Order Picking Method for Multi-entity Cooperation in Picking Warehouses

by

Jingwei Liu

A dissertation submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Auburn, Alabama
August 5, 2023

Keywords: Order Picking, Assistive Robots, Routing Method, Mixed Integer Linear Programming, Ant Colony Optimization, Simulation

Approved by

Jeffrey Smith, Chair, Joe W. Forehand Jr. Professor of Industrial and Systems Engineering
Alice Smith, Joe W. Forehand/Accenture Professor of Industrial and Systems Engineering
Konstantinos Mykoniatis, Assistant Professor of Industrial and Systems Engineering
Gregory Purdy, Assistant Professor of Industrial and Systems Engineering

Abstract

In recent years, different kinds of autonomous robots have been widely studied and applied in warehouses to meet the increasing demand for dealing with more customer orders, providing shorter delivery time, and achieving higher system scalability. As the core operation of the modern warehousing system, order picking is an area in which many researchers are trying to find good strategies to apply robots to improve the performance and efficiency of order picking. This dissertation provides a Multi-entity Cooperative Order Picking (MCOP) strategy in which two types of entities (pickers and transporters) collaborate in a picker-to-parts warehouse to complete order picking. We first define MCOP, and describe the components, the working pattern, and the operating data structure involved in MCOP. An animated, data-generated, and data-driven simulation model is developed to represent a visualized realization of the processes in MCOP and provide a testbed for performance evaluation under different configurable settings. Next, we develop a MILP model to find the optimal operational decisions about workloads and routes for all entities in MCOP. Then, to overcome the inefficiency of using the proposed model to deal with the situation that a large number of items need to be retrieved in a pick wave, we develop an alternative algorithm called Hetero-ACO to find the operational decisions in a short time. A two-stage structure is applied to search the operational decisions for pickers and transporters, respectively. Lastly, we explore different combinations of certain distributions of product picking times and entity traveling speeds to analyze the impact of the variabilities on the makespan of MCOP and the wait time of pickers. The finding of this work can potentially help warehouse owners find the improvement in upgrading manual order pickings with assistive robots and guide the warehouse owners in how to apply robots to assist order pickings.

Acknowledgments

I want to express my gratitude to my academic advisor, Jeffrey Smith, for providing significant guidance, assistance, and financial support throughout my Ph.D. study at Auburn University. I thank him for advising me in academic research and supporting me to build up my knowledge and experience toward a Ph.D. degree.

I want to thank Dr. Alice Smith for her support and advice on my research and this dissertation. I would also like to thank the other committee members, Dr. Konstantinos Mykoniatis and Dr. Gregory Purdy, for their enriching insights and suggestions during my dissertation process. In addition, I would like to express my appreciation to Dr. Levent Yilmaz, for being the University reader.

I want to thank my friends: Yaoxuan Luan, Gaoxiang Li, Jiaxiang Ren, Xuyang Hu, and Dr. Jianzhou Mao, for all kinds of help. Finally, I want to give my deepest gratitude to my family for their love, encouragement, and accompany that supported me throughout my life.

Contents

List of Figures

List of Tables

List of Abbreviations

ABS	Agent-Based Simulation
ACO	Ant Colony Optimization
AGV	Automated Guided Vehicle
AMR	Autonomous Mobile Robot
DES	Discrete Event Simulation
GA	Genetic Algorithm
MCOP	Multi-entity Cooperative Order Picking
MILP	Mixed Integer Linear Programming
PPL	Picker Pick List
PSO	Particle Swarm Optimization
RPL	Required Pick List
SA	Simulated Annealing
SKU	Stock Keeping Unit
TCL	Transporter Carry List
TS	Tabu Search
WODS	Warehouse Operations Data Structure

Chapter 1

Introduction

1.1 Problem Description and Background

In a modern logistics system, warehousing is a required part for most businesses that transport products. In the past, traditional warehouses provided nothing more than storage for in-transit and inventoried goods. However, as the need to save money and boost the system's productivity increases, more complex warehouses that can provide better inventory management, efficient packing/processing, superior customer services, and other additional functions are preferred by modern businesses. Based on the research conducted in U.S. Logistics Industry, the rise of e-commerce sales requires warehouses to process online orders quickly and efficiently [1]. In addition, as a result of COVID-19 pandemic, many e-commerce retailers experienced an unprecedented rise in the number of online shoppers. So, the Intralogistics sector is now facing challenges that how to balance profit with product variability, short delivery time, and the scalability of the system. To overcome those challenges, in this research, we focus on applying automation technologies to assist order picking in a picking warehouse that can accelerate the fulfillment of customer orders.

The target picking warehouse is a picker-to-parts system that involves several types of work tasks such as receiving and dispatching orders, picking and transporting products, and packing and shipping products. In a picking warehouse, order picking involves picking and transporting products from specific storage locations to the depot (s), order consolidation, and order packing in the depot (s) to fulfill customer orders. Traditionally, order picking is a non-value-adding activity. But since this process involves significant money/time costs and eventually affects customer satisfaction levels, the realization of the order picking has substantial

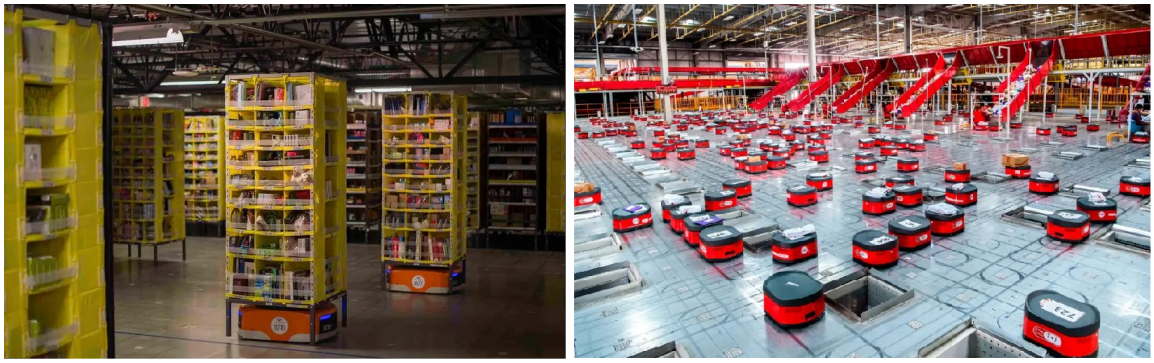
impacts on the warehousing performance. Existing studies indicated that the cost of order picking is estimated to be as much as 55% of the total expense of operating the warehouse [2], and human pickers involved in order picking use about 50% - 70% of their time traveling through the warehouse workspace [3, 4]. Therefore, order picking, as the core of warehousing, has been studied by many researchers to improve its performance and efficiency. What's more, as automation technology advances, especially in the robotic field, many researchers have started to find strategies that incorporate using robots to further improve the performance and efficiency of order pickings.

Automated guided vehicles (AGVs) are mobile robots that follow predefined paths to transport materials and were first brought to market in the 1950s by Barrett Electronics [5]. Figure 1.1 shows the first AGV introduced to the industry and an example of modern AGVs. From the figure, we can find that some modern AGVs don't change much after the AGV is invented. AGVs have become quite capable of performing transporting tasks throughout decades of development. Therefore, large warehouses prefer to use AGVs to accelerate their daily order picking even though installing AGVs needs a large investment. Figure 1.2 shows Amazon's AGV solution and Jingdong's AGV solution [6, 7]. Also, compared to the first AGV, those AGVs can do loading and unloading tasks automatically. However, in those applications, AGVs and human workers do not share their workspace, which means those solutions cannot be used in a picker-to-parts warehouse. From the late 1990s until recently, many researchers have focused on the area of Human-Robot-Interaction, and one major application is the assistive robot system that seeks to provide various support to people such as carrying objects, navigating through spaces, and interacting with people [8]. What's more, due to the vast amount of research on robotics-related topics in recent years, autonomous mobile robots (AMRs) are more cost-effective and flexible than AGVs for supporting human pickers because AMRs can move to any reachable place in a building rather than being constrained to predefined routes and locations. Therefore, some companies have already started to use AMRs to assist human order pickers in their warehouses. Figure 1.3 shows how Crocs™ uses AMRs to support order pickers for carrying products and navigation [9].

(a) (b)

Figure 1.1: (a) The first AGVs were introduced to the industry in the 1950s, by Barrett Electronics of Northbrook, Illinois [5] and (b) An example of modern AGVs

Currently, few existing works in the literature focus on order picking with assistive robots. One important research artifact that we use focused on minimizing the makespan of order pickings for a single picker and multiple assistive AGVs [10]. They extended the algorithm proposed in [11] and integrated a dynamic programming approach to find the optimal route for the picker. In [12], they focused on multiple picking and transporting robot cooperation to complete order picking. They used a Mixed Integer Linear Programming (MILP) model to create interdependent schedules for all picking and transporting robots to minimize the total makespan of the order picking. Another research focused on minimizing the makespan of AMR-assisted order picking in a two-block warehouse [13]. In their work, they use some predefined handover locations where the pickers place the picked items on AMRs. In Chapter 2, we will discuss these works in detail.



(a) (b)

Figure 1.2: AGV solutions in warehouses: (a) Amazon's Kiva system for a fulfillment center in Tracy, California [6], and (b) Jingdong's unmanned logistic warehouse [7]

Figure 1.3: Crocs™ uses AMR solution in its warehouse to assist order pickers [9]

Based on the research conducted in [1], warehouse employment in the U.S. experienced 37% growth between 2014 and 2017. They also predicted that employment would continue to grow by 21% until 2026 due to the rise of e-commerce sales. So inspired by the increasing need for more efficient order picking and the current state of the literature, in this dissertation, we propose a method for performing multi-entity (human pickers and AMRs) cooperative order picking (MCOP) in picker-to-parts systems. This MCOP involves picker entities that pick and place products from storage shelves to transporters and transporter entities that take and transport products to a depot where further consolidation and/or packing operations take place. While these are important warehouse-related tasks, we do not consider those operations performed in the depot. The proposed method aims to provide good workloads (the partitions of the products needed to be picked) and routes for all entities that minimize the total makespan. Specifically, this work investigates the following open questions:

1. The conceptual design questions focus on identifying characteristics of MCOP such as components involved, behavior logic of components, the data structure used, etc.

2. The operational control questions focus on determining workloads and routes for all pickers and transporters to efficiently complete MCOP in different operational conditions such as insufficient numbers of pickers, insufficient numbers of AMRs, etc.

3. The tactical questions focus on the performance of operational control methods in different tactical settings such as different numbers of entities, different numbers of items needed to be retrieved, etc.

4. The strategic questions focus on the performance of operational control methods in different warehouse configurations such as different sizes, different layouts, etc.

5. How does variability in picking, placing, and travel times impact the MCOP operations?

1.2 Research Methodology and Contributions

To answer the questions listed above, we develop different mathematical and simulation models and conduct experiments using these models. Our work aims to provide a set of approaches from the conceptual design phase to the practical operation phase to systematically examine the overall performance of MCOP in different tactical and strategy settings and under stochastic and changing conditions. The methodologies and contribution of this work are concluded as follows:

1.2.1 Definition of MCOP and The Design of A Corresponding Simulation Model

In Chapter 3, a general analytical approach is established to support the discussion about the conceptual design phase in MCOP. We first give a detailed definition of MCOP. This includes the components involved in MCOP and the behavior logic of each entity in MCOP. Then, we discussed the operating data structure used to complete MCOP. The data structure is split into four groups to represent the data needed in different aspects of MCOP such as the information on warehouse layout, storage, customer orders, and entities. Finally, an animated, data-generated, and data-driven simulation model based on Simio is built to represent the whole operation of MCOP. From the design perspective, the simulation model is the digital twin of MCOP and provides a visualized realization of the processes in MCOP. Also, in the subsequent chapters, the simulation model will act as a cross-validation tool of the mathematical models and a testbed that enables the detailed analysis and performance evaluation under certain scenarios involving various product picking times and entity traveling speeds.

1.2.2 Exact Method for Operational Decisions

In Chapter 4, we use MILP to build a mathematical model for the pick wave of MCOP to find optimal decisions about workloads and routes of all pickers and transporters from the operational perspective. In our study, we treat the pick wave of MCOP as a joint order batching and picker routing problem. To make our method effectively handle MCOP in different operational conditions (insufficient number of pickers or AMRs), inspired by the work in [12], we extend their model that picking and placing an item can occur in different places. In addition, the model can decide the times of depot visits for each transporter rather than using predefined values. However, to reduce the complexity, we relax our model so that it will no longer take the battery usage of AMRs into consideration. In this chapter, the needed parameters, decision variables, and constraints are abstracted, formulated, and augmented step by step along with the assumptions.

We also conduct experiments to compare the performance of our model with the one in [12] under different settings such as the different numbers of pickers, different numbers of transporters, different numbers of items needed to be retrieved, different transporter capacities, and different warehouse layouts. One important finding from our work is that when there are fewer transporters with lower capacities, picking and placing items in different places can lead to a shorter makespan for the pick wave of MCOP. And we also find that for a larger number of retrieved items, using the MILP model is not efficient which may make the model less appealing for real warehouse owners.

1.2.3 Fast Method for Operational Decisions

In Chapter 5, we develop an alternative algorithm called Hetero-ACO to find operational decisions fast for MCOP based on Ant Colony Optimization (ACO) [14]. Since pickers and transporters can cooperate with multiple partners (transporters and pickers respectively) in MCOP, infeasible decisions could lead MCOP to a deadlock where all pickers are waiting for transporters and all transporters are waiting for pickers. So, to avoid generating those infeasible decisions, Hetero-ACO will first generate the workloads and routes for pickers and then uses

an ACO-based algorithm to find the workloads and routes for transporters that minimize the makespan. Simulated Annealing (SA) and Tabu Search (TS) [15] are also tested in alternating workloads and routes for pickers to check which method can improve the performance of our algorithm. In addition, we develop a much simpler MILP model that only involves pickers. We extract the intermediate solutions containing the workloads and routes for pickers from solving this simpler MILP model and use those solutions to accelerate the convergence of our proposed Hetero-ACO algorithm. Evaluation is performed by comparing our algorithm with a heuristic method. By this comparison, we find that when the number of transporters is less than the number of pickers, our proposed algorithm can provide a better makespan than the heuristic method. What's more, if the needed items are spread widely in the warehouse, our algorithm usually performs better.

1.2.4 Analysis on The Impact of Variabilities in MCOP

In Chapter 6, we conduct experiments to check the impact of picking, placing, and travel time variability on our proposed methods for MCOP. By exploring different combinations of distributions of product picking times and entity traveling speeds in our simulation model, we provide a brief analysis of the influence on the makespan of MCOP and the wait time of pickers. Based on our experiment results, the variabilities will negatively impact the decisions from our proposed methods when the ratio of transporter number to picker number is smaller than 2. that is, the real makespan becomes more likely to be greater than the baseline expected value. This finding shows sophisticated pickers are important to our method if we want to reduce the negative impact of the variabilities.

1.3 Summary

Due to the variety, complexity, and timing requirements brought by the rising e-commerce, coherent and flexible decisions for operating the order picking in a picker-to-parts warehouse are needed in modern warehousing systems. As the more cost-effective mobile robots, assistive AMRs show great potential in improving the warehousing efficiency by taking advantage of their fast movement.

To better use AMRs and to improve the overall performance of warehousing, a multi-entity order picking method is proposed. Also, by exploring research approaches involving simulation modeling, mathematical programming and modeling, statistical and stochastical analysis, etc., we expect our work to deliver useful insights to warehouse owners who are seeking to use robotic technology to improve warehousing systems and contribute to the current body of literature as well.

Chapter 2

Literature Review

2.1 Introduction

Over the last decades, a vast body of research has focused on improving order picking processes and facility designs. Due to the complexity of order picking, many aspects of a warehouse system can affect its performance. Recent reviews [2, 16, 17] discuss the aspects of the warehouse system that can impact order picking, the current research state, and future research directions. Although every aspect discussed in those reviews is important for improving order picking, in this work, we will only focus on several aspects and just give a brief introduction to those we don't focus on.

The warehouse layout design is a factor that can affect order picking. In general, it concerns two problems: one is called the *facility layout problem* which focuses on deciding where to locate various departments (storage, picking, packing, shipping, etc.). The other one is called the *internal layout problem* focuses on determining the storage blocks and aisles in each block. A lot of research has been done in these general areas [18, 19, 20, 21, 22].

The warehouse *storage assignment* is another factor. It generally focuses on the placement of products (items) before they can be retrieved to fulfill order picking. Currently, even there are many existing storage policies such as random storage, dedicated storage, closest open storage, class-based storage, and turnover-based storage, many researchers are still trying to find more efficient and effective storage policies to better support order picking [23, 24, 25, 26].

The remainder of this chapter presents an analysis of the gaps and strengths of the research on four main areas, namely batching strategy, routing method, robotic applications in order picking, and simulation modeling in order picking.

2.2 Batching Strategy

Order batching is the process of grouping customer orders into batches and releasing them for order picking. That is, for a single picker, before backing to the depot and unloading the orders, the picker can pick multiple orders during the picking tour. Usually, the main objective of the batching strategy is to reduce the traveling time or traveling distance of order pickers.

In many studies, researchers tried using heuristics to solve order batching problems. Seed algorithms form a big group of those heuristics. They usually consist of two steps: Choosing the seed (or initial) order and then adding other orders to the seed order until the order picker is filled to capacity. For the variety of seed algorithms, they use different seed selection rules and seed order addition rules to perform those two steps. For example, using a random order as the seed order and then selecting the order with the property that the sum of the distances between every item of the seed and the closest item in the order is minimized [27]. Using the order with the largest number of items as the seed order and selecting the order with the property that the sum of the distances between every item of the order and the closest item in the seed order is minimized [28]. In some methods, the added order is chosen by minimizing the additional aisles visited or checking the difference between the center of gravity (COG) of added orders and the COG of the seed order [29]. Another big group of heuristics is called *savings algorithms* which focus on the time savings that can be obtained by combining two orders. A very basic variant is called C&W(i). In that algorithm, savings of all possible order pairs are calculated. Then, the pair with the highest saving is selected and judged by some rules to be inserted into an existing route or forming a new route. To improve this algorithm, the other two variants C&W(ii) and C&W(iii), add a recalculation feature when one or two orders have been clustered and a limitation of the number of batches, respectively [30]. Elsayed and Unal [31] proposed an algorithm called EQUAL which combines seed algorithms and savings algorithms. De Koster et al. [32] compared the seed algorithms and saving algorithms with different routing methods.

They found that seed algorithms have better performance when the capacity of the pick device is large, and savings algorithms have better performance when the capacity of the pick device is small. J. Zhang et al. [33] proposed an integrated online order batching method called IOOPDS which can deal with dynamically arriving orders. Their algorithm assigns orders into picking batches to minimize the makespan and maximize the number of completed orders before departure (depart from the warehouse for customers). Their work focuses on a typical single-block warehouse which is shown in Figure 2.1. Also, they used the S-shape tour as the routing strategy. At the start of the IOOPDS method, it uses a MILP model to solve an offline version of the order batching problem. Then, it applies rule-based solutions to deal with future orders. One rule applies seed algorithms in [34] and C&W(ii) based on order similarity. For two orders, the similarity is calculated by taking the ratio of the numbers of identical picking aisles between two orders and the total number of picking aisles that the picker needs to visit if the two orders are combined. Another rule modifies the seed algorithms and C&W(ii) and takes the completion time of the orders and departure time into consideration.

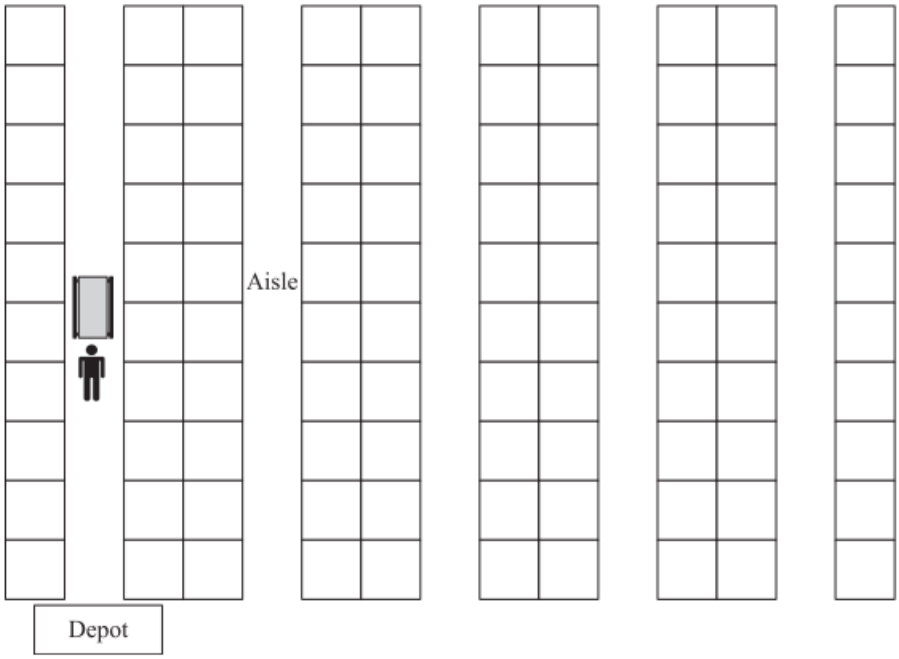


Figure 2.1: The depicted warehouse layout configuration in [33]

Hsieh and Huang [35] introduced two new batching heuristics based on data-mining concepts to minimize the traveling distance of pickers. One method is called Self-organization

Map Batching (SOMB), in which orders act as the artificial neurons, and each order has three factors: the number of SKUs in the order; the number of aisles covered in the order; the number of the same aisles covered between two orders (must be redefined before use). Based on these three factors, orders can be marked as a three-dimensional (3-D) node grid. Then, to reduce the computation complexity, the SOMB method uses Order relativity, a linear combination of the number of SKUs in an order and the number of the same aisles covered between two orders to project the 3-D grid to a 2-D grid. Finally, the order containing maximum SKUs becomes the core order to form a new batch, and other orders will be added to the batch based on the node distance in the 2-D grid. When a batch reaches its capacity, a new batch will start to form. Another method is called K-means Batching (KMB), in which the order with the minimum traveling distance is the core of a new batch. Then, an order that causes a minimum increment in total traveling distance will be added to this batch. When a batch reaches its capacity, a new batch will start to form. In their work, they also showed that order batching is the critical strategy under small and medium order numbers, while storage assignment is the critical strategy under large order numbers.

Other than heuristics, metaheuristics are also widely studied for order batching problems. One methodology was presented in [36] to study the joint order batching and scheduling optimization problem. The method is based on a revised similarity index used to form the batch and then combined the genetic algorithm (GA) [37] with a two-echelon encoding structure to determine the sequence of the orders. They also adopted several initialization methods to increase the overall efficiency of the GA such as methods where the picking line with the earliest completion time has the priority to be scheduled, or the batch with a larger amount of items is more likely to be assigned to high-efficiency picking lines. Henn and Wäscher [38] proposed order batching methods using two metaheuristics: Tabu search and attribute-based hill climber (ABHC) to reduce the travel distance in a picker-to-parts warehouse. For their two metaheuristics, the configurations are represented by three arguments: the initial solution, the neighborhood structure, and the neighborhood selection strategy. They provided two initial solution methods: one is generated entirely randomly, and one is generated by a recalculation time-saving algorithm. They also provided three neighborhood structures: one is called *swap*,

in which the set of solutions can be obtained from interchanging two customer orders from different batches, another is called *shift*, in which the set of solutions can be generated by assigning one order to a different batch, and the remaining one is a combination of the previous two structures. Their selection strategy tried two options: exploring all neighborhoods or exploring part of the neighborhood. They also provided class-based demand and uniform-based demand experiments, which reveal that these two methods have similar performances. Moreover, based on their experiments, they found that if they use the combination neighborhood structure, the impact of the initial solution is negligible.

2.3 Routing Method

According to the research in [3], traveling through the warehouse contributes about 50% of the picker's total order picking time. So, reducing the traveling time is usually the first candidate for improving the order-picking process. The routing problem in warehouse order picking is a special case of the well-known Travelling Salesman Problem (TSP). After order pickers receive a picklist in a warehouse, they start at a location (usually a depot), then visit all item locations in the picklist and finally return to the starting location. There are different methods developed in recent decades to handle this routing problem, and we will discuss several works focusing on the routing problems.

2.3.1 Static Routing Method

One common assumption in many research is that after the routes for pickers are determined, the routes should not be adjusted during the order picking. So, in static routing methods, if new orders arrive while the pickers follow the existing routes, the new orders can only be included in new routes. For solving static routing problems, many researchers focus on using heuristic methods or metaheuristic methods or both.

Heuristic routing methods use efficient heuristics to find an acceptable solution quickly instead of the optimal one. For rectangular warehouses that have parallel picking aisles and crossover aisles only at the top and bottom shown in Figure 2.2, Ratliff and Rosenthal [11] developed an algorithm called *Optimal* to construct a minimum length tour for a single order

picker to pick all items in a pick list based on some pre-defined route segments they called partial tour subgraph (PTS). In their work, they converted the warehouse into a graph in which vertices represent the depot, the requested items, and the ends of each picking aisle. Also, each adjacent pair of vertices is connected by an unlimited number of parallel arcs and the pre-defined PTSs are some possible combinations of arcs. By using dynamic programming, the algorithm can construct the tour with minimum traveling distance. Their algorithm provides a very efficient procedure to solve the routing problem for the desired warehouse layout configuration. However, if there are more crossover aisles in the warehouse, the number of the pre-defined possible arc configuration classes could increase rapidly. For rectangular warehouses that have multiple cross aisles (at the top, bottom, and in between), an "Aisle-by-aisle" routing method to find the minimum traveling distance is investigated in [39]. In this method, order pickers will visit every picking aisle one by one. For every picking aisle, if the indices of the entered and exited cross aisle are given, the method can compute the traveling distance to pick all required items in that picking aisle. Therefore, this policy can recursively find a picking tour with minimum traveling distance by using dynamic programming.

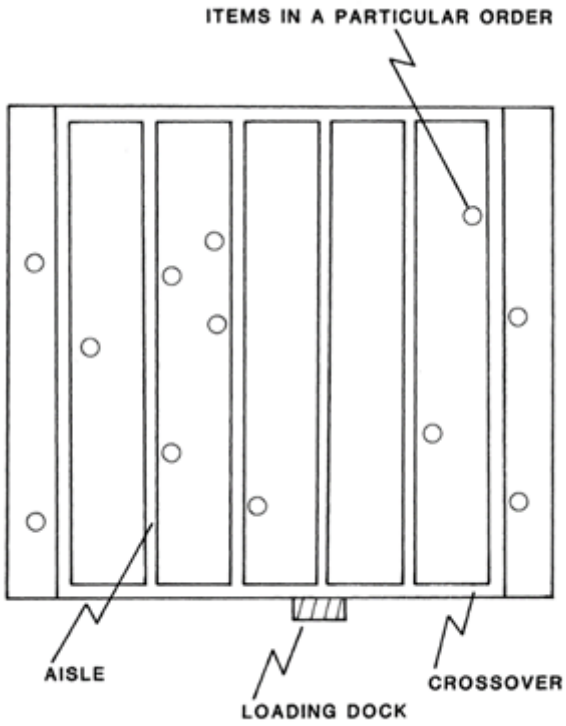


Figure 2.2: The depicted warehouse layout configuration in [11]

Roodbergen and Koster [40] compared four different heuristic routing methods and proposed a combined heuristic that extends the *Optimal* algorithm in [11] to fit warehouses with multiple cross aisles. The four compared routing heuristics are S-shape, Largest Gap, Aisle-by-aisle, and the *Optimal* algorithm. The S-shape heuristic is that the picker will traverse the entire picking aisle when the aisle contains at least one required item. The Largest Gap heuristic tracks the gap (distance) between any two adjacent required items within a picking aisle or between a required item and a cross aisle. So, the Largest Gap heuristic will determine whether it is necessary to traverse the entire picking aisle. The Aisle-by-aisle and the *Optimal* algorithm are the exact work we discuss above. Figure 2.3 provides example routes for these four routing heuristics. For their proposed Combined heuristic, it divides the warehouse into several blocks. Each block consists of two adjacent cross aisles and the picking aisles between the two cross aisles. Then, starting from the farthest block to the nearest block (from the depot) and starting from the left picking aisle to the right picking aisle, the Combined heuristic will construct the route using dynamic programming extended from the *Optimal* algorithm. Heuristic methods are also compared in [41]. The authors analyzed the S-shape, Return, Midpoint, Largest Gap, Combined, and the *Optimal* algorithms in warehouses which are similar to the one in Figure 2.2. Based on the analysis, they found the route distances of those methods heavily depend on the size and shape of the warehouse and the size of the pick list. In addition, a properly selected routing heuristic could result in routes that are only a few percent over the optimal route.

