carrier, this can be seen in Figure 18 where clearly less vehicles have to be deployed to reach the throughput. However, a vehicle carrying 5 product carriers needs more floor space, and will potentially give problems if the speed is low. Furthermore, this vehicle also requires additional software to optimize routing and to decide what product carriers to pick. It is important to realize that this vehicle might be more expensive, and that both types of vehicles need a different type of storage (height), which gives a different cost. It is up to the reader to decide if the price (for both the vehicle and additional resources such as software) is worth the extra performance.
Configuration 7, where we have a vehicle carrying a rack and a put wall, seems optimal in the number of vehicles, but still requires a lot of workstations (more than when using a small workstation), meaning we have less automation. Furthermore, the racks require a larger floor space ($1 \cdot 1m$) and therefore a larger warehouse (in floor space) is required, which can be very costly. Therefore, from the results we expect that this option is only preferred if floor space and man power is not an issue, and if we would aim for a low number of vehicles.

We will now give a compact overview of the conclusions. First, we mentioned that we used many assumptions, and therefore the simulation and its results are viewed as groundwork. The results from the simulation are;

- Aiming for high automation, the small workstation is considered to be the best workstation choice.

- With the used parameters, the vehicle carrying 5 product carriers is considered to have the best performance.

6 Conclusions and recommendations

The goal of this research was to compare different types of non-stationary, autonomous GtP systems, where every retrieval action can only utilize one vehicle. We started by determining what specific types of systems we wanted to compare, and the final configurations were stated in Table 2. To start investigating the configurations, we first created a simple mathematical queuing model. In this model, we computed the mean and variance of the service time. With these numbers, we approximated the mean waiting time; the time an order has to wait until a vehicle starts retrieving the product carriers required to fulfill this order. With this waiting time, we determined the optimal number of vehicles such that the mean waiting time is low, but not too many vehicles were used. However, making this mathematical queuing model more complex turned out to be very challenging, and therefore we continued with a simulation model.

We created a model and simulation to investigate all described configurations. This simulation works differently for single order picking and batch picking. For single order picking, all vehicles are requested to retrieve product carriers, and at the workstation, orders of arriving product carriers are reserved on the spot. Furthermore, only product carriers of reserved orders are moved on the conveyor in front of the workstation. For batch picking, we create a batch (group of tactically chosen orders with overlap in SKUs) for each workstation, and vehicles are requested to retrieve product carriers for all these orders. Before running the main simulation, we do some "test runs", which is a small portion of the day, to find the practical minimum number of vehicles and workstations to achieve the required throughput. With this minimum, we ran the main simulation for all configurations, while also varying some parameters.

The best configuration for a workstation mostly depends on the preferences of the warehouse owners and the true parameters. The results showed that if automation is a high priority, configurations that use a small workstation are preferred, since these configurations have the lowest total number of workstations (and therefore operators). However, if few vehicles are of greater priority; the results showed that the configuration where a vehicle carries a rack requires the least vehicles. However, this configuration requires more manpower and floor space, which can be costly. From the results, we also derived that although a vehicle carrying 5 product carriers needs more floor space, it performs slightly better than a vehicle carrying 1 product carrier. We still keep in mind that this result depends on the chosen parameters, such as the vehicle speed. Furthermore, although one vehicle might perform better (and thus less vehicles are required), this vehicle might be more expensive. Therefore, the reader can decide if the relative price is worth the extra performance.
However, the simulation still used a lot of assumptions, and therefore we viewed the simulation and its results as a groundwork. The results showed some clear issues with some of the assumptions, and showed how this affects the performance. An example of this is the test runs; right now, an unlucky test run gave a higher number of vehicles (than necessary), which resulted in either less workstation idle time and a higher throughput than necessary, or in more vehicle idle time (if the workstation is a bottleneck).

From this research, we have the following recommendations. This research with the simulation is recommended to be used as a groundwork to compare multiple specific types of

systems as mentioned above. It is recommended to not use the results to directly make decisions on which system is the most optimal. However, the results can be used to gain an insight on which factors might be important to model, and how certain assumptions influence the results. Furthermore, almost all parameters are variable in the simulation, meaning the reader can use the simulation to investigate the performance if certain parameters are different. It is possible to create a variable operator time or pickup/dropoff time, however this simply has not been done yet due to a lack of data and time.
Furthermore, it is recommended to remove a lot of assumptions to create a better and more realistic model. The most crucial points to improve have been described in the discussion. With a more realistic model, we believe that this simulation and the future results can be used to gain a far more accurate prediction on the performance, and would therefore be a very helpful tool to decide what type of system is the most optimal for a warehouse, or to decide what type(s) of system(s) to further develop.

