

DYNAMIC ORDER-BASED SCHEDULING
ALGORITHMS FOR AUTOMATED RETRIEVAL
SYSTEM IN SMART WAREHOUSES

LIU JIALEI

MASTER OF SCIENCE (COMPUTER SCIENCE)

FACULTY OF INFORMATION AND
COMMUNICATION TECHNOLOGY
UNIVERSITI TUNKU ABDUL RAHMAN
DECEMBER 2021

**DYNAMIC ORDER-BASED SCHEDULING ALGORITHMS FOR
AUTOMATED RETRIEVAL SYSTEM IN SMART WAREHOUSES**

By

LIU JIALEI

A dissertation submitted to the Department of Computer Science,
Faculty of Information and Communication Technology,
Universiti Tunku Abdul Rahman,
in partial fulfillment of the requirements for the degree of
Master of Science (Computer Science) in December 2021

ABSTRACT

DYNAMIC ORDER-BASED SCHEDULING ALGORITHMS FOR AUTOMATED RETRIEVAL SYSTEM IN SMART WAREHOUSES

LIU JIALEI

With the rapid development of logistics industry, Smart Warehouse, which aims to automate the tasks of storage, picking, packaging, delivery, etc., has become a very important part in the logistics system. To automate and speed up the item retrieval process, a Smart Warehouse usually employs a management system, called the Automated Retrieval System (ARS), to control and schedule the retrieval jobs. However, most of the existing ARS scheduling algorithms handle the retrieval jobs of items independently, but do not consider the integrality of orders. Thus, the overall delay of orders cannot be optimized. In this dissertation, we introduce the concept of Order Tag to the ARS scheduling algorithms. First, we verify whether the Order Tag strategy can reduce the overall delay in the case of "Static Order Arrival". We propose two static algorithms, namely Static Order-Based Scheduling Algorithm – I (SOB-I) and Static Order-Based Scheduling Algorithm II (SOB-II). Simulation results demonstrate that these two strategies can reduce the total retrieval delay by approximately 30% compared to the existing algorithms, such as Order-Based Random Out Algorithm (OBRO), Item-Based Shortest-Job-First Algorithm (IB-SJF). Next, we study the case of "Dynamic Order Arrival". Instead of assuming that all

orders arrive to the system before processing, the algorithm considers orders arrive dynamically and it handles each new order once received. This makes the warehouse more flexible and efficient, but it also has higher requirements on the scheduling algorithms. To minimize the average delay and ensuring the fairness, two algorithms are proposed. They are named as Dynamic Order-Based (DOB) and Dynamic Order-Based with Threshold (DOBT) Scheduling Algorithms, respectively. Compared with the First-Come-First-Serve and other approaches, the simulation results show that DOB and DOBT are able to reduce the average order retrieval delay by at least 30%, and generate less backlog pressure to the downstream operations.

ACKNOWLEDGEMENT

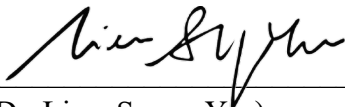
First of all, I would like to show my great appreciation to my main supervisor, Dr. Liew Sounq Yue, who has been helping me all along the way of my Master's study at UTAR. Dr. Liew has been of great help to me both academically and in life. In terms of life, as an international student, when I first came to Malaysia, he helped me get familiar with the environment here and often cared about my life like a good friend. Academically, he often pointed me in the direction with his rich experience and professional knowledge when I was confused. I couldn't have made it this far without his help. At the same time, I would also like to thank my co-supervisor, Dr. Ooi Boon Yaik, who often provides some good ideas and suggestions for our research and promotes our research process at critical moments. Similarly, I must also thank my external co-supervisor, Prof. Qin Donghong. It was with his encouragement that I embarked on my study trip to UTAR, which is an unforgettable and meaningful experience. I also want to thank him for his concern about my life and study, which often gives me warmth and motivation to persist.

Finally, I'd like to thank everyone I've met at UTAR for all the trouble I've put you through. I really appreciate your patience and help.

APPROVAL SHEET

This dissertation/thesis entitled “**Dynamic Order-based Scheduling Algorithms for Automated Retrieval System in Smart Warehouses**” was prepared by LIU JIALEI and submitted as partial fulfillment of the requirements for the degree of Master of Master of Computer Science at Universiti Tunku Abdul Rahman.

Approved by:

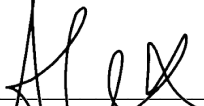


(Dr. Liew Soung Yue)

Date: 17/5/2022

Main Supervisor

Department of Computer and Communication Technology
Faculty of Information and Communication Technology
Universiti Tunku Abdul Rahman



(Dr. Ooi Boon Yaik)

Date: 17/5/2022

Co-supervisor

Department of Computer Science
Faculty of Information and Communication Technology
Universiti Tunku Abdul Rahman



(Prof. Qin Donghong)

Date: 17/5/2022

Co-supervisor

Department of Network Engineering
Faculty of Artificial Intelligence
Guangxi University for Nationalities

SUBMISSION SHEET

FACULTY OF INFORMATION AND COMMUNICATION
TECHNOLOGY

UNIVERSITI TUNKU ABDUL RAHMAN

Date: 16/5/2022

SUBMISSION OF FINAL YEAR PROJECT /DISSERTATION/THESIS

It is hereby certified that LIU JIALEI (ID No: 19ACM07125) has completed this dissertation entitled “ Order-based Scheduling Algorithms for Automated Retrieval System in Smart Warehouses ” under the supervision of Dr. Liew Soung Yue (Supervisor) from the Department of Computer and Communication Technology, Faculty of Information and Communication Technology , and Dr. Ooi Boon Yaik (Co-Supervisor) from the Department of Computer Science, Faculty of Information and Communication Technology, and Prof. Qin Donghong (Co-Supervisor), from the Department of Network Engineering, Faculty of Artificial Intelligence, Guangxi University for Nationalities.

I understand that University will upload softcopy of my dissertation in pdf format into UTAR Institutional Repository, which may be made accessible to UTAR community and public.

Yours truly,



(LIU JIALEI)

*Delete whichever not applicable

DECLARATION

I hereby declare that the dissertation is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at UTAR or other institutions.

Name  _____
(LIU JIALEI)

Date 16/5/2022

TABLE OF CONTENTS

	Page
ABSTRACT	ii
ACKNOWLEDGEMENT	iv
APPROVAL SHEET	v
SUBMISSION SHEET	vi
DECLARATION	vii
TABLE OF CONTENTS	viii
LIST OF TABLES	x
LIST OF FIGURES	xi
LIST OF ABBREVIATIONS	xiii

Chapter

1.0 INTRODUCTION	1
1.1 Background	1
1.2 Problem Statement	3
1.3 Objectives	4
1.4 Organization of Dissertation	5
2.0 RELATED WORKS	6
2.1 Utilization of AGV in Smart Warehouses	7
2.2 Utilization of Automated Stackers in Smart Warehouses	11
2.3 Algorithms for Scheduling the Stackers' Jobs	13
2.4 Layout and Workflow of Smart Warehouses	18
3.0 MATHEMATICAL MODELLING OF THE PROBLEM	21
3.1 Retrieval Time	21
3.2 Notations and Definitions	22
3.3 Objective Function	25
4.0 Static Scheduling Algorithms	27
4.1 Order-Based Random Out Algorithm (OBRO)	27
4.2 Item-Based Shortest-Job-First Algorithm (IB-SJF)	27
4.3 Static Order-Based Scheduling Algorithm – I (SOB-I)	29
4.3.1 Job Set, Schedule Sequence and Order Tag	29
4.3.2 Implementation Procedure	30
4.4 Static Order-Based Scheduling Algorithm – II (SOB-II)	30
4.4.1 Order Set	30
4.4.2 Implementation Procedure	30
4.5 Comparison between SOB-I and SOB-II	31

4.6	Methodology for Performance Study	32
5.0	SIMULATION and DISCUSSION I	33
5.1	Simulation Settings	33
5.2	Simulation Results	33
5.3	Discussion	35
6.0	DYNAMIC SCHEDULING ALGORIHTMS	36
6.1	First-Come-First-Serve (FCFS) Scheduler	36
6.2	Last-Come-First-Serve (LCFS) Scheduler	37
6.3	Shortest-Job-First (SJF) Scheduler	37
6.4	Dynamic Ordered-Based (DOB) Scheduler	38
6.4.1	Flow of DOB Scheduling Algorithm	41
6.4.2	Discussion of DOB	43
6.5	Dynamic Order-Based with Threshold (DOBT) Scheduler	43
6.6	Methodology for Performance Study	44
7.0	SIMULATION and DISCUSSION II	45
7.1	Simulation Models	45
7.1.1	Random Model for Generating Order Arrivals	45
7.1.2	Random Model for Generating Item Quantity	45
7.1.3	Random Models for Generating Shelf Locations and Retrieval Times 46	
7.2	Simulation Settings	47
7.3	DOB VS. other Schedulers	47
7.3.1	Average Retrieval Delay of Orders	49
7.3.2	Maximum Retrieval Delay of Orders	50
7.3.3	Downstream Backlog Pressure	52
7.4	DOBT vs. DOB and FCFS	54
7.5	Discussion	59
8.0	CONCLUSION	61
8.1	Proposed Approaches and Achievement	61
8.2	Future Work	62
	REFERENCES	64
	APPENDIX A: EXPERIMENT DATA	70