The word *metaheuristic* was first mentioned in [42] to refer to a high-level problem-independent algorithmic framework that provides a set of guidelines or strategies to develop heuristic optimization algorithms [43]. Compared with heuristics, metaheuristics generally perform better and can be considered a higher-level strategy to modify heuristics to generate solutions. Also, unlike heuristics that often end up with local optimal solutions, well-defined metaheuristics can often find the global optimal solution. So, in recent years, many researchers have started focusing on using metaheuristics to solve routing problems in order picking. Tsai et al. [44] proposed a two-layer GA to solve the batch-picking problem in a picker-to-parts warehouse. Their objective is to find a minimum travel cost that combines the total traveling distance and a penalty related to tardiness (or earliness) of fulfilling all orders. They assumed

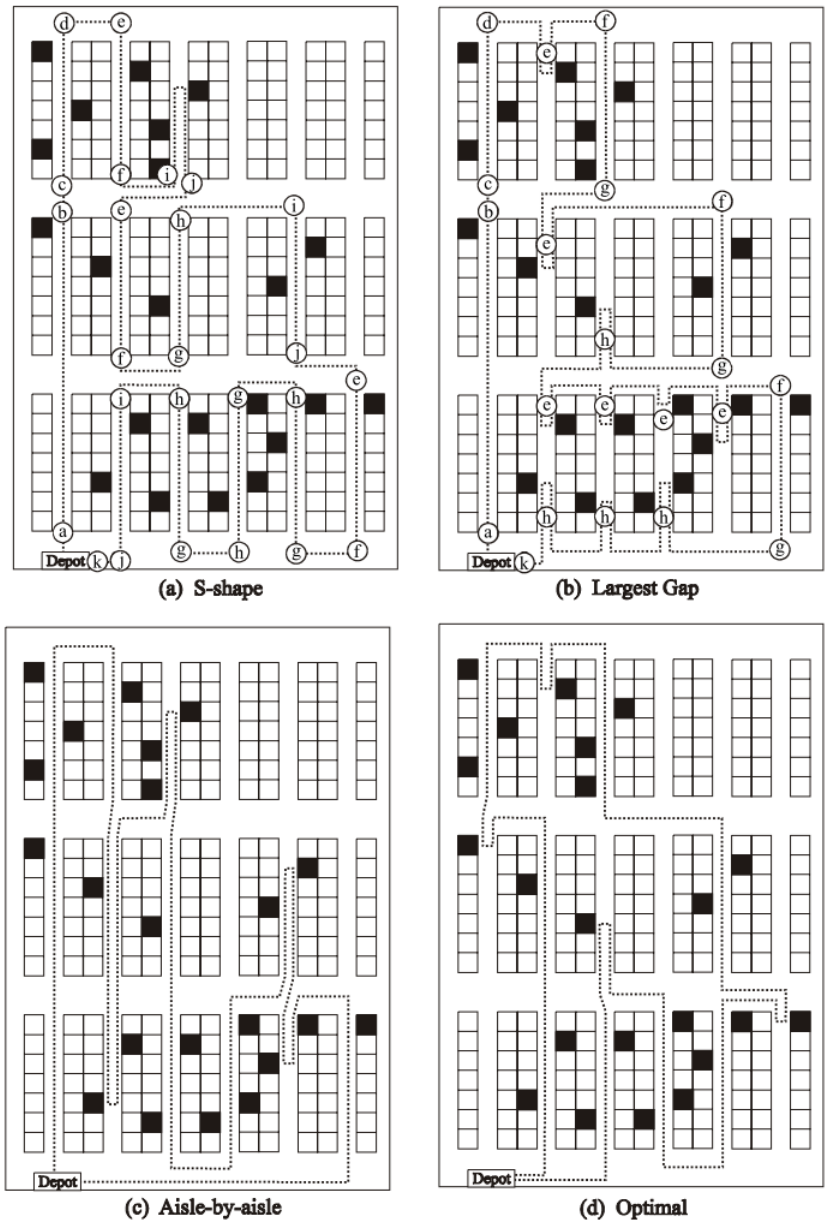


Figure 2.3: Example routes for the four routing heuristics [40]

that each order was dividable so that a batch could contain parts of an order. In their proposed method, the first layer of GA is to solve a batch problem. A population of chromosomes in which the length of each chromosome equals the number of items in all orders. Each gene in the chromosome represents the batch number. After having the chromosomes, the method uses crossover and mutation operations to generate offspring. Figure 2.4 and Figure 2.5 show the example of the chromosome encoding strategy and the updating operations, respectively. The second layer of GA is to solve a TSP. The population initialization, selection, mutation, and

surviving rules are the same as in the first layer of GA. Kulak et al. [45] combined TS and clustering algorithms to solve order batching and picker routing problems jointly. In their work, they first used a MILP to model the problems and provided two modified TS methods to find the solutions. The first TS method constructs the solution by integrating the Nearest Neighbor + Or-opt heuristic [46] and the TS algorithm. The second method constructs the solution by integrating the Savings + 2-opt heuristic [47] and the TS algorithm. They also developed a seed-based clustering algorithm to generate initial batches. Chen et al. [48] introduced a hybrid algorithm to solve joint order batching and picker routing problems. Their work focuses on minimizing the total tardiness of all customer orders in a picker-to-parts warehouse. The hybrid algorithm first applies a hybrid-coded GA to make decisions about batch size and order sequence in a batch. After deciding the batch sizes and order sequences, the proposed algorithm uses ACO to construct the route for each batch. A two-stage algorithm is also proposed in [49] which uses Particle Swarm Optimization (PSO) to first determine the order batches and then use ACO to identify the shortest picking distance for each batch.

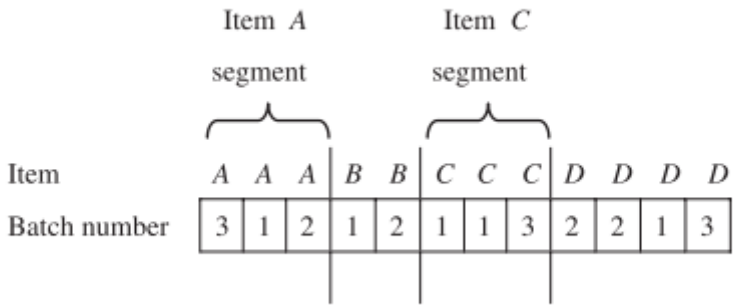


Figure 2.4: An example of the chromosome encoding strategy [44]

2.3.2 Dynamic Routing Methods

Unlike static routing problems, many problems focus on order picking where routes of pickers can be adjusted for newly incoming orders during an order picking. Those problems are usually treated as dynamic vehicle routing problems (DVRPs). As warehouses face the challenge of dealing with more and more orders based on the rise of e-commerce sales, dynamic routing

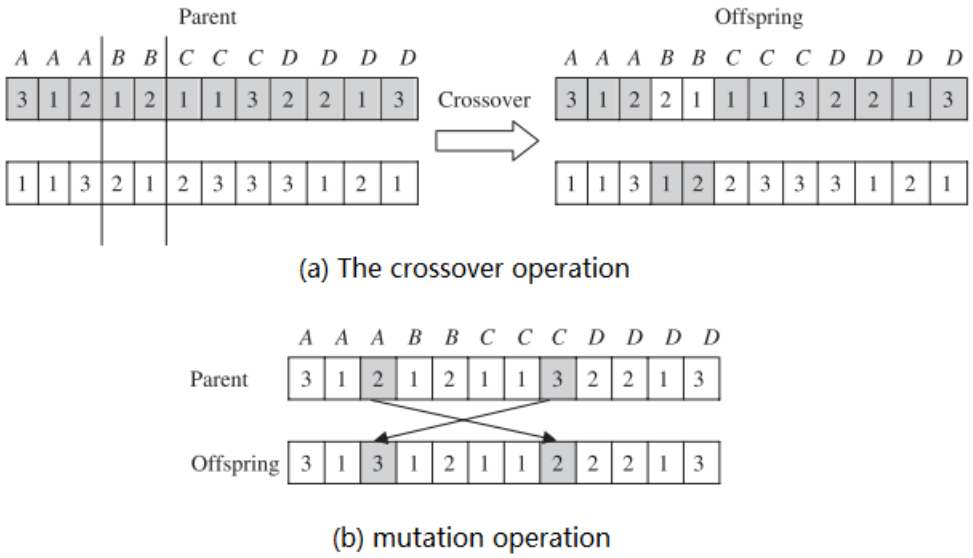


Figure 2.5: Examples of the crossover and mutation operations [44]

methods may provide quicker responses to customer orders. Therefore, studies in this area are increasing in recent years.

Giannikas et al. [50] proposed an Interventionist Routing Algorithm (IRA), which extends the *Optimal* algorithm to deal with new arriving orders during the picking process. Their work allowed the picking operation to be interrupted by newly arrived orders even though the required items are not further downstream in a picker's current route. To achieve the dynamic feature, they first defined two traveling areas: One-way and Round-trip. For a One-way area, the picker can travel through aisles in either a forward or backward direction but not both. For a Round-trip area, the picker can travel through aisles in both directions. Also, they introduced seven new arc configuration classes to build partial tour subgraphs. If new orders arrive, the IRA will construct a new route to deal with new orders. Therefore, the picking operation can be supplemented by any order received. The authors also provided simulation tests based on their proposed IRA in [51]. They compared their IRA with the *Optimal* algorithm and the Largest Gap heuristic. They found that when the rate of new orders is lower than some value, IRA outperforms the other two in terms of completion time. As the rate of new orders increases, the performance of the IRA and the *Optimal* algorithm are close but still better than the Largest Gap heuristic. Xu et al. [52] introduced an Enhanced Ant Colony Optimization (E-ACO) to

solve the multi-vehicle dynamic vehicle routing problem (DVRP). In their work, they used the K-means clustering algorithm to reduce the size of the optimization problem. Then for each cluster, they use their E-ACO to construct the routes. In their E-ACO, they used an ant-weight pheromone updating strategy shown in formula 2.1.

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{Q}{K*L} * \frac{D^k - d_{ij}}{m^k * D^k}, & \text{if edge (i, j) visited by the } k\text{th ant} \\ 0, & \text{otherwise} \end{cases} \tag{2.1}$$

Where $\Delta\tau_{ij}^{k}$ is the pheromone left by ant k. Q is a constant and L is the sum of all routes' lengths, that is $L = \sum_k D^k$. D^k is the route length of kth ant, d_{ij} is the length of the edge (i, j), and m^k is the number of locations visited in the kth route. Also, several local search methods (crossover operation, swap operation, etc.) are included in their E-ACO to enhance the solution. Based on their experiments, the E-ACO outperforms other metaheuristics like TS and GA-based algorithms when dealing with multi-vehicle DVRP.

2.4 Robotic Applications in Order Picking

Azadeh et al. [53] presented an overview of the recent trends in using robotic technology in warehousing to fulfill orders. In their work, they provided analytical models to represent and evaluate the operations in warehouses. As an overview, they discussed several autonomous picking systems in three categories: system analysis, design optimization, and operations planning and control. Also, they mentioned that human picker in collaboration with support AGVs is becoming increasingly popular in practice in order picking operation but has not been adequately studied for scientific investigations.

Löffler et al. [10] extended the *Optimal* algorithm in [11] to provide a routing strategy for an AGV-assisted order-picking system. Their work optimizes the routing of a single picker when he (she) is cooperating with AGVs to minimize the makespan of the order picking. In their research, they assumed that when an AGV is assigned to a human picker before the human picker finishes his picking tour, the AGV cannot assist other human pickers at the same time. When a human picker finishes his picking tour, his assigned AGV will be released and can be

assigned to another human picker if needed. They also conducted experiments and indicated that AGV-per-picker-ratios of about 1.5 seems to be a realistic rule of thumb in their AGV-assisted order picking with multiple pickers. Lee and Murray [12] developed a MILP model to represent two kinds of robots: pickers and transporters that cooperate in order picking. In their model, an item should first be picked by a picker and then placed on a transporter at the same place. Therefore, pickers will only focus on picking items and placing items, and transporters will only focus on holding and transferring the items to the depot. They also assumed that a picker could cooperate with multiple transporters simultaneously and vice versa. In addition, only successfully transferring an item from a picker to a transporter, both the picker and the transporter can move to another place. Their work also considers the usage of the batteries of robots. They evaluated the performance of different combinations of pickers and transporters and different warehouse layout designs. Based on the results, it shows that when there are more picking aisles or fewer cross aisles, using robots in order-picking operations offers the most significant improvement over traditional human order-picking.

Giulia et al. [13] investigated an application of a hybrid picker-to-parts order-picking system, in which human pickers collaborate with AMRs. In their study, the application warehouse is characterized by two blocks, as depicted in Figure 2.6, and the warehouse has some handover locations where pickers put collected items at a handover location, which are visited by an AMR. The AMR then picks and transports the items to the depot. Based on their study, they found that letting picking aisles share the handover locations can lead to more efficient solutions in terms of tardiness when compared with separating handover locations for each picking aisle. They also indicated in this application, increasing AMR fleet size provides better performance than increasing AMR fleet speed, in terms of tardiness. Žulj et al. [54] proposed a two-stage heuristic consisting of adaptive large neighborhood search (ALNS) to solve the AMR-assisted order-picking problem, which focuses on grouping customer orders into batches, assigning batches to AMRs, and finding the sequence of these batches processed by the order pickers and the AMRs such that the total tardiness of all customer orders is minimized. In their work, they find that compared to increasing the AMR fleet size, increasing the speed ratio between AMRs and order pickers can lead to a larger reduction in total tardiness.

Takayoshi [55] proposed an algorithm that generates a sub-optimal cooperation schedule for both AGVs and human pickers. The algorithm combines two heuristic rules to reduce the size of the search space: assigning a picker to an AGV that is closest to the picker or assigning a picker to an AGV who is most behind in the schedule. By using these two heuristic rules, the proposed algorithm is almost linear to the number of AGVs used in the warehouse.

Since human pickers are still the core of robot-assisted order pickings, how applying robots influences the pickers is an important area as well. A recent study suggests, to assess the effect of the introduction of more automation in intralogistics jobs, one needs to consider the specific system design characteristics[56]. Pasparakis et al. [57] investigated how the experience of autonomy affects the pickers' satisfaction with the job, which may affect the turnover intentions. In their work, job satisfaction and self-evaluations including self-esteem and self-efficacy are the metrics for assessing the long-term success of a human-robot collaborative system. In addition, the work evaluates two human-robot collaboration strategies: human leading the robot and human following the robot. Based on their experiments, the work finds that the introduction of collaborative robots increases job satisfaction for both collaboration strategies. But when the pickers are guided by the robots, there is a greater positive effect on job satisfaction. The work also finds that self-esteem and self-efficay are stable in the transition from manual to collaborative order picking.

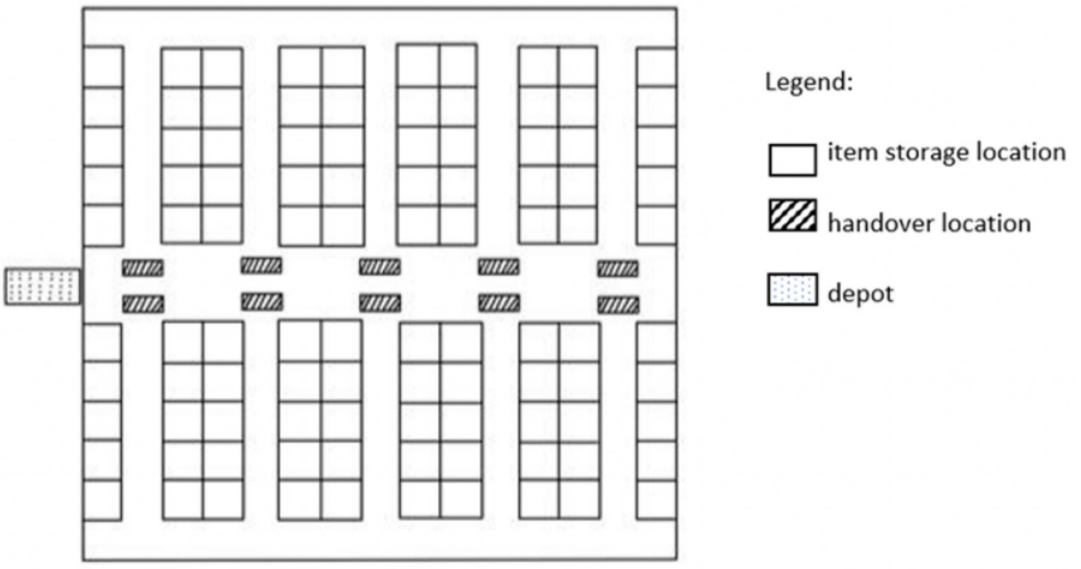


Figure 2.6: The layout of the two-block warehouse [13]

2.5 Simulation Modeling in Order Picking

Compared to analytical models, well-defined simulation models have several advantages: 1) simulation models can handle some complex systems which are difficult to analyze using mathematical methods; 2) simulation models allow the modeling of uncertainty and the generation of probability distributions for model outputs; 3) simulation models can help to generate a deeper understanding of the system being modeled, by allowing the user to see the behavior of the system over time and under different conditions; 4) simulation models allow for experimentation, such as "what-if" analysis, that can be difficult or impossible to perform with analytical models. In this work, we are focusing on the papers that describe both simulation development methodology and the use of simulation to evaluate performance. Currently, there are many simulation modeling methodologies. Multi-paradigm modeling and simulation framework (MPMF) is proposed in [58] to model real-world systems. In this methodology, a complex system is divided into several subsystems in their work, and each subsystem could use one or multiple modeling and simulation (M&S) methods (Discrete Event (DE), System Dynamics (SD), and Agent-Based (AB) methods) to model. Because many subsystems may need to exchange information, this methodology also provides a four-step structure to help identify the proper method to use and how they interact to exchange information in a simulation model. A three-step methodology is discussed in [59]. In this work, to develop a simulation model, domain modeling which describes the problem domain, conceptual modeling which describes the domain in language-independent simulation terms, and simulation modeling are applied.

The ASDI methodology (Analysis, Specification, Design, Implementation) was first introduced in [60] and is used for the design, development, and implementation of a simulation environment, which could be an existing domain (class of system) or a system that still has to be conceived [61]. Figure 2.7 shows the ASDI methodology [60]. Based on ASDI methodology, the final simulation model is obtained by repeatedly constructing the knowledge model and action model. The ASDI methodology recommends the construction of a knowledge model of the system through *Analysis* and *Specification* phases, which allow the modeler to identify the objects in the system and the corresponding behaviors. Then, based on the methodology,

through *Design* and *Implementation* phases, the knowledge model is translated to an action model that provides performance criteria for the system.

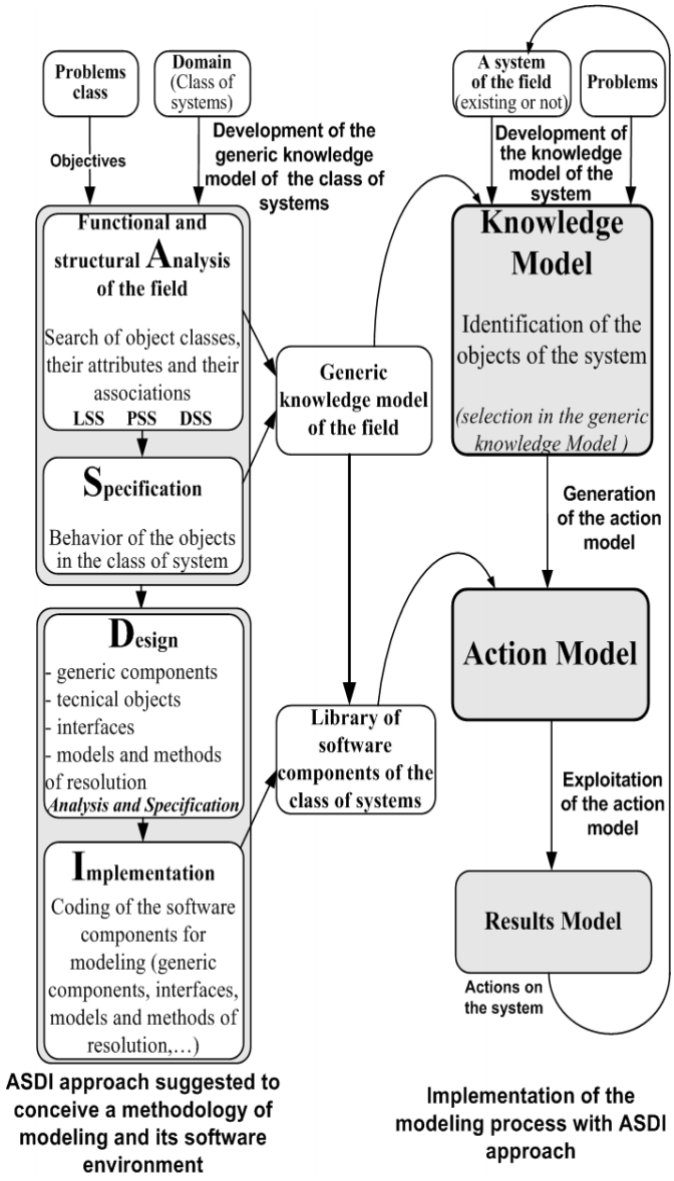


Figure 2.7: The ASDI modeling methodology [60].

Warehouse Operations Data Structure (WODS) was first introduced in [62] as a generalized data structure used in real-world order pickings and simulation areas. The idea is that this data structure can serve as a "starting point" for research work involving order picking. In their work, the data structure consists of six tables: Orders, SKUs, LineItem, Nodes, Arcs, and Slots. The Orders table holds information about customer orders. The SKUs table holds information related to each Stock Keeping Unit (SKU). The LineItem table stores the information related

to each line of orders and the required SKU. The Nodes table and the Arcs table store information related to graph representations of the warehouse layout. The Slots table is related to the storage that connects the representation of the node with SKUs. WODS can be implemented in any database software by having this design, and it is also easily extendable based on the real problem.

A simulation-based comparison of an Autonomous Storage and Retrieval System (AS/RS) and a Kiva system is conducted in [63]. In their work, they examined the performance of the two systems on the basis of expected throughput and expected container retrieval times under different settings such as different numbers of SKUs, different numbers of open pickers, and different warehouse layouts, etc. A multi-agent simulation of the logistics warehouses is proposed in [64]. They designed a self-contained agent architecture that makes an agent can be executed as a single program, and the internal attributes (variables, functions, and methods) are not directly referred by the other programs. They also designed a unified architecture of agent messages to make the development of agents' communication easier. An event-based simulation model of manual order picking is discussed in [65]. In this work, An event graph is developed to represent the event and description of the system logic. The simulation model tested three delivery policies from same-day delivery to 48-hour delivery. Under the three policies, they found that specifying different time windows for different delivery policies outperforms the mixture of different delivery policies in terms of picker utilization and makespan of order picking.

Bahrami et al. [66] carried out an intensive simulation study to examine the performance of order picking under different picking policies in a picker-to-parts system. Their work measures the system based on the total traveled distance, the number of collisions between pickers, and the order lead time. The work also characterizes the system based on four factors: batching rules, routing policies, sorting methods, and storage strategies. According to a full factorial experiment, the work finds congestion has a direct impact on order lead time and a random batching rule performs poorly than other batching rules for collisions and order lead time. Klodawski et al. [67] also focused on using simulation to analyze congestion situations in a picker-to-parts system with very narrow aisles. In their work, the order picking system is

characterized by different storage policies, the number of pickers, and picking strategies. The work explains seven congestion situations that may occur during order picking. Based on their experiments, although increasing the number of pickers can increase the picking efficiency, more pickers lead to congestion and cuts the picking efficiency.

Winkelhaus et al. [68] developed a simulation model to study a hybrid order picking where autonomous robots and human pickers work together within a shared workspace. In their work, autonomous robots are capable of picking items and are considered as a different type of pickers that can pick large and standardized goods. To develop the simulation model, the work applies AB and DE methods to build its three components: agents and behaviors, interactions, and working environment. The work uses eight basic actions to depict the logic of human pickers and autonomous robots. In addition, the work illustrates six blocking configurations and discusses the process of solving the blocking situations. Because the proposed hybrid order picking is rarely used in practice, the work validates the simulation model based on related research and observations.

2.6 Summary

This chapter discusses many research works that focus on different aspects related to order picking. Because of the complexity of order picking, those aspects are always interdependent, and therefore, the recent trend of using robots to assist human pickers in cooperative order picking raises many research questions that are open for discussion.

Many researchers focus on order picking that only involves human pickers. However, fulfilling the cooperative order picking that applies robots is different from the order pickings that only have human pickers. Currently, how to assign robots to assist human pickers, and whether the existing batching strategies and routing methods are still effective in cooperative order picking are still not fully studied. In addition, for the warehouse layout and storage assignment methods, whether those methods still have similar performance in cooperative order picking is needed to be discussed.

Some researchers have already proposed several cooperative order-picking methods in batching strategies and routing methods. However, those works only focus on the static situation that they assume the pickers and robots will follow the exact schedules without any variation. In reality, human pickers and robots may face unforeseen delays (need more time to find the items, need more time to place the items, etc.) when fulfilling the order picking. Existing studies in the literature haven't considered these unforeseen delays when they determine the batches and routes for pickers and robots. So, studies on analyzing the influence of unforeseen delays on cooperative order pickings and designing methods to handle those unforeseen delays are needed.

According to the literature review, there are many works focused on using simulation techniques to model order pickings, but none of the studies consider building a universal simulation model to test methods in different aspects of multi-entity cooperative order picking. Although there are works that provide frameworks to build complex systems, there is no general testbed for cooperative order picking among the existing studies in the literature.

Chapter 3

Definition of MCOP and Design and Development of Simulation Model

3.1 Introduction

This chapter first aims to provide an introduction to the Multi-entity Cooperative Order Picking (MCOP) problem. We will discuss the picker and transporter entities, the warehouse layout, the dispatching rule, the data structure used, and the assumptions we use to study MCOP. Then, A data-driven and data-generated simulation model is built based on ASDI methodology to support the development of both design configuration and operational control of MCOP. Since the operations in MCOP includes entity movements, loading/unloading, as well as the dynamics in order, entity, and Stock Keeping Unit (SKU) information, etc., those operations are simulated based on a hybrid methodology which combines Discrete Event Simulation (DES) and Agent-Based Simulation (ABS) techniques. The simulation model is developed in Simio [69], which is a commercial multimethod simulation modeling tool.

3.2 Problem Description of MCOP

In MCOP, a required pick list (RPL) is the list of all items that must be retrieved from the warehouse storage and delivered to a depot (packing station). The list of items may consist of multiple customer orders, and it is known before the start of order picking and is fixed during the whole operation period. In MCOP, there are two types of entities that perform different tasks: pickers and transporters. Pickers' duties are picking items from picking locations and placing items on transporters, while transporters' duties are accepting items from pickers and transporting them to the depot. In general, the pickers provide the dexterity required for picking

different items from different shelf locations/slots and the transporters provide the transport speed required for efficient operation. At the start of a pick wave in MCOP, each picker will have a picker pick list (PPL) that describes the required items this picker needs to pick in this pick wave. Also, each transporter will have a transporter carry list (TCL) that describes the required items this transporter needs to deliver to the depot in this pick wave. Each item in the RPL should appear and only appear in one PPL and one TCL. Since there could be multiple same items in the RPL belonging to different customer orders, those items could appear in different PPLs and TCLs. Using these definitions, it is clear that combining all PPLs (or TCLs) will be the exact RPL (i.e., the PPLs form a partitioning of the RPL).

In MCOP, the warehouse consists of storage racks, picking and cross aisles, picking locations, hand-off spots, and home and depot locations. Figure 3.1 shows an example of the warehouse used in MCOP. In the figure, each storage rack consists of multiple picking slots, and several picking slots have a corresponding picking location where the pickers pick items. The "Home" represents the rest area for pickers and the charging area for transporters. Sometimes it is possible that the depot also acts as the home location. Also, there are some locations called hand-off spots at which pickers place items on transporters. In this work, pickers follow an individual picking strategy. That is, when a picker finishes picking the needed item(s) at a picking location, the picker should move to a hand-off spot and place the item(s) on a transporter before picking the next item(s). Ideally, the optimal hand-off spot for each item should be determined by incorporating the states of all entities, which means we need to calculate the optimal hand-off spot for each item in the warehouse during the pick wave. However, to reduce the complexity of the problem, we use sets of pre-defined hand-off spots in this work. In addition, in order to let order picking be closer to reality, as depicted in [12], every picking location is a hand-off spot by default. When a picker arrives at the assigned hand-off spot with the picked item(s) ahead of the assigned transporter, the picker must wait at the hand-off spot. Similarly, when the transporter arrives at the hand-off spots earlier, it must wait for the assigned picker. In actual MCOP, instead of waiting at the hand-off spots, it is reasonable for pickers to dynamically alter their positions when they "see" their assigned partner moving toward them. However, in this work, we let both pickers and transporters wait at the hand-off spots before

their assigned partner arrives to simplify MCOP. Therefore, only after finishing the placement of the picked item(s) can the assigned picker and transporter leave the hand-off spot and move to their next place for other activities. And in Chapter 6, we track the waiting time by pickers and transporters and use them as performance metrics to analyze MCOP.

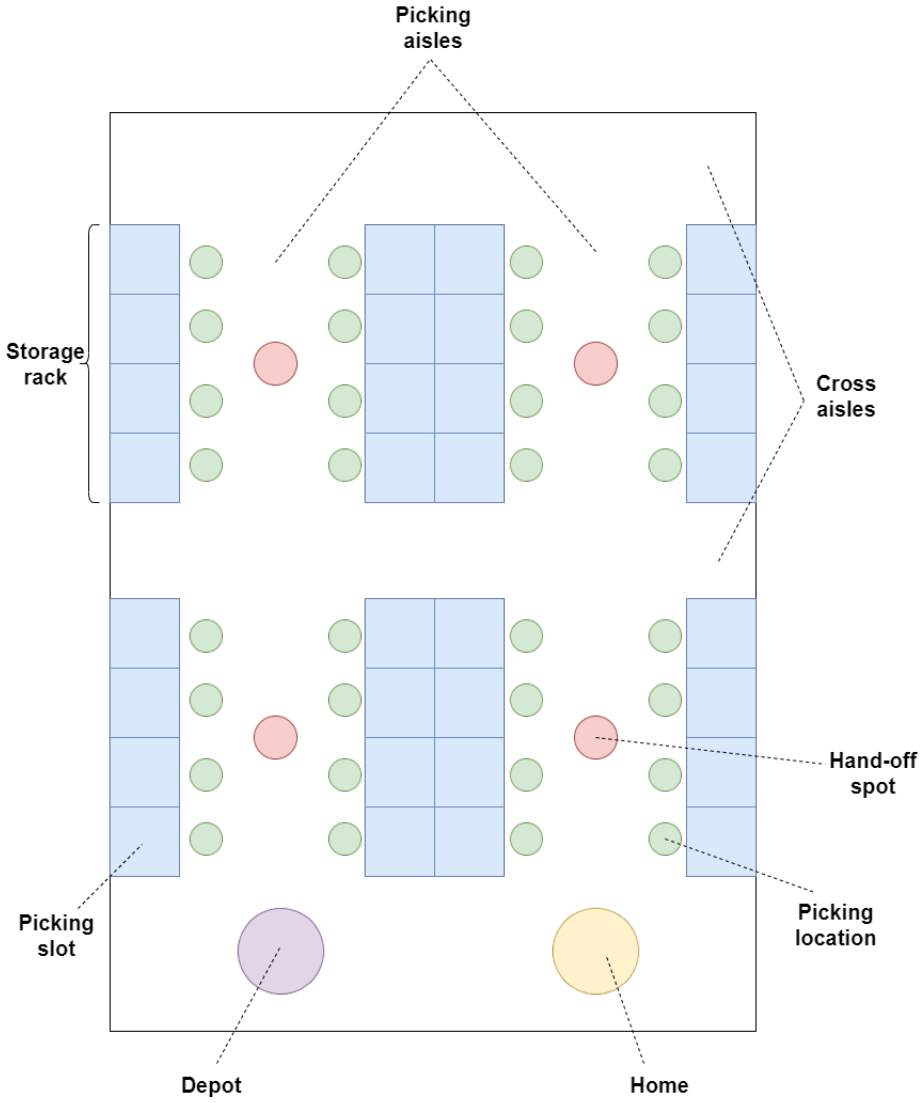


Figure 3.1: An example of the warehouse layout in MCOP

To accomplish MCOP, pickers and transporters will cooperate and pick up all the items in the RPL and transport them to the depot. Here, we make no assumption about the cooperation type. That is, each picker can cooperate with all transporters while each transporter can cooperate with all pickers during the pick wave. All transporters have a limited capacity and may require to drop their carried items at the depot multiple times during the order picking. So, at

the beginning of MCOP, each picker should have its PPL and the corresponding route represents the picking sequence of the items in PPL. While each transporter should have its TCL and the corresponding route represents the carrying sequence of the items in TCL. Then, the goal of this work is to determine the workload (PPLs and THLs) for pickers and transporters and the route for pickers and transporters to finish their workload to accomplish the pick wave of MCOP in a minimum makespan. In the following of this work, we will use the word "schedules" to represent the workloads and routes for all pickers and transporters. In this work, we have the following assumptions to better study MCOP:

1. The depot has an additional consolidation process, and therefore, we can split orders into individual items and let different pickers and transporters pick and transport the associated items.