6.1 Discussion

As mentioned in the results and conclusion, we view the simulation and its results mainly as groundwork for a more accurate, complex and realistic simulation. This simulation can then be used to compare different types of non-stationary autonomous GtP systems, where only one vehicle can be used per retrieval action. We mentioned many assumptions throughout the report, and we will now give a compact overview of the most important improvements that can be made to create a more realistic model.

Instant time at workstation Right now, waiting and arriving vehicles at the workstation drop off the product carrier instantly, and furthermore, this product carrier can also be instantly reached by an operator. Both these assumptions are recommended to be removed to create a more realistic scenario in front of the workstation.

Optimized pipeline For single order picking, we can optimize which product carriers are requested to be retrieved, and depending on the (expected) location of the vehicle decide what new product carriers are requested. Furthermore, product carriers can be moved on the workstation sooner, if all product carriers for pending orders have been moved on the conveyor already. For batch picking, a batch can be released when a batch is (expected to be) almost done, to prevent vehicle idle time before this batch is released, or workstation idle time right after a new, empty put wall is placed. Clearly, we also want to implement that a vehicle can carry less product carriers than its maximum, which would prevent workstations waiting for each other.

Better test runs Test runs can be improved by simulating a larger section of a day, giving a larger morning list to give enough orders to the batching algorithm, and more accurately decide whether the tests have failed or succeeded.

Choosing workstations Improvements can be made so vehicles retrieve product carriers for a more tactically chosen workstation. In this way, we aim to prevent vehicle idle time because all put walls are full at the same time, or choose more tactically when to retrieve a product carrier that is further away from the workstations.

Routing of vehicles It would be beneficial to optimize the selection of which product carriers a vehicle visiting multiple different locations in storage is going to retrieve. Especially when the size of the workstation increases, the vehicle visiting multiple locations has a large travel distance, which can be optimized if product carries located in close range are retrieved.

Realistic storage It is recommended to implement storage more realistically, specifically that product carriers only have a limited number of items inside. Especially if all items are stored in racks, product carriers will exhaust and can not be used for an unlimited number of orders. Furthermore, a model that takes newly arriving items throughout the day into account is also recommended. Not only is this necessary for modelling exhausting product carriers, but this would also give more insight in the effort, time and manpower it takes to create chaotic storage in a warehouse. Additionally, the discard conveyor could also be taken into account to make the model even more realistic, and update storage locations properly.

Realistic vehicles As mentioned before, it is possible that vehicles are blocking paths and therefore interfere with other vehicles. This traffic should be taken into account, which would also show why adding more and more vehicles is not always a viable solution. Charging of vehicles is also a realistic aspect to take into account. Furthermore, acceleration, deceleration and other vehicle maneuvers should also be taken into account to better predict and model the vehicle travel time.

Batch picking algorithm We used a simple batch picking algorithm since creating a complex algorithm was not in the scope of this research. However, creating better batches will certainly increase simulation performance. This includes batch picking with a larger horizon if a lot of orders are ready in the morning (giving very efficient performance the first part of the day), and choosing orders of a batch based on the location of product carriers, giving opportunities for smaller travel distances because of smarter batch picking.

Realistic parameters Finally, it would be recommended to find more accurate parameters for the model. This means, for example, a variable time for an operator, and possibly some breaks for an operator. Furthermore, the time to pickup/dropoff an item would also be more realistic if it was variable, and for example dependent on how high the vehicle needs to reach.

References

[1] B. Knill, "Automated storage makes a comeback," *Modern Materials Handling [Warehouse Management Edition]*, vol. 60, no. 13, pp. 40–41, 2005.

[2] R. De Koster, T. Le-Duc, and K. J. Roodbergen, "Design and control of warehouse order picking: A literature review," *European journal of operational research*, vol. 182, no. 2, pp. 481–501, 2007.

[3] M. Oitzman, "Toyota industries corporation launches global autonomous vehicle software development company," *The Robot Report*, April 20, 2021.