LIST OF TABLES

Table	Page
3.2.1 Notations and Definitions	24
4.2.2 Retrieval time of three orders	29
4.2.3 Schedule of three stackers	29
6.4.1. Definitions of Variables	41
7.4.1 Average Delays of Different Algorithms with Different Thresholds	55

LIST OF FIGURES

Figure	Page
2.2.1 Top view of shelves and stackers (J. Cheng, 2019)	12
2.3.1 Comparison of ACO and Hybrid algorithm. (H. Hu, et al., July 2018)	15
3.1 (a) Layout of the smart warehouse. (b) Workflow of the warehouse	19
3.2 Structure of a shelf.	20
4.1.1 Coordinate map of the shelf.	22
4.2.1 Mapping of the ordered items to the jobs of stackers.	24
4.2.2 An example of item-job mapping.	25
4.2.3 The job queue of a stacker.	25
5.2.1 Order continuity on the same stacker	28
6.2.1 Total Delay in Seconds over the Number of Orders with five stackers.	34
6.2.2 Total Delay in Seconds over the Number of Orders with ten stackers.	34
6.2.3 Total Delay in Seconds over the Number of Orders with fifteen stackers.	34
7.4.1 Flow of Dynamic Order-Based Algorithm.	39
8.3.1.1 Comparison of average delays with 6 stackers	49
8.3.1.2 Comparison of average delays with 12 stackers	50
8.3.2.1 Comparison of max delays with different order rates	51
8.3.3.1 Comparison of average order pressures with different order rates	53
8.3.3.2 Comparison of max order pressures with different order rates	53
8.3.3.3 Comparison of average item pressures with different order rates	53
8.3.3.4 Comparison of max item pressures with different order rates	54
8.4.1 Average Delays of DOBT with different threshold values.	55
8.4.2 Max order delays of DOBT under different threshold values ($\underline{\lambda}=110$ orders/hour).	56
8.4.3 Average order pressures of DOBT under different threshold values ($\underline{\lambda}=110$ orders/hour).	57
8.4.4 Max order pressures of DOBT under different threshold values ($\underline{\lambda}=110$ orders/hour).	58

- 8.4.5 Average item pressures of DOBT under different threshold values ($\lambda=110$ orders/hour). 58
- 8.4.6 Max item pressures of DOBT under different threshold values ($\lambda=110$ orders/hour). 59

LIST OF ABBREVIATIONS

SW	Smart Warehouse
WMS	Warehouse Management System
AS/RS	Automated Storage and Retrieval System
ASS	Automated Storage System
ARS	Automated Retrieval System
DOB	Dynamic Order-Based
DOBT	Dynamic Order-Based with Threshold
FCFS	First-Come-First-Serve
LCLS	Last-Come-First-Serve
SJF	Shortest-Job-First
AGV	Automated Guided Vehicles
TSP	Travelling Salesman Problem
ACO	Ant Colony Optimization
PSO	Particle Swarm Optimization
OBRO	Order-Based Random Out Algorithm
IB-SJF	Item-Based Shortest-Job-First (IB-SJF) Algorithm
SOB-I	Static Order-Based Scheduling Algorithm – I (SOB-I)
SOB-II	Static Order-Based Scheduling Algorithm – II (SOB-II)

CHAPTER 1

INTRODUCTION

1.1 Background

More and more customers choose to shop online because they can reach a great variety of products conveniently with just a few clicks. In addition, online shopping can also reduce people's contact and ensure their safety while they are shopping for the daily needs. This is particularly important during the pandemic of COVID-19. With the growth of e-commerce businesses, on the other hand, the merchants would need to provide quality products and good services in order to attract customers to buy from them. To reduce their business costs, many merchants choose to store their products in public warehouses, because this can streamline their backend storing and delivering processes so that they can be more focused on the frontend promotion and selling processes [1], [2]. Such a trend also puts higher demands on the management capabilities of those large public warehouses.

In addition to more frequent import and export operations, e-commerce merchants need to handle a large quantity of customers and take care of customers' purchasing experiences carefully. Delay in deliveries will have negative impacts on the image of a merchant, as this will influence the customers' satisfaction. When the

qualities and prices of the same products provided by different suppliers are more or less in the same range, the delay will make a subtle influence on customers choices. The faster the delivery the better the customer experience. What more is, there will always be some people who would like to pay extra fees for faster delivery for various reasons.

The function of warehouse plays a definitely important role in the whole logistics process. In contrast to the traditional warehouses, with the development of automation technology, more and more Smart Warehouses appear. They change the operation pattern of warehouses from “people to goods” to “goods to people” through automated equipment, which greatly improved the efficiency of the warehouse [3].

A Smart Warehouse has a different management approach. For example, Warehouse Management System (WMS) and Automated Storage and Retrieval System (AS/RS) are introduced, which can manage inventory information and perform storage and retrieval operations automatically. In this way, it can not only save the labor cost under the traditional warehouse sorting mode, but also reduce the errors caused by manual operation and reduce the potential safety hazards for workers, so as to greatly improve the work efficiency in all aspects of warehousing, and provide reliable warehousing services for large-scale e-commerce activities. Typically, an AS/RS employs automated stackers to store and retrieve items. With this technology, the Smart Warehouse can handle higher shelves, which means it can have a greater storage capacity as compared with the traditional warehouses. However, the performance of AS/RS operations relies on the careful designs and implementations

of the relevant process flows and algorithms of the system.

1.2 Problem Statement

In typical warehouse operations, upon receiving the orders from customers, the purchased items need to be retrieved from shelves and then packaged accordingly for delivery. These operations need to be properly managed in order to guarantee excellent experience for customers [4], [5].

There are many factors that may affect the customer satisfaction level. Among these factors, “delay in delivery” has a devastating impact on the reputation of merchants, and it greatly influence the customers’ choices on the preferred merchants. As a matter of fact, there are even customers who are willing to pay more for faster delivery [6]. Although a differentiated handling policy can be implemented at a warehouse to reduce the delivery delay for a group of high-priority customers [7], the overall performance of the warehouse is still very important to ensure the satisfaction of most customers [8]. Therefore, there is a need for optimizing every operation of the warehouse in order to provide better service for all.

The AS/RS can be divided into two subsystems, namely Automated Storage System (ASS) and Automated Retrieval System (ARS). In this dissertation, we focus on the process and operation of ARS because the product retrieval speed directly affects the subsequent processes in the Smart Warehouse, such as packaging and delivery, and thus bringing significant impact to the entire transaction duration between merchants and customers. In particular, the items ordered by the same

customer should be considered as an integral part; because if such an integrality is not considered by the ARS in the retrieval process, then the entire order may experience an unnecessary delay because the order from the merchant used to contain multiple items, and the items that come to the packing area first needs to wait for other items.

In the past, the integrality of order has not received much attention in the parallel retrieval process of multiple stackers. Most of the research focuses on how to reduce the travel distance of the stacker, neglecting that one order usually contains multiple items. To take the integrality into account, this dissertation proposes using an Order Tag to label all the items that belong to the same order. The Order Tags are then used to schedule the item retrieval sequence of each stacker in order to achieve the required performance. In other words, the way of calculating the Order Tags will determine the scheduling discipline of the ARS.

1.3 Objectives

The main objective of this dissertation is to study the integration of the concept of Order Tag into the scheduling of ARS in order to minimize the average retrieval delay of all orders and ensure the fairness among the orders to improve the overall shopping experience of customers. In order to achieve this main objective, first of all, we need to design a suitable mathematical model to describe and evaluate the job scheduling problem of stackers. The whole study is divided into two scenarios: the first stage verifies whether the order-based algorithms can really reduce the delay in the case of “Static Order Arrival”; the second stage designs the dynamic Order-

based algorithms for the case of Dynamic Order Arrival, and then verifies the performance of the algorithms through experiments.

1.4 Organization of Dissertation

The remainder of this dissertation is organized as follows. In Chapter 2, we review some relevant works and point out the difference of our proposed algorithms, and discuss the layout and workflow of the smart warehouse. The mathematical model of the retrieval process of stackers is derived in Chapter 3. We explain the procedure of static scheduling algorithms in Chapter 4. Simulation results are discussed in Chapter 5. We elaborate on the design of our dynamic scheduling algorithms, and compare them with the FCFS and other approaches in Chapter 6. In Chapter 7, the simulation models and assumptions are presented. We also discuss the simulation settings and analyzes the performances of the abovementioned scheduling algorithms. Finally, Chapter 8 presents the conclusions.

CHAPTER 2

RELATED WORKS

This Chapter provides an overview of two kinds of implementation of “goods to person” retrieval process in the Smart Warehouse, and analyses their advantages and disadvantages. The application and working principles of automated stackers in the Smart Warehouse are also introduced. Besides, we make a thorough review of the research of other scholars on stackers’ scheduling algorithm and compare them with our research.