2. All entities initially start from the home location. When finishing the order picking, all entities will return to the home location.

3. Traveling speeds of all entities, the picking and placing time of all pickers are deterministic and known for modeling MCOP. In Chapter 6, we will also experiment with some distributions for those values.

4. One type of item can only be stored in one picking slot in a storage rack, and one slot can store multiple types of items, as defined in [62].

5. The capacity of transporters could be related to many factors (item volume, item weight, item number, etc.). For simplification, in this work, each transporter has a number-related capacity that each transporter can only carry a limited number of items before back to the depot.

3.3 Operating Data Structure in MCOP

In MCOP, an operating data structure is needed to complete the operation. We develop the operating data structure based on WODS. WODS provides a general way to organize warehouse data and offers the flexibility to add extra information when applying to different applications.

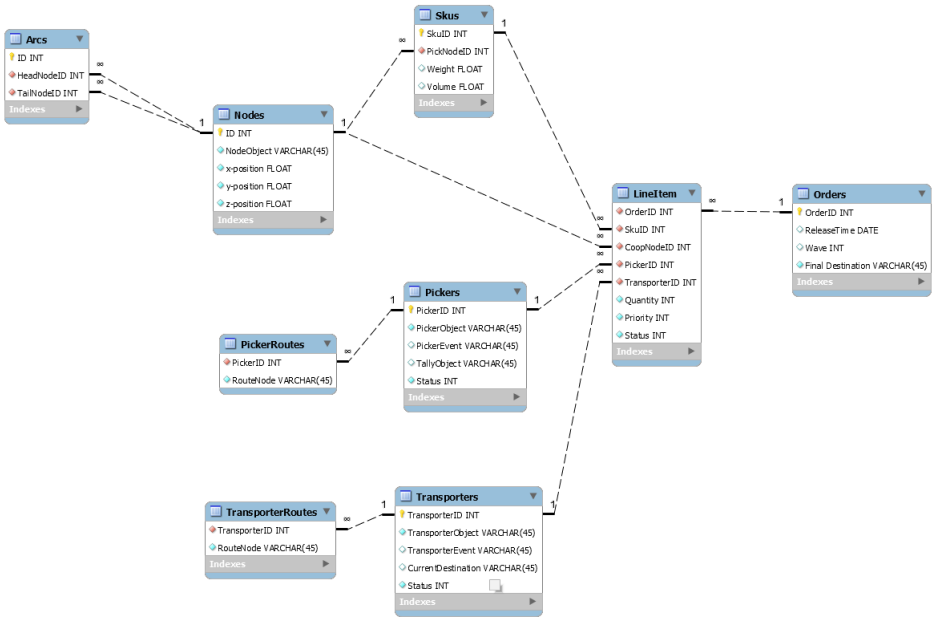


Figure 3.2: Entity-relationship diagram of the operating data structure for MCOP

Figure 3.2 shows the entity-relationship diagram (ERD) of the operating data structure for MCOP. In this chapter, our simulation model will use this data structure to simulate MCOP. The data structure contains multiple tables and can be broken down into four main groups based on the roles in MCOP. We will explain those groups in this section, but we will not discuss them in detail again for those groups already discussed in [62].

For the experiments in Chapters 4 and 5, we use randomly generated storage slots for SKUs and randomly generated customer orders, which are the values in the *Skus* table and *Orders* table from our data structure. However, if there is any "real data", we could easily apply it to our data structure and then use it.

3.3.1 Warehouse Layout Group

This group consists of *Nodes* table and *Arcs* table. Using these tables, we can represent the warehouse layout and the movement graph for pickers and transporters. Figure 3.3 shows an example of how nodes and arcs represent the warehouse [62]. In the figure, the circles are the nodes, and arcs are the paths between adjacent circles. Pickers and transporters may stop at these nodes to perform tasks like picking items or cooperating with others. Also, all circles in

Figure 3.1 are treated as nodes. In the following of this chapter, we will use PickNodes and CoopNodes, which represent the picking locations and hand-off spots, respectively.

Nodes – This table contains node information like node ID, node object type (picking locations, hand-off spots, etc.), and the position represented by their X, Y, and Z values with the corresponding unit under a specific coordinate system.

Arcs – This table contains arc information like arc ID, the head node (starting point), and the tail node (ending point). The head and tail nodes make each arc a unidirectional path. So, to simulate the bidirectional property of the paths in the warehouse, for each pair of nearby nodes, two arcs that have opposite head and tail nodes are applied.

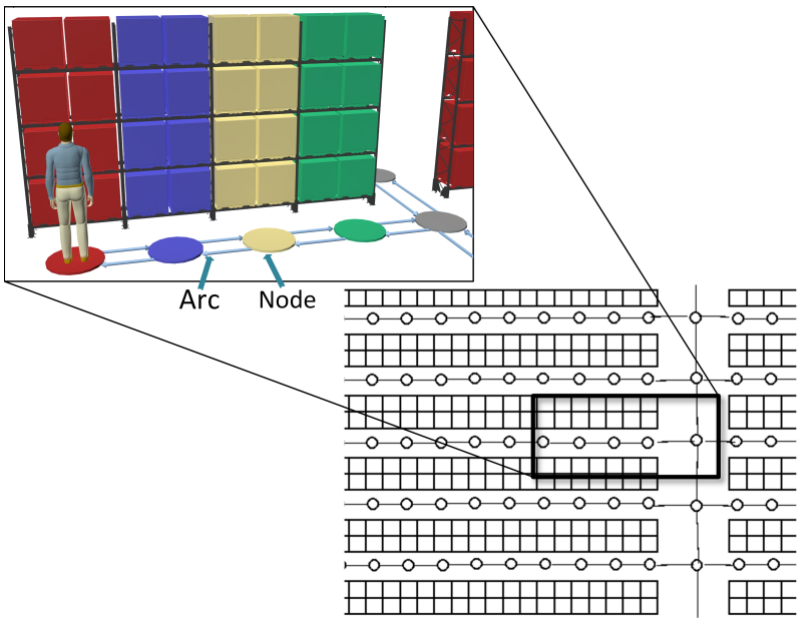


Figure 3.3: An example shows how nodes and arcs represent the warehouse layout [62].

3.3.2 Stock Keeping Unit Group

This group consists of *Skus* table. It holds information about each SKU, which is also the type of items stored in the picking slot. We've already indicated the capacity of the transporters is based on the number of carried items. However, to make the data structure available for more common use, the table *Skus* also contains weight or volume (or both) information of the SKU. So, in Figure 3.2, the *Skus* table includes Sku ID, the PickNode (picking location), the item's weight (optional), and the item's volume (optional).

3.3.3 Customer Order Group

This group consists of *Orders* table and *LineItem* table. To have a clear data structure, we use different tables to store information about customer orders and the associated items. Also, the information about the workloads of entities is stored in this group.

Orders – This table contains order information like order ID, the destination (where to be packed), order release time (optional), and order release wave (optional).

LineItem – This table contains the item and some solution information. The item information includes the SKU ID and the quantity of that SKU. The workload information includes the assigned picker and transporter, priority of the transporter (items with higher priority will be picked first), CoopNode (hand-off spot), and the current status of the items (indicating whether the item is picked or not).

3.3.4 Entity Group

This group consists of *Pickers* table, *PickerRoutes* table, *Transporters* table, and *TransporterRoutes* table. It contains information about all entities and their related routes. In addition, for simulation purposes, this group will also hold some simulation information.

Pickers – This table contains information like picker ID, the corresponding picker simulation components, which provide the features to interact with the simulation system.

PickerRoutes – This table contains information like picker ID and the corresponding node sequence of its route.

Transporters – This table contains information like transporter ID, the corresponding picker simulation components, which provide the features to interact with the simulation system.

TransporterRoutes – This table contains information like transporter ID and the corresponding node sequence of its route.

3.4 Simulation Model Development

A well-defined simulation model can mimic the actual order picking in warehouses. Since there is no real application for MCOP, to develop the simulation model for MCOP, we decide to use the ASDI modeling methodology. Based on the ASDI shown in Figure 2.7, we can develop the simulation model through iterative constructions of the knowledge and action models.

3.4.1 Development of Knowledge Model

Since MCOP is a complex system, to obtain its knowledge model, ASDI recommends a systemic decomposition of the system into three communicating subsystems [70]:

1. The Physical Subsystem (PSS) defines the physical entities set (which concerns different fields, such as the production, storage, handling and transport fields), their geographical distribution and the links between them.

2. the Logical Subsystem (LSS) represents the flows of entities which have to be handled by the system, along with the set of operations concerning these flows, and the nomenclatures which refer to this set.

3. the Decision-making Subsystem (DSS) contains the management and working rules of the system.

To develop the knowledge model of MCOP, we need to identify these three subsystems, which are all objects involved and their behaviors and interactions in MCOP. Based on the analysis of MCOP, the system has nine different types of objects. *Items* is the basic object that represents multiple products of the same SKU needed in customer orders. *Home* is the place where all pickers and transporters should start at the beginning of MCOP and park at the end of MCOP. *Depot* is the place where all transporters drop the carried items. *Picker* and *Transporter* are the entities that cooperate to pick and carry the items from picking slots to the depot. *PickNode, CoopNode, TransferNode*, and *Path* are places that form part of the movement graph of the warehouse. *Picker* can pick *Items* at *PickNode*, place *Items* at *CoopNode*, and

move to any node through *Path*. *Transporter* can get *Items* at *CoopNode* and move to any node through *Path*.

Since *Home, TransferNode, Path* are only used to represent the warehouse layout, those objects don't have additional behaviors. For the other objects, *Depot* should receive the *Items* carried by *Transporters*, *PickNode* should store the corresponding *Items*, *Picker* and *Transporter* are objects with complex behaviors, so their behavior logics are listed below. Table 3.1 summarizes the major objects and their behaviors identified for the knowledge model.

Behavior Logic 1 Procedures for Pickers

1: Based on the route, move to the next *PickNode*.
2: Pick the desired *Items* at the corresponding *PickNode*.
3: Move to the corresponding *CoopNode*.
4: If the assigned *Transporter* does not wait at the *CoopNode*, wait for the assigned *Transporter*. Otherwise, go to step 5.
5: Put the picked *Items* on the assigned *Transporter*.
6: If there are *Items* that still need to be picked at this *PickNode*, go to step 2. If no *Items* need to be picked at this *PickNode*, but there is a next *PickNode* in the route, go to step 1. Otherwise, go to step 7.
7: Move to *Home* and park there.

Behavior Logic 2 Procedures for Transporters

1: Based on the route, move to the next *CoopNode*.
2: Wait at *PickNode* to get the needed *Items*.
3: After obtaining the all *Items* at this *PickNode*, if the next node in the route is *Depot*, go to step 4. Otherwise, go to step 1.
4: Move to *Depot* and unload all carried *Items*. After finishing unloading, if there is a next *CoopNode* in the route, go to step 1. Otherwise, go to step 5.
5: Move to *Home* and park there.

3.4.2 Development of Action Model

In our process of translating the knowledge model into the action model, we apply DE and AB methods to support the realization of the behaviors of objects and the features for performance criteria in the action model. In the action model, *Items* is derived from the default *ModelEntity* object in Simio. To make each *Picker* follows a correct sequence to pick all *Items* in its PPL, instead of creating complicated logic, we add a customized *Source* object for each *Picker* in Simio that can assign the *Items* to *PickNode*. Therefore, instead of creating all *Items* related to

Table 3.1: The summary of objects involved in MCOP.

Object Type	Primary Parameters	Primary Behaviors	Object Description
Items			This object represents the items of customer orders
Home	Position X, Y, Z		The start location of all entities at the beginning of a pick wave
Depot	Position X, Y, Z	Receive Items	The place where the transporters drop the items
PickNode	Position X, Y, Z	Store Items	This object represents a specific item picking location
CoopNode	Position X, Y, Z		This object represents a hand-off spot
TransferNode	Position X, Y, Z		This object is used to form the paths between different nodes
Path	Head and Tail node		This object represents the aisles in the warehouse
Picker	Move speed; Picking time Placing time	Pick Items; Place Items; Move among nodes; Wait at nodes	This object represents a specific picker.
Transporter	Move speed; Capacity;	Carry Items; Move among nodes; Wait at nodes	This object represents a specific transporter.

a PPL at the beginning of the simulated MCOP, the *Source* create and assign *Items* sequentially. That is, when the simulation starts, a *Source* will assign the first *Items* to its *PickNode*. After the *Picker* picks the *Items*, according to the route of that *Picker*, the next *PickNode* and corresponding *Items* are known. Then the *Source* will assign the corresponding *Items* to the next *PickNode*. The *Source* will repeat these procedures until no more *Items* need to be assigned. The *Picker* and *Transporter* objects are derived from the default *Vehicle* object in Simio. We add customized processes to achieve the behavior logics we discussed in Section 3.4.1. The *Home, PickNode, CoopNode*, and *TransferNode* are all node objects in Simio. *Home* is derived from the default *BasicNode* and the other three is derived from the default *TransferNode*. The *Depot* object is derived from the default *Sink* object in Simio that also provides a corresponding node object which can be connected to other node objects. Finally, we use exactly the same *Path* object in Simio as the *Path* object in our simulation model.

As we discussed in Section 3.2, the objective of this work is to find schedules with the minimum makespan. The computation of the schedules is not included in this action model. We will discuss those in Chapter 4 and Chapter 5. So, the action model only stands for the physical operations of MCOP. Since no actual application applies MCOP, the simulation model

provides a visualized realization of the processes in MCOP. Also, the simulation model can act as a cross-validation tool of the mathematical models discussed in the subsequent chapters and a testbed when involving various product picking times and entity traveling speeds. The makespan is the primary performance criterion metric. In addition, we add two other metrics: the average wait time of all pickers and the max wait time of all pickers. These two metrics are helpful for us to examine the other qualities of the schedules when we consider the impact of variabilities in Chapter 6.

After developing the exact objects in Simio and determining the criteria metrics. We could use the data based on the data structure discussed in Section 3.3 to generate action models with different warehouse layouts and different combinations of pickers and transporters. Figure 3.4 shows an example of our final simulation (action) model with 2 pickers, 3 transporters, 24 picking locations, and 12 hand-off spots.

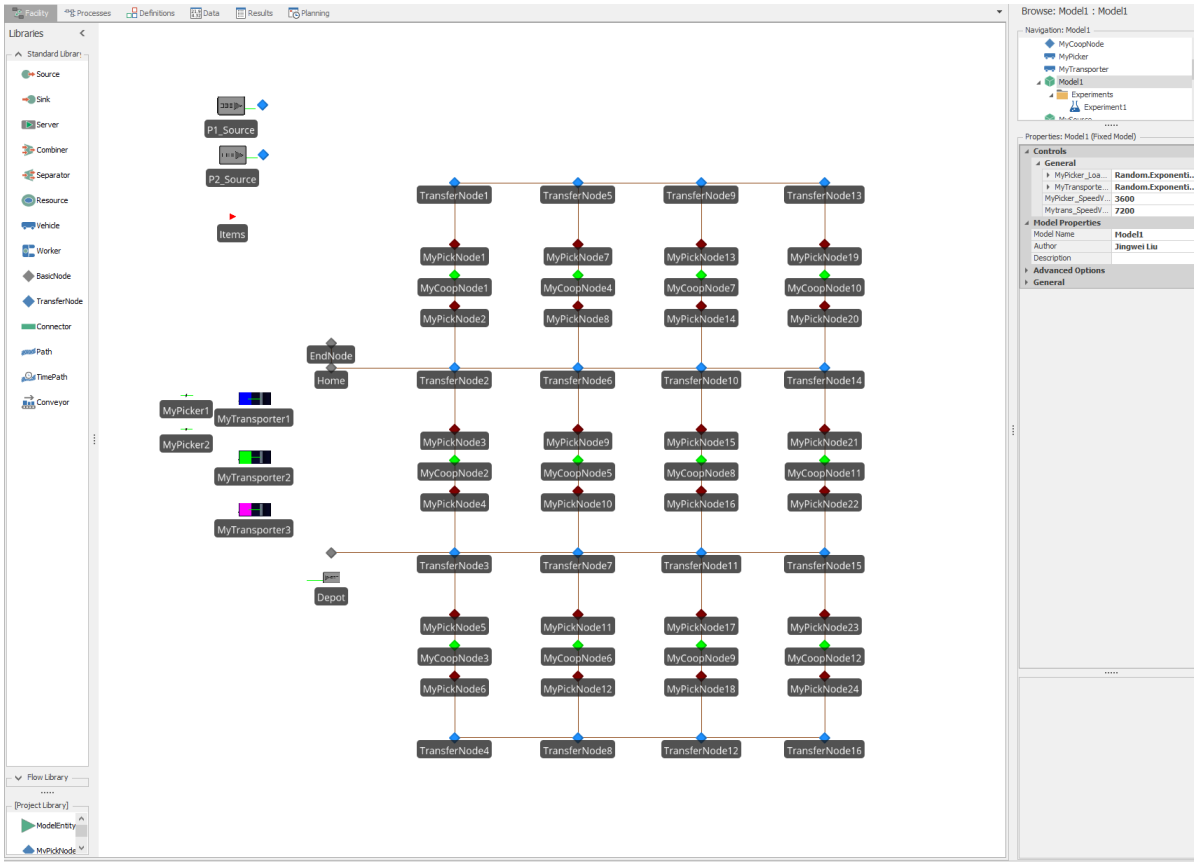


Figure 3.4: An example of the simulation model with 2 pickers, 3 transporters, 24 picking locations, and 12 hand-off spots.

3.4.3 Model Verification and Validation

Verification and validation (V&V) of the simulation model are performed using the aforementioned ASDI modeling methodology. Verifying a simulation model is the process of assessing the design and implementation of the model to ensure that it accurately represents the system that is intended to simulate. The validation of a simulation model is the process of determining the degree to which a simulation model accurately represents the real world from the perspective of the intended uses of the model. In this work, the model verification is performed by developing submodels that represent different aspects of MCOP and examining the submodels through data collection from the models and visual animation. Those submodels help verify the cooperation of entities, the realization of picking PPLs for pickers, the sequence of getting TCLs for transporters, etc.

Since there is no actual application of MCOP, we could not find real data to help the validation process. Therefore, the model validation is performed by analyzing system parameters of similar industrial applications [9] and similar research in the order picking area [10, 12, 13, 72, 73, 74, 75]. Table 3.2 shows the summary of the system parameters based on the analysis. Therefore, our simulation model with parameters of any value combinations within the range could be a valid representation of a future possible MCOP.

3.4.4 Use Case of the Simulation Model

According to a survey conducted in [71], the application of simulation in manufacturing systems can be classified into three classes: 1) for system design that involves long-term decisions including facility layout and system capacity/configuration and the analysis of design alternatives; 2) for system operations involving short-term decisions such as operations planning and scheduling, real-time control, operating policies, and performance analysis; and 3) simulation language/software package development. In this work, the simulation model is primarily used for supporting the first two classes: the system design and the system operation designs of MCOP. Those topics will be discussed in detail in Chapter 6. What's more, although the simulation model is developed based on the commercial software Simio, the modeling approaches

Table 3.2: Summary of system parameters of order pickings.

Model Parameter	Value Range
Required Pick List	5 - 100 items [10, 12, 13]
Picking Slots	450 - 2500 [9, 10, 12, 13, 72, 73, 74, 75]
Number of Pickers	1 - 40 [10, 12, 13]
Number of Transporters	1 - 100 [10, 12, 13]
Number of Picking Aisles	2 - 10 [12, 13, 75]
Number of Cross Aisles	2 - 4 [12, 13, 75]
Picker Speed	0.5 - 1 m/s [10, 12, 13, 72, 73, 74, 75]
Transporter Speed	0.8 - 4 m/s [9, 10, 12, 13]
Width of Picking Aisle	1 - 3 meters [12, 13, 75]
Width of Cross Aisle	1 - 3 meters [12, 13, 75]
Width of Storage Rack	0.2 - 1 meters [12, 13, 75]
Width of Storage Rack	0.2 - 1 meters [12, 13, 75]
Picking Time of Pickers (Per Item)	1 - 5 seconds [10, 12, 13, 74, 75]
Picking Time of Pickers (Per Location)	10 - 80 seconds [72, 73]
Placing Time of Pickers (Per Item)	1 - 4 seconds [10, 12, 13, 74, 75]
Unload Time of Transporter	0 - 90 seconds [72, 73]

and techniques applied are generalized. Therefore, following the model development methods discussed in this section, one can easily re-develop the simulation model in other simulation platforms.

3.5 Summary

This chapter explains the components, the operations, and the development of a simulation model of MCOP. In MCOP, pickers and transporters cooperate with each other to pick and transport all needed items to the depot.

We first discuss the concepts of RPL, PPL, and TCL, as well as the concepts in warehouse layout. The operations of pickers and transporters are also identified. Then, a data structure consisting of four groups - warehouse layout, stock keeping unit, customer order, and entity - is discussed for managing the data during the whole operation of MCOP. Finally, a simulation model is designed and developed based on ASDI modeling methodology. System objects, objects' behaviors and processes are modeled through the construction of the knowledge model and action model. Also, the model is verified and validated based on small models reflecting different aspects of MCOP and analysis of similar order-picking applications/research. The

simulation is expected to support the system design and operation design of MCOP, to be further discussed in Chapter 6.

Chapter 4

Development of MILP Model for Schedules of All Entities

4.1 Introduction

In this chapter, we will develop a mathematical model using MILP for MCOP to find optimal decisions about workloads and routes of all entities that minimize the makespan of the pick wave from the operational perspective. The workload of a picker (transporter) represents all the products in the corresponding PPL (THL). While the route of a picker (transporter) indicates the exact location sequence that the picker (transporter) follows to finish its PPL (THL).

The MILP model is an exact method since it uses mathematical formulae to characterize the pick wave of MCOP. Also, the MILP model is capable of providing the optimal solution as long as its runtime is long enough. As discussed in Chapter 1, our MILP model is inspired by the work in [12], which uses MILP to model two kinds of robots cooperating to complete order picking. We extend their work so that picking and placing items of pickers can occur in different places. In addition, the model can decide the times of depot visited by each transporter rather than using predefined values. But, to reduce the complexity of the model, we relax our model so that the usage of robot batteries is no longer considered. This no-battery usage assumption seems reasonable since the use of backup robots, hot battery swapping, and wireless charging are increasing in popularity for high-volume facilities.

The rest of this chapter is organized as follows. In Section 4.2 we present the needed parameters used for developing the MILP model. Section 4.3 describes the decision variables needed in our MILP model. In Section 4.4, we build and explain the mathematical formulae that characterize the pick wave of MCOP. Section 4.5 discusses the settings of our experiments

and results to compare our MILP model with the model proposed in [12]. In section 4.6, we summarize our findings.

4.2 Parameters and Notation

In the previous chapter, we have already described the pick wave of MCOP. To develop a MILP model to represent the pick wave, we first need to introduce the parameters in the model (see Table 4.1). Since the RPL may contain multiple items of a single SKU (eg. customers order 10 same small toys), it is reasonable for a picker to pick those items together and then bring them to a transporter as long as the picker can carry those items. Therefore, in our mathematical model, we treat those items as a single "item", and in the rest of this chapter, we will use the term *item* to represent multiple items of a single SKU. Then, the RPL is denoted as $I = \{1, ..., |I|\}$ which means there are total $|I|$ items that need to be picked and delivered to the depot. It is necessary to mention here that if there are too many products of a single SKU in the RPL a picker cannot carry at once, it is possible we have multiple items related to that SKU in I. In addition, the determination of items is done by a pre-processing process so that the MILP model will just use that information.

Considering there are two types of available entities in MCOP, P represents the set of pickers, and D represents the set of transporters. The set of all entities can be denoted as $E = P \cup D$. Because transporters have a number-related capacity, for each transporter $d \in D$, its capacity is denoted as C_d, which means the transporter can carry no more than C_d items before dropping its load at the depot. In addition, for each entity $e \in E$, the traveling speed of that entity is denoted as V_e. There are three types of action time that need to be identified. First is the pick time for a picker $p \in P$ to pick an item $i \in I$ from the storage which is denoted as $T_{p,i}^{pick}$. Then, we use $T_{p,i}^{place}$ to represent the placing time that a transporter gets an item $i \in I$ from a picker $p \in P$. Last, we use T_d^{drop} to represent the time needed for a transporter $d \in D$ to drop all loaded items at the depot. We assume the transporters use totes to carry the items. Therefore, T_d^{drop} is independent of the number of items carried by a transporter.

This work uses a node network described in section 3.2 as the warehouse to help represent the movement of all entities in the MILP model. The network consists of four types of nodes:

(1) the home nodes as the home location for all entities, (2) the picking nodes as the picking locations for all items, (3) the depot replicas as the depot, and (4) the hand-off replicas as the hand-off spots. For the home node of an entity $e \in E$, it is denoted as HN_e. The picking node of each item $i \in I$ is denoted as R_i. Here we need to mention that although some items may share the same physical picking location and all entities share the exact home location, in the MILP model, each HN_e and each R_i are unique. In addition, the set of all home nodes is denoted as HN, and the set of all picking nodes is denoted as R. In MCOP, there is only one physical depot. However, since each transporter may visit the depot multiple times during the pick wave, to make sure a node in the MILP model can only be visited no more than once, we use multiple depot replicas to represent the physical depot. So, the set of depot replicas for a transporter $e \in E$ is denoted as DN_e where $DN_{e_1} \cap DN_{e_2} = \emptyset$ for all $e_1 \neq e_2$. Because only transporters visit the depot, when $e \in P$, $DN_e = \emptyset$. The set of all depot replicas is denoted as DN. Since in our work, the exact number of depot replicas used by each transporter is determined based on the solution from our MILP model, before running the model, the model only knows the maximum number of depot replicas can be used by all transporters, which is denoted as DN_c. After an item is picked by a picker, it could be placed on a transporter at different hand-off spots. Also, based on the same reason for the depot, hand-off replicas are used in the MILP model to make sure each replica is visited no more than once. Then, HR_i is used to represent the set of hand-off replicas of item $i \in I$. That is, the set of hand-off replicas HR_i is a copy of the possible hand-off spots for item $i \in I$. The set of all hand-off replicas is denoted as HR. Therefore, in the MILP model, all nodes in the network that the entities could visit in the pick wave are denoted as $N = HN \cup R \cup DN \cup HR$. For any two nodes $k, k1 \in N$, the distance between the two nodes is denoted as $L_{k,k1}$, which is known.

There are two additional notations that characterize the movement of the entities. Given a node $k \in N$, for an entity $e \in E$, we define $N^-_{e,k}$ as the set of nodes that could be visited immediately before node k. For example, if $k \in R$, then $N^-_{e,k} \subset HR \cup HN$ for any given picker $e \in P$. The other notation is $N^+_{e,k}$, which represents the set of nodes that could be visited immediately after node k. For example, if $k \in R$, then $N^+_{e,k} \subset HR$ for any given picker $e \in P$.Table 4.1 summarizes the parameter notations of our MILP model.

Table 4.1: Summary of the parameters and notations

Parameter Notation	Parameter Notation Description
I	The set of all items in the RPL
P	The set of all pickers
D	The set of all transporters
E	The set of entire entities $E = P \cup D$
C_d	The capacity of a transporter $d \in D$
V_e	The traveling speed of an entity $e \in E$
$T_{p,i}^{pick}$	The time needed for a picker $p \in P$ to pick an item $i \in I$
$T_{p,i}^{place}$	The time needed for a transporter to get an item $i \in I$ from a picker $p \in P$
T_d^{drop}	The time needed for a transporter $d \in D$ to drop all carried items at the depot
HN_e	The home node for entity $e \in E$
HN	The set of all home nodes for all entities
R_i	The picking node of item $i \in I$
R	The set of all picking nodes
DN_e	The set of depot replicas for an entity $e \in E$
DN	The set of all depot replicas
DN_c	The total number of depot replicas
HR_i	The set of hand-off replicas of item $i \in I$
HR	The set of all hand-off replicas
N	The set of all nodes
$N_{e,k}^-$	The set of nodes that could be visited immediately before node $k \in N$ for an entity $e \in E$
$N_{e,k}^+$	The set of nodes that could be visited immediately after node $k \in N$ for an entity $e \in E$
$L_{k,k1}$	The distance between two nodes $k, k1 \in N$

4.3 Decision Variables

There is a variety of decision variables encountered in the MCOP (see Table 4.2). First, for each entity, we have a binary decision variable $Y_{e,k,k1} = 1$ if an entity $e \in E$ moves directly from node k to node $k1$ where $k, k1 \in N \; and \; k \neq k1$. For each picker, we also have a binary decision variable $X_{p,i,j} = 1$ if that picker $p \in P$ pick item j just after picking item i where $i \in \{I \cup \{0\}\}, j \in \{I \cup \{-1\}\} \; and \; i \neq j$. To make the relation between $Y_{e,k,k1}$ and $X_{p,i,j}$ easier for every picker, we add two virtual items $0 \; and - 1$ where 0 represents the virtual item located at the home node that each picker needs to pick first. While -1 represents the virtual item located at the home node that each picker needs to pick last.