[4] J. Stilgoe, "Machine learning, social learning and the governance of self-driving cars," *Social studies of science*, vol. 48, no. 1, pp. 25–56, 2018.

[5] Y. Gu, J. C. Goez, M. Guajardo, and S. W. Wallace, "Autonomous vessels: state of the art and potential opportunities in logistics," *International Transactions in Operational Research*, vol. 28, no. 4, pp. 1706–1739, 2021.

[6] B. Van Meldert and L. De Boeck, "Introducing autonomous vehicles in logistics: a review from a broad perspective," *FEB Research Report KBI_1618*, 2016.

[7] A. Rimélé, P. Grangier, M. Gamache, M. Gendreau, and L.-M. Rousseau, "E-commerce warehousing: learning a storage policy," *arXiv preprint arXiv:2101.08828*, 2021.

[8] N. Boysen, K. Stephan, and F. Weidinger, "Manual order consolidation with put walls: the batched order bin sequencing problem," *EURO Journal on Transportation and Logistics*, vol. 8, no. 2, pp. 169–193, 2019.

[9] B. Cals, "The order batching problem: A deep reinforcement learning approach." `https://research.tue.nl/en/studentTheses/the-order-batching-problem`, 2019.

[10] B. Cals, Y. Zhang, R. Dijkman, and C. van Dorst, "Solving the order batching and sequencing problem using deep reinforcement learning," *arXiv preprint arXiv:2006.09507*, 2020.

[11] A. Michelen, "Exotec solutions announced shelf-climbing autonomous robots," *Engineering 360*, 2017.

[12] J.-t. Li and H.-j. Liu, "Design optimization of amazon robotics," *Automation, Control and Intelligent Systems*, vol. 4, no. 2, pp. 48–52, 2016.

[13] S. Hur, Y. H. Lee, S. Y. Lim, and M. H. Lee, "A performance estimation model for as/rs by m/g/1 queuing system," *Computers & Industrial Engineering*, vol. 46, no. 2, pp. 233–241, 2004.

[14] B. Y. Ekren and S. S. Heragu, "Performance comparison of two material handling systems: Avs/rs and cbas/rs," *International Journal of Production Research*, vol. 50, no. 15, pp. 4061–4074, 2012.

[15] M. Küçükyaşar, B. Y. Ekren, and T. Lerher, "Cost and performance comparison for tier-captive and tier-to-tier sbs/rs warehouse configurations," *International transactions in operational research*, vol. 28, no. 4, pp. 1847–1863, 2021.

[16] Y. A. Bozer and F. J. Aldarondo, "A simulation-based comparison of two goods-to-person order picking systems in an online retail setting," *International Journal of Production Research*, vol. 56, no. 11, pp. 3838–3858, 2018.

[17] P. Yang, Z. Zhao, and H. Guo, "Order batch picking optimization under different storage scenarios for e-commerce warehouses," *Transportation Research Part E: Logistics and Transportation Review*, vol. 136, p. 101897, 2020.

[18] 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.

[19] B. Yetkin Ekren, "Graph-based solution for performance evaluation of shuttle-based storage and retrieval system," *International Journal of Production Research*, vol. 55, no. 21, pp. 6516–6526, 2017.

[20] Z. Sui, L. Duan, T. Hou, and T. Zhang, "Modeling and scheduling of tier-to-tier shuttle-based storage and retrieval systems," in *2019 Chinese Control Conference (CCC)*, pp. 2247–2253, IEEE, 2019.

[21] R. Manzini, M. Gamberi, and A. Regattieri, "Design and control of an as/rs," *The International Journal of Advanced Manufacturing Technology*, vol. 28, no. 7-8, pp. 766–774, 2006.

[22] Y. A. Bozer and M. Cho, "Throughput performance of automated storage/retrieval systems under stochastic demand," *IIE Transactions*, vol. 37, no. 4, pp. 367–378, 2005.

[23] D. Roy, A. Krishnamurthy, S. Heragu, and C. Malmborg, "Stochastic models for unit-load operations in warehouse systems with autonomous vehicles," *Annals of Operations Research*, vol. 231, no. 1, pp. 129–155, 2015.

[24] F. Piroird and B. Dale, "The importance of lead time control in the order fulfilment process," *Production Planning & Control*, vol. 9, no. 7, pp. 640–649, 1998.