Warehouses in the modern logistics industry are getting more inclined to the adoption of automation technology, which makes the efficiency and effectiveness of warehouses higher. Amazon uses Automated Guided Vehicles (AGV) to transport the entire shelf to the sorter [9], thus liberating the legs of the sorter, which has led many scholars to pay attention to the utilization of AGV and other automated equipment to deliver the items to the sorter to improve the performance of the total warehouse. This new type of work style makes the packagers just need to wait at the Packaging Station and wait for the products (or the whole shelf) come to them. Using this principle, various automated shelves have been designed. There are two main kinds of shelves adopted by modern automated warehouse. With reference to Fig. 2.1, the first one is

movable shelves. Generally, the movable shelves are smaller and lighter than fixed shelves, and their height is not very high, usually a few meters, so that AGVs can carry them. The grid of items stored on each layer of the shelves is not very large so the shelves are often used to store smaller items. In this case, an AGV is able to carry the whole shelf to the packaging station even there is only one product needed to be picked. With reference to Fig. 2.2, another is the fixed shelves, which are not movable but we can utilize some automated equipment to pick and retrieve items automatically. Then we can use AGVs or conveyor belts to transport the goods to the Packaging station.

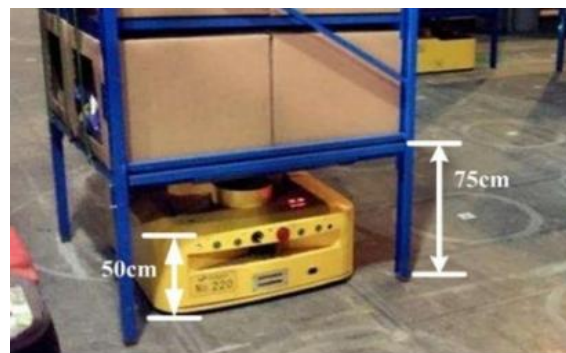


Figure 2.1. Movable shelf and AGV [9]



Figure 2.2 Fixed Shelf and Stacker [19]

2.1 Utilization of AGV in Smart Warehouses

The environment of warehouse for the AGV is always considered as a grid

map and each grid is a reachable location for AGV. Since multiple AGVs move together at the same time, the competition for resource or the right to use of each grid will occur. There are various collisions that may occur in real time. To simplify the problem many research teams studied the corresponding Collision Model. Zheng's team [10] classify the collision issues into four kinds: head-on collision, cross collision, node-occupancy collision, and shelf-occupancy collision, as the following figure 2.1.1 shows. The greatest thing of their work is that they set some parameters and attached each situation with an acute mathematical formula so that the supervisor system level can compute each kind of collisions in advance and formulate corresponding solutions.

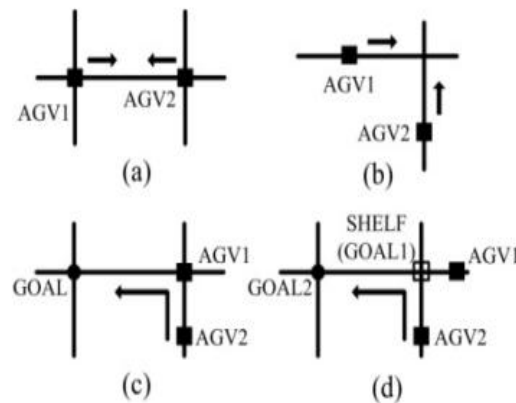


Figure 2.1.1 Collision Types [10]

In the case of carrying the whole shelf to the sorting station, the AGV has two states: load state means it is carrying one shelf and unload state means it's going to carry one shelf or charge or do something else [10]. There are many routing algorithms that have been designed and tested to plan routes for AGVs, such as particle swarm optimization [11], colony optimization [12] and Conflict-Based Search [13]. Zheng Zhang's team proposed an algorithm called Collision-Free Route

Planning. The theory behind it is a typical off-line [14] routing planning algorithm. They use improved Dijkstra's algorithm [15] to plan the complete path for the AGV in advance. Then they utilize the collision detection formula to compute whether there are any possible collisions may occur between the current AGV's route and previous AGVs' routes. If there may occur collisions. The system will choose one or two solutions from the three alternative solutions until the collision disappear. The next AGV will repeat these processes. It schedules the path of AGVs previously and use the collision detection to ensure there are no collisions will happen in the actual operation.

In contrast, Yijing Guo's team provided a Dynamic Unlock Algorithm [16], which doesn't need to process the collisions at the path planning stage. They liken the AGVs contention for nodes to the contention of computer resources by multiple threads in system. They first plan the routes for every AGV without considering whether there will be collisions among these routes. After the first step, they make lists of each site to record all the AGVs that need the same site in a queue as the Figure 2.1.2 shows. Each time to a specific site, only the head number of in its queue can use that site, means it can move through the site. That means even other AGVs may arrive before the head AGV, they all need to wait until head AGV arrive. This algorithm is one kind of on-line algorithm [17] of path planning for AGV. This kind of algorithms always resolve the collision during the real running of AGVs.

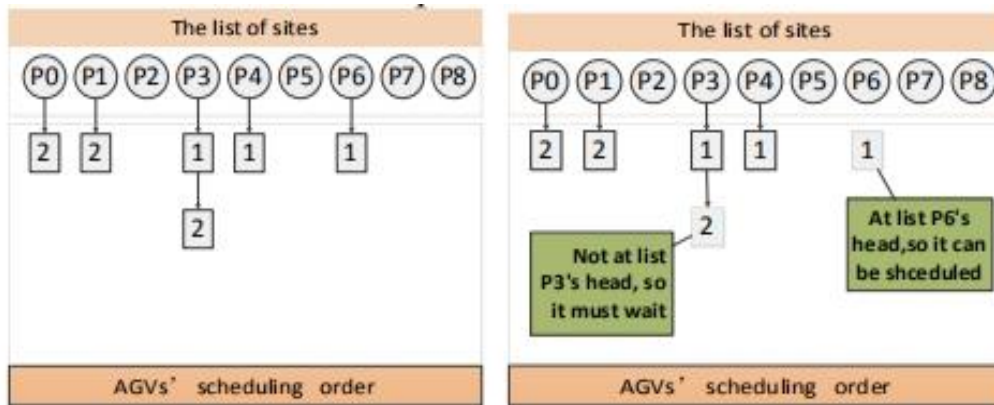


Figure 2.1.2 List of each site [17]

Actually, whether it's Zheng Zhang's or Yijing Guo's team or other research teams, they all abstracted the collision model very well and provided corresponding solutions to each kind of collisions, which can be used in any scenario where AGV is required. However, the algorithms provided by them still have some drawbacks. For example, Collision-Free Route Planning provided by Zheng Zhang's team takes a large amount of time when planning the path route for the new AGV. Because each time the system needs to compare the new path with the previously planned paths to detect whether there are collisions. If there is any, it will have to replan the route. In addition, an off-line algorithm cannot deal with any accident that occurs at real operation. Besides, in Dynamic Unlock Algorithm, each AGV needs to wait until the head AGV of queue arrive. If there are two AGVs whose routes are completely converse, then the deadlock may occur, just like the threads in system.

From the perspective of the practical application of their papers and in an automated warehouse scenario, what we really want to do is deliver the items that need to be picked to the sorting station. However, if we are moving the entire shelf to a packaging station just for picking one or two products from it, the work efficiency is very low from the perspective of AGVs. What's more, a shelf, with a few meters high,

greatly reduce the speed at which the AGV can run. That's one of main reasons that we start paying attention on the fixed shelves and ARS.

Consider that if we can fetch the products we need from shelves and the AGV just need to deliver the useful products to the Packaging Station. Not only there will be less limitations on the speed of AGV, but also can we make the full use of the space in the warehouse. That is because if we need to move the whole shelf, then the kinds of products we can store are very limited due to the size of shelf's structure and the weight AGV can bear.

2.2 Utilization of Automated Stackers in Smart Warehouses

With the advancement of automation technology, many new facilities have been developed for automating the warehouse operations to enhance the production processes and management approaches [18]–[20]. One of these facilities is the Automated Stacker, which can be used to retrieve items from a shelf automatically in a Smart Warehouse [21]. In order to increase the storage capacity, on the other hand, the Smart Warehouse is normally equipped with multiple parallel shelves of extremely large size. The Smart Warehouse can then employ a management system, called the Automated Retrieval System (ARS), to control and schedule the retrieval jobs of multiple stackers on these parallel shelves [22], [23].

The concept of the stacker first appeared in the 1960s and was put into industrial use in the United States [24]. At present, the maximum height of the automated stacker in advanced countries can reach more than 50 meters, the

maximum horizontal movement can reach 300 - 400 m / min, the horizontal acceleration can reach 5 m / s²; the vertical lifting speed can reach 100 m / min and the load can reach 10T, the positioning error can be controlled within 5mm [25], [26]. Considering its running speed, load, and positioning accuracy, automated stacker is a relatively mature automation technology. It can basically meet the storing and retrieval ability we need.