We have continuous decision variables $t_{e,k} \geq 0$ determine when an entity $e \in E$ arrives at a node $k \in N$ to conduct an activity related to the items. This definition represents when a picker arrives at a picking node or starts placing the picked item at a hand-off replica. While for a transporter, this definition represents when it starts receiving the item at a hand-off replica or begins to drop all items at a depot replica. For coordinating all entities, the binary decision variables $a_{p,d,i}$ represent the pairing between two entities for a particular item so that $a_{p,d,i} = 1$ if a picker $p \in P$ and a transporter $d \in D$ are assigned to pick and deliver the item $i \in I$. For each transporter, integer decision variables $S_{d,k,k1} \geq 0$ represent the payload of that transporter $d \in D$ leaving node $k1$, and have traveled from node k where $k \neq k1$. Therefore, if a transporter $d \in D$ travels from node k to node $k1$, $S_{d,k,k1}$ will include all items loaded after leaving node k plus the item added at node $k1$. Finally, the makespan of the pick wave, which is to be minimized, is defined by the continuous decision variable $m \geq 0$. In this work, it is the time the depot has received all items in the RPL. Table 4.2 summarizes the decision variables of our model.

Table 4.2: Summary of the decision variables

Decision Variable	Decision Variable Description
$a_{p,d,i}$	$a_{p,d,i} = 1$ if a picker $p \in P$ and a transporter $d \in D$ are assigned to pick and deliver the item $i \in I$
m	The makespan of the pick wave
$t_{e,k}$	The time when an entity $e \in E$ at a node $k \in N$ to conduct an activity related to the items
$S_{d,k,k1}$	The total number of items carried by the transporter $d \in D$ after leaving node $k1$, and have traveled from node k
$X_{p,i,j}$	$X_{p,i,j} = 1$ if a picker $p \in P$ pick item j just after picking item i where $i, j \in I \; and \; i \neq j$
$Y_{e,k,k1}$	$y_{e,k,k1} = 1$ if an entity $e \in E$ move from node k to node $k1$ where $ik, k1 \in N \; and \; k \neq k1$

4.4 Mathematical Formulation

After having the parameters and the decision variables, the mathematical formulation of the objective and constraints are shown as follows.

$$\text{Min } m \tag{4.1}$$

$$\text{s.t.} \quad m \geq t_{d,k} + T_d^{drop} \quad \forall d \in D, k \in DN, \tag{4.2}$$

$$\sum_{p \in P} \sum_{k \in N^-_{p,R_i}} Y_{p,k,R_i} = 1 \quad \forall i \in I, \tag{4.3}$$

$$\sum_{p \in P} \sum_{k_1 \in HR_i} Y_{p,R_i,k_1} = 1 \quad \forall i \in I, \tag{4.4}$$

$$\sum_{k_1 \in R} Y_{p,HN_p,k_1} = 1 \quad \forall p \in P, \tag{4.5}$$

$$\sum_{k \in HR} Y_{p,k,HN_p} = 1 \quad \forall p \in P, \tag{4.6}$$

$$\sum_{d \in D} \sum_{k \in N^-_{d,k_1}} Y_{d,k,k_1} \leq 1 \quad \forall k1 \in \{HR \cup DN\}, \tag{4.7}$$

$$\sum_{k \in N^-_{d,k_1}} Y_{d,k,k_1} = \sum_{k_2 \in N^+_{d,k_1}} Y_{d,k_1,k_2} \quad \forall k1 \in \{HR \cup DN\}, d \in D, \tag{4.8}$$

$$\sum_{k \in DN} Y_{d,k,HN_d} = 1 \quad \forall d \in D, \tag{4.9}$$

$$\sum_{k_1 \in HR} Y_{d,HN_d,k_1} = 1 \quad \forall d \in D, \tag{4.10}$$

$$\sum_{d \in D} \sum_{k \in N^-_{d,k_1}} \sum_{k_1 \in HR_i} Y_{d,k,k_1} = 1 \quad \forall i \in I, \tag{4.11}$$

$$\sum_{p \in P} \sum_{j \in \{I \cup \{-1\}\}} X_{p,i,j} = 1 \quad \forall i \in I, \tag{4.12}$$

$$\sum_{p \in P} \sum_{i \in \{I \cup \{0\}\}} X_{p,i,j} = 1 \quad \forall j \in I, \tag{4.13}$$

$$\sum_{j \in I} X_{p,0,j} = 1 \quad \forall p \in P, \tag{4.14}$$

$$\sum_{i \in I} X_{p,i,-1} = 1 \quad \forall p \in P, \tag{4.15}$$

$$\sum_{i \in \{I \cup \{0\}\}} X_{p,i,j} = \sum_{l \in \{I \cup \{-1\}\}} X_{p,j,l} \quad \forall p \in P, j \in I, \tag{4.16}$$

$$2X_{p,i,j} \leq \sum_{k \in HR_i} (Y_{p,R_i,k} + Y_{p,k,R_j}) \quad \forall i \in I, j \in I, p \in P, \tag{4.17}$$

$$X_{p,i,j} + 1 \geq \sum_{k \in HR_i} (Y_{p,R_i,k} + Y_{p,k,R_j}) \quad \forall i \in I, j \in I, p \in P, \tag{4.18}$$

$$2X_{p,0,i} \leq Y_{p,HN_p,R_i} + \sum_{k \in HR_i} Y_{p,R_i,k} \quad \forall i \in I, p \in P, \tag{4.19}$$

$$X_{p,i,j} + 1 \geq Y_{p,HN_p,R_i} + \sum_{k \in HR_i} Y_{p,R_i,k} \quad \forall i \in I, p \in P, \tag{4.20}$$

$$2X_{p,i,-1} \leq \sum_{k \in HR_i} (Y_{p,R_i,k} + Y_{p,k,HN_p}) \quad \forall i \in I, j \in I, p \in P, \tag{4.21}$$

$$X_{p,i,-1} + 1 \geq \sum_{k \in HR_i} (Y_{p,R_i,k} + Y_{p,k,HN_p}) \quad \forall i \in I, j \in I, p \in P, \tag{4.22}$$

$$Y_{p,R_i,k} \leq Y_{p,k,R_j} + 2(1 - X_{p,i,j}) \quad \forall p \in P, i \in I, j \in I, k \in HR_i, \tag{4.23}$$

$$Y_{p,k,R_j} \leq Y_{p,R_i,k} + 2(1 - X_{p,i,j}) \quad \forall p \in P, i \in I, j \in I, k \in HR_i, \tag{4.24}$$

$$Y_{p,R_i,k} \leq Y_{p,k,HN_p} + 2(1 - X_{p,i,-1}) \quad \forall p \in P, i \in I, j \in I, k \in HR_i, \tag{4.25}$$

$$Y_{p,k,HN_p} \leq Y_{p,R_i,k} + 2(1 - X_{p,i,-1}) \quad \forall p \in P, i \in I, j \in I, k \in HR_i, \tag{4.26}$$

$$\sum_{p \in P} \sum_{d \in D} a_{p,d,i} = 1 \quad \forall i \in I, \tag{4.27}$$

$$Y_{p,R_i,k_1} \leq \sum_{k \in N^-_{d,k_1}} Y_{d,k,k_1} + 2(1 - a_{p,d,i}) \quad \forall p \in P, d \in D, i \in I, k_i \in HR_i, \tag{4.28}$$

$$\sum_{k \in N^-_{d,k_1}} Y_{d,k,k_1} \leq Y_{p,R_i,k_1} + 2(1 - a_{p,d,i}) \quad \forall p \in P, d \in D, i \in I, k_i \in HR_i, \tag{4.29}$$

$$t_{p,k} \geq t_{p,R_i} + T^{pick}_{p,i} + \frac{L_{R_i,k}}{V_p} - M(1 - Y_{p,R_i,k}) \quad \forall p \in P, i \in I, k \in HR_i, \tag{4.30}$$

$$\begin{gathered} t_{p,R_j} \geq t_{p,k} + T^{coop}_{p,i} + \frac{L_{k,R_j}}{V_p} - M(2 - X_{p,i,j} - Y_{p,R_i,k}) \\ \forall p \in P, i \in I, j \in I, k \in HR_i, \end{gathered} \tag{4.31}$$

$$t_{p,R_i} \geq \frac{L_{R_i,k}}{V_p} - M(1 - Y_{p,HN_p,R_i}) \quad \forall p \in P, i \in I, \tag{4.32}$$

$$t_{d,k_1} \geq t_{d,k} + T^{coop}_{p,i} + \frac{L_{k,k_1}}{V_d} - M(2 - a_{p,d,i} - Y_{d,k,k_1}) \quad \forall p \in P, d \in D, i \in I, k \in HR_i, k_1 \in N^+_{d,k}, \tag{4.33}$$

$$t_{d,k_1} \geq t_{d,k} + T^{drop}_{d} + \frac{L_{k,k_1}}{V_d} - M(1 - Y_{d,k,k_1}) \quad \forall d \in D, i \in I, k \in DN, k_1 \in HR, \tag{4.34}$$

$$t_{d,k} \geq \frac{L_{HN_d,k}}{V_d} - M(1 - Y_{d,HN_d,k}) \quad \forall d \in D, k \in HR, \tag{4.35}$$

$$t_{d,k_1} \geq t_{p,k_1} - M(2 - a_{p,d,i} - Y_{p,k,k_1}) \quad \forall p \in P, d \in D, i \in I, k_1 \in HR_i, k \in N^-_{p,k_1}, \tag{4.36}$$

$$t_{p,R_j} \geq t_{d,k} + T^{coop}_{p,i} + \frac{L_{k,R_j}}{V_p} - M(3 - a_{p,d,i} - X_{p,i,j} - Y_{p,k,R_j}) \quad \forall p \in P, d \in D, i \in I, j \in I, k \in HR_i, \tag{4.37}$$

$$S_{d,k,k_1} \leq Y_{d,k,k_1} \quad \forall d \in D, k_1 \in \{HR \cup DN\}, k \in N^-_{d,k_1}, \tag{4.38}$$

$$S_{d,HN_d,k} = Y_{d,HN_d,k} \quad \forall d \in D, k \in HR, \tag{4.39}$$

$$S_{d,k,k_1} = Y_{d,k,k_1} \quad \forall d \in D, k \in DN, k_1 \in HR, \tag{4.40}$$

$$S_{d,k_1,k_2} \geq \sum_{k \in N^-_{d,k_1}} S_{d,k,k_1} + Y_{d,k_1,k_2} - C_d(1 - Y_{d,k_1,k_2}) \quad \forall d \in D, k_2 \in HR, k_1 \in N^-_{d,k_2}, \tag{4.41}$$

$$m \geq 0, \tag{4.42}$$

$$t_{e,k} \geq 0 \quad \forall e \in E, k \in N, \tag{4.43}$$

$$S_{d,k,k_1} \geq 0 \quad \forall d \in D, k_1 \in N, k \in N^-_{d,k_1}, \tag{4.44}$$

$$a_{p,d,i} \in \{0,1\} \quad \forall p \in P, d \in D, i \in I, \tag{4.45}$$

$$X_{p,i,j} \in \{0,1\} \quad \forall p \in P, i \in I, j \in I, \tag{4.46}$$

$$Y_{e,k,k1} \in \{0,1\} \quad \forall e \in E, k_1 \in N, k \in N^-_{e,k_1}, \tag{4.47}$$

The objective function 4.1 is to minimize the makespan when all items from the RPL are delivered to the depot, as limited by Constraint 4.2. Constraint 4.3 ensures each picking node can be visited by only one picker one time. Constraint 4.4 ensures a picker will move to one of the hand-off replicas of the item after picking that item at the picking node. Constraints 4.5 and 4.6 specify a picker will start from its home node and return to its home node after finishing the pick wave. Constraint 4.7 ensures that no more than one transporter will visit each hand-off replica and depot replica. Constraint 4.8 guarantees a transporter will leave a hand-off replica or a depot replica if it enters that node. Constraints 4.9 and 4.10 ensure a transporter will start from its home node and return to its home node after finishing the pick wave. Constraint 4.11 makes sure that each item can only be placed on one transporter at one hand-off replica.

Constraints 4.12 to 4.29 coordinate the items picking sequence with the node sequence of each picker and transporter. Constraints 4.12 and 4.13 ensure each item can only be picked by one picker. Constraints 4.14 to 4.16 set the proper picking sequence of the items for each picker and guarantee each picker will start from picking item 0 and end from picking item -1. Constraints 4.17 to 4.22 set the appropriate node sequence for each picker based on the picking sequence. Constraints 4.23 to 4.26 ensure when a picker enters a hand-off replica, it will leave that node. Constraints 4.27 to 4.29 set the appropriate value of $a_{p,d,i}$ to pair pickers and transporters and ensure each item can only be placed on one transporter.

Constraints 4.30 to 4.37 are related to the route time of all entities. Constraint 4.30 indicates that if a picker moves from the picking node R_i to a hand-off replica k, then the arrival time to k cannot be before the arrival time to R_i plus the picking time at R_i plus the travel time from R_i to k. The value of M is a sufficiently large number. In this model, one possible value could be calculated as the maximum cumulative time of traveling through all nodes' pairs plus the summation of the maximum time of picking, placing, and dropping times the number of items. Constraint 4.31 indicates that if a picker moves from the hand-off replica k to the picking node R_j, then the arrival time to R_j cannot be before the arrival time to k plus the placing time at k plus the travel time from k to R_j. Constraint 4.32 states the time a picker moves from the home node to a picking node. Constraint 4.33 ensures that the time a transporter starts getting the item at a hand-off replica k_1 cannot be earlier than the time the transporter gets the previous

item at k plus the travel time. Constraints 4.34 and 4.35 state the time a transporter starts getting item when the transporter travels from a depot replica or the home node. Constraint 4.36 indicates a transporter can start getting the item only when the associated picker is ready at the hand-off replica. Constraint 4.37 indicates a picker cannot move to the next picking node until the placement of the previous item is completed at the hand-off replica.

Constraints 4.38 to 4.41 are related to the capacity limitation of the transporters. Constraint 4.38 ensures the total payload on a transporter cannot exceed the limit at any node. Constraint 4.39 and 4.40 guarantee that a transporter's payload is 1 when that transporter leaves a hand-off replica and has traveled from the home node or a depot replica. Constraint 4.41 forces the payload to be at least the summation of the payload when a transporter leaves node k_1 plus the item loaded at node k_2. Constraints 4.42 to 4.47 are related to the definition of the decision variables.

4.5 Performance Comparison

This section compares the performance of our above-discussed MILP model with the baseline model in [12] based on the results from experiments under different settings. Both models are solved by Gurobi 9.1.2 [76] via Python version 3.8.12. All computational work was conducted on a PC with an Intel(R) Xeon(R) W-2123 processor and 128 GB RAM running Microsoft Windows 10 in 64-bit mode.

To better evaluate these two models, we apply five main different settings: (1) 3 different numbers of pickers from 1 picker to 3 pickers, (2) 3 different numbers of transporters from 1 transporter to 3 transporters, (3) 2 different RPL sizes $|I| = 5$ items and $|I| = 10$ items, (4) 3 different transporter capacities, and (5) 6 different warehouse layouts which have different numbers of picking aisles (PA) and cross aisles (CA). Therefore, there are 324 experiment scenarios in total.

The 3 transporter capacity settings are *Low*, *Medium*, and *High*. The *Low* capacity indicates the transporter cannot carry more than 2 items. The *Medium* capacity is related to the RPL size that the transporter cannot carry more than $\lceil |I|/2 \rceil$ items. The *High* capacity is also related to the RPL size that the transporter cannot carry more than $|I|$ items. For the warehouse

layout settings, we test warehouses with 4 and 8 picking aisles and 0 to 2 cross aisles (top and bottom aisles are not included). All 6 warehouses are rectangular and their picking aisles are perpendicular to the cross aisles like the warehouse shown in Figure 3.1. In that figure, we can also find that if the picking slots in a storage rack are close enough, it is reasonable to use only one location to represent the picking locations of all picking slots in that storage rack. In addition, based on our analysis of the literature shown in Table 3.2, the width of the picking aisles is from 1m to 3m, which we think is small. So, using one picking location for both side storage racks of a picking aisle is practical as well. Then, the warehouses used in the experiments have the following properties settings: (1) the width of the picking aisles is 2m, (2) the width of the cross aisles is 3m, (3) there are 96 storage racks, and each is 2m wide, 3m long and contains 10 picking slots, (4) the picking location for the two storage racks of a picking aisle locates in the center of the aisle as shown in Figure 4.1. (5) the depot, which also acts as the home location, is at the center of the bottom aisle, (6) there are 4 additional hand-off spots in the warehouses, which are located at the top aisles, bottom aisle, or cross aisles. Figure 4.2 shows one layout of our tested warehouses with 4 picking aisles and 1 cross aisle. In this figure, each red node is the picking location for the nearby two storage racks. Each green node is the additional hand-off spot. Each blue node is used to form the aisles and the black node is the depot. The other 5 layouts are shown in Appendix A.

We have some other settings for the model parameters. Each Picker travels at a speed of 1 m/s, while each transporter travels at 2 m/s. Based on our definition of item, the picking time of pickers is based on per location shown in Table 3.2. So, we choose 30 seconds/item as the picking time. Considering placing multiple items in a tote will not differ much from placing just one item, in our experiment, we set 5 seconds/item as the placing time. The tote drop-off time of each transporter is 60s. The total number of depot replicas is 10. For each item, its hand-off replicas contain 5 nodes: the corresponding picking node and the other 4 hand-off replicas that are not the picking locations. In addition, for each experiment scenario, we test 50 replications where each replication has a different RPL and in each replication, both models use the same RPL. The cut time of both models is set to 300s for 5 items RPL and 600s for 10 items RPL. So, all experimental settings are summarized in Table 4.3.

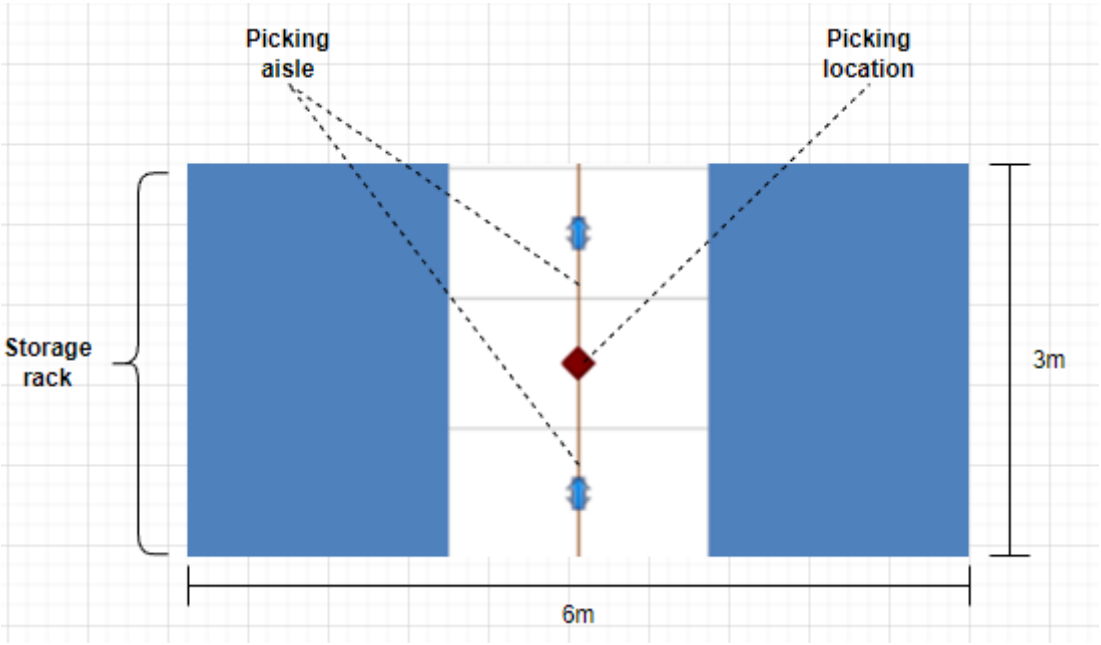


Figure 4.1: An example shows the positions of picking location and storage racks in the experiments.

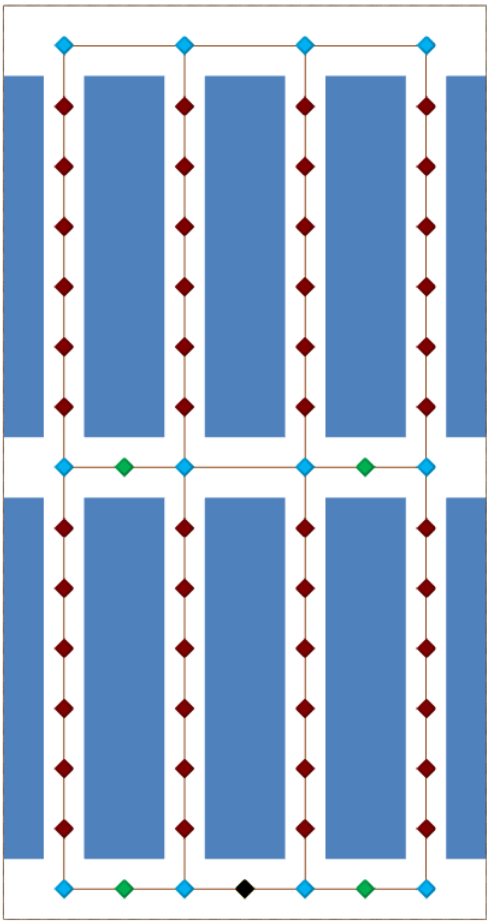


Figure 4.2: The layout of a warehouse with 4 picking aisles and 1 cross aisle.

Table 4.3: Summary of the experimental settings

Setting	Value	Levels
Number of Pickers		1, 2, 3
Number of Transporters		1, 2, 3
Size of RPL		5 items, 10 items
Transporter Payload Capacity		*Low* (2 items), *Medium* ($\lceil \lvert I \rvert / 2 \rceil$), *High* ($\lvert I \rvert$)
Number of PAs		4, 8
Number of CAs		0, 1, 2
Cut Time		300s, 600s
Number of Storage Racks	96	
Number of Picking Slots	960	
Number of Picking Locations	48	
Width of PA	2 m	
Width of CA	3 m	
Picker Speed	1 m/s	
Transporter Speed	2 m/s	
Picking Time	30 /item	
Placing Time	5 /item	
Total Number of DN	10	
Size of HR_i	5	
Tote Drop-off Time	60 s	
Replications of main settings combination	50	

In each replication, both models generate schedules for all entities. Then, the makespan and average wait time of pickers are computed as the comparison metrics to analyze these two models. Here, the wait time is the time used for a picker waiting at the hand-off spots. After finishing all replications, we find that for the scenarios when the RPL has 5 items, the baseline model can always find the optimal solutions before the cut time (300s). While for all other scenarios, both models cannot find the optimal solutions before the cut time.

Table 4.4 shows when the RPL has 5 items, the average percentage increase of the makespan of the schedule obtained from our model compared to the makespan obtained from the baseline model under scenarios of different numbers of pickers, numbers of transporters, numbers of PAs, and transporter capacities. In the table, 1P1T means the scenarios use 1 picker and 1 transporter. Also, a negative value in the table means our model can find better schedules for the scenarios since the average makespan of all corresponding replications is smaller. From the table, we can find that when there are fewer transporters with *Low* or *Medium* capacity,

schedules from our method are better. That means in those scenarios, pickers often need to find different locations to place the items. While if there are more transporters with *High* capacity, the baseline can provide better schedules because the model does not need to find different hand-off spots for pickers to place the picked items since the transporters in those scenarios are capable of waiting for the pickers at the picking locations. Table 4.5 shows when the RPL has 5 items, the percentage of the replications in different scenarios that our model provides a better result than the baseline model on makespan under different combinations of the number of pickers, number of transporters, number of PAs, and transporter payload capacities. From the table, we can find that when the transporters with *Low* or *Medium* capacity are fewer than the pickers, our model can find better schedules for almost every replication in those scenarios. Considering the schedules found by the baseline model are the optimal solutions that model can get in those scenarios, we can conclude that during the pick wave, there are always some pickers that need to move after picking an item to shorten the makespan. Table 4.6 shows when the RPL has 5 items, the average wait time decrease per item by changing from the baseline model to our model under different combinations of the number of pickers, number of transporters, number of PAs, and transporter payload capacities. This table indicates reducing the wait time of placing each item is very important for a pick wave in MCOP to have shorter makespans.

Table 4.4: When the RPL has 5 items, the average percentage (%) increase of the makespan obtained from our model compared to the value obtained from the baseline model in different scenarios.

Entity Combination	4PAs			8PAs		
	Low	Medium	High	Low	Medium	High
1P1T	**-6.0**	**-3.2**	0.8	**-6.2**	**-3.3**	0.3
1P2T	0.9	0.6	0.5	0.4	0.4	0.4
1P3T	0.2	0.2	0.3	0.2	0.2	0.2
2P1T	**-15.2**	**-9.4**	2.6	**-14.0**	**-7.3**	1.3
2P2T	3.6	3.5	3.9	2.8	2.7	2.8
2P3T	2.5	2.1	2.1	2.3	2.6	1.5
3P1T	**-21.0**	**-17.4**	0.0	**-16.0**	**-14.8**	**-2.0**
3P2T	**-6.5**	3.9	3.7	**-5.8**	4.2	4.7
3P3T	2.8	2.0	1.8	2.2	2.3	2.4

Figure 4.3 shows the percentage of average makespan from scenarios with 3CAs and 4CAs compared to average makespan from scenarios with 2CAs in our model. From the figure, we

Table 4.5: When the RPL has 5 items, the percentage (%) of replications in different scenarios that our model provides a better result than the baseline model on makespan.

Entity Combination	4PAs			8PAs		
	Low	Medium	High	Low	Medium	High
1P1T	**96.0**	**85.3**	0.0	**98.0**	**90.7**	0.0
1P2T	0.0	0.0	0.0	0.0	0.0	0.0
1P3T	0.0	0.0	0.0	0.0	0.0	0.0
2P1T	**100.0**	**98.0**	**18.7**	**100.0**	**96.7**	**30.0**
2P2T	**19.3**	0.0	0.0	**19.2**	0.0	0.0
2P3T	0.0	0.0	0.0	0.0	0.0	0.0
3P1T	**100.0**	**100.0**	**38.0**	**100.0**	**99.3**	**59.3**
3P2T	**82.0**	0.7	1.3	**83.3**	0.0	0.0
3P3T	0.0	0.0	0.0	0.0	0.0	0.0

can find that increasing the number of cross aisles may not always shorten the makespan. This is consistent with the findings in [12, 40] that increasing the number of CAs might not improve performance despite creating more picking route options.

For those scenarios when RPL has 10 items, the results are similar to what we find from scenarios with 5 items, and the detailed tables are shown in Appendix A. However, we find that there is about 20 percent of the overall replications using our model cannot find a feasible solution before the cut time. So, from a practical standpoint, when there are more items in the RPL, it is important to find a fast method to generate the schedule for all entities. And that schedule could be used as the initial solution for our MILP model.

Table 4.6: When the RPL has 5 items, the average wait time decrease per item by changing from the baseline model to our model in different scenarios.

Entity Combination	4PAs			8PAs		
	Low	Medium	High	Low	Medium	High
1P1T	**3.3s**	1.4s	0.0s	**2.8s**	1.2s	0.0s
1P2T	0.1s	0.0s	0.0s	0.0s	0.0s	0.0s
1P3T	0.0s	0.0s	0.0s	-0.1s	0.0s	0.0s
2P1T	**15.9s**	**9.5s**	0.4s	**13.2s**	**6.9s**	0.9s
2P2T	1.8s	0.1s	0.4s	1.7s	0.8s	0.6s
2P3T	0.0s	0.3s	0.4s	0.7s	0.6s	0.2s
3P1T	**22.0s**	**10.3s**	1.2s	**13.6s**	**7.1s**	1.7s
3P2T	**6.1s**	0.6s	0.6s	**4.8s**	0.5s	-0.1s
3P3T	0.1s	1.0s	0.3s	0.1s	0.2s	0.7s

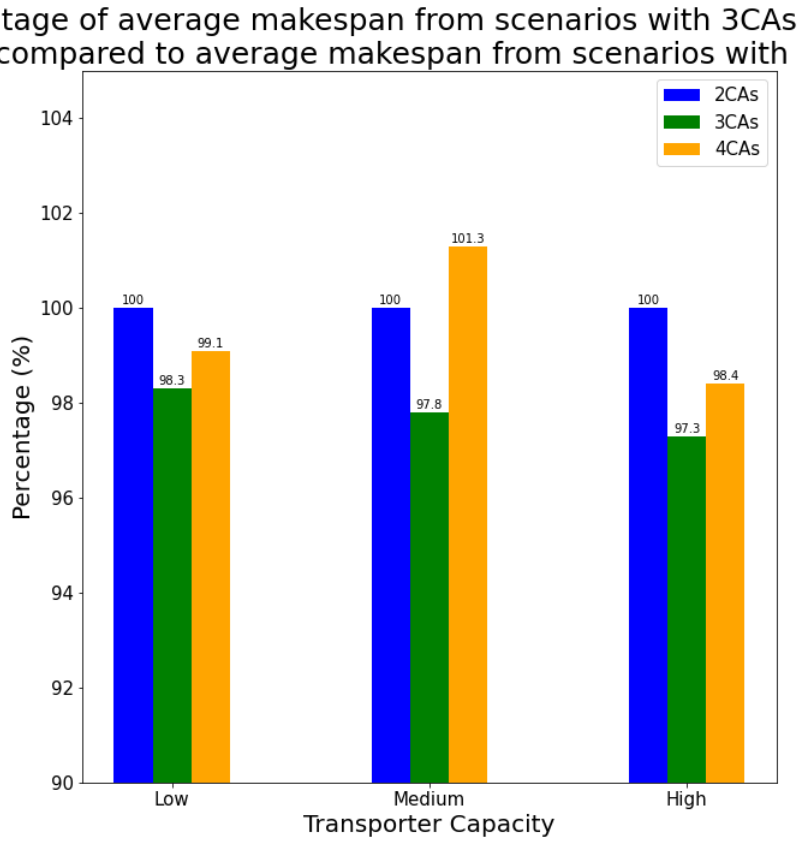


Figure 4.3: Percentage of average makespan from scenarios with different CAs in our model.