[25] M. Ten Hompel and T. Schmidt, *Warehouse management*. Springer, 2008.

[26] I. Adan and J. Resing, *Queueing Theory: Ivo Adan and Jacques Resing*. Eindhoven University of Technology. Department of Mathematics and Computing, 2001.

[27] J. Puente, D. d. l. Fuente, P. Priore, and R. Pino, "Abc classification with uncertain data. a fuzzy model vs. a probabilistic model," *Applied Artificial Intelligence*, vol. 16, no. 6, pp. 443–456, 2002.

[28] P. Hokstad, "Approximations for the m/g/m queue," *Operations Research*, vol. 26, no. 3, pp. 510–523, 1978.

[29] T. Lamballais, D. Roy, and M. De Koster, "Estimating performance in a robotic mobile fulfillment system," *European Journal of Operational Research*, vol. 256, no. 3, pp. 976–990, 2017.

[30] "Bin-to person picking." `https://www.geekplus.com/robot/c-robot`.

[31] H. Robotics, "Hai robotics product manual." `https://static1.hairobotics.com/EN-ProductManual2021.pdf`.

[32] Exotec, "Skypod systems." https://www.exotec.com/en/skypod-system/.

[33] Invia, "invia picker: Small mighty autonomous mobile robots." https://www.inviarobotics.com/our-system/invia-picker-robots/.

[34] C. Roser, "The amazon robotics family: Kiva, pegasus, xanthus, and more...." https://www.allaboutlean.com/amazon-robotics-family/.

[35] E. Guizzo, "Kiva systems: Three engineers, hundreds of robots, one warehouse." https://spectrum.ieee.org/robotics/robotics-software/three-engineers-hundreds-of-robots-one-warehouse.

[36] "Goods-to-person picking." https://www.geekplus.com/robot/p-robot.

A Proofs of mathematical model

Lemma A.1. $\mathbb{E}[G_i] = \frac{p_i}{1-g}$

Proof. First we use the law of total probability, starting at $k = 1$ since $G > 0$. We write out the definition of the expectation, and write out the probability mass function of the geometric distribution, which gives $\mathbb{P}(G = k) = (1 - g)g^{k-1}$. For the probability $\mathbb{P}(G_i = j|G = k)$, notice that this represents the binomial distribution with a success (choosing the item i when selecting one item) probability of p_i, with a total of k tries, since we condition the probability on $G = k$. Therefore, we use the probability mass function of the binomial distribution, in this case giving $p_i^j(1 - p_i)^{k-j}\binom{k}{j}$. At $(*)$, notice that this is a known infinite sum that has a closed expression.

$$
\begin{aligned}
\mathbb{E}[G_i] &= \sum_{k=1}^{\infty} \mathbb{E}[G_i|G = k]\mathbb{P}(G = k) \\
&= \sum_{k=1}^{\infty}\sum_{j=0}^{\infty} j\mathbb{P}(G_i = j|G = k)\mathbb{P}(G = k) \\
&= \sum_{k=1}^{\infty}\sum_{j=0}^{k} jp_i^j(1 - p_i)^{k-j}\binom{k}{j}(1 - g)g^{k-1} \\
&= \frac{1 - g}{g}\sum_{k=1}^{\infty} g^k \sum_{j=0}^{k} jp_i^j(1 - p_i)^{k-j}\binom{k}{j} \\
&= \frac{1 - g}{g}p_i\sum_{k=1}^{\infty} g^k k \\
&\overset{(*)}{=} p_i\frac{1 - g}{g}\frac{g}{(1 - g)^2} \\
&= \frac{p_i}{1 - g}
\end{aligned}
$$

□

Lemma A.2. $\mathbb{P}(G_i = 0) = \frac{(1-g)(1-p_i)}{1-g(1-p_i)}$

Proof. Similar to lemma A.1, we start by writing out the probability mass functions of the geometric and binomial distribution. We see a known sum appear and since $0 < g < 1$ and $0 < 1 - p_i < 1$, we know $|g(1 - p_i)| < 1$ and therefore we can write the infinite sum into its

closed expression.