Since between every two shelves there is a stacker. Figure 2.2.1 is a top view of the stackers and shelves. That means all the products that store on the adjacent shelves can only be accessed by only one stacker. This may cause some “congestion”, when there are a large number of items in the adjacent shelves that need to be taken out. Therefore, for the three-dimensional fixed shelves, the placement of items also needs to be adjusted appropriately, and some related attributes of the items need to be considered, such as turnover rate, or other methods need to be introduced to reduce this “congestion” [27]. What’s more, there is limitation on the loading capacity of a stacker, so when a pile of items arrives at the stacker, which one to be retrieved first that means the retrieval order of items may make different sense in the reality.

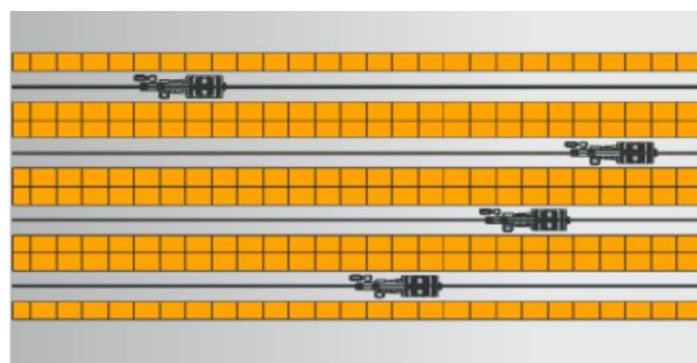


Figure 2.2.1 Top view of shelves and stackers [27]

In short, the AGV make the packagers relief their legs. However, carrying the

whole shelf seems not an advisable choice due to the limitation both to the storage capacity of shelves and the efficiency of the stacker. It seems a feasible scheme that we combine stacker (storage capacity of the fixed shelf) and AGV to reach a more efficient Smart Warehouse.

2.3 Algorithms for Scheduling the Stackers' Jobs

Many existing works attempt to improve the overall performance of Smart Warehouses. For example, Kung et al. studied the warehouse system with multiple stackers on a common rail to reduce chances of collision among stackers and improve the work efficiency of warehouse significantly [28]. At present, the leading research on the retrieval process of Smart Warehouses is to optimize the moving path of the stacker according to the storage location of the items so as to minimize the total completion time of orders [29]–[31].

Most of the above works assume that a stacker can travel through the corresponding shelves to retrieve multiple items in a trip. They further formulate the retrieval jobs of stackers as combinatorial optimization problems, such as the Travelling Salesman Problem (TSP) [32]–[39], so that many optimization algorithms can be applied to solve the stacker scheduling problem. TSP is one of the most common problems been researched, it is an np-hard problem that we cannot solve the problem in polynomial time. These algorithms include HGA-VNS Algorithm [33], Combination of Free Search and Amendment Circle Algorithm [34], Genetic Algorithm [35], Non-dominant Sequencing Genetic Algorithm with Elite Strategy [36]

and Genetic Particle Swarm Algorithm [37]. Li et al. studied the utilization of stackers in the tobacco industry and solved the coordination problem of the Automated Storage System and Retrieval System [38]; through the engineering test cases, the effectiveness of their designed algorithm has been demonstrated. Besides, there are also some teams that research different types of stacker aisles. Yu analyzed the ant colony system and parthenogenetic algorithm, and proposed a parthenogenetic ant colony algorithm, which greatly reduces the time of order picking to improve warehouse efficiency [39].

However, with the increasing number of orders, it takes exponential time for the above optimization algorithms to get acceptable results, and yet it is with no guarantee of getting an optimal solution in a fixed number of iterations. Furthermore, the design of an optimization algorithm to reach an optimal or suboptimal solution can sometimes be complicated and the parameters need to be subtly adjusted, which is not practical in the real situations. For example, the Fee Search is a heuristic algorithm which not only take advantages of Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO), but also the characteristic of advanced animals like sense and mobility [34]. Amendment-circle algorithm is not efficient enough nowadays, but its feasibility makes it useful in Huaining Hu's team's research. They make numeric simulations. The result of their experiments shows that the hybrid algorithms is better than ACO in the solution of ordering picking problem under their assumptions.

However, as the Figure 2.3.1 shows, through the experiment data, when the

problem size extend to 40, the speed of hybrid algorithm declines sharply. When the test size achieves 800, the time cost reach 576 seconds. What's more, all the heuristic algorithms cannot assure the optimal solution, they have the possibility to stuck into a local problem. And the research didn't show us the probability we can hit the optimal solution or the deviation between the result of their algorithms and the real optimal solutions that we can reach. More practicable and more flexible algorithms need to be proposed to face different needs of automated warehouses.

Size	S-ACO	T-ACO	S-FSAC	T-FSAC
40	1608	27.9504	1602.5	1.5327
80	3255.5	221.3211	3233	1.5856
120	4709	807.8908	4663.5	5.5830
200	7872.5	4230.8707	7792.5	5.8546
400	15885.5	53155.1873	15677	48.9001
800	-	-	3061.75	576.1129

Figure 2.3.1 Comparison of ACO and Hybrid algorithm. [34]

Another issue of most of the existing approaches is that, they focus on the travelling path of a stacker, but do not pay much attention to the integrality of orders across multiple parallel stackers. It should be noted that an order from merchants usually contains multiple items. And an order can be packaged into a parcel only when all its items have been retrieved. Therefore, the order may experience unnecessary waiting time if one of its items encounters a much larger retrieval delay than other items do.

In this dissertation, we focus on the scenario whereby an ARS is employed in a Smart Warehouse to manage the retrieval jobs of multiple stackers. We further assume that each order from a customer may contain multiple items distributed over

different shelves, and a stacker can only pick up one item at a time. However, parallel processing of multiple stackers is possible to accomplish the retrieval task [40]. It should also be noted that when an order is waiting for the last item to be retrieved, the rest of its items that have been retrieved will have to occupy the temporary storage of the downstream operation, such as at the packaging station. Such pending order will then become the backlog and generate pressure to the system.

For this reason, the retrieval sequence of items of different orders in different stackers will affect the overall performance [41], [42]. That is, a proper scheduling approach can not only reduce the average retrieval delay, but also reduce the backlog pressure.

Guo et al. [43] considered the integrality of order in the situation where the arrivals of all orders are known in advance. They focused more on how to resolve the contention for stacker resources to reduce the total retrieval time and get the optimal retrieval sequence with the Ant Colony Algorithm. Through their experiments, they enhanced about 10% efficiency compared with the traditional optimization algorithms. Unfortunately, they did not consider that in a real-world scenario, a warehouse normally receives orders dynamically.

We integrate the order integrality into a label, named Order Tag, for performing the job scheduling of stackers. First step, we demonstrate work showed that the integration of Order Tag in the job scheduling algorithms can effectively reduce the average retrieval delay for the case of Static Order Arrival. Next, we further extend the work to deal with Dynamic Order Arrival, and proposes two

dynamic order-based scheduling algorithms, namely DOB and DOBT, respectively. The details of the two proposed algorithms will be discussed in the subsequent sections.

There are also some naive algorithms proposed in logistic-related researches, such as Order-Based Random Out Algorithm [44], [45], Shortest-Job-First Algorithm [46], [47], First Come First Serve Algorithm [48], [49], Last Come First Serve Algorithm [50], [51].

Although these native algorithms are rarely studied alone, they can be used as a comparison to reflect the improvement of our algorithms from different angles. For example, Order-Based Random-Served is a naive algorithm which does not consider the Item Retrieval Time. On the other hand, Item-Based Shortest-Job-First is another naive algorithm which does not consider the integrality of the order. Our approach combines two parameters (1) Order-Tag and (2) Item Retrieval Time for job scheduling. If one of the parameters is removed from the algorithm, then it becomes a naive algorithm which cannot achieve the required performance.

2.4 Layout and Workflow of Smart Warehouses

Typically, there are multiple types of built-in facilities in a Smart Warehouse that need to operate together to accommodate customers' demands, as shown in Figure 2.4.1. These facilities are responsible for different processes, which include picking/retrieving the ordered items from shelves, transferring the retrieved items to packaging stations, and packaging the items into parcels for shipping. The layout of these facilities has to be well designed in order to streamline the shelf-to-package operations of the warehouse. There are two mainstream automated shelf-to-package approaches, and they are described as follows.

The first approach is to use heavy-duty Automated Guided Vehicles (AGVs) to lift and transport shelf units to the pickers at the packaging stations; then the pickers manually select the items ordered by customers from the shelf units, and package the items into parcels [52], [53]. However, many challenges may arise in warehouse management with this approach, such as the avoidance of collisions of the AGVs [54]. Moreover, as the entire shelf unit must be lifted and moved, the speed of the AGVs ought to be low and thus it limits the throughput of the warehouse. Another issue is that, since a shelf unit may contain many other items which are not required by the picker, moving the entire shelf unit may result in the wastage of energy. The high maintenance cost of heavy-duty AGVs and the size limitation of the shelf unit are two other potential problems with this approach.