4.6 Summary

This chapter proposed a mathematical model based on MILP to find schedules for all entities in a pick wave of MCOP. Compared to the existing work, our model gives pickers the ability to place the picked items at a place where it is different from the picking location. In addition, unlike the existing work, our model can decide how many times each transporter should visit the depot.

We performed a set of experiments to investigate the performance of our model with the existing work under different settings in terms of makespan and picker wait time. The results show that the makespan from our model is smaller than the one from the existing work when there are fewer transporters with a small capacity. The results also indicate reducing the pickers' wait time to place the picked items is an important reason that our model can provide better results in some experiment settings. From our experiments, we found increasing the number of CAs may lead to a higher makespan, which is consistent with the findings from other research

as well. We found the mathematical model is not efficient when the RPL size is large. So, to use this model in practice, a more efficient method is needed to provide an initial solution to our model. In the next chapter, we will discuss our proposed efficient method to generate schedules for all entities.

Chapter 5

Development of Fast Method for Schedules of All Entities

5.1 Introduction

In this chapter, to overcome the disadvantage of the MILP model which is insufficient to handle MCOP with large required pick list (RPL), we develop an alternative algorithm called Hetero-ACO to find schedules for all entities. The purpose of the proposed Hetero-ACO is to try to find good schedules which result in a small pick wave makespan in a reasonable amount of computational time. Unlike the MILP we discussed in Chapter 4, Hetero-ACO is not a global optimization algorithm which means the best schedules obtained in Hetero-ACO are not guaranteed to be the global optimal schedules no matter how long you run this algorithm. Also, since our Hetero-ACO is developed based on ACO, it is a swarm intelligence (SI) optimization algorithm with the merits of self-learning ability, easy implementation, and a simple framework.

Because pickers and transporters act asynchronously and each can cooperate with multiple partners during the pick wave in MCOP, improper schedules for all entities could lead the pick wave to a deadlock where all pickers are waiting for transporters and all transporters are waiting for pickers. Therefore, a dedicated algorithm that can avoid generating those infeasible schedules is important since even for a fast algorithm, we don't want to waste computational resources on infeasible solutions. Unlike some SI-based algorithms, which rely on modifying the complete solution, ACO employs the probabilistic technique and gradually constructs the complete solution. Thus, this feature of ACO provides the ability to avoid infeasible solutions during the construction of the solutions.

The organization of this chapter is as follows. In section 5.2, we discuss the overview of Hetero-ACO which includes the workflow of the algorithm and the encoding strategy used in the algorithm. Section 5.3 presents the methods involved in the construction of schedules for all transporters when all schedules of pickers are given. Then, we describe and compare the methods used in the algorithm for searching the schedules for all pickers in section 5.4. In section 5.5, we conduct experiments and compare our Hetero-ACO with a heuristic to evaluate the performance in MCOP. Finally, section 5.6 summarizes our findings.

5.2 Overview of Hetero-ACO

To better present our proposed algorithm, we first must illustrate our encoding strategy. Figure 5.1 shows an example of the coded representation of entities' schedules. In this figure, for a picker $p \in P$, a sequence of item indices is used to represent its schedule S_p, and the picker will follow this sequence to pick the assigned items. In this coded representation, the hand-off spot for each item is the picking location at the starting point for the algorithm and may change as the algorithm iterates. While for a transporter $d \in D$, the sequence of item indices is also used to represent its schedule S_d. Here, we will use the symbol Dt to represent the depot. Therefore, the coded representation of a transporter's schedule consists of item indices and the depot symbols. In addition, we use S_P and S_D to denote the schedules for all pickers and all transporters, respectively. The variables used in the rest of this chapter are listed in Table 5.1

Figure 5.2 shows the flowchart of our Hetero-ACO. After initializing the algorithm parameters (see Table 5.2 and Table 5.3) and the alternative MILP, we first generate the schedules for all pickers. In this algorithm, generating random schedules with equal-sized sequences, adjusting existing schedules with some operators, and using intermediate solutions from small MILP models are the methods we applied. Those methods are discussed in detail in section 5.4. When schedules for all pickers are determined, the algorithm will construct S_D. There are three terms *Group*, *Squad*, and *Ant*, which are important for understanding the construction processes. An *Ant* in the algorithm represents a virtual transporter, and each *Ant* will construct its schedule. A *Squad* represents a group of *Ants*, and the number of *Ants* equals the number of transporters

Table 5.1: variables involved in Hetero-ACO

Variable Notation	Parameter Notation Description
S_p	Schedules of a picker $p \in P$
S_d	Schedules of a transporter $d \in D$
S_P	Schedules of all pickers
S_D	Schedules of all transporters
$p^s_{i,j,k}$	The probability for the sth *Ant* in the *Squad* s of adding item j after adding item i to its schedule
$\tau_{i,j}$	Pheromone value of item index pair (i, j)
$\Delta\tau^s_{i,j}$	Pheromone increment on item index pair (i, j) left by *Squad* s
$\eta_{i,j}$	Heuristic function value of item index pair (i, j)
$time_{gap}$	The maximum time reduction in makespan the algorithm gets through the dynamic local search after completing dl *Groups*
m^s	The makespan with *Squad* s
t'_i	Time the corresponding Ant leaves the hand-off spot of item i
t'_j	Time the corresponding Ant reaches the hand-off spot of item j
S^t_P	Schedules of all pickers at iteration t
S^t_D	Schedules of all transporters at iteration t
T_t	Temperature value used in SA at iteration t
m_t	Makespan at iteration t
p_t	Probability of accepting the new S^t_P or not in SA at iteration t
$tabu$	The tabu list in TS

used in MCOP. In addition, the schedules for the *Ants* in a *Squad* form a S_D. A *Group* represents a group of *Squads*, each *Squad* in a *Group* shares the same pheromone matrix. For a given S_P, there will be an enormous number of possible S_D. So, by using *Groups* and *Squads*, our algorithm can construct multiple S_D that increase the chances of finding relatively good solutions. Therefore, the algorithm constructs schedules *Squad* by *Squad* and *Group* by *Group*. After Constructing the schedules for a *Squad*, we can compute the makespan since we have a set of schedules for all entities. As discussed above, each item's hand-off spot is the picking location by default in our coded representations. After computing the makespan, the algorithm may apply a dynamic local search to find different hand-off spots for each item to decrease the makespan. When all *Squads* in a *Group* finish the schedule construction and local search, then the algorithm will update the pheromone matrix for the next *Group*. When all *Groups* finish, our algorithm completes an iteration for finding schedules for all entities. Then, the algorithm starts a new iteration with a new S_P or stops and returns the best schedules found with the

smallest makespan. The criterion of our algorithm is the maximum running time. Section 5.3 provides a more detailed explanation of how the algorithm constructs S_D.

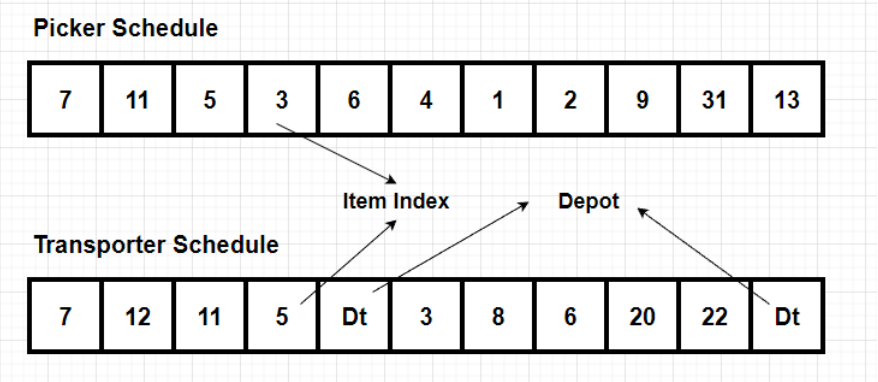


Figure 5.1: A example of the coded representation of entities' schedules

5.3 Construction of Schedules for Transporters

To better understand this algorithm, we will first explain the details of the construction of S_D. The parameters involved in this construction process are shown in Table 5.2. We also use some parameters discussed in section 4.2. After a S_P is given, our algorithm will generate gp_{max} *Groups*, and in each *Group*, there are sd_{max} *Squads*. To construct the schedules for each *Squad*, the strategy is sequentially letting an *Ant* in the *Squad* add an item index or the depot symbol until all item indices are assigned. For example, if the *Squad* has two *Ants*, then first *Ant* will add its first item index; the second *Ant* add its first item index; the first *Ant* add its second item index and so on. The strategy of picking the next item for each *Ant* depends on the given S_P and a probabilistic rule that considers the pheromone value and a heuristic function. The following formula shows how the kth *Ant* decides the probability of adding item j after adding item i to its schedule.

$$p^s_{i,j,k} = \begin{cases} \frac{G(w_i)*(\tau_{i,j})^\alpha*(\eta_{i,j})^\beta}{\sum_j (\tau_{i,j})^\alpha*(\eta_{i,j})^\beta}, & i \in I, j \in Sub_I \subset I \\ 1 - G(w_i), & j \; is \; Dt \end{cases} \tag{5.1}$$

In this formula, $p^s_{i,j,k}$ is the probability for the kth *Ant* in the *Squad* s of adding item j after adding item i. $\tau_{i,j}$ is the pheromone value of index pair (i, j), and $\eta_{i,j}$ is the heuristic function

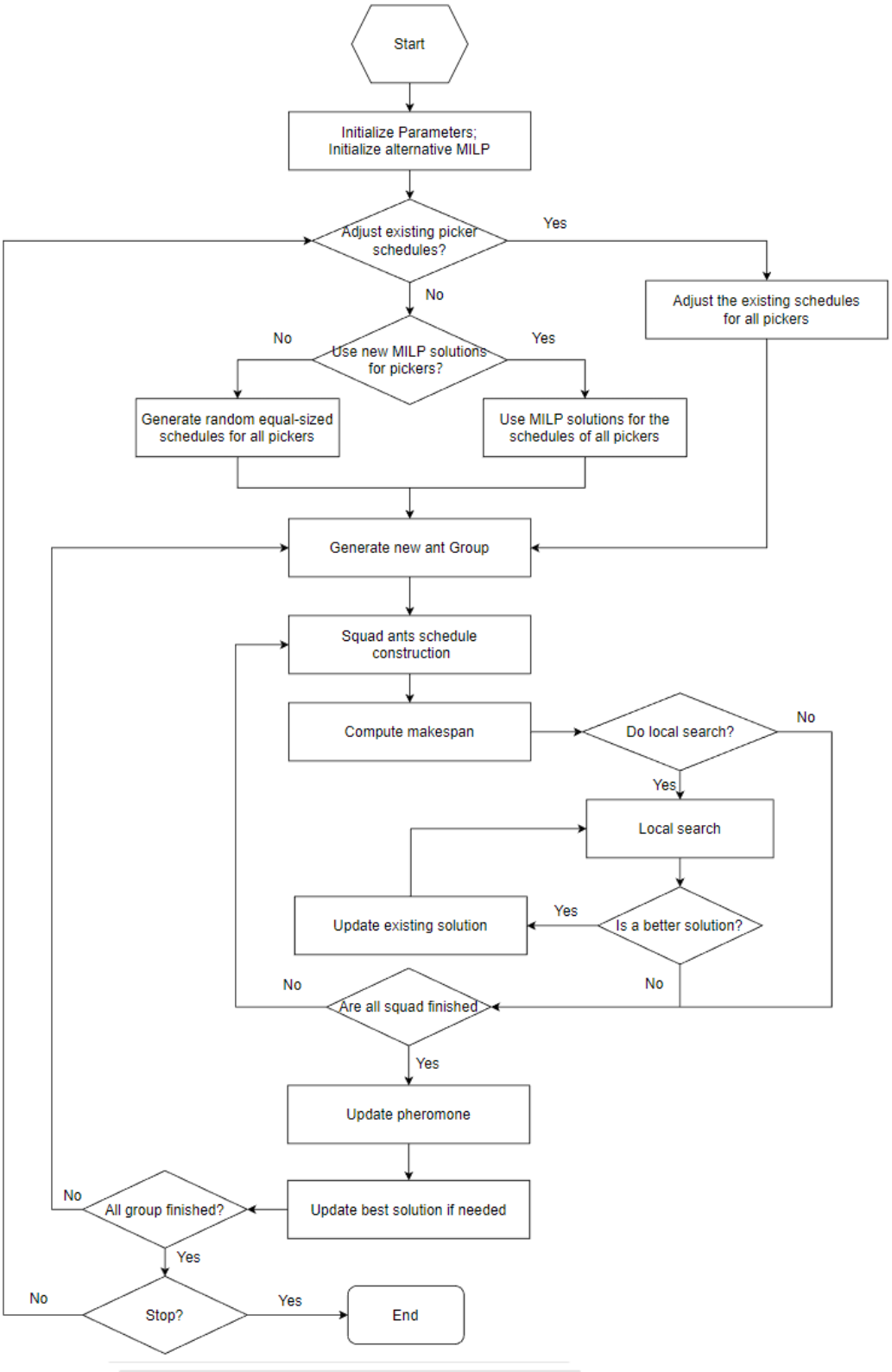


Figure 5.2: The flowchart of the Hetero-ACO

value of index pair (i, j). α and β are the relative importance of the pheromone density and the heuristic function value, respectively. w_i is the payload of the *Ant* after taking item i, and $G(w_i)$

is a function that related to the payload. The higher the payload, the lower the $G(w_i)$ value. So $G(w_i)$ can control when an *Ant* needs to visit the depot to unload all carried items. In this work, when the *Ant* doesn't reach the payload capacity, $G(w_i) \in [0.9, 1]$ is a linear function with w_i, which means when the payload is zero, $G(w_i) = 1$. And when the payload is the *Ant* capacity minus one, $G(w_i) = 0.9$. And when the *Ant* reaches payload capacity, $G(w_i) = 0$. In this formula, when j is not the depot symbol, j belongs to a subset of all items. Figure 5.3 explains how the subset is built, and changes as the *Ants* add item indices to their schedules. In this figure, 3 pickers and 2 transporters cooperate to retrieve 9 items. When the S_P is determined, the initial subset consists of the first item of all pickers, that is items {1,4,7}. If *Ant1* add item 4 to its schedule, then, the subset will be items {1,5,7}. After *Ant2* add item 7 to its schedule, the subset then changes to items {1,5,8} and *Ant1* can add an item which is item 8 from this subset. So, the subset always consists of the first item that is not added by *Ants* in each picker's schedule. By using this rule, we can guarantee the constructed schedules are feasible.

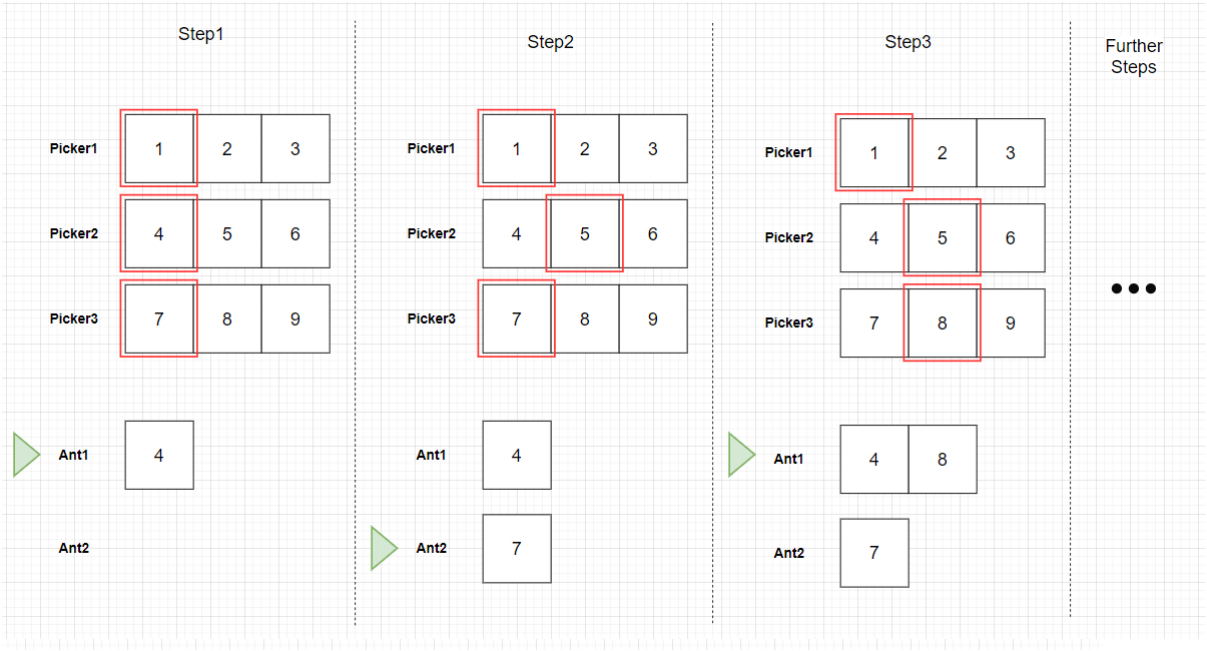


Figure 5.3: An example of how the subset of items is formed based on the schedules of pickers for *Ants* to add to schedules

Table 5.2: Parameters involved in constructing schedules of all transporters in Hetero-ACO

Parameter Notation	Parameter Notation Description
α	Relative importance of pheromone
β	Relative importance of heuristic
ρ	Pheromone evaporation rate
τ_{init}	Initial pheromone density
gp_{max}	Maximum number of *Groups* used
sd_{max}	Maximum number of *Squads* in each *Group*
Q_h	A constant value used in the heuristic function
$HS_{i,j}$	The set of all alternative hand-off spots along the path between the picking locations of item i and item j
f	The dynamic local search will stop if f consecutive items are tested with no improving makespan
dl	The number of *Groups* used to compute the maximum time reduction in the dynamic local search
Q_τ	A constant value used in the computation of increased pheromone of each *Squad*
τ_{min}	The lower bound of the pheromone in each index pair
τ_{max}	The upper bound of the pheromone in each index pair

5.3.1 The Heuristic Function

In the ACO-based algorithms, a good heuristic is helpful for constructing good solutions. In MCOP, for intuition, an item picked first should have a higher probability of being placed on a transporter first to reduce pickers' wait time, which may lead to a smaller makespan. In addition, another good intuitive strategy is the nearer the item, the higher the probability it is added by the *Ant* to reduce the traveling time between items. The formula of the heuristic function is shown below:

$$\eta_{i,j} = \frac{Q_h}{\phi(j)! * L_{i,j}} \quad i,j \in I \tag{5.2}$$

where $\eta_{i,j}$ is the heuristic function value of index pair (i,j), Q_h is a constant value, and $L_{i,j}$ is the distance between the hand-off spot of item i and the picking locations of item j. $\phi(j)$ is the relative position in the pickers' schedules. For example, in Figure 5.3, at Step 1, since the items {1,4,7} are all first item in the pickers' schedules, $\phi(1) = \phi(4) = \phi(7) = 0$. At Step 2, item 5 is the second item in schedule of Picker2, now $\phi(1) = \phi(7) = 0,\ and\ \phi(5) = 1$. At Step 3, $\phi(1) = 0,\ and\ \phi(5) = \phi(8) = 1$.

5.3.2 Dynamic Local Search

After a *Squad* s finishes constructing the schedules, our algorithm performs a dynamic local search to potentially find better solutions. This dynamic local search aims to find alternative hand-off spots for some items to decrease the makespan. Since we know where all picking locations and potential hand-off spots are, all possible paths between a hand-off spot and a picking location are known in advance. Therefore, when the warehouse layout is determined, we denote all alternative hand-off spots along the paths between the hand-off spots of item i and the picking location of item j as $HS_{i,j}$. Figure 5.4 gives an example of the alternative hand-off spots.

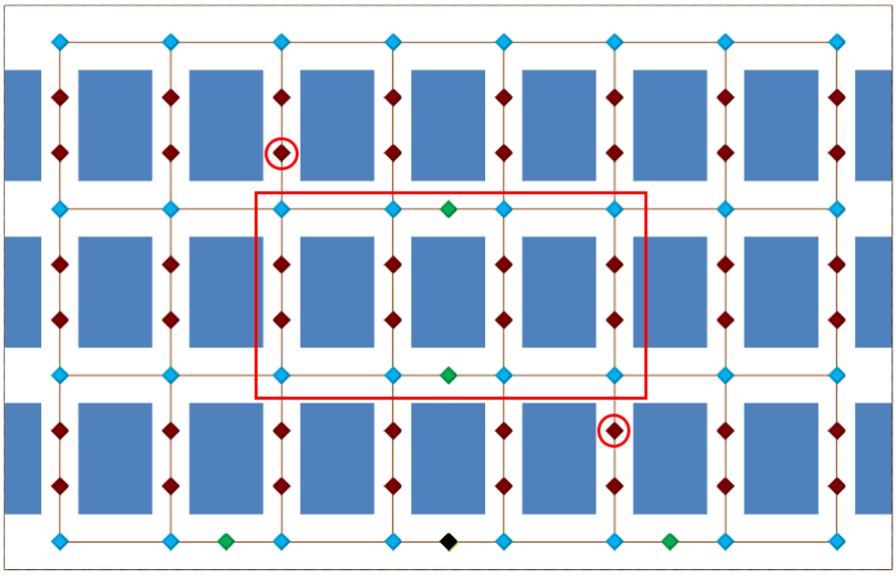


Figure 5.4: An example shows all alternative hand-off spots between a hand-off spot and a picking location. The nodes in the red circles are the corresponding hand-off spot and picking location. The nodes (red and green) in the red box are all alternative hand-off spots since the possible paths between the two nodes in the red circles pass those nodes in the red box.

The strategy of choosing which items to test the alternative hand-off spots is based on a probabilistic rule that the higher the time a picker waits at the assigned hand-off spot to place the item, the higher the probability the item is chosen to test the alternative hand-off spots. After determining the item i, the algorithm will try all alternative hand-off spots in $HS_{i,j}$ along the paths that a picker moves from its assigned hand-off spot of this item i to the picking location of the next item j. If using new hand-off spots for that item i makes the makespan smaller, then

the hand-off spot provides the largest time reduction in makespan is the new assigned hand-off spot to item i. The local search will continue until f consecutive items are tested without obtaining a smaller makespan.

From our analysis in Chapter 4, if there are relatively large numbers of transporters in MCOP, the pickers usually place the picked items at the picking locations. While, if there are fewer transporters, the pickers tend to place the picked items at alternative hand-off spots. So, the dynamic feature in our local search is achieved by the following settings: (1) During generating schedules for the first dl *Groups*, when a *Squad* finishes the local search, the time reduction in makespan before and after the local search is determined. The maximum time reduction in makespan $time_{gap}$ the algorithm gets through the local search is known after completing dl *Groups*. (2) For the rest *Groups*, when the makespan of a *Squad* finishes constructing the schedules is smaller than the shortest makespan $m_{shortest}$ the algorithm currently obtains plus $time_{gap}$, the *Squad* will perform the local search. Otherwise, the *Squad* skips the local search. The dynamic local search can obtain high efficiency using these settings under different entity combinations.

5.3.3 Update of Pheromone Matrix

Updating the pheromone is one of the essential steps in ACO-based algorithms. This technique enables the algorithms to achieve a self-learning feature which helps obtain high-quality solutions. To simulate the process of pheromone evaporation and ensure no index pair has unique advantages, we use the following formula to update the pheromone matrix:

$$\tau_{i,j}^{new} = \rho * \tau_{i,j}^{old} + \sum_{s=1}^{sd_{max}} \Delta\tau_{i,j}^{s} \quad i,j \in I \tag{5.3}$$

where $\tau_{i,j}^{new}$ is the new pheromone for index pair (i,j), $\tau_{i,j}^{old}$ is the previous pheromone for index pair (i,j) before the update. $\rho \in (0,1)$ is a constant value that determines the evaporation rate, s is the index of a *Squad* in a *Group*, and $\Delta\tau_{i,j}^{s}$ is the increased pheromone of index pair (i,j) left by the sth *Squad*. Our Hetero-ACO uses a similar strategy of computing the increased pheromone as in [52]:

$$\Delta\tau_{i,j}^{s} = \begin{cases} \frac{Q_\tau}{m^s} * \frac{m^s - (t'_j - t_i)}{m^s}, & \text{if index pair } (i,j) \text{ in the } sth \textit{ Squad} \\ 0, & \text{otherwise} \end{cases} \tag{5.4}$$

where $i, j \in I$, Q_τ is a constant value and m^s is the makespan of the sth *Squad* in a *Group*. t'_j is the time the corresponding *Ant* reaches the hand-off spot of item j, and t_i is the time the corresponding *Ant* leaves the hand-off spot of item i. So, this pheromone updating strategy includes two parts: (1) The global pheromone increment Q_τ/m^s is related to the makespan of that *Squad*. (2) The local pheromone increment $(m^s - (t'_j - t_i))/m^s$ is related to the index pair contribution to the makespan. That is, the shorter the time the *Ant* moves between the hand-off spots of items i and j, the larger the local pheromone increment the index pair (i, j) can provide. In addition, a user-defined lower and upper bound of the pheromone $[\tau_{min}, \tau_{max}]$ will be applied in our work. These bounds can help avoid the solution being stuck to a local optimal.

5.4 Searching Schedules for Pickers

The parameters involved in searching schedules for pickers are shown in Table 5.3. As illustrated in section 5.2, Hetero-ACO uses three methods to find S_P. The purpose of using these three methods is to try to let the algorithm balance the exploration and exploitation of the search process and let another model provide promising directions for the search process. As shown in Figure 5.2, the alternative MILP model is built and solved at the start of Hetero-ACO. Whenever the solver finds an incumbent solution, Hetero-ACO will store that solution. The random generation of S_P is performed after the initialization of Hetero-ACO, then at each iteration, the algorithm tends to adjust the existing S_P. If the algorithm doesn't find a smaller makespan after adjusting existing S_P for Re iterations, the algorithm will generate new S_P as a re-start strategy. If there are unused solutions from the alternative MILP model, the algorithm will first use those solutions to generate new S_P. Otherwise, the random generation method is used to generate new S_P.

Table 5.3: Parameters involved in searching schedules of all pickers in Hetero-ACO

Parameter Notation	Parameter Notation Description
Re	The maximum iterations of non-improving adjustment of pickers' schedules before using randomly generated schedules
T_{max}	The maximum temperature after using randomly generated schedules
T_{min}	The minimum temperature before using randomly generated schedules
θ	Coefficient for updating temperature at each iteration
Len_{TB}	The length of the tabu list
Len_{NB}	The size of the partial neighborhood

5.4.1 Generation of Random Schedules

In this method, we first group the items in the same picking location into a cluster. Then, for each picker, the method sequentially adds a random cluster to the pickers' schedule. When the size of the schedule exceeds $|I|/3$, the method will start to generate the schedule for another picker. After generating the schedules for all pickers, if the sizes of those schedules are different, we will remove several items in the largest-sized schedule and add them to other schedules to make all pickers' schedules equal-sized. The rule for selecting those items is based on the size of the items' corresponding cluster. Those items from smaller clusters have a high priority to be selected to make equal-sized schedules.

5.4.2 Adjusting Existing Schedules

In Hetero-ACO, we use three operators to adjust the existing schedules: crossover operator, 2-opt operator, and relocation operator.

For the crossover operator, it comes from GA (Genetic Algorithm), where the operator randomly chooses two pickers and swaps two item indices between their coded representations for their schedules. After that, the two new schedules replace the original schedules. Figure 5.5 shows an example of how the crossover operator adjusts two pickers' schedules. The 2-opt operator is a classical local search heuristic, which was first introduced in [77]. The main idea is to exchange the two neighborhood locations to find better solutions. In our work, we will randomly select a picker and swap two item indices in the picker's schedule. Then, the original schedule will be replaced by this newly generated one. Figure 5.6 shows an example of how the

2-opt operator works. The relocation operator will cut off one item index in a picker's schedule with the longest time and insert that item index into a target picker's schedule with the shortest time. The item from smaller clusters have a high priority to be cut off. If the target schedule has items belonging to the same cluster of the inserted item, then the inserted item will be inserted adjacently to those items. Otherwise, the cut-off item is inserted into a random position in the target schedule. Using this operator, the algorithm allows schedules with different sizes for pickers to decrease makespan. Figure 5.7 provide an example of this relocation operator.

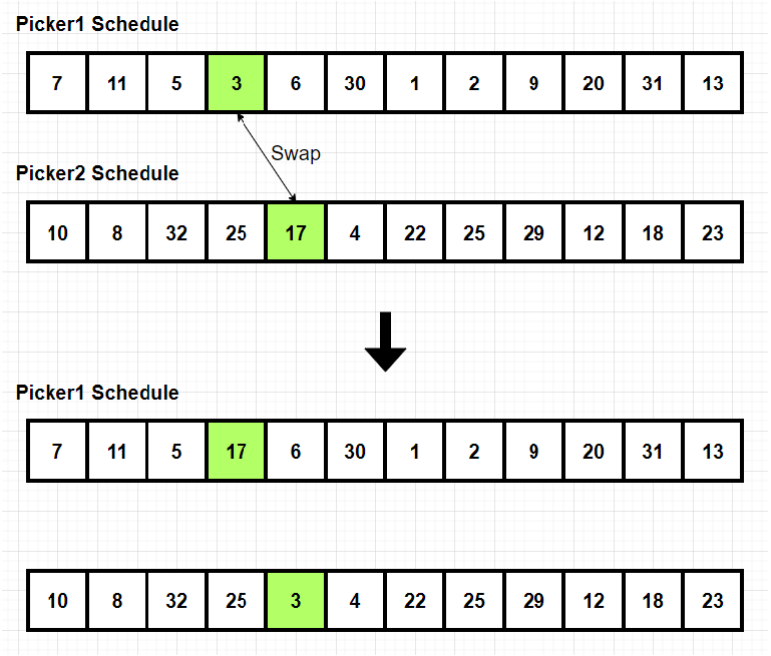


Figure 5.5: An example of how the crossover operator adjusts two pickers' schedules

To improve the performance of the three operators, we also test applying strategies from two other metaheuristics SA (Simulated Annealing) and TS (Tabu Search). The algorithms for applying those strategies are shown below.