$$\begin{aligned}
\mathbb{P}(G_i = 0) &= \sum_{k=1}^{\infty} \mathbb{P}(G_i = 0 | G = k)\mathbb{P}(G = k) \\
&= \sum_{k=1}^{\infty} (1 - p_i)^k (1 - g) g^{k-1} \\
&= \frac{1 - g}{g} \sum_{k=1}^{\infty} (g(1 - p_i))^k \\
&= \frac{1 - g}{g} (\sum_{k=0}^{\infty} (g(1 - p_i))^k - 1) \\
&= \frac{1 - g}{g} (\frac{1}{1 - g(1 - p_i)} - 1) \\
&= \frac{1 - g}{g} \frac{g(1 - p_i)}{1 - g(1 - p_i)} \\
&= \frac{(1 - g)(1 - p_i)}{1 - g(1 - p_i)}
\end{aligned}$$

□

Corollary A.2.1. *(Result of lemma A.2)*
$\mathbb{P}(G_i > 0) = \frac{p_i}{1 - g(1 - p_i)}$

Proof.

$$\begin{aligned}
\mathbb{P}(G_i > 0) &= 1 - \mathbb{P}(G_i = 0) \\
&= 1 - \frac{(1 - g)(1 - p_i)}{1 - g(1 - p_i)} \\
&= \frac{1 - g(1 - p_i) - (1 - p_i) + g(1 - p_i)}{1 - g(1 - p_i)} \\
&= \frac{1 - (1 - p_i)}{1 - g(1 - p_i)} \\
&= \frac{p_i}{1 - g(1 - p_i)}
\end{aligned}$$

□

B Information and parameters of the simulation

B.0.1 Coding details of the simulation

In this section, we will describe how the pre-computed dictionary for the travel distances is created, and how it can be used.

Pre-computed travel distances Retrieving X slow movers or X fast movers has a different travel distance, because of the occurrence of these product carriers. Therefore we will create a dictionary part per unique combination of movers. So for example, if a vehicle can only carry 1 product carrier, there are three different dictionary parts, but if a vehicle can carry 5 product carriers, there are 21 dictionary parts, where we call each part a *key*. For each key, we write down the occurrence that belongs to this type of mover. So, for example, the key of one dictionary is $(5, 5, 1, 1, 1)$ while another dictionary has key $(5, 3, 3, 1, 1)$. If the vehicle can only carry one product carrier, the keys are (5), (3) and (1). For each key, we decided to have 5000 (carrying capacity one) or 500 (carrying capacity five) instances where we compute the travel distance for each instance. When simulating, if a vehicle is requested to retrieve a medium mover, then we can look at the dictionary with key (3) and randomly draw a travel distance.

Computing total path length To determine the travel distance for one instance, we use the Travelling Salesman Problem (TSP), which states *"Given a list of cities the distances between them, what is the shortest possible route to visit all cities exactly once?"*. A vehicle retrieving only one product carrier is a very simple version of TSP, since the list of "cities" is only one product carrier location, which means there is only one possible path, which is consequently the shortest. If a vehicle is retrieving five product carriers, we have a TSP with a list of five "cities". However, if the occurrence is higher than one, there are multiple instances of TSP. In this case, the solution of the TSP that gives the shortest path is chosen. So say a vehicle is requested to retrieve a medium mover (occurrence 3), and the distances of TSP are x, y, and z, then the distance $min(x, y, z)$ is chosen for this product carrier. Choosing the minimum of different TSP's means we assume that no product carrier exhausts, and there are always enough products in a product carrier to complete the request. This part of the simulation is shown in detail in Algorithm 4.

B.1 Tables used in the simulation

Carrying capacity	**# SKU**	$Xcomparts$	# Small lanes
1	10.000	1	2
1	100.1000	2	12
5	10.000	1	6
5	100.000	4	15
Rack	10.000	1	1
Rack	100.1000	2	8