With reference to Figure 2.4.1a, the other shelf-to-package approach can be divided into three sub-operations [55], [56]. The first sub-operation is to use

automated stackers to automatically pick and retrieve the ordered items from the shelves in the Retrieval Area; the retrieved items are then placed at the transfer stations in the Transfer Area. In the second sub-operation, a fleet of lightweight AGVs or a conveyor belt is employed to transport the items from the Transfer Area to the Packaging Area. In the last sub-operation, there are several packaging stations in the Packaging Area, which are in charge of packaging the items ordered by customers into parcels. Note that the items from the same order have to be packaged by the same packaging station. With this approach, only the ordered items will be transported. Thus, not only that it optimizes energy usage, but also the entire operation, including picking process, can be automated.

The workflow of Smart Warehouse is summarized and shown in Figure 2.4.1b.

In this dissertation, we focus on the retrieval process in the Retrieval Area.

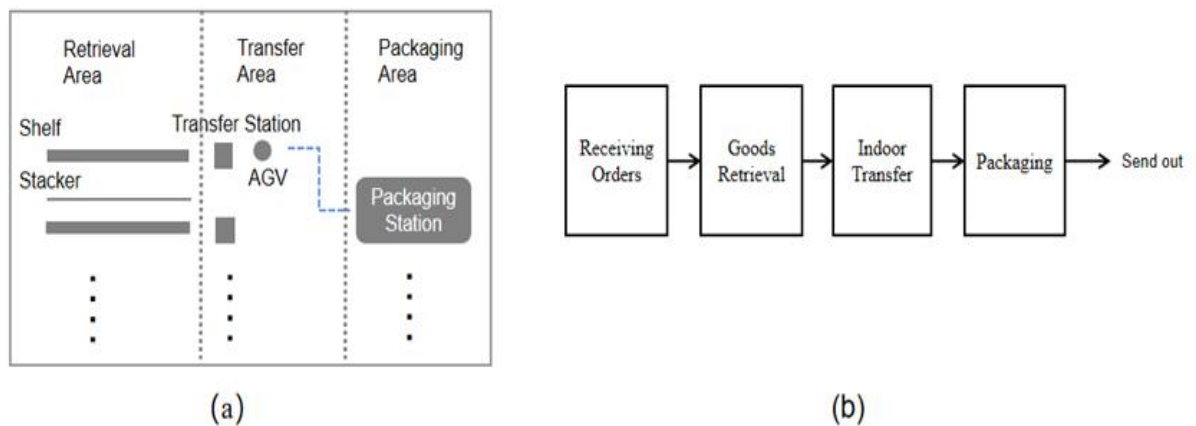


Figure 2.4.1 (a) Layout of the smart warehouse. (b) Workflow of the warehouse

With reference to Figure 2.4.2, each shelf in the Retrieval Area can be divided into multiple levels vertically, and each level has a height of h . In each level, space is further divided into several storage units horizontally, and each storage unit has a width of w . A storage unit can store one kind of product only. Moreover, each shelf is

associated with an automated stacker composed of a movable cargo platform and a robotic picker. The movable cargo platform can move to any of the storage units of the shelf so that the robotic picker can take out an item from the storage unit at a time, and then transport the item back to the transfer station. Based on this operation setting, we derive the mathematical model of the retrieval process of stackers in the next Section.

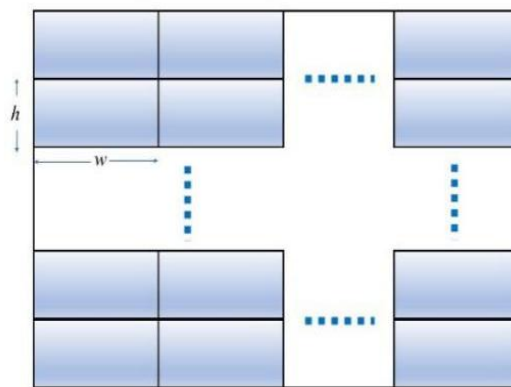


Figure 2.4.2 Structure of a shelf.

CHAPTER 3

MATHEMATICAL MODELLING OF THE PROBLEM

3.1 Retrieval Time

Without the loss of generality, let the position of a storage unit be denoted by (x, y) , where x th and y th are the ordinal numbers of the storage unit along the horizontal and vertical directions, respectively, in the shelf. We further assume that the parking position of the stacker as well as the item transfer station are at location $(0, 0)$. Note that the stacker can move along horizontal and vertical directions to reach any of the storage units to retrieve item(s). An example is as shown in Figure 3.1.1. That is, if the stacker desires to retrieve an item from the storage unit located at $(4, 3)$, then it needs to first travel from $(0, 0)$ to $(4, 3)$. Subsequently, the stacker can take out the desired item from the storage unit $(4, 3)$ and transport the item back to $(0, 0)$. The whole retrieval process is completed after the stacker drops the desired item at the transfer station. Note that the horizontal unit distance is w , and the vertical unit distance is h . In addition, we assume that the horizontal and vertical travelling speeds of the stacker are v_x and v_y , respectively. As shown in (1), the traveling time of the stacker from $(0, 0)$ to (x, y) is the maximum time needed to travel along with both directions.

$$\text{Time from } (0,0) \text{ to } (x, y) = \max\left(\frac{x \cdot w}{v_x}, \frac{y \cdot h}{v_y}\right) \quad (3.1)$$

The common retrieval time function can then be obtained by calculating the entire time needed for a stacker to perform a one-time pick-up job from point (x, y) . Let T denote the sum of the constant times needed for the stacker to take out an item from a storage unit, drop the item to the transfer station, etc. Thus, for the item stored in (x, y) , we have the retrieval time function, $F(x, y)$, as shown below.

$$F(x, y) = \max\left(\frac{x \cdot w}{v_x}, \frac{y \cdot h}{v_y}\right) \times 2 + T \quad (3.2)$$

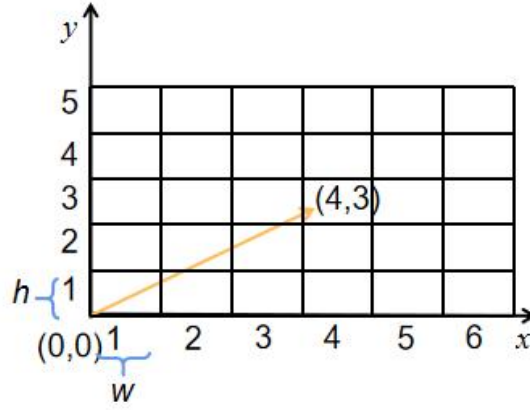


Figure 3.1.1 Coordinate map of the shelf.

3.2 Notations and Definitions

With reference to Figure 3.1.1 and Table 3.2.1, let $\mathcal{Q} = \{\mathcal{O}^1, \mathcal{O}^2, \dots, \mathcal{O}^K\}$ be the order set that includes all the orders received by the system, where K is the total number of orders in the order set, and order \mathcal{O}^i the i th order in \mathcal{Q} , for $1 \leq i \leq K$.

The order \mathcal{O}^i further contains several items requested by the customer. Let $L_i \geq 1$ be the total number of items in order \mathcal{O}^i , then $\mathcal{O}^i = \{I_1^i, I_2^i, \dots, I_{L_i}^i\}$, where I_j^i is the j th item of order i and $1 \leq j \leq L_i$. Note that each item I_j^i should be converted into a retrieval job of the corresponding stacker.

On the other hand, assume that there are N stackers in the warehouse, and each stacker will receive a set of retrieval jobs corresponding to the orders received by the system. Let $\mathbf{R} = \{\mathbf{S}^1, \mathbf{S}^2, \dots, \mathbf{S}^N\}$ be the stacker set, where \mathbf{S}^n is the set of retrieval jobs allocated to stacker n , for $1 \leq n \leq N$.

Let Z_n be the total number of retrieval jobs allocated to stacker n and J_p^n be the p th retrieval job in \mathbf{S}^n , where $1 \leq p \leq Z_n$. Then we have $\mathbf{S}^n = \{J_1^n, J_2^n, \dots, J_{Z_n}^n\}$, which represents the retrieval job sequence that stacker n needs to handle.

Note that, each retrieval job J_p^n in the stacker set \mathbf{R} is corresponding to an item I_j^i in the order set \mathbf{Q} . Such a relationship is actually a one-to-one mapping from \mathbf{Q} to \mathbf{R} .

In other words, the retrieval job scheduling problem of stackers can be formulated by two functions, $n(i, j)$ and $p(i, j)$ in such a way that there is a one-to-one mapping from I_j^i of \mathbf{Q} to $J_{p(i, j)}^{n(i, j)}$ of \mathbf{R} . Note that, $n(i, j)$ denotes which stacker is assigned to retrieve item I_j^i , and $p(i, j)$ denotes the position of item I_j^i in the retrieval job sequence of stacker $n(i, j)$. The objective of a retrieval job scheduling algorithm is then to determine the details of these two mapping functions so as to achieve the performance desired while fulfilling the retrieval requirement (e.g., shelf $n(i, j)$ must contain item I_j^i).