To test the strategy in SA, we use two parameters T_{max} and T_{min}, which represent the maximum and minimum temperatures, respectively. After starting (re-starting) using the randomly generated schedules, a temperature value T_t at iteration t resets to T_{max} and decrease iteratively. T_{min} is the lower bound of T_t. The equation below shows how T_t changes at each iteration. In the equation, $\theta \in (0, 1)$ is a constant value that controls the decreasing process of T_t.

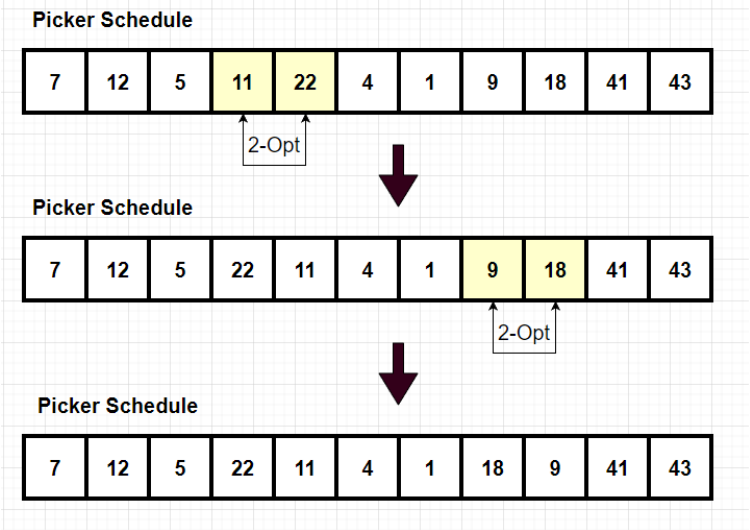


Figure 5.6: An example of how the 2-opt operator adjusts a picker's schedule

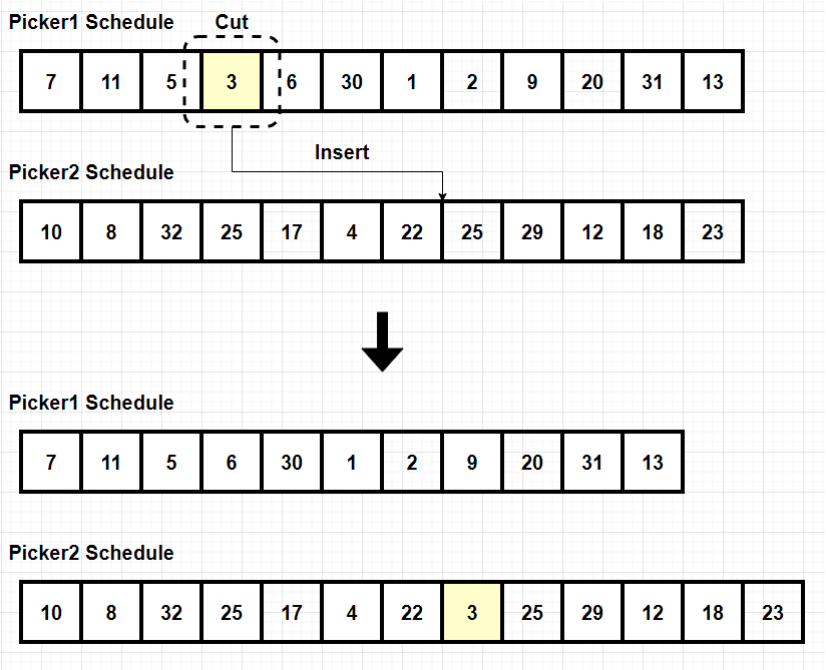


Figure 5.7: An example of how the relocation operator adjusts two pickers' schedules

$$T_t = \theta * T_{t-1} \tag{5.5}$$

After completing iteration $t-1$, for the existing schedules of all pickers S_P^{t-1}, the algorithm can get the schedules of all transporters S_D^{t-1} with the shortest makespan m_{t-1}. At iteration t, a random operator is used to create new schedules of pickers S_P^t. Then we can get the shortest

makespan m_t with S_D^t. Here, like SA, we have a probability of accepting the S_P^t or not at each iteration. The probability equation at iteration t is shown below:

$$p_t = \begin{cases} 1, & \text{if } m_t < m_{t-1} \\ e^{\frac{m_{t-1}-m_t}{T_t}}, & \text{otherwise} \end{cases} \tag{5.6}$$

From the equation, the algorithm will always accept new schedules S_P^t if the corresponding shortest makespan m_t is smaller than the makespan m_{t-1} in the previous iteration. When $m_t \geq m_{t-1}$, if we accept S_P^t, at iteration $t+1$, the operators will operate S_P^t. Otherwise, the operators will operate S_P^{t-1}.

Algorithm SA strategy in Hetero-ACO

Initialize T_{min}, T_{max}, Re, θ
$t \leftarrow 0$
$T_t \leftarrow T_{max}$
$count \leftarrow 0$ (Non-improvement iteration count)
Generate random S_P^t and find corresponding S_D^t and m_t according section 5.3
While not stop:
    $t++$
    $count++$
    $T_t \leftarrow \theta * T_{t-1}$
    if $T_t < T_{min}$:
        $T_t \leftarrow T_{min}$:
    if $count > Re$:
        $S_P^t \leftarrow$ randomly generated S_P
        $count \leftarrow 0$, $T_t \leftarrow T_{max}$
    else:
        choose an operator σ
        $S_P^t \leftarrow \sigma(S_P^{t-1})$
        find corresponding S_D^t and m_t
        if $m_t < m_{t-1}$:
            $count \leftarrow 0$
        else:
            $count++$
            $U \leftarrow$ Uniform(0,1)
            if $U < e^{\frac{m_{t-1}-m_t}{T_t}}$:
                $S_P^t \leftarrow S_P^{t-1}$
    if exceed maximum running time:
        Stop algorithm

To test the strategy in TS, we use a fixed-length tabu list, and its length is denoted as Len_{TB}. The tabu list stores the item indices that prohibit the operators from selecting those indices in the tabu list. So this could prevent operators from making "cycle" moves on S_P, which may benefit our algorithm. Here, at iteration t, the tabu list is denoted as L^t_{TB}. After starting (re-starting) using the randomly generated schedules, the algorithm empties the tabu list. In our algorithm, after completing iteration $t-1$, the algorithm has the corresponding S_P^{t-1}, S_D^{t-1}, and m_{t-1}. At iteration t, the algorithm selects a random operator and uses that operator to create a partial neighborhood of S_P^{t-1}. We need to mention that the neighborhood size is denoted as Len_{NB}, and the operator cannot use the items indices in the tabu list to find the partial neighborhood. Then, the algorithm uses the methods discussed in section 5.3 to find the corresponding S_D and makespan for all S_P in the neighborhood. Then, the algorithm will rank those S_P based on the makespans, and S_P with the best makespan in the neighborhood will be S_P^t. After this, the algorithm adds the items indices used by the operator to create S_P^t from S_P^{t-1} into the tabu list. If the tabu list reaches the max length, then the oldest items indices are dropped and new ones are added.

5.4.3 Solutions from An Alternative MILP

In MCOP, if there are relatively large numbers of transporters, there is no need to consider those schedules for transporters since transporters are enough to guarantee the picker can place the picked item at the picking location. In that situation, we can focus on finding good schedules for all pickers. However, even if there are not that many transporters, those schedules for pickers may still provide some useful information. For example, in a pick wave, ten transporters are needed to guarantee the hand-off spots for all items are their picking locations. Then, if only nine transporters are available, the schedules for all pickers cooperating with nine transporters may share some similarities with the schedules for all pickers cooperating with ten transporters. Therefore, we develop an alternative MILP, which only involves pickers to finish the pick wave. We assume a transporter is always waiting at the picking location in this MILP.

The parameters and decision variables used in this MILP are similar to those we discussed in Chapter 4, so we will not explain them again but list them in Table 5.4 and Table 5.5.

Algorithm TS strategy in Hetero-ACO

Initialize Len_{TB}, Len_{NB}, Re
$t \leftarrow 0$
$tabu \leftarrow []$ (tabu list)
$count \leftarrow 0$ (Non-improvement iteration count)
Generate random S_P^t and find corresponding S_D^t and m_t according section 5.3
While not stop:
    $t++$
    $count++$
    if $count > Re$:
        $S_P^t \leftarrow$ randomly generated S_P
        $count \leftarrow 0$
    else:
        choose an operator σ
        find partial neighborhood with Len_{NB} elements using $\sigma(S_P^{t-1})$ and $tabu$
        for S_P in the partial neighborhood:
            find corresponding S_D and makespan
        rank all S_P based on the corresponding makespans
        $S_P^t \leftarrow S_P$ with shortest makespan
        $m_t \leftarrow$ the shortest makespan
        if $m_t < m_{t-1}$:
            $count \leftarrow 0$
        else:
            $count++$
        if length of $tabu$ equals Len_{TB}:
            drop the oldest items indices
        add new items indices to $tabu$
    if exceed maximum running time:
        Stop algorithm

Table 5.4: Summary of the parameters for the alternative MILP

Paramter Notation	Paramter Notation Description
I	The set of all items in the RPL
P	The set of all pickers
R_i	The picking node of item $i \in I$
R	The set of all picking nodes
V_p	The traveling speed of a picker $p \in P$
V_D	The traveling speed of transporters
$T_{p,i}^{pick}$	The time need for a picker $p \in P$ to pick an item $i \in I$
$T_{p,i}^{place}$	The time need for a picker $p \in P$ to place an item $i \in I$ on a transporter
Dp	The depot node
HN_p	The home node for a picker $p \in P$
HN	The set of all home nodes
N	The set of all nodes where $N = R \cup HN$
$N_{p,k}^{-}$	The set of nodes that could be visited immediately before node $k \in N$ for a picker $p \in P$
L_{k,k_1}	The distance between two nodes $k, k_1 \in N$

Table 5.5: Summary of the decision variables for the alternative MILP

Decision Variable	Decision Variable Description
m	The makespan of the pick wave
$t_{p,k}$	The time when a picker $p \in P$ reach node $k \in N$ to pick the related item
$Y_{p,k,k1}$	$y_{p,k,k1} = 1$ if a picker $p \in P$ move from node k to node $k1$ where $k, k1 \in N \; and \; k \neq k1$
$a_{p,i}$	$a_{p,i} = 1$ if picker p pick item $i \in I$

The mathematical formulation of the objective and constraints for this alternative MILP are shown as follows.

$$\text{Min } m \tag{5.7}$$

$$\text{s.t.} \quad m \geq t_{p,R_i} + T_{p,i}^{pick} + T_{p,i}^{place} + \frac{L_{R_i,Dp}}{V_D} - M(1 - Y_{p,R_i,HN_p}) \quad \forall p \in P, i \in I, \tag{5.8}$$

$$\sum_{p \in P} a_{p,i} = 1 \quad \forall i \in I, \tag{5.9}$$

$$\sum_{p \in P} \sum_{k \in N_{p,R_j}^{-}} Y_{p,k,R_j} = 1 \quad \forall j \in I, \tag{5.10}$$

$$\sum_{k \in N^-_{p,R_j}} Y_{p,k,R_j} = \sum_{k \in N/R_j} Y_{p,R_j,k} \quad \forall p \in P, j \in I, \tag{5.11}$$

$$\sum_{j \in I} Y_{p,HN_p,R_j} = 1 \quad \forall p \in P, \tag{5.12}$$

$$\sum_{i \in I} Y_{p,R_i,HN_p} = 1 \quad \forall p \in P, \tag{5.13}$$

$$a_{p,i} \leq \sum_{k \in N^-_{p,R_j}} Y_{p,k,R_j} \quad \forall p \in P, j \in I, \tag{5.14}$$

$$\begin{gathered} t_{p,R_j} \geq t_{p,R_i} + T^{pick}_{p,i} + T^{place}_{p,i} + \frac{L_{R_i,R_j}}{V_p} - M(1 - Y_{p,R_i,R_j}) \\ \forall p \in P, i \in I \cup 0, j \in I \cup -1, \end{gathered} \tag{5.15}$$

The objective function 5.7 is to minimize the makespan when all items are delivered to the depot. Constraints 5.8 state that the makespan is no earlier than the time an item reaches to depot. Constraints 5.9 indicate that each item can only be picked by one picker. Constraints 5.10 guarantee a picking location can only be visited once by only one picker. Constraints 5.11 ensure a picker enters a picking location and will leave that picking location. Constraints 5.12 and 5.13 ensure each picker starts from the home node and returns the home node at the end. Constraints 5.14 are use to create relations between decision variable $a_{p,i}$ and $Y_{p,k,k1}$. Constraints 5.15 states the time a picker moves from one node to another node.

5.5 Performance comparison

We first compare the performance of makespan among our Hetero-ACO with SA to adjust S_P, Hetero-ACO with TS to adjust S_P, and Hetero-ACO without any metaheuristic strategiy to adjust S_P. Table 5.6 shows our test settings. In this test, the RPL contains 50 items. The warehouse layout is the warehouse with 4 picking aisles and 2 cross aisles shown in Figure A.4 in Appendix A. We fix the number of pickers to 2 and let the number of transporters be 1, 2, or 3. We think this could reflect the situations where MCOP uses enough transporters or not enough transporters. The transporter payload capacities are set to 3, 5, and 10 items for Low, $Medium$, and $High$, respectively. Since in TS, the length of the tabu list may have a significant impact

on the performance, we test two tabu list lengths $Len_{TB} = 10$ or $Len_{TB} = 20$. Also, based on some small preliminary tests, we found that $gp_{max} = 10$, $sq_{max} = 15$ are good for both SA and TS strategies. For the other parameters in Hetero-ACO, we set $\alpha = 1$, $\beta = 1$, $\rho = 0.5$, $\tau_{init} = 2$, $\tau_{max} = 3$, $\tau_{min} = 0.5$, $Q_h = 10$, $Q_\tau = 1000$, $f = 10$, $Re = 4$, $dl = 3$, $T_{max} = 200$, $T_{min} = 20$, $\theta = 0.8$,$Len_{NB} = 20$. For each combination of these settings, we test 10 replications and the cut time is set to 600 seconds. For the other parameters related to MCOP like the traveling speed of entities, the picking time and placing time, and so on, those settings are the same as those values in Table 4.3.

Table 5.6: Summary of the test settings for comparing Hetero-ACO using different metaheuristic strategies

Setting	Value	Levels
Number of Transporters		1, 2, 3
Transporter Payload Capacity		*Low* (3 items), *Medium* (5 items), *High* (10 items)
Len_{TB}		10, 20
Number of PAs	4	
Number of CAs	2	
Size of RPL	50 items	
Number of Pickers	2	
α	1	
β	1	
ρ	0.5	
τ_{init}	2	
gp_{max}	10	
sd_{max}	15	
Q_h	10	
f	10	
dl	3	
Q_τ	1000	
τ_{min}	0.5	
τ_{max}	3	
Re	4	
T_{max}	200	
T_{min}	20	
θ	0.8	
Len_{NB}	20	
Replications of main settings combination	10	
Cut Time	600s	

Table 5.7 shows the comparison of the average makespan decrease across replications in each scenario in percentage (%). In this table, the values are the Hetero-ACO with SA or TS compared to Hetero-ACO without any metaheuristics. A negative value means using a metaheuristic strategy provides a smaller makespan than without a metaheuristic strategy. From the table, we can find using the strategy from SA has a little better performance when there are not enough transporters. The reason could be at each iteration, the partial neighborhood is not good, and for a S_P that is not good in the neighborhood, our TS strategy still needs to find the corresponding S_D. Also, from the table, we can find that when there are many transporters, the performance of using metaheuristic strategies or not using a metaheuristic strategy is close. This is because of the alternative MILP model since the intermediate solutions from that model play a decisive role in those situations.

Table 5.7: Comparison of the average makespan decrease across replications in each scenario in percentage (%)

Entity Combination	Hetero-ACO with SA			Hetero-ACO with TS					
				short list			long list		
	Low	Medium	High	Low	Medium	High	Low	Medium	High
2P1T	**-1.9**	**-2.2**	**-3.2**	**-0.5**	**-2.1**	**-1.3**	**-1.3**	**-2.0**	**-1.4**
2P2T	**-1.6**	**-1.2**	**-0.3**	**-0.6**	0	0	0	2.5	0.1
2P3T	**-0.2**	0	0.1	0.1	0.2	0.1	0	0.2	0.2

We also compare the performance of makespan between the Hetero-ACO with SA and a simple heuristic as the number of transporters increases from small to large. The heuristic is that when a transporter is assigned to a picker, it cannot collaborate with other pickers before it delivers the current assigned picker's items to the depot. After unloading the carried items at the depot, that transporter can be re-assigned to other pickers. In addition, the hand-off spot of an item in the heuristic is the picking location for that item. The assignment of transporters in the heuristic is determined using the following procedures: When a transporter d is available to be assigned, for a picker p that is not assigned a transporter, let t_p be the time that the picker gets the next item i in S_p. Then, the transporter will be assigned to the picker with the smallest t_p. If a picker gets the item, but no transporter is available. The picker will wait at the picking location. In this test, the S_P are obtained from the alternative MILP model discussed in section 5.4.3. If we denote t'_p be the moment in time the picker p get the last item j to fill up the assigned

transporter d, then the actual time interval of d occupied by p is $[t_k - \frac{L_{Dt,R_i}}{V_d}, t'_k + \frac{L_{R_j,Dt}}{V_d} + T_d^{drop}]$ where $t_k - \frac{L_{Dt,R_i}}{V_d}$ is the time d departs from the depot and $t'_k + \frac{L_{R_j,Dt}}{V_d} + T_d^{drop}$ is the time d finish unloading at the depot.

Table 5.8 shows the experiment settings in this comparison. The number of pickers is set to be 3 or 6. When using 3 pickers, we set RPL with 60 items, and we will test the makespan for the number of transporters ranging from 1 to 6. When using 6 pickers, we set RPL with 150 items, and we will test the makespan for the number of transporters ranging from 1 to 12. The transporter payload capacity is set to 5 items. The warehouses used in this test are the same as the experiments we discussed in section 4.5. Therefore, 6 warehouse layouts with 4 or 8 picking aisles and 0 to 2 cross aisles are used. In our warehouse settings, there are 48 picking locations. In this test, we let the picking locations used for the items in RPL be 10 or 48 to check how the two methods perform when the items in RPL are concentrated in several picking locations or are spread widely in the warehouse. For each combination of these settings, we test 30 replications, and the cut time is set to 300 seconds. For all other parameters related to MCOP or the Hetero-ACO, we use the same values in Table 4.3 and Table 5.6.

Table 5.8: Summary of the test settings for comparing Hetero-ACO with heuristic

Setting	Value	Levels
Number of Pickers		3, 6
Number of Transporters		1-6 (3 pickers), 1-12 (6 pickers)
Size of RPL		60 items (3 pickers), 150 items (6 pickers)
Number of PAs		4, 8
Number of CAs		0, 1, 2
Picking location used for RPL		10, 48
Transporter payload capacity	5 items	
Replications of main settings combination	30	
Cut Time	300s	

Figure 5.8 and Figure 5.9 show the comparison of average makespan across replications of each scenario for 3 and 6 pickers, respectively. The red bar is the makespan of the heuristic which we set to 100 percent, the blue bar is the Hetero-ACO applied on 10 picking locations, and the green bar is Hetero-ACO applied on 48 picking locations. From the figures, we can find that when the number of transporters is less than the number of pickers, our Hetero-ACO

always provides smaller makespans. When the number of transporters exceeds the number of pickers, the heuristic sometimes provides slightly smaller makespans. Also, Hetero-ACO applied on 48 picking locations usually provides smaller makespans than applied on 10 picking locations, which means our Hetero-ACO performs better when the items in RPL spread widely in the warehouse. This is because when the items in RPL are concentrated in several picking locations, pickers will spend a relatively long time at those picking locations. Therefore, instead of letting different transporters collaborate with a picker, it is efficient letting a single transporter collaborate with a picker until the transporter reaches the payload capacity.

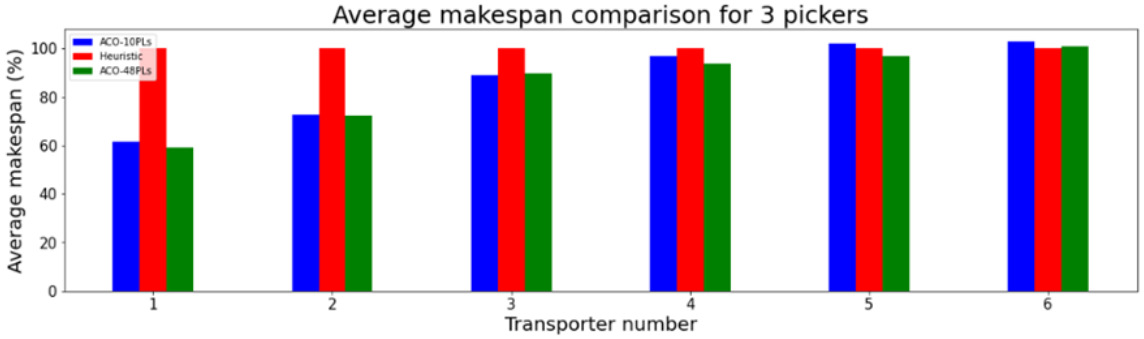


Figure 5.8: The comparison of average makespan across replications of each scenario for 3 pickers

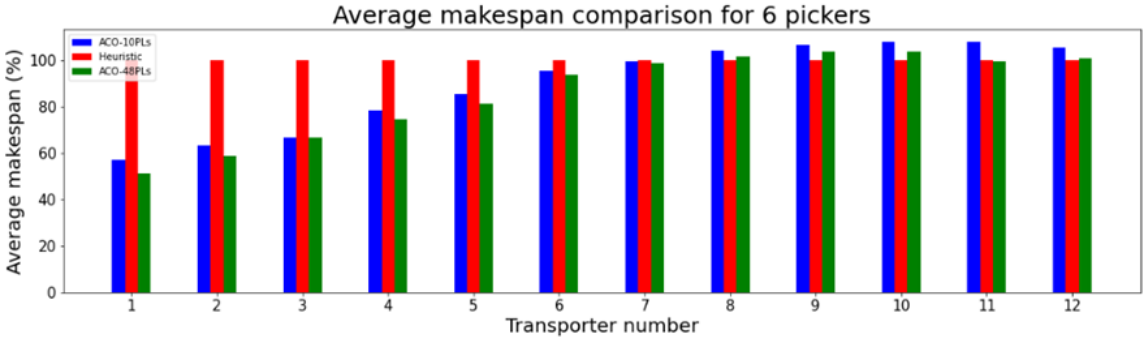


Figure 5.9: The comparison of average makespan across replications of each scenario for 3 pickers

In our experiments, we informally observed that when there are relatively more transporters, our algorithm's convergence rate is faster than when there are relatively fewer transporters. We think one reason for this observation is that our alternative MILP model can provide good schedules for pickers, and the final solutions (schedules for both pickers and transporters) of our Hetero-ACO are found in the neighborhood of the MILP solutions when there are more

transporters. However, when there are fewer transporters, the final solutions of our Hetero-ACO are found in the neighborhood of the randomly generated schedules for pickers. In this work, we don't make any analysis of the computational effort of our proposed algorithm. However, a detailed computational analysis is one avenue for future work.

In Hetero-ACO, we use random numbers to help find good schedules for all entities. However, we don't test the sensitivity of the random number seeds in our algorithm. The reasons are: 1) In the transporter schedule construction phase, we use multiple *Groups* and *Squads* to help determine the schedules for all transporters, which can reduce the influence caused by using random seeds in this phase. 2) In the picker schedule searching phase, the algorithm will generate multiple random solutions and use the operators multiple times for the neighborhood search. Using this setting, we can also reduce the influence caused by random seeds in this phase. But, when the problem size increases, if the computation time is limited, since the algorithm may not have enough time to do the random searching or construction, using a heuristic solution as the starting point could reduce the influence of using random seeds.

5.6 Summary

This chapter proposes a fast algorithm called Hetero-ACO to find schedules for all entities in a pick wave of MCOP. This algorithm applies strategies from basic ACO to construct schedules for all transporters. A dynamic local search is also integrated into the algorithm to improve the schedules of all transporters by trying different hand-off spots for some items. In addition, the algorithm applies three methods for searching the schedules of all pickers. The random schedule generation method, the schedule adjustment based on three operators, and the alternative MILP model ensure the algorithm is efficient under different picker and transporter combinations. We test applying two strategies from the metaheuristics SA and TS for the schedule adjustment method. Based on the test results, we find using the strategy from SA has slightly better performance in terms of makespan. Moreover, we compare our Hetero-ACO with a heuristic under different picker and transporter combinations. The results show Hetero-ACO outperforms the heuristic when the transporters are no more than the pickers. And Hetero-ACO

and the heuristic have similar performance when the transporters are more than the pickers. Experiment results also show that Hetero-ACO performs better when the items in RPL are spread widely in the warehouse than items in RPL are concentrated in several picking locations.

In this work, the Hetero-ACO can find similar or better schedules when the RPL is smaller than 150 items, the number of pickers is smaller than 6, and the number of transporters is smaller than 12. However, we don't test the robustness for larger problems. For larger problems, our algorithm can always provide feasible schedules for the entities. But, since a larger problem will also increase the search space of all solutions, the larger the problem, the more time is needed for the algorithm to find a relatively good solution. We tested several different parameter combinations to determine the final values used in this work. But we don't do detailed parameter tuning. In general, for problems with more transporters, we could use more Groups and Squads to find the schedules for all transporters since we can obtain relatively good schedules for all pickers from our alternative MILP. While for problems with fewer transporters, we could use less Groups and Squads to let the algorithm search for more schedules for all pickers from random generation and neighborhood search. Detailed parameter tuning analysis could be a good direction for future work to strengthen the algorithm.

Chapter 6

Analysis on The Impact of Variability in MCOP

In Chapter 4 and Chapter 5, our proposed MILP model and Hetero-ACO algorithm use constant values for the traveling speed of the entities, and the time needed for picking and placing an item. However, in an actual pick wave in the order picking, those values could vary due to many reasons. For example, an order picker who is less experienced with a storage rack may need much more time to search for the items on the shelves than an experienced picker [72]. Similarly, a picker who is not familiar with the warehouse layout and stored items may need more time to move from one picking location to another picking location. In addition, since in MCOP, pickers and transporters share the workspace, they often must adjust their speed to avoid collisions and congestion. So, because of the variabilities caused by those reasons, there may be unforeseen delays during the actual pick wave in MCOP. Since we cannot predict what will happen before the pick wave and the entities cannot avoid the variability during the pick wave, in the chapter, we will use our simulation model discussed in Chapter 3 to conduct experiments to check the impact of adding variabilities to some parameters in the pick wave of MCOP. We need to mention here that this work does not complete a thorough test to check the impact of all variabilities that may occur during the pick wave of MCOP. We only add variabilities to some parameters that we think those values are likely to vary during the pick wave. In section 6.1, we will discuss the experiment settings in detail. Section 6.2 analyzes the results and interprets the impact of adding variabilities on the makespan and wait time of pickers. Section 6.3 summarizes our findings.

6.1 Experiment Settings

In our experiments, the warehouse layout is the warehouse with 4 picking aisles and 2 cross aisles shown in Figure A.4 in Appendix A. We tested 3 entity combinations: 1P1T (1 picker 2 transporters), 2P3T (2 pickers 3 transporters), and 5P8T (5 pickers 8 transporters). The 1P2T is used to test the situation when we have enough transporters to guarantee the picking location and the hand-off spot for each item are the same. 2P3T and 5P8T are used to test when we have relatively small numbers of transporters. The reason to choose these three combinations is we want to focus on the situations where the ratio of transporters to pickers is no more than 2. For situations where the ratio is more than 2, the impact of adding variabilities in a pick wave will be similar to the impact of adding variabilities to 1P2T. This conclusion is intuitive because we can always assign two different transporters for each picker if the ratio is greater than 2. In addition, the payload capacity of each transporter is set to 10 items. We also test 3 different RPL sizes in 9 cases: 50 items for cases 1 to 3, 100 items for cases 4 to 6, and 150 items for cases 7 to 9. The items in the RPL for each case are randomly generated.

To add variabilities in the pick wave, we apply 4 different settings: *Base, SpeedVary, LoadVary, AllVary*. In each setting, we use constant values or some distributions to represent 4 parameters: the picking time and placing time for pickers, and the traveling speed for both pickers and transporters. For the picking time, we will use 60 s/item or exponential distribution with $\lambda = 1/60$ seconds. For the placing time, we will use 30 s/item or exponential distribution with $\lambda = 1/30$ seconds. For the pickers' speed, we will use 1 m/s or a triangular distribution with 0.9 m/s, 1 m/s, and 1.1 m/s for min, mode, and max, respectively. For the transporters' speed, we will use 2 m/s or a triangular distribution with 1.8 m/s, 2 m/s, and 2.2 m/s for min, mode, and max, respectively. The details of the 4 settings are shown in Table 6.1. The reason we choose exponential distributions for the picking and placing time is that we want to add large variances to the pick wave. While the reason we choose triangular distributions for the speed is that we want to add small variances to the pick wave. Therefore, we can check how small and large variances impact the pick wave.

Therefore, we will have 108 unique experiment scenarios. Except those scenarios contain *Base* setting, we will run 200 replications for each scenario. Based on our preliminary result, 200 replications can provide a relatively precise result under reasonable computation time. For those scenarios containing *Base* setting, we will only run 1 replication since the pick wave is deterministic. The schedules for all entities are computed using our Hetero-ACO. The parameters we discussed in this chapter are summarized in Table 6.2. The parameters related to Hetero-ACO and MCOP that are not discussed in this section are set based on Table 4.3 and Table 5.6.

Table 6.1: The name and values of the 4 parameters used to add variabilities in pick wave

Setting Name	Picking Time (second/item)	Placing Time (second/item)	Picker Speed (meter/second)	Transporter Speed (meter/second)
Base	60	30	1	2
SpeedVary	60	30	Triang(0.9,1,1.1)	Triang(1.8,2,2.2)
LoadVary	Exp(1/60)	Exp(1/30)	1	2
AllVary	Exp(1/60)	Exp(1/30)	Triang(0.9,1,1.1)	Triang(1.8,2,2.2)

Table 6.2: Summary of the test settings for testing the impact of adding variabilities in pick wave

Setting	Value	Levels
Number of Pickers		1, 2, 5
Number of Transporters		2 (1 pickers), 3 (2 pickers), 8 (5 pickers)
Size of RPL		50 items (3 cases), 100 items (3 cases), 150 items (3 cases)
Variability Setting		*Base, SpeedVary, LoadVary, AllVary*
Transporter payload capacity	10 items	
Number of PAs	4	
Number of CAs	2	
Replications	200 (except *Base* setting)	

6.2 Analysis of The Results

In our experiment, for each replication, we will record the makespan, the average wait time of pickers at hand-off spots, and the maximum wait time of pickers at hand-off spots, which are

the same metrics we discussed in section 3.4.2. This section will show the results from our experiments and analyze how the added variabilities impact the metrics.