Table 24: Parameters of the different warehouse layouts

Algorithm 4 Creating dictionary for travel distances

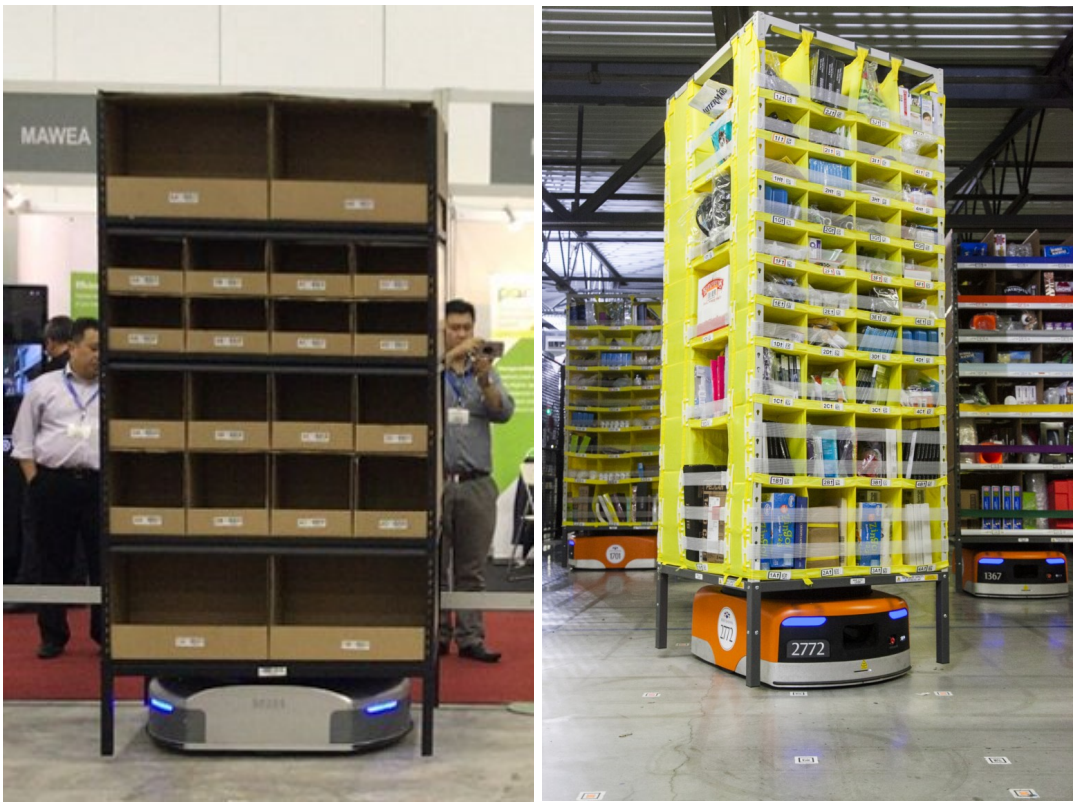
```
1: procedure MAKE DICTIONARY
2:     keys = all unique combinations of occurrences
3:     travel mean is zero
4:     for key in keys do
5:         Create list of distances
6:         for trial in trials do
7:             min length = runTrial()
8:             add min length to list of distances
9:         add list of distances to dictionary of key
10:        travel mean += mean of distances · likelihood of key(*)
11:    Save the total travel mean.
12: procedure RUNTRIAL
13:    for Each index (i) of key do
14:        Create list: locations(i)
15:        Draw |occurrence of key(i)| locations
16:        Add locations to the list
17:    toteCombinations = all unique combinations with 1 value per i from locations(i)
18:    minimum distance = 10.000
19:    for c in toteCombinations do
20:        minimum distance = min(minimum distance, TSP(c))
21:    return the minimum distance
22: procedure TSP(list of locations)
23:    all permutations = all combinations of locations
24:    minimum path = 10.000, current location = (0,0.85)
25:    for p in all permutations do
26:        pathweight = 0
27:        for city in p do
28:            pathweight += distance(current location, city)
29:            current location = city
30:        pathweight += distance(current location, (0,0.85))
31:        minimum path = min(pathweight, minimum path)
32:    return minimum path
```