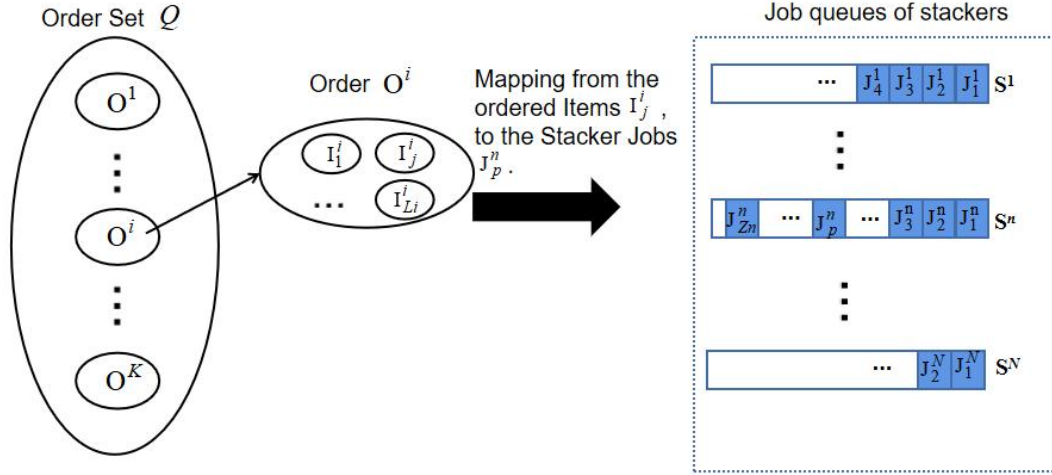


Figure 3.2.1 Mapping of the ordered items to the jobs of stackers.

Table 3.2.1 Notations and Definitions

Notation	Definition
$\mathcal{Q} = \{\mathcal{O}^1, \mathcal{O}^2, \dots, \mathcal{O}^K\}$	The set of orders received by the system, which contains K orders
$\mathcal{O}^i = \{I_1^i, I_2^i \dots I_{L_i}^i\}$	The i th order, which contains L_i items
I_j^i	The j th item of the i th order,
$\mathcal{R} = \{\mathcal{S}^1, \mathcal{S}^2, \dots, \mathcal{S}^N\}$	The set of stackers used in the warehouse, which contains N stackers
$\mathcal{S}^n = \{J_1^n, J_2^n, \dots, J_{Z_n}^n\}$	The set of jobs of the n th stacker, which contains Z_n retrieval jobs.
J_p^n	The p th job in the n th stacker,

An example is given in Figure 3.2.2, where the sixth order contains three items, and these three items will be mapped to the retrieval jobs of stackers 1, 2, and 3, respectively.

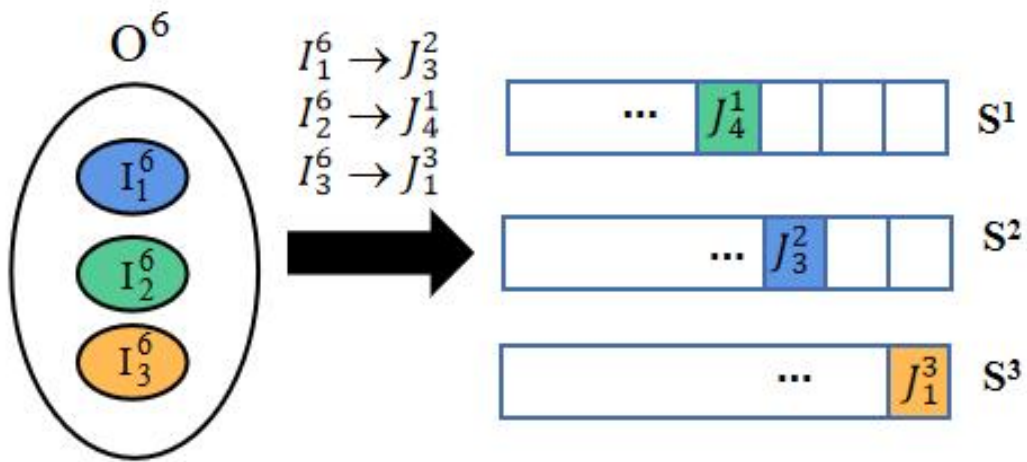


Figure 3.2.2 An example of item-job mapping.

With reference to Figure 3.2.3, the retrieval jobs of each stacker can actually be formulated as a single-server priority non-preemptive queue. In this example, $J_{p(i,j)}^{n(i,j)}$ is the fifth item in the job queue to be processed by $S^{n(i,j)}$. When a new retrieval job is assigned to the stacker, the sequence of the existing jobs in the queue may be affected and need to be re-arranged based on the recalculated priorities (which will be discussed later); however, such a queuing model is said to be non-preemptive because the job which is being executed will not be interrupted by the arrival of a higher priority job.

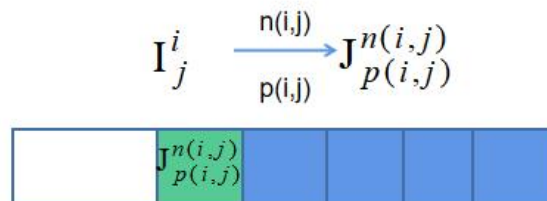


Figure 3.2.3 The job queue of a stacker.

3.3 Objective Function

Since the work presented in this dissertation focuses on the performance of ARS, in the following discussion, the term "delay" is referred to as the delay incurred

in the retrieval process unless otherwise specified.

Let the objective of the two-dimensional mapping function $(i, j) \rightarrow (n, p)$ be minimizing the average delay of the orders received by the system. Such an objective is actually equivalent to minimizing the total delay of all orders.

Considering the order integrality, the retrieval delay of an order is measured as the duration from the arrival of the order until the retrieval of the last item of the order. In other words, the delay of an order is equal to the maximum delay of the item belonging to the order, as indicated in (3).

$$\text{Delay of } O^i = \max_{1 \leq j \leq Li} (\text{Delay of } I_j^i) \quad (3.3)$$

Let τ_i be the arrival time of order O^i , and $DT(i, j)$ be the departure time of item I_j^i . Note that the departure time of item I_j^i is defined as the time when the item reaches the transfer station from the shelf. The delay of item I_j^i can then be represented by

$$\text{Delay of } I_j^i = DT(i, j) - \tau_i \quad (3.4)$$

Combining (3.3) and (3.4), we can get the following.

$$\text{Delay of } O^i = \max_{1 \leq j \leq Li} (DT(i, j) - \tau_i) \quad (3.5)$$

It should be noted that the value of $DT(i, j)$ is determined by variables n and p , and it is actually the output of the scheduler. Thus, the objective function of the scheduler can then be formulated as, finding the mapping of $(i, j) \rightarrow (n, p)$ in order to

$$\text{Minimize } \left\{ \sum_{i=1}^K \max_{1 \leq j \leq Li} [DT(i, j) - \tau_i] \right\} \quad (3.6)$$

CHAPTER 4

Static Scheduling Algorithms

In this chapter, some existing algorithms will be discussed (Sections 4.1 and 4.2), and then our order-based algorithms will be proposed (Sections 4.3 and 4.4) for the scenario of static order arrivals.

4.1 Order-Based Random Out Algorithm (OBRO)

Assume that a warehouse receives a bunch of orders from their management system, then OBRO is the simplest strategy to handle these orders because when any stacker is available, the warehouse will just randomly select an order with an item at the stacker to serve. The rest of the items of this order will also be distributed to other corresponding stackers. In this case, the sequence of orders to be served is random, and the computation cost is small.

4.2 Item-Based Shortest-Job-First Algorithm (IB-SJF)

In the Item-Based Shortest-Job-First Algorithm, the item with shorter retrieval time gets higher execution priority. In addition, if there are multiple items from the same order that will be assigned to the same stacker, then they can actually be combined as a “bigger item” for that stacker. Assume that we have only one

stacker in our smart warehouse which is the simplest case in our study. To each “item”, it is obvious that every time executing the “item” with shortest retrieval time can minimize the total delay of all the “items”. For example, there are three orders O^1 , O^2 and O^3 (suppose that they have only one item). And retrieval time is respectively 5, 6, 7 units of time. If we take the IB-SJF strategy, we will make the shortest retrieval time first and get the job sequence: I_1^1, I_1^2, I_1^3 . The delay of each order is 5, 11, 18 units of time respectively, which definitely brings the lowest order delay.

The reason that we combine the items from the same order as a “bigger item”, is illustrated in Table 4.2.1, which means consecutively completing items from the same order. Assume that we have two orders (each of them has two items) on the same stacker, the total delay of two orders brought by schedule two is always less than schedule one.

Schedule one			
I_1^1	I_1^2	I_2^1	I_2^2
Schedule two			
I_1^1	I_2^1	I_1^2	I_2^2

Figure 4.2.1 Order continuity on the same stacker

However, when it comes to multiple stackers, the IB-SJF doesn’t work as well as before. There is some space to improve them. For example, we have three orders (O^1 has two items, O^2 has one item and O^3 has three items) and three stackers. Let $RT^n(i, j)$ be the retrieval time of I_j^i from the respective stacker and assume that the retrieval time of each item is given in Table 4.2.2. If we take the IB-SJF strategy as mentioned before, we will get the schedule shows in Table 4.2.3. We find that, in the

job sequence S^l , I_1^2 needs to wait for I_1^3 and the meantime, in the S^3 , I_3^3 need to wait for I_2^1 . So, if we interchange the order of I_1^2 and I_1^3 in the S^l , the total delay will be reduced.