6.2.1 Impacts on The Makespan

For scenarios with 1P2T, we find that although adding variabilities increases the dispersion of the makespan, the average and median values of the makespan among each scenario are very close. Figure 6.1 shows the boxplot of the makespan for the scenarios with RPL sizes in Case1, Case4, and Case9 of 1P2T. The green triangle represents the average makespan, and the orange line represents the median value of the makespan. The boxplots for all cases are shown in Figure B.1 in Appendix B.

For scenarios with 2P3T and 5P8T, we find that when the variances are large, adding variabilities increases the dispersion of the makespan and the average and median values of the makespan. That is, for settings *LoadVary* and *AllVary* in all cases, more than half of the replications will result in a larger makespan compared to the makespan from setting *Base*. Figure 6.2 shows the boxplot of the makespan for the scenarios with RPL sizes in Case1, Case4, and Case9 of 2P3T. Figure B.2 in Appendix B shows the boxplots for all cases. Figure 6.3 shows the boxplot of the makespan for the scenarios with RPL sizes in Case1, Case4, and Case9 of 5P8T and Figure B.3 in Appendix B shows the boxplots for all cases. Also, since we are not using normal distributions, we will use Welch's t-test and Wilcoxon signed-rank test to statistically compare the mean and median makespan among settings *Base, LoadVary* and *AllVary*. Table 6.3 shows the p values of the hypotheses that the mean makespans from settings *LoadVary, AllVary* are equal to the makespan in setting *Base* for 1P2T, 2P3T, and 5P8T. From the table, we can reject the hypotheses for 2P3T and 5P8T, which means the mean makespans from settings *LoadVary, AllVary* are statistically different from the makespan in setting *Base*. Table 6.4 shows the p values of the hypotheses that the median makespans from settings *LoadVary, AllVary* are equal to the makespan in setting *Base* for 1P2T, 2P3T, and 5P8T. From the table, we can also reject the hypotheses for 2P3T and 5P8T.

Table 6.3: The p values of the hypotheses that the mean makespans from settings *LoadVary, AllVary* are equal to the makespan in setting *Base* for 1P2T, 2P3T, and 5P8T

	Hypothesis	Case1	Case2	Case3	Case4	Case5	Case6	Case7	Case8	Case9
1P2T	$\mu_{Base} = \mu_{AllVary}$	0.46	0.62	0.85	0.58	0.95	0.47	0.25	0.51	0.08
	$\mu_{Base} = \mu_{LoadVary}$	0.90	0.84	0.87	0.92	0.60	0.86	0.36	0.50	0.71
2P3T	$\mu_{Base} = \mu_{AllVary}$	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01
	$\mu_{Base} = \mu_{LoadVary}$	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01
5P8T	$\mu_{Base} = \mu_{AllVary}$	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01
	$\mu_{Base} = \mu_{LoadVary}$	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01

Table 6.4: The p values of the hypotheses that the median makespans from settings *LoadVary, AllVary* are equal to the makespan in setting *Base* for 1P2T, 2P3T, and 5P8T

	Hypothesis	Case1	Case2	Case3	Case4	Case5	Case6	Case7	Case8	Case9
1P2T	$M_{Base} = M_{AllVary}$	0.13	0.82	0.99	0.84	0.57	0.25	0.09	0.61	0.12
	$M_{Base} = M_{LoadVary}$	0.64	0.56	0.62	0.95	0.74	0.97	0.37	0.51	0.68
2P3T	$M_{Base} = M_{AllVary}$	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01
	$M_{Base} = M_{LoadVary}$	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01
5P8T	$M_{Base} = M_{AllVary}$	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01
	$M_{Base} = M_{LoadVary}$	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01

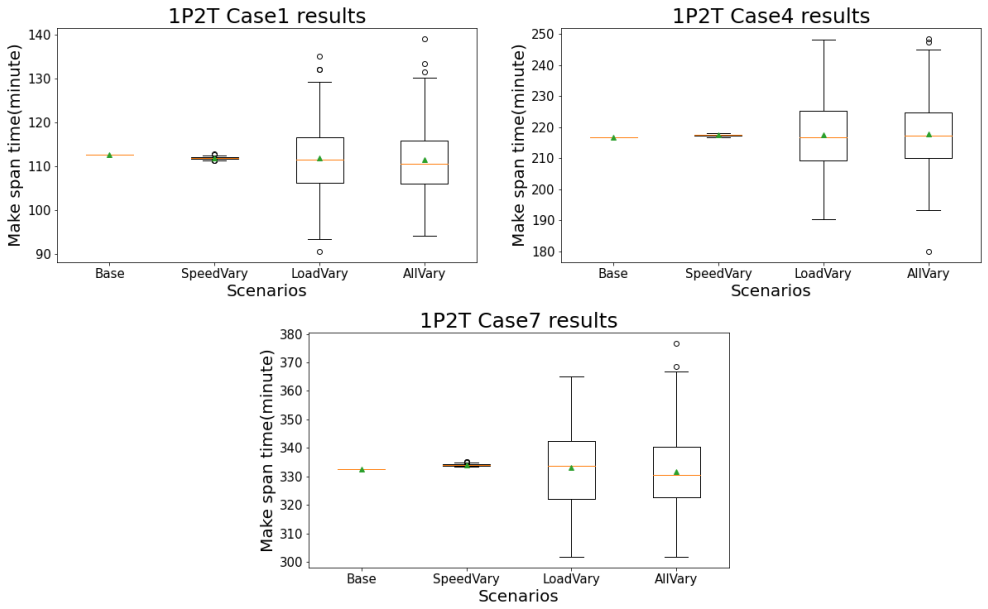


Figure 6.1: Boxplot of the makespan for the scenarios with RPL sizes in Case1, Case4, and Case9 of 1P2T

6.2.2 Impacts on The Wait Time of Pickers

For scenarios with 1P2T, the average wait time for all pickers is almost 0 for every scenario in all cases, which means for most replications in those scenarios, pickers can place the item

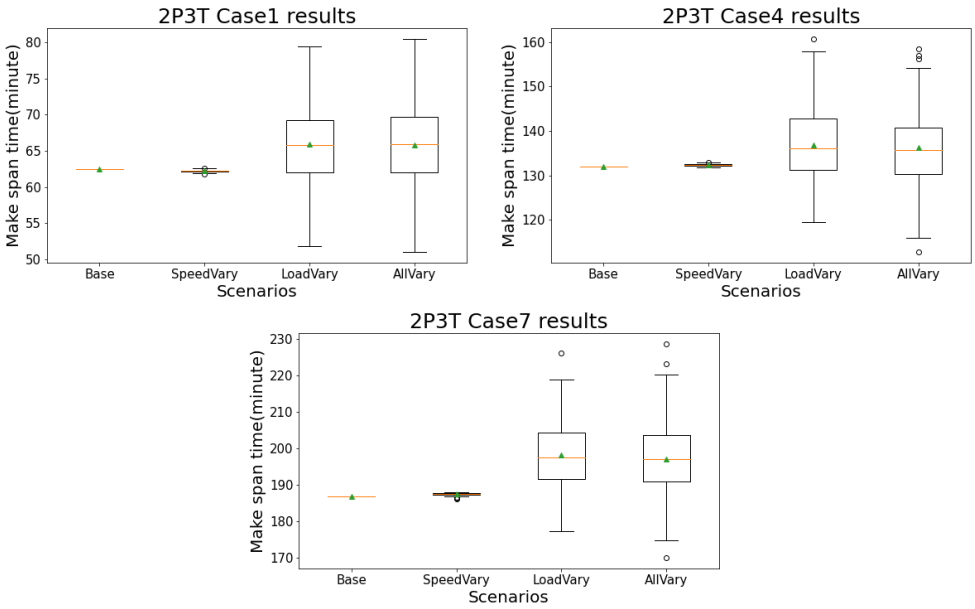


Figure 6.2: Boxplot of the makespan for the scenarios with RPL sizes in Case1, Case4, and Case9 of 2P3T.

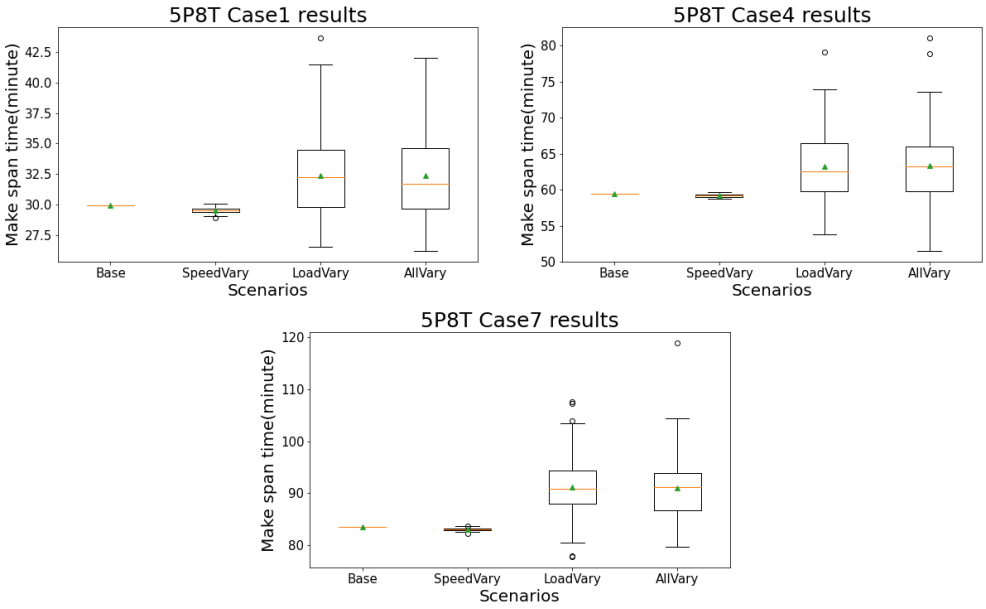


Figure 6.3: Boxplot of the makespan for the scenarios with RPL sizes in Case1, Case4, and Case9 of 5P8T.

immediately on the transporter when they reach the hand-off spots. For scenarios with 2P3T and 5P8T, we find that the average wait time of pickers for settings *Base* and *SpeedVary* are

almost identical in all cases. In addition, the average wait time of pickers for settings *LoadVary* and *AllVary* are almost identical and larger than the wait time from the other two settings in all cases. We also find similar results for the max wait time of pickers. Figure 6.4 and Figure 6.5 show the average and max wait times of pickers for all scenarios with 2P3T in all cases, respectively. Figure B.4 and Figure B.5 in Appendix B show the average and max wait times of pickers for all scenarios with 5P8T in all cases, respectively.

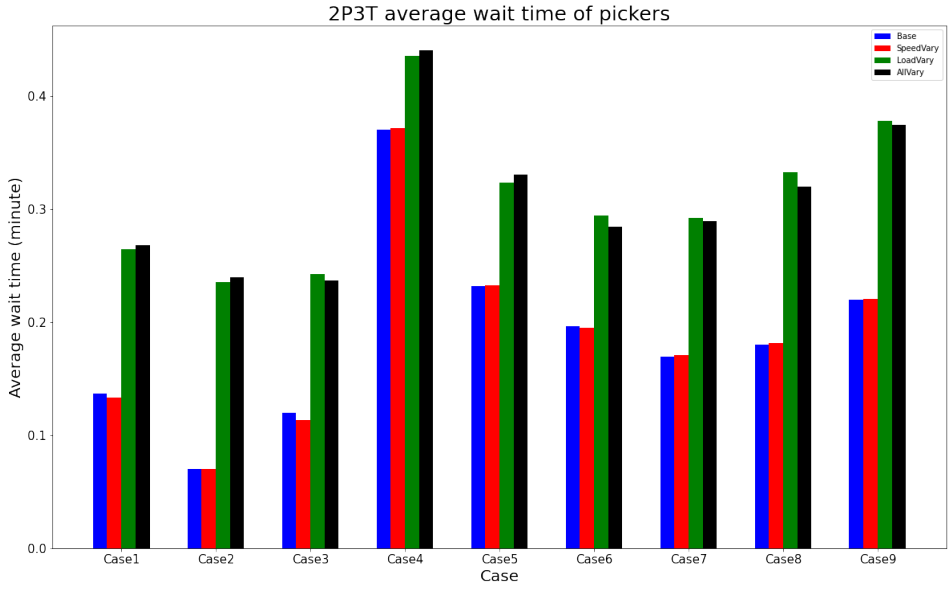


Figure 6.4: Average wait time of pickers for all scenarios with 2P3T in all cases

We also check the variance in wait time from settings *LoadVary* and *AllVary* versus the wait time from setting *Base* in all cases for scenarios with 2P3T and 5P8T. For each scenario with settings *LoadVary* and *AllVary*, all entities follow the same schedules in the 200 replications. Therefore, for placing a certain item, we can compute the variance of the wait time and compare that variance with the wait time when there is no variance (*Base* setting). Figure 6.6 and Figure 6.7 show the wait time variance versus wait time for scenarios with 2P3T and 5P8T, respectively. The figures show that the picker may need to wait to place an item even if the wait time in setting *Base* is zero. We can also find that there seems to be a trend that the higher the wait time to place an item is, the higher the potential variance will be.

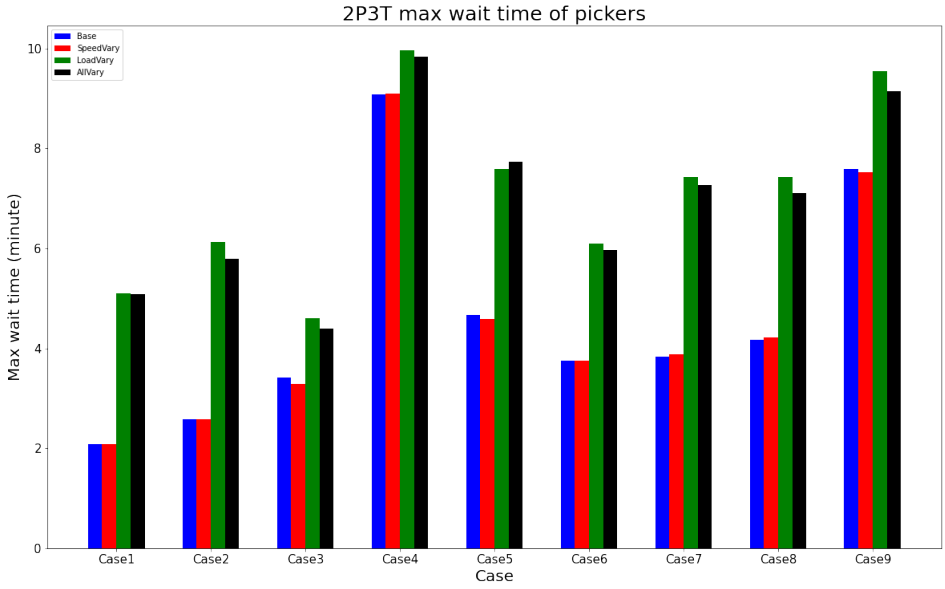


Figure 6.5: Max wait time of pickers for all scenarios with 2P3T in all cases

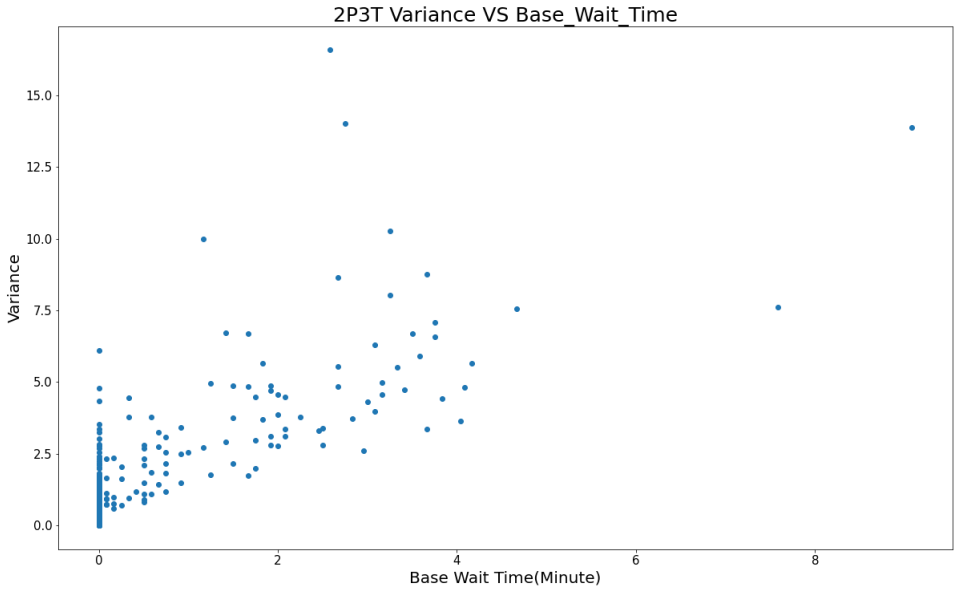


Figure 6.6: The variance in wait time from settings *LoadVary* and *AllVary* versus the wait time from setting *Base* in all cases for scenarios with 2P3T

6.3 Summary

In this chapter, we used our simulation model developed in Chapter 3 as a testbed and combined the Hetero-ACO we proposed in Chapter 5 to check the impact on the pick wave when

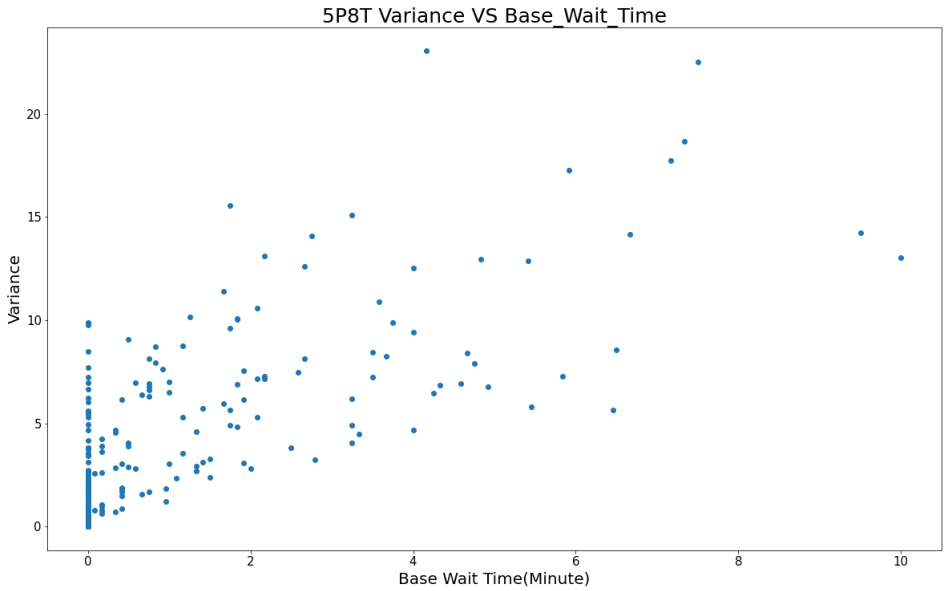


Figure 6.7: The variance in wait time from settings *LoadVary* and *AllVary* versus the wait time from setting *Base* in all cases for scenarios with 5P8T

adding variabilities to several parameters in MCOP. The main focus was trying to check how the variabilities influence the makespan and the pickers' wait time. Our experiments investigated certain distributions of pickers' picking and placing times and entities' traveling speeds. The results showed that when the ratio of transporters to pickers is smaller than 2, adding variabilities may lead to a situation where the chance the makespan is larger than the makespan with no variabilities is greater than 50 %. In addition, we found that as the RPL size increases, that chance also rises. Our experiments also observed that the pickers' average and max wait time would increase when the picking and placing times have high variance distributions. And if a picker needs to wait to place an item based on the schedule from the Hetero-ACO, the longer the wait time, the higher the variance of wait time the picker may face when adding variabilities. Therefore, from our results, sophisticated pickers are important to reduce the negative impacts of the variabilities that may occur in the pick wave of MCOP. However, our tests only involve several distributions for several parameters, and the variabilities in the real-world pick wave could differ from our test settings. Since our objective is to make a brief analysis,

our results can still provide useful insights into the impacts on the pick wave of MCOP caused by the variabilities.

For our simulation model, besides the application discussed in this chapter, there could be more use cases, for example:

1. In our algorithms for finding the schedules for all entities, battery usage is not considered. But in our simulation model, we could add the battery usage and evaluate how different charging policies (changing conditions, changing durations, etc.) impact the schedules obtained from models that don't consider battery usage.

2. Since the simulation model can act as a data provider, we could integrate the optimization model into the simulation model. In actual order picking, pickers or transporters may be temporarily unavailable (shutdown, injured, etc.) The simulation model can easily add features to represent the situation. If that happens in the simulation, the simulation model can execute the optimization model and re-schedule the remaining pickers and transporters.

Chapter 7

Conclusions and Future Research

This dissertation studied methods for cooperative order picking to minimize the makespan of a pick wave. According to our study on the current body of literature and the industrial applications, the outcome shows three research gaps that need to be addressed in cooperative order picking. First, the strategy of the cooperation logic is not well studied. Consequently, we proposed a multi-entity cooperative order picking strategy that provides different cooperation logic based on the characteristics of the pick wave. That is, the locations for placing the picked items are determined based on the storage locations of the items, the warehouse layout, and the number and ratio of different types of entities. So, when there are fewer pickers in the pick wave, pickers will tend to place the items at the picking location. When there are more pickers in the pick wave, pickers will tend to place the items at some alternative locations. In addition, we developed a data-driven and data-generated simulation model to represent the pick wave of MCOP. By applying a customized data structure, our simulation model can be a testbed for performance evaluation of pick wave in MCOP under different warehouse layouts, storage policies, and batching and routing methods.

The second gap is how to effectively and efficiently make the operational decisions, including the workload and routes for all entities of the pick wave in MCOP. Therefore, we proposed an exact method based on MILP that uses mathematical formulae to represent the whole operation of the pick wave and solved the model using a commercial solver. Experiment results show our model can provide a smaller makespan than the one from the existing work when there are fewer transporters with a small capacity. This performance improvement is mainly achieved by letting pickers choose alternative locations to place items to reduce the wait time

of pickers. Due to the limitation of the size of the pick wave the exact model can handle in practice, we also demonstrate a fast algorithm to find the operational decisions by integrating different optimization methods. We combined three methods to find the operational decisions for all pickers, including random generation, adjustment by three searching operators, and usage of intermediate solutions from a small alternative MILP model. We also tested applying two strategies from Simulated Annealing and Tabu Search to improve the adjustment methods for the decisions of all pickers. Besides that, an Ant Colony Optimization based strategy is used to construct operational decisions for all transporters. Lastly, a dynamic local search is integrated to improve the decisions by searching alternative locations for placing some items.

The last gap is none of the current literature considers the impact of variabilities on the pick wave of MCOP. We used our proposed simulation model as the testbed and the fast algorithm to find operational decisions for all entities to check the impact on the pick wave when the pickers' picking and placing times and entities' traveling speeds have certain distributions. Our experiment results reveal some of our tested distributions may increase the expected makespan and may lead to a situation where the chance the makespan is larger than the makespan with no variabilities is greater than 50 % when the ratio of transporters to pickers is smaller than 2. We also observed from our experiments that the average and max wait times of pickers increase when the picking and placing times have high variance distributions. Therefore, sophisticated pickers are essential to reduce the negative impacts of the variabilities in MCOP.

This dissertation aimed to provide a comprehensive study on cooperative order picking. We believe our proposed algorithms and models can provide insights and help warehouse owners find the improvement in upgrading manual order pickings with assistive robots and decide the best combination of pickers and robots. In future studies, we can extend our work in several directions, including:

1. Making a detailed computational analysis of our Hetero-ACO algorithm. Such as collecting data to analyze the "quality" of the solution associated with each phase (constructing each *Squad* and the local search), analyzing how Simulated Annealing contributes to the algorithm in the neighborhood search, analyzing how the alternative MILP contribute to

the algorithm at different ratio of transporters to pickers, and analyzing whether the alternative MILP can still accelerate the convergence when the size of the problem is larger (PRL size larger than 150 items).

2. Analyzing the robustness of the Hetero-ACO under different problem sizes. This can help the determination of the effectiveness of the algorithm for a certain size of a problem. Furthermore, detailed parameter tuning can indicate each parameter's effect and help select the parameter settings.

3. Computing dynamic hand-off spots in continuous space during the pick wave in MCOP. In a real pick wave, based on the locations of pickers and transporters, using dynamic hand-off spots can further reduce the wait time of pickers to improve the makespan. Also, using dynamic hand-off spots may let the paired picker and transporter reach a hand-off spot at the same time, which is more realistic since pickers and transporters can move toward their assigned partner when they "see" it.

4. Applying additional strategies to handle new coming orders. During a pick wave, new customer orders may arrive. If the current pick wave can be intervened and re-scheduled to include the new orders, this kind of extension may further improve the efficiency of the order picking and customer satisfaction.

5. Extending the problem also to facilitate re-stocking activities. So, robots bring items from the depot back into the warehouse, and pickers return items to the shelves. This could help warehouses get rid of individual re-stocking activities and leave more time for order picking to increase the daily throughput.

Bibliography

[1] Beth Gutelius and Nik Theodore, The Future of Warehouse Work: Technological Change in the U.S. Logistics Industry. UC Berkeley Center for Labor Research and Education and Working Partnerships USA, 2019.

[2] René de Koster, Tho Le-Duc, and Kees Jan Roodbergen, Design and control of warehouse order picking: A literature review. European Journal of Operational Research, 182(2):481-501, 2007.

[3] James A Tompkins, John A White, Yavuz A Bozer, and J. M. A. Tanchoco, Facilities Planning (3rd ed.). John Wiley and Sons, 2003.

[4] Ivona Bajor, Antonia Glasnović, and Josip Habazin, Order Picking Process in Warehouse: Case Study of Dairy Industry in Croatia. PROMET - Traffic&Transportation, 29(1):57-65, 2017.

[5] Günter Ullrich, Automated Guided Vehicle Systems – A Primer with Practical Applications. Springer, 2015.

[6] Amelia Smith, Amazon Reveals Robot Army in Its Warehouses, 2014.https://www.newsweek.com/amazon-reveals-robot-army-its-warehouses-288124

[7] ShaSha Lai, JD.Com Opens Doors to Unmanned Warehouse for First Time, 2018. https://www.yicaiglobal.com/news/jdcom-opens-doors-to-unmanned-warehouse-for-first-time

[8] Alan C Schultz and Michael A Goodrich, Human-Robot Interaction: A Survey. Foundations and Trends® in Human-Computer Interaction, 1(3):203-275, 2007.

[9] 6 RIVER SYSTEMS, CROCS POP-UP WAREHOUSE - READY IN TIME FOR PEAK, 2020. https://6river.com/case-study/crocs-pops-up-warehouse-for-peak/

[10] Maximilian Löffler, Nils Boysen, and Michael Schneider, Picker Routing in AGV-Assisted Order Picking Systems. INFORMS Journal on Computing, 34(1):440-462, 2022.

[11] Donald H. Ratliff and Arnon S Rosenthal, Order-Picking in a Rectangular Warehouse: A Solvable Case of the Traveling Salesman Problem. Operations Research, 31(3):507-521, 1983.

[12] Hung-Yu Lee and Chase C Murray, Robotics in order picking: evaluating warehouse layouts for pick, place, and transport vehicle routing systems. International Journal of Production Research, 57(18):5821-5841, 2018.

[13] Giulia Pugliese, Xiaochen Chou, Dominic Loske, Matthias Klumpp, and Roberto Montemanni, AMR-Assisted Order Picking: Models for Picker-to-Parts Systems in a Two-Blocks Warehouse. Algorithms, 15(11), 2022.

[14] Marco Dorigo, Optimization, Learning and Natural Algorithms. Ph.D. Thesis, Politecnico di Milano, 1992.

[15] Fred Glover, Tabu Search—Part I. ORSA Journal on Computing, 1(3): 190-206, 1989.

[16] Johan Oscar Ong and Don Thomas Joseph, A Review of Order Picking Improvement Methods. J@Ti Undip: Jurnal Teknik Industri, 9(3):135-138, 2014.

[17] Nils Boysen, René de Koster, and Felix Weidinger, Warehousing in the e-commerce era: A survey. European Journal of Operational Research, 277(2):396-411, 2019.

[18] Semih Önüt, Umut R. Tuzkaya, and Bilgehan Dog˘ac, A particle swarm optimization algorithm for the multiple-level warehouse layout design problem. Computers & Industrial Engineering, 54(4): 783-799, 2008.

[19] Abbas Ahmadi, Mir Saman Pishvaee, and Mohammad Reza Akbari Jokar, A survey on multi-floor facility layout problems. Computers & Industrial Engineering, 107: 158-170, 2017.

[20] Vassilios Vrysagotis and Patapios Alexios Kontis, Warehouse layout problems : Types of problems and solution algorithms. Journal of Computations & Modelling, 1(1):131-152, 2011.

[21] Shahab Derhami, Jeffrey S. Smith, and Kevin R. Gue, Optimising space utilisation in block stacking warehouses. International Journal of Production Research, 55(21):6436-6452, 2017.

[22] Kevin R. Gue and Russell D. Meller, Aisle Configurations for Unit-Load Warehouses. IIE Transactions, 41(3):171-182, 2009.

[23] B. Rouwenhorst, B. Reuter, V. Stockrahm, G.J. van Houtum, R.J. Mantel, and W.H.M. Zijm, Warehouse design and control: Framework and literature review. European Journal of Operational Research, 122(3): 515-533, 2000.

[24] Rong Yuan, Tolga Cezik, and Stephen C. Graves, Stowage decisions in multi-zone storage systems. International Journal of Production Research, 56(1-2): 333-343, 2018.

[25] Yipeng Zhang, Correlated Storage Assignment Strategy to reduce Travel Distance in Order Picking. IFAC-PapersOnLine, 49(2): 30-35, 2016.