Option	Value	Source
Throughput	2000 items/hour or 10.000 items/hour	[12] mentions 10K-20K order lines per day.
Number of SKUs	10K or 100K	
Occurrences (not carrying rack)	5 (fast mover) 3 (medium mover) 1 (slow mover)	
Occurrences (carrying rack)	13 (fast mover) 8 (medium mover) 2 (slow mover)	(2)
Pareto curve (1)	Stock: 20/30/50 Chosen: 80/15/5	[27] mentions this Pareto curve, [29] is extremely similar close to the chosen curve.
Vehicle carrying capacity (multiple product carriers)	5	A Geek+ vehicle [30] has 5 carriers. A Hai Robotics vehicle [31] has up to 8 carriers. Meaning 5 is a common choice.
Vehicle carrying capacity (rack)	75	(2)
Vehicle speed (carrying capacity 1)	2.2 m/s	Skypod [32] moves up to 4 m/s. Invia Picker Robots [33] moves up to 2.2 m/s, but appears to be rather slow. To compensate for the acceleration and deceleration we found 2.2 m/s to be reasonable.
Vehicle speed (carrying capacity 5)	1.6 m/s	A Geek+ vehicle [30] has a maximum speed of 1.8 m/s (no load), a Hai Robotics vehicle [31] has a stable (with/without load) speed of 1.8 m/s.
Vehicle speed (carrying a rack)	1.3 m/s	An Amazon Robotics vehicle [29, 34, 35] has a speed of 1.3-1.4 m/s. A Geek+ vehicle [36] has maximum speed of 1.6 m/s (full load) to 2 m/s (no load).
Vehicle pickup/ dropoff time (carrying capacity 1)	36s	From a video, we deduce; 5 seconds to drive up or down a storage rack, 5 seconds to take a product carrier from a storage rack. 3 seconds to dropoff or pickup a carrier from a workstation. All actions are done twice; to bring back a carrier and to retrieve a new product carrier.
Vehicle pickup/ dropoff (carrying capacity 5)	144s	(3)
Vehicle pickup/ dropoff (rack)	12 s	Lamballais et al [29] mention a pod lifting time of 1s, a Geek+ vehicle [36] takes minimally 3s, this action is done twice per pod, and for both a discarded and new pod.
Horizon	5000 orders	
Arrival intensity	6/5	
# items in order	1/(1-0.55)	

Table 25: Parameters used for the simulation

(1) We use the Pareto curve to determine two properties. The first one is abbreviated as "Chosen", which is the probability that an item is chosen in an order. The second one is "Stock", which is the ratio of SKUs that is this type of mover.

(2) In Figure 21a, the layout of a rack that a Geek+ vehicle carries is shown, where a rack contains 20 product carriers. However, when a vehicle carries a rack, we will assume that the product carriers are mixed skew carriers, which means more types of SKUs are located in one product carrier or compartment of a rack. In the case of a rack that an Amazon Robotics vehicle carries (Figure 21b), we also have mixed skew carriers (this can be seen in the figure; there are multiple different products in one yellow compartment). We can do a simplified computation to compute the number of different SKUs. Assume that all product carriers are 50% filled on average, and then assume we have 225 items with 3 items per SKU, then we have a total of 75 different SKUs in a rack. However, we want to keep the total number of items in the warehouse constant. A regular $0.6 \cdot 0.4m$ product carrier is estimated to have 15 items as its maximum, meaning that if we want to keep the number of items in a warehouse constant, we need to increase the occurrence of each item type by (7.5/3). This ultimately led to the assumption of an occurrence of 13, 8 and 2.



(21a) Rack layout in a Geek+ system (21b) Rack in an Amazon Robotics system

Figure 21: Different approaches to a rack layout.

(3) A geek+ vehicle [30] has a minimum lifting time of 12 seconds. This action is done ten times in total; we put five discarded product carriers back in storage, and then retrieve five new product carriers. Furthermore, all product carriers are loaded or unloaded at the

workstation. We assume that we have a specialized docking station on multiple heights, so all product carriers can be (un)loaded simultaneously. Therefore, we assume that this process takes 24 seconds in total.

Option	Value	Source [4]
# pick-to locations small workstation	1	FM-GtP of Vanderlande
# pick-to locations medium workstation	5	[5]
# put wall compartments	50	Put wall size of the Leanpick solution of Vanderlande
Pick time (small workstation)	3s	Look and read screen for information, move hands to product carrier, grab (first) item, lift first item, lift item out of product carrier, place item in order carrier, retrieve hand, hand to button, press button.
Product exchange time (small workstation)	1s	Exchange time per item. The MTM already compensates for actions that can be done simultaneously.
Order exchange time (small workstation)	1s	Exchange at order start.
Pick time (medium workstation)	4.7 s	Same actions as in small workstations, but with additional actions: Step to pick-from location, step to pick-to location, additional time to comprehend screen instructions.
Product exchange time (medium workstation)	1s	Exchange time per item.
Order exchange time (medium workstation)	1s	Exchange at order start.
Pick time (put wall)	5.258s	[6]
Product exchange time (put wall)	3.1s	Product carrier exchange and push away empty product carrier.
Product exchange time (put wall, rack)	5.258s	[7]
Putwall exchange time	40s	Directly given by MTM study.