Table 4.2.2 Retrieval time of three orders

Order Job Sequence	O^1	O^2	O^3
S^l	0	$RT^1(I_1^2)=7$	$RT^1(I_1^3)=5$
S^2	$RT^2(I_1^1)=5$	0	$RT^2(I_2^3)=3$
S^3	$RT^3(I_2^1)=5$	0	$RT^3(I_3^3)=6$

Table 4.2.3 Schedule of three stackers

S^l	I_1^3	I_1^2
S^2	I_1^1	I_2^3
S^3	I_2^1	I_3^3

4.3 Static Order-Based Scheduling Algorithm – I (SOB-I)

4.3.1 Job Set, Schedule Sequence and Order Tag

Assume that we have two sequences for each stacker, one is *Job Set* containing the items allocated to that stacker, another is *Schedule Sequence* containing the items' execution order. The *Order Tag* of an order is the maximum delay of items in it. Since none item is selected at the beginning, all the items' waiting time is 0 and their delay is equal to their retrieval time. The items belong to the same order share the same tag (that's why they are called *Order Tag*).

4.3.2 Implementation Procedure

1. Initialize each order's Order Tag equal to the maximum retrieval time of the items in it and allocate all of items to corresponding stacker's *Job Set*.
2. Traverse every stacker to find the remaining item with smallest Order Tag.
3. Add the item selected from the *Job Set* of stacker into the *Schedule Sequence*.
4. Modify waiting time of the remaining items in the same stacker's *Job Set*.
5. If the modified waiting time of item plus its retrieval time is greater than its original *Order Tag*, change that Order Tag with this new value.
6. Check whether there are items need to be completed. If still has, return back to step 2. Otherwise, go to step 7.
7. Finish scheduling algorithm.

4.4 Static Order-Based Scheduling Algorithm – II (SOB-II)

4.4.1 Order Set

SOB-II shares the same concepts of *Job Set*, *Schedule Sequence* and *Order Tag* with SOB-I. However, its core idea is the *Order Set*, which contains all orders received by warehouse. Each iteration in the algorithm, we select one order from *Order Set* and complete its items.

4.4.2 Implementation Procedure

1. Initialize each order's Order Tag equal to the maximum retrieval time of

items in it and add all orders to *Order Set*.

2. Traverse each order in the *Order Set* to find the order with smallest order tag.
3. Add all the items contained by the order selected to the *Schedule Sequence* of the corresponding stacker.
4. Modify waiting time of the items in the same stacker's *Job Sequence*.
5. If the modified waiting time of job plus its retrieval time is greater than original Order Tag, change that Order Tag with this new value.
6. Check whether there are orders need to be completed. If still has, return back to step 2. Otherwise, go to step 7.
7. Finish scheduling algorithm.

4.5 Comparison between SOB-I and SOB-II

The core idea applied in both two greedy strategies is dynamically maintaining the Order Tag of each order. Since only if all the items contained by an order been retrieved then that order can be sent out, we need to take consideration of each part together. What's more, the items belong to different orders can affect each other, so we adjust the Order Tag each time we select items. The difference between two algorithms is that the SOB-I stands on the perspective of each stacker. Each iteration SOB-I can select one item from each stackers' *Job Sequence* (if its *Job Sequence* is not null). The SOB-II is from the angle of order entities. Each iteration we start from *Order Sequence* and find the order with smallest Order Tag. The difference for

implementation between two algorithms is that SOB-I usually only take care of the items on the same stacker whereas the SOB-II considering all the orders.

4.6 Methodology for Performance Study

In order to compare the above static algorithms, we will carry out the simulation study in the next Chapter. We will ensure that the order information processed by each algorithm is consistent in each simulation experiment, reducing the impact of other variables. To collect sufficient data from the experiment for statistical analysis, we will generate a large number of orders in the beginning of each simulation, and then we use the uniform distribution to randomly generate the number of items in each order and their shell locations. These serve as the inputs to different algorithms. After the orders being processed with different algorithms, we collect the retrieval delay of each item and of each order under these algorithms to calculate the Average Retrieval Delay and Maximum Retrieval Delay to compare the pros and cons of these algorithms.

CHAPTER 5

SIMULATION and DISCUSSION I

5.1 Simulation Settings

We make a large number of simulations with JAVA language to compare the performance among these four strategies: OBRO, IB-SJF, SOB-I and SOB-II. We simulate three cases: warehouse with 5 stackers, 10 stackers and 15 stackers. For each case, we test every strategy with 100 orders, 200 orders ... until 1000 orders and randomly generate the items amount and retrieval time for each item.

5.2 Simulation Results

Here are data tables and line charts drew from simulations. The line charts Figure 5.2.1, 5.2.2 and 5.2.3 (raw data can be found in Appendix A), which shows the trend of total delay's change with the number of orders and stacker increasing, as a result of the different scheduling outcome computed by different algorithms.

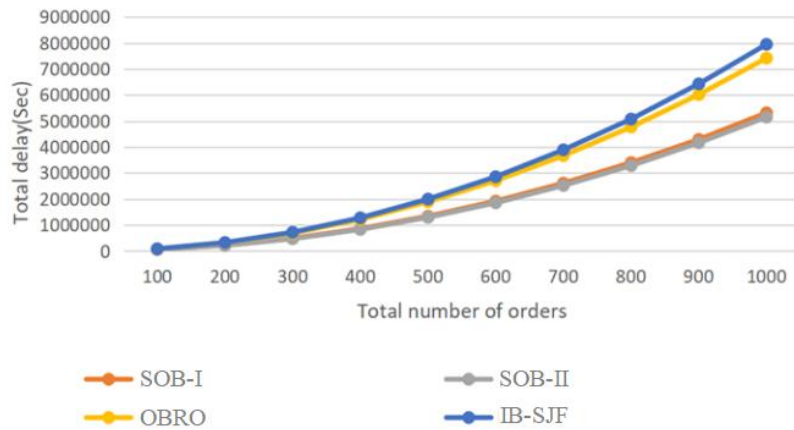


Figure 5.2.1 Total Delay in Seconds over the Number of Orders with five stackers.



Figure 5.2.2 Total Delay in Seconds over the Number of Orders with ten stackers.

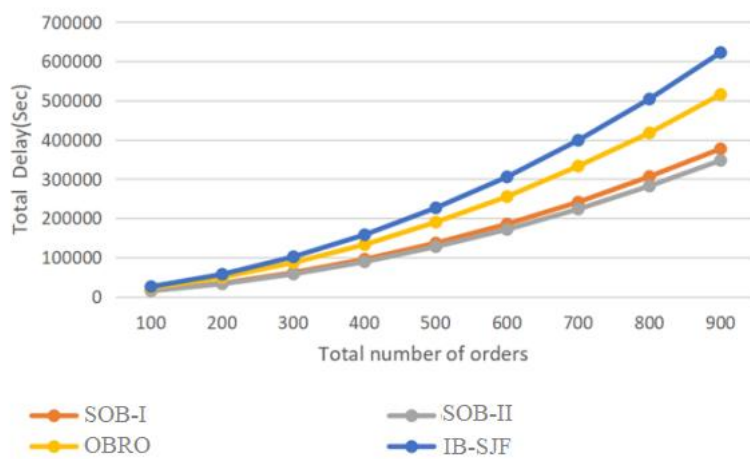


Figure 5.2.3 Total Delay in Seconds over the Number of Orders with fifteen stackers.

The simulation results from Figure 5.2.1~5.2.3 prove that the SOB-I and SOB-

II can distinctly reduce the total delay of orders. To each case, we find the SOB-II's performance is always slightly better than SOB-I. And the delay computed by SOB-I and SOB-II can always reduce 30% of delay computed OBRO and IB-SJF, no matter the number of orders is large or small. In the case of five stackers, the total delay increases sharply when the number of orders reaches 500. Comparing different cases, when we use ten stackers, the total retrieval delay can drop sharply when processing the same amount of orders with five stackers. Comparing Figure 5.2.2 and Table 5.2.3, when the amount of orders reaches to 400, the gap of total delay stands out between 10 stackers and 15 stackers.

5.3 Discussion

In this Chapter, we have proposed two Order-Based scheduling algorithms. We counted the integrality of orders and proposed the demand for minimizing the total delay of orders. Through the simulation results, we compared these four strategies: OBRO, IB-SJF, SOB-I and SOB-II. And the improvement of the algorithms we have proposed is significant. The best application scenario is when we can realize shipping each order in units which will definitely decrease total retrieval delay. This research is meaningful to those warehouses who attach importance to the satisfaction of customers. Actually, in many real warehouses, orders come dynamically not the same as the case of our research. In next Chapter, we will study the circumstance where bunches of orders dynamically arrive at the warehouse and deal with the conflicts between different batches.

CHAPTER 6

DYNAMIC SCHEDULING ALGORITHMS

With the objective function given in Chapter 4, in this Chapter we discuss some existing schedulers, such as FCFS, LCFS, SJF, etc. We then propose two new scheduling algorithms, DOB and DOBT, which are able to better approach the objective of minimizing the average delay of the orders.