[26] Mohammadnaser Ansari and Jeffrey S. Smith, Gravity Clustering: A Correlated Storage Location Assignment Problem Approach. Paper presented at the 2020 Winter Simulation Conference, 2020.

[27] D.R. Gibson and G.P. Sharp, Order batching procedures. European Journal of Operational Research, 58: 57-67, 1992.

[28] C-H Pan and S-Y Liu, A comparative study of order batching algorithms. Omega International Journal of Management Science, 23: 691-700, 1995.

[29] M. B. Rosenwein, A comparison of heuristics for the problem of batching orders for warehouse selection. International Journal of Production Research, 34: 657-664, 1996.

[30] G. Clarke and W. Wright, Scheduling of vehicles from a central depot to a number of delivery points. Operations Research, 12: 568-581, 1964.

[31] E.A. Elsayed and I.O. Unal, Order batching algorithms and travel-time estimation for automated storage/retrieval systems. International Journal of Production Research, 27: 1097-1114, 1989.

[32] De Koster, M.B.M., Van der Poort, and M. Wolters, Efficient orderbatching methods in warehouses. International Journal of Production Research, 37(7): 1479-1504, 1999.

[33] Jun Zhang, Xuping Wang, and Kai Huang, Integrated on-line scheduling of order batching and delivery under B2C e-commerce. Computers & Industrial Engineering, 94: 280-289, 2016.

[34] Y.-C. Ho and Y.-Y. Tseng, A study on order-batching methods of order-picking in a distribution centre with two cross-aisles. International Journal of Production Research, 44(17): 3391-3417, 2006.

[35] Ling-Feng Hsieh and Yi-Chen Huang, New batch construction heuristics to optimise the performance of order picking systems. International Journal of Production Economics, 131(2): 618-630, 2011.

[36] Xiaochun Feng and Xiangpei Hu, A Heuristic Solution Approach to Order Batching and Sequencing for Manual Picking and Packing Lines considering Fatiguing Effect. Scientific Programming, 2021.

[37] Wolfgang Banzhaf, Peter Nordin, Robert E. Keller, and Frank D. Francone, Genetic programming: an introduction on the automatic evolution of computer programs and its applications. Morgan Kaufmann Publishers Inc., 1998.

[38] Sebastian Henn and Gerhard Wäscher, Tabu search heuristics for the order batching problem in manual order picking systems. European Journal of Operational Research, 222(3): 484-494, 2012.

[39] T. S. Vaughan, The effect of warehouse cross aisles on order picking efficiency. International Journal of Production Research, 37(4): 881-897, 1999.

[40] Kees Jan Roodbergen and René de Koster, Routing methods for warehouses with multiple cross aisles. International Journal of Production Research, 39(9): 1865-1883, 2001.

[41] Goran Dukić and Čedomir Oluić, Order-Picking Routing Policies: Simple Heuristics, Advanced Heuristics or Optimal Algorithm. Journal of Mechanical Engineering, 50(11):530-535, 2004.

[42] Fred Glover, Future paths for integer programming and links to artificial intelligence. Computers & Operations Research, 13(5): 533-549, 1986.

[43] Kenneth Sörensen and Fred Glover, Metaheuristics. Encyclopedia of Operations Research and Management Science, pages 960–970. Springer, New York, 2013

[44] C.-Y. Tsai, J. J. H. Liou, and T.-M. Huang, Using a multiple-GA method to solve the batch picking problem: considering travel distance and order due time. International Journal of Production Research, 46(22): 6533-6555, 2008.

[45] Osman Kulak, Yusuf Sahin, and Mustafa Egemen Taner, Joint order batching and picker routing in single and multiple-cross-aisle warehouses using cluster-based tabu search algorithms. Flexible Services and Manufacturing Journal, 24(1): 52-80, 2011.

[46] Ilhan Or, Traveling Salesman-Type Combinatorial Problems and Their Relation to the Logistics of Regional Blood Banking. Ph.D. Thesis, Northwestern University, Illinois, 1967.

[47] Martin Grunow, Hans-Otto Günther, Martin Schleusener, and Ihsan Onur Yilmaz. Operations planning for collect-and-place machines in PCB assembly. Computers & Industrial Engineering, 47(4): 409-429, 2004.

[48] Tzu-Li Chen, Chen-Yang Cheng, Yin-Yann Chen, and Li-Kai Chan, An efficient hybrid algorithm for integrated order batching, sequencing and routing problem. International Journal of Production Economics, 159: 158-167, 2015.

[49] Chen-Yang Cheng, Yin-Yann Chen, Tzu-Li Chen, and John Jung-Woon Yoo, Using a hybrid approach based on the particle swarm optimization and ant colony optimization to solve a joint order batching and picker routing problem. International Journal of Production Economics, 170: 805-814, 2015.

[50] Vaggelis Giannikas, Wenrong Lu, and Duncan McFarlane, Interventionist Routing Algorithm for single-block warehouse: Application and Complexity, 2015.

[51] Wenrong Lu, Duncan McFarlane, Vaggelis Giannikas, and Quan Zhang, An algorithm for dynamic order-picking in warehouse operations. European Journal of Operational Research, 248(1): 107-122, 2016.

[52] Haitao Xu, Pan Pu, and Feng Duan, Dynamic Vehicle Routing Problems with Enhanced Ant Colony Optimization. Discrete Dynamics in Nature and Society, 2018:1-13, 2018.

[53] Kaveh Azadeh, René de Koster, and Debjit Roy, Robotized Warehouse Systems: Developments and Research Opportunities. SSRN Electronic Journal, 2017.

[54] Ivan Žulj, Hagen Salewski, Dominik Goeke, and Michael Schneider, Order batching and batch sequencing in an AMR-assisted picker-to-parts system. European Journal of Operational Research, 298(1):182-201, 2022.

[55] Takayoshi Yokota, Min-Max-Strategy-Based Optimum Co-Operative Picking with AGVs in Warehouse. 58th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE), Hiroshima, Japan, 2019.

[56] Sven Winkelhaus, Christoph H. Glock, and Eric H. Grosse, Job Satisfaction: An Explorative Study on Work Characteris- tics Changes of Employees in Intralogistics 4.0. Journal of Business Logistics, 43(3): 343-367, 2022.

[57] Alexandros Pasparakis, Jelle de Vries, and René de Koster, Assessing the impact of human–robot collaborative order picking systems on warehouse workers. International Journal of Production Research, 2023.

[58] Konstantinos Mykoniatis and Anastasia Angelopoulou, A modeling framework for the application of multi-paradigm simulation methods. Simulation, 96(1): 55-73, 2019.

[59] Donghuang Li, Comprehensive Methodology for the Design Configuration and Operational Control of Shuttle-based Storage and Retrieval Systems. Ph.D. Dissertation, 2022.

[60] Michel Gourgand and Patrick Kellert, An object-oriented methodology for manufacturing system modelling. In Summer Computer Simulation Conference, 1123–1128. Reno, Nevada, USA, 1992.

[61] Michelle Chabrol, Michel Gourgand, and Sophie Rodier, A modeling methodology and its application to the design of decision-making aid tools dedicated to the hospital systems. In RCIS'08:International Conference on Research Challenges in Information, 2008.

[62] Mohammadnaser Ansari and Jeffrey S. Smith, Warehouse Operations Data Structure (WODS): A data structure developed for warehouse operations modeling. Computers & Industrial Engineering, 112: 11-19, 2017.

[63] Yavuz A. Bozer and Francisco J. Aldarondo, A simulation-based comparison of two goods-to-person order picking systems in an online retail setting. International Journal of Production Research, 56(11): 3838-3858, 2018.

[64] Takumi Kato and Ryota Kamoshida, Multi-Agent Simulation Environment for Logistics Warehouse Design Based on Self-Contained Agents. Applied Sciences, 10(21), 2020.

[65] Pasquale Legato, Massimiliano Matteucci, and Rina Mary Mazza, Event-based modeling and simulation for optimizing order picking. 21st International Conference on Modelling and Applied Simulation, 2022.

[66] Behnam Bahrami, El-Houssaine Aghezzaf, and Veronique Limere, Using Simulation to Analyze Picker Blocking in Manual Order Picking Systems. Procedia Manufacturing, 11: 1798-1808, 2017.

[67] M. Klodawski, R. Jachimowski, I. Jacyna-Golda, and M. Izdebski, Simulation Analysis of Order Picking Efficiency with Congestion Situations. International Journal of Simulation Modelling, 17(3): 431-443, 2018.

[68] Sven Winkelhaus, Minqi Zhang, Eric H. Grosse, and Christoph H. Glock, Hybrid order picking: A simulation model of a joint manual and autonomous order picking system. Computers & Industrial Engineering, 167, 2022.

[69] Simio LLC. Simio Reference Guide. 2021. https://www.simio.com/

[70] A.Talhi, JC. Huet, V. Fortineau, and S. Lamouri, Towards a Cloud Manufacturing systems modeling methodology. IFAC-PapersOnLine, 48(3): 288-293, 2015.

[71] Jeffrey S. Smith, Survey on the use of simulation for manufacturing system design and operation. Journal of manufacturing systems, 22(2): 157-171, 2003.

[72] Eric H. Grosse and Christoph H. Glock, An experimental investigation of learning effects in order picking systems. Journal of Manufacturing Technology Management, 24(6): 850-872, 2013.

[73] Eric H. Grosse, Christoph H. Glock, Mohamad Y. Jaber, and W. Patrick Neumann, Incorporating human factors in order picking planning models: framework and research opportunities. International Journal of Production Research, 53(3): 695-717, 2014.

[74] Melonee Wise, Michael Ferguson, Daniel King, Eric Diehr and David Dymesich, Fetch & Freight: Standard Platforms for Service Robot Applications. The IJCAI 2016 Workshop on Autonomous Mobile Service Robots, New York City, NY, USA, 2016.

[75] Mengfei Yu and René de Koster, Enhancing performance in order picking processes by dynamic storage systems. International Journal of Production Research, 48(16): 4785-4806, 2009.

[76] Gurobi Optimization, Inc. Gurobi Optimizer Reference Manual. 2021. https://www.gurobi.com/

[77] G. A. Croes, A Method for Solving Traveling-Salesman Problems. Operations Research, 6(6): 791-812, 1958.

Appendices

Appendix A

Additional Materials for MILP Model Experiments

In this part, we present the additional materials for the experiments comparing the performance of the two MILP models in section 4.5.

A.1 The remaining 5 warehouse layouts used in the experiments

Figure A.1 to A.5 show the other 5 warehouse layouts used in the experiments. Table A.1 to A.3 shows the experiments results when RPL has 10 items.

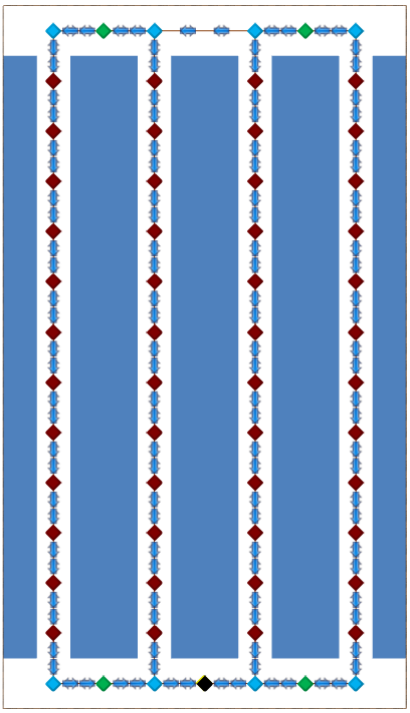


Figure A.1: The layout of a warehouse with 4 picking aisles and 0 cross aisles.

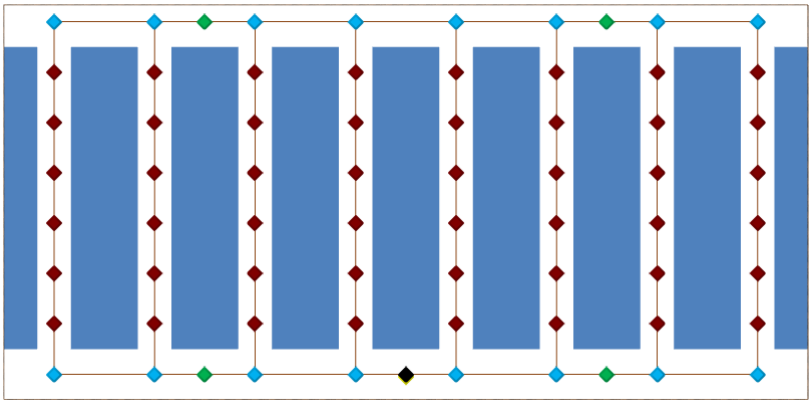


Figure A.2: The layout of a warehouse with 8 picking aisles and 0 cross aisles.

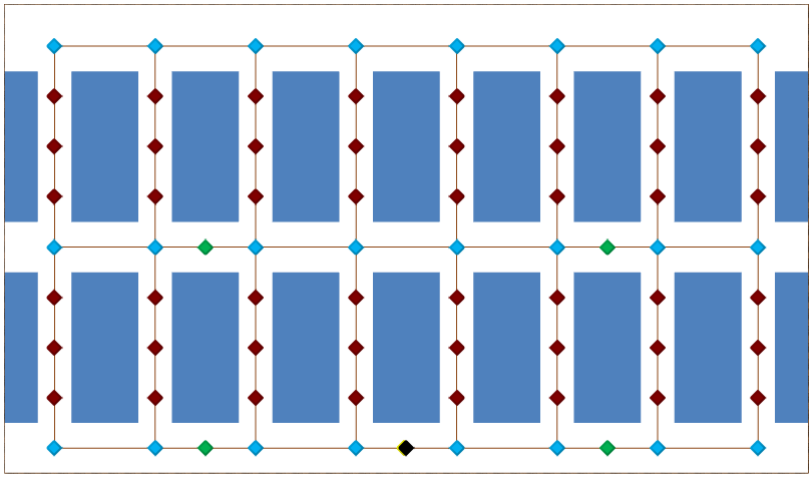


Figure A.3: The layout of a warehouse with 8 picking aisles and 1 cross aisle.

Table A.1: When the RPL has 10 items, the average percentage (%) increase of the makespan obtained from our model compared to the value obtained from the baseline model in different scenarios.

Entity Combination	4PAs			8PAs		
	Low	Medium	High	Low	Medium	High
1P1T	**-4.9**	**-3.4**	1.0	**-7.1**	**-3.0**	0.8
1P2T	1.2	0.9	0.3	0.2	0.6	0.5
1P3T	0.2	0.4	0.4	0.5	0.7	0.2
2P1T	**-12.2**	**-7.4**	2.6	**-14.8**	**-7.1**	2.0
2P2T	5.2	4.4	3.2	4.3	3.9	3.7
2P3T	4.1	3.3	3.6	2.5	3.8	2.2
3P1T	**-19.5**	**-16.2**	0.0	**-16.8**	**-13.9**	0.3
3P2T	**-5.7**	4.1	3.0	**-4.2**	3.5	4.0
3P3T	2.0	3.1	1.5	2.5	2.9	3.0

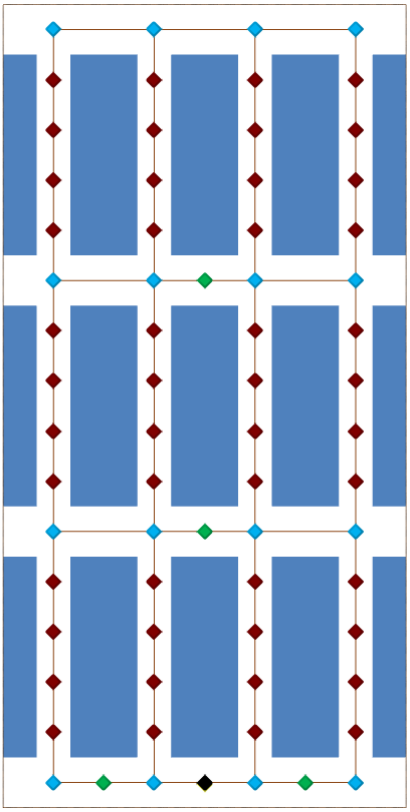


Figure A.4: The layout of a warehouse with 4 picking aisles and 2 cross aisles.

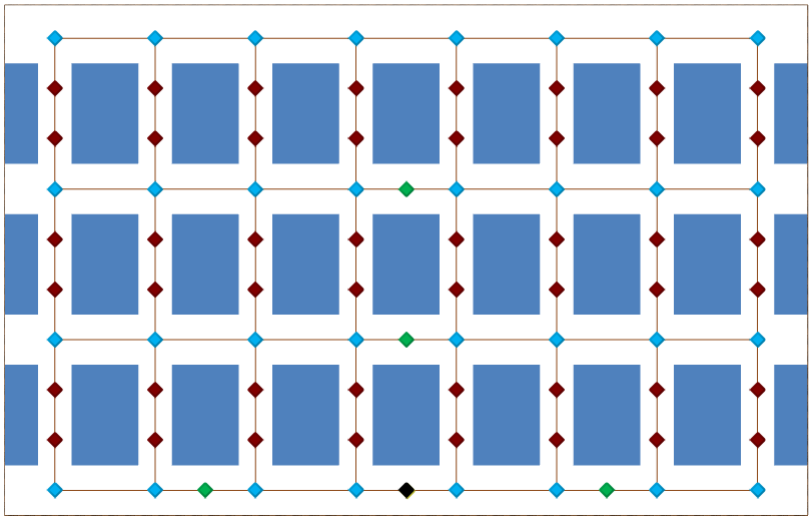


Figure A.5: The layout of a warehouse with 8 picking aisles and 2 cross aisles.

Table A.2: When the RPL has 10 items, the percentage (%) of replications in different scenarios that our model provides a better result than the baseline model on makespan.

Entity Combination	4PAs			8PAs		
	Low	Medium	High	Low	Medium	High
1P1T	**95.2**	**85.7**	0.0	**95.0**	**90.0**	0.0
1P2T	0.0	0.0	0.0	0.0	0.0	0.0
1P3T	0.0	0.0	0.0	0.0	0.0	0.0
2P1T	**100.0**	**95.0**	**19.0**	**100.0**	**93.0**	**34.1**
2P2T	**20.0**	0.0	0.0	**18.2**	0.0	0.0
2P3T	0.0	0.0	0.0	0.0	0.0	0.0
3P1T	**100.0**	**100.0**	**40.0**	**100.0**	**98.0**	**60.0**
3P2T	**84.0**	2.0	1.5	**82.0**	0.8	0.0
3P3T	0.0	0.0	0.0	0.0	0.0	0.0

Table A.3: When the RPL has 10 items, the average wait time decrease per item by changing from the baseline model to our model in different scenarios.

Entity Combination	4PAs			8PAs		
	Low	Medium	High	Low	Medium	High
1P1T	**2.8s**	1.2s	0.0s	**3.0s**	1.1s	0.0s
1P2T	0.1s	0.1s	0.0s	0.0s	0.0s	0.0s
1P3T	0.0s	0.0s	0.0s	0.0s	0.0s	0.0s
2P1T	**13.5s**	**8.7s**	0.2s	**14.1s**	**6.3s**	0.2s
2P2T	-0.6s	-0.3s	0.1s	0.8s	0.1s	0.0s
2P3T	0.0s	-0.2s	0.0s	0.2s	0.1s	0.1s
3P1T	**20.7s**	**10.0s**	1.5s	**12.2s**	**7.0s**	textbf2.0s
3P2T	**5.3s**	0.0s	0.2s	**3.9s**	0.0s	0.0s
3P3T	0.1s	0.1s	0.4s	0.1s	0.1s	-0.3s

Appendix B

Additional Materials for Variability Experiments

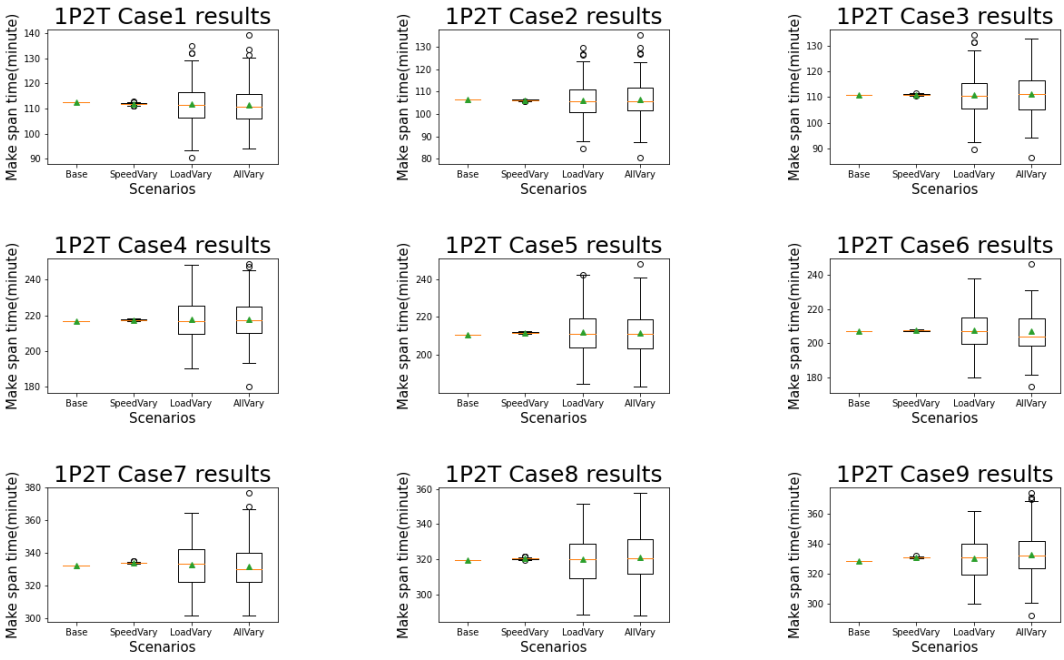


Figure B.1: Boxplot of the makespan for the scenarios with RPL sizes in all cases of 1P2T

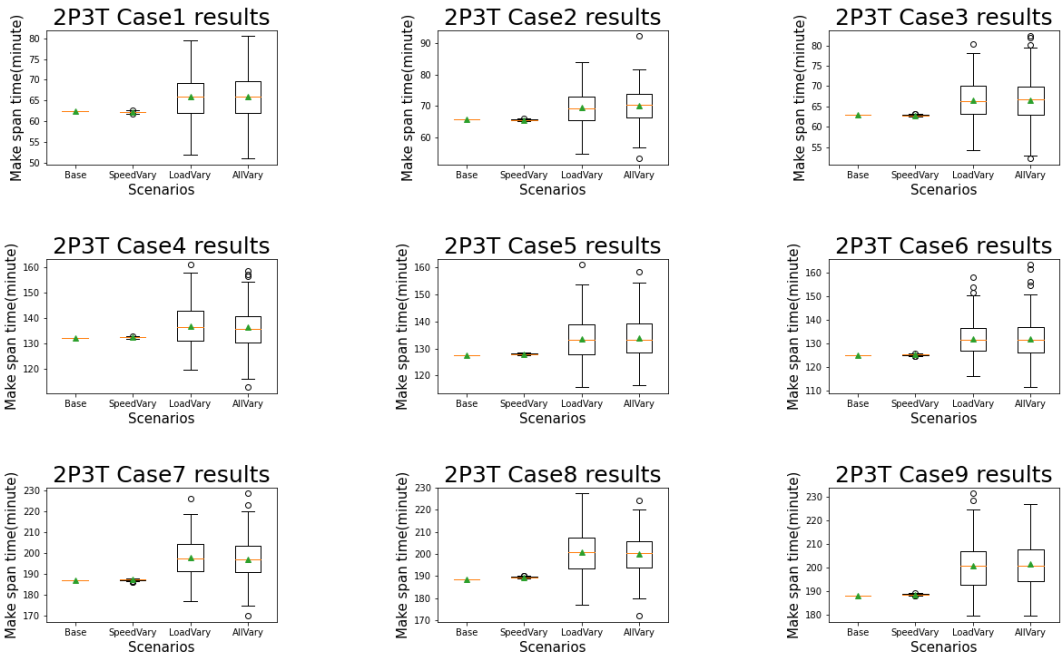


Figure B.2: Boxplot of the makespan for the scenarios with RPL sizes in all cases of 2P3T

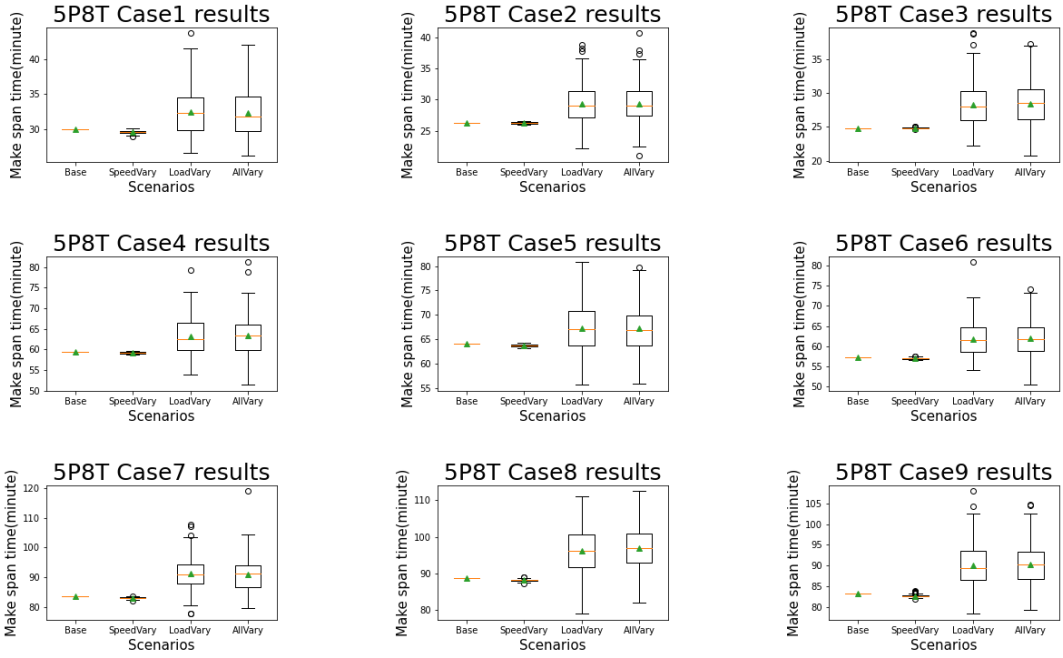


Figure B.3: Boxplot of the makespan for the scenarios with RPL sizes in all cases of 5P8T

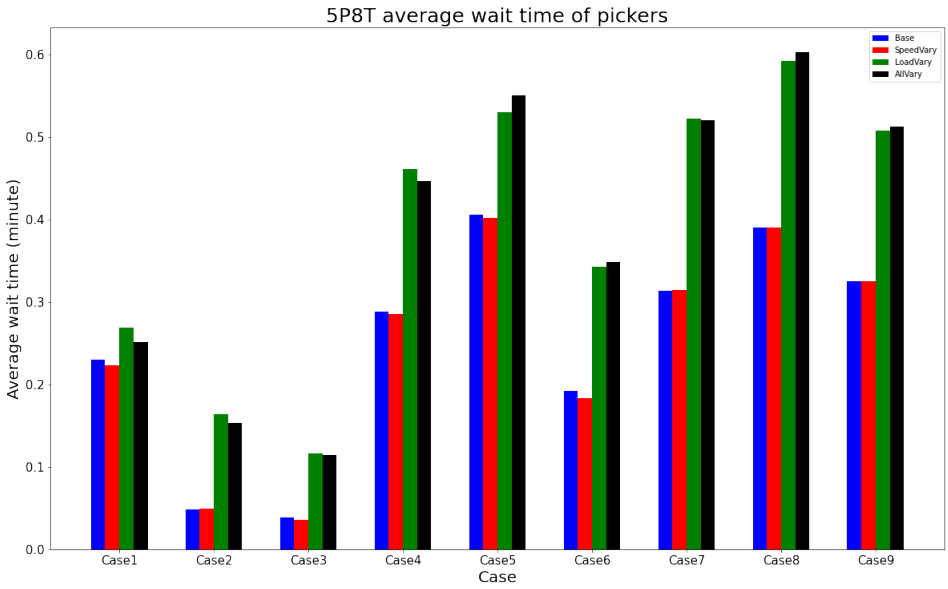


Figure B.4: Average wait time of pickers for all scenarios with 5P8T in all cases

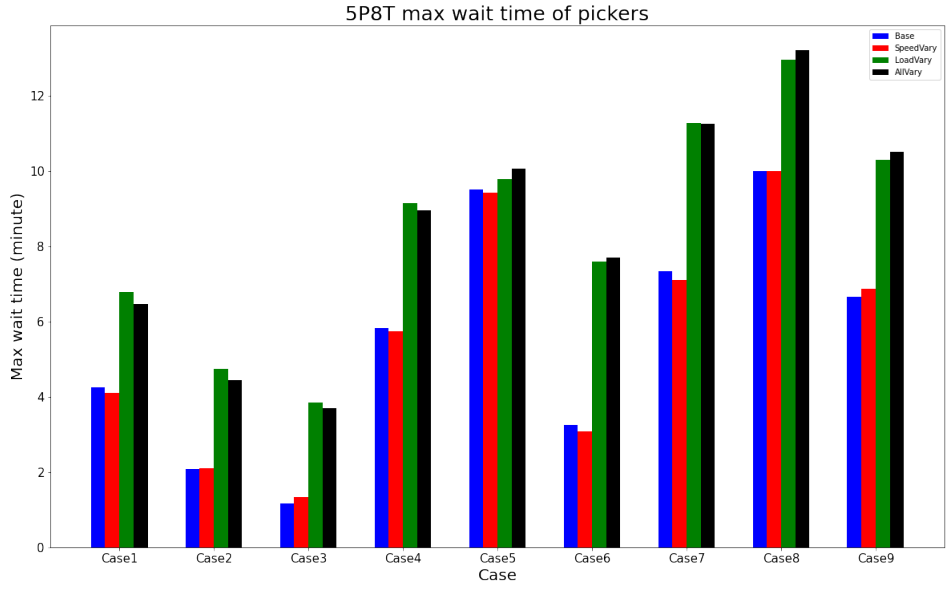


Figure B.5: Max wait time of pickers for all scenarios with 5P8T in all cases