Table 26: Parameters for the workstation, used for the simulation

[4] All time-based parameters from the workstations are based on a few given MTM (Methods Time Measurement) studies. An MTM is a predetermined motion time system, and this can be used to estimate the duration of an operation or task done by an operator, taking the basic (combination of) motions of this activity into account. Furthermore, MTMs are internationally approved methods to compute the operator capacity, and the calculations are done by the Simulation Department of Vanderlande. For each action, for example "look and read screen for information", there is a given time, for example this action is estimated to take 0.504 seconds. To explain how a certain operator time is determined, we will indicate in the table which tasks have been taken into account.

[5] There are multiple $1:n$ workstations at Vanderlande, where $1:n$ means we pick an

item from 1 location to n potential other locations. Most $1:n$ stations have $4 \leq n \leq 6$; a higher number is suboptimal since the walking distance is large, and for a lower number a $1:1$ station is more likely to be optimal. Therefore, we chose the mean value for n which is 5.

(6) This time was hard to estimate, since it depends on the number of items the operator could grab from the product carrier. If the operator is holding two items (because two items are obtained from one product carrier, which can be used for two orders), the operator can walk from the first compartment to the second, and does not have to walk back to the put wall. This would give a smaller pick time per item, and this heavily depends on the synergy of this instance. Furthermore, the items for single-item orders can be moved on a separate conveyor, which also reduces the picking time (for example, the walking action is now reduced). Although we attempted to reduce the picking time from a given MTM for a put wall, the picking time is a less accurate approximation.

(7) We consider product exchange time in the case where the operator picks items from a rack to a put wall. In this case, the exchange time will be considered higher than picking from a product carrier on a conveyor, since the operator has to find the correct item in the rack, move to this rack, and take the item. We could not find any data for these actions, and therefore we decided to take the picking time for this product exchange time. The thought behind this, is that the two processes (taking item from conveyor, finding put wall compartment, walking to put wall, placing item, versus finding rack compartment, walking to rack, taking item from rack) are very similar in steps, and therefore we decided to assume the times are equal.

C Results of the simulation: varying pickup/dropoff time

Speed multiplier → Configuration ↓	· **0.5**	·**0.75**	·**0.9**	·**1**	·**1.1**	·**1.25**	·**1.5**
1	0.105	0.087	0.078	0.074	0.077	0.070	0.058
2	0.000	0.000	0.000	0.000	0.000	0.000	0.000
3	0.070	0.055	0.048	0.046	0.041	0.039	0.034
4	0.000	0.000	0.000	0.000	0.000	0.000	0.000
5	0.019	0.053	0.009	0.030	0.061	0.061	0.045
6	0.062	0.191	0.228	0.208	0.442	0.237	0.565
7	0.039	0.114	0.175	0.092	0.088	0.083	0.140

Table 27: Ratio of idle vehicles during the day

Speed multiplier → Configuration ↓	· **0.5**	·**0.75**	·**0.9**	·**1**	·**1.1**	·**1.25**	·**1.5**
1	0.246	0.252	0.253	0.254	0.237	0.239	0.257
2	0.190	0.194	0.195	0.168	0.197	0.198	0.199
3	0.207	0.217	0.234	0.224	0.242	0.229	0.233
4	0.196	0.199	0.177	0.200	0.188	0.172	0.177
5	0.012	0.007	0.025	0.014	0.013	0.016	0.023
6	0.065	0.083	0.094	0.101	0.105	0.120	0.120
7	0.023	0.019	0.016	0.021	0.022	0.023	0.021

Table 28: Ratio of idle workstations during the day

Pickup/dropoff multiplier → Configuration ↓	· **0.5**	·**0.75**	·**0.9**	·**1**	·**1.1**	·**1.25**	·**1.5**
1	2035	2021	2018	2013	2060	2054	2005
2	2047	2037	2033	2102	2028	2025	2022
3	2142	2114	2069	2094	2046	2082	2072
4	2031	2024	2078	2020	2052	2091	2080
5	2501	2512	2468	2496	2498	2490	2473
6	2370	2324	2297	2279	2268	2233	2234
7	2128	2136	2143	2331	2130	2127	2132

Table 29: Achieved throughput (items/h) during the day.