6.1 First-Come-First-Serve (FCFS) Scheduler

FCFS is the simplest algorithm to perform the item-job mapping function. At each time when there are orders received by the ARS, the FCFS scheduler will always arrange the order which arrives earlier to be retrieved first. The advantages and disadvantages of FCFS are obvious. First, it has a very low time complexity. Considering that the maximum number of items in an order is L_{\max} , then the time complexity of scanning through the order to place its items into the corresponding stacker queues is $O(L_{\max})$. It should be noted that if L_{\max} is a constant value, then this scanning process has time complexity $O(1)$. Since there is no need for rearranging the retrieval job sequence when a new item arrives, the overall FCFS algorithm's complexity is $O(1)$.

However, the FCFS scheduler does not take the order integrality and overall

delay into account; thus, it is not efficient in reducing the total delay of all orders. For example, if an order (say O^i) contains an item (say I_j^i) that has a long delay time at one of the stackers (say stacker n), even if the rest of the items (say I_k^i , where $k \neq j$) can be retrieved earlier at other stackers (say stacker n' , where $n' \neq n$), it would not help as all other items (i.e. I_k^i) will still need to wait for the last item (i.e. I_j^i) to be retrieved before they can be packaged. In such a case, a more efficient scheduler should schedule the jobs from other orders (say $O^{i'}$) to be executed earlier without strictly following FCFS, aiming to reduce the total delay of all orders.

6.2 Last-Come-First-Serve (LCFS) Scheduler

LCFS is the opposite algorithm to FCFS. When a new order arrives, stackers always execute the items from this new order first. That is, LCFS schedules the job retrieval sequence based on the waiting times that the pending orders have experienced. The longer time that an order has waited, the lower priority it has to be serviced. The disadvantage of this service discipline is obvious, because the service to the pending items will be further delayed while there is a new arrival. As a result, it can be expected that LCFS will have a poor performance in terms of overall delay and fairness.

6.3 Shortest-Job-First (SJF) Scheduler

If an item has a long service (retrieval) time at a stacker, the rest of the items in the queue will have to wait and experience the same large amount of waiting time.

With this observation, SJF is a simple greedy algorithm that schedules the shortest job in each queue (stacker) to be serviced first. This is done in the hope to minimize the overall waiting time experienced by all items in the queue. Consider a sorted queue, when a new retrieval job arrives to the queue, SJF needs to insert the job to the queue based on its retrieval time. If the binary insertion algorithm is used, the complexity of SJF is $O(\log Z)$, where Z is the number of existing jobs in the queue when the new job arrives.

However, SJF does not consider the integrality of order over multiple queues. Therefore, even some items of an order have been retrieved, they will still need to wait for the last item of this order to come before they can be packaged. As a result, the overall order delay may not be improved with SJF.

6.4 Dynamic Ordered-Based (DOB) Scheduler

We design a dynamic algorithm, named the Dynamic Order-Based (DOB) Scheduling Algorithm, with the consideration of order integrality and overall delay. In our previous work, we have proved that in the case of Static Order Arrival, the Order-Based scheduling algorithm can reduce the total delay of orders significantly. We extend the algorithm to perform the scheduling task for stackers with Dynamic Order Arrival.

In general, we introduce the concept of Order Tag to the scheduling algorithm as follows. When a new order arrives to the system, it will be assigned an Order Tag so that all the items in this order will receive the same Order Tag. On the other hand,

at any time, each stacker will execute the item retrieval process based on the ascending order of the items' Order Tags in the queue. Note that once an Order Tag has been assigned to an item, the tag will remain until another new order arrives. Then the Order Tag may need to be recalculated, depending on the criteria of the scheduler.

The general flow of the Order-Based Algorithm is illustrated in Figure 6.4.1, where there are three main parallel processes. The first process is to detect whether there is a new order arriving to the system. If there is, the second process will be triggered to calculate/recalculate the Order Tags for all items in the queues and schedule the retrieval sequences of stackers accordingly. Meanwhile, the stackers will continue to retrieve items (in the third process) as long as the job queues are not empty.

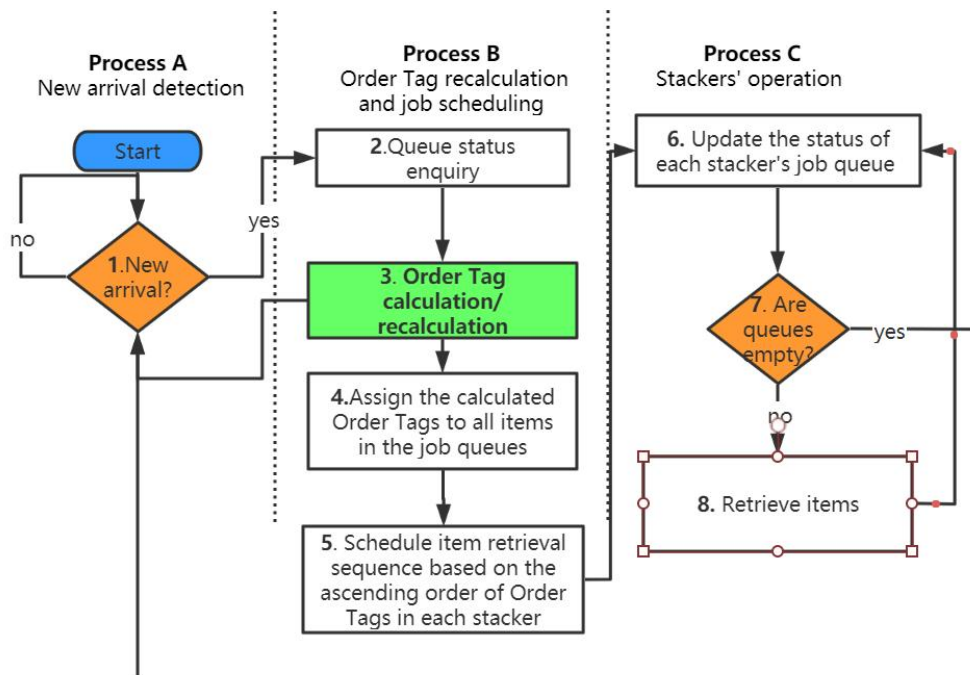


Figure 6.4.1 Flow of Dynamic Order-Based Algorithm.

The concept of Order Tag can actually be implemented with different schedulers, such as FCFS. That is, in FCFS, the Order Tag of each order is set to be

the arrival time of that order, and once the Order Tag has been assigned, it will not be changed throughout the whole retrieval process. In this case, however, even if subsequently there arrive some new items with shorter retrieval times, these new arrivals can only be placed in the queue after the existing items, and thus they may experience large delay. In other words, the performance of FCFS could be adversely affected by the items which arrive earlier but with very long retrieval times.

To resolve the problem that occurs in FCFS, we propose that when a new order arrives, other than that its Order Tag needs to be calculated, the Order Tags of the existing items in the stackers' job queues may also need to be recalculated. The objective of the Order Tag calculation/recalculation is to allow the orders that can complete the retrieval jobs earlier to get higher priority (lower Order Tag values). Thus, the Order Tag calculation/recalculation should take the following factors into account, namely the orders' arrival times, item retrieval times, and expected delays. It should also be noted that the principles of Order Tag calculation/recalculation actually determine the service disciplines, and also the complexities, of the respective schedulers.

With reference to Table 6.4.1, consider the status of the job queue of stacker n at time t . Let $\text{Residue}^n(t)$ be the residual retrieval time of the job being executed by the stacker, $\mathbf{V}^n(t)$ the set of waiting jobs in the queue, and $q^n(t) = |\mathbf{V}^n(t)|$ the number of the jobs in the queue.

Let τ_l be the arrival time of order \mathbf{O}^l , where $0 < \tau_l < \tau_{l+1}$ for all $l \geq 1$. Note that the Order Tag calculation/recalculation procedure will be triggered only at τ_l ,

then the values of Order Tags will remain the same in the time interval $[\tau_l, \tau_{l+1})$.

At time τ_l , the scheduler will first scan through the items in O^l and distribute the retrieval jobs of O^l to the corresponding stacker queues. Upon receiving the new job requests, stacker n will update its $V^n(\tau_l)$ and $q^n(\tau_l)$. Then the next scheduling task is to calculate/recalculate the Order Tags of all items in $V^n(\tau_l)$, for $1 \leq n \leq N$.

Without the loss of generality, let $I_j^i \in V^n(\tau_l)$ be a waiting job in the queue of stacker n , $RT^n(i, j)$ the Retrieval Time of item I_j^i .

To explain the entire flow of DOB, we further define other variables as shown in Table 6.4.1. Note that $OrderTag(i)$ is the output of the Order Tag Calculation/Recalculation Procedure for all unfinished order O^i .

Table 6.4.1. Definitions of Variables

Variables	Definition
$Residue^n(t)$	The residual retrieval time of the job being executed by the stacker n at time t .
$V^n(t)$	The set of jobs waiting to be retrieved by the stacker n at time t .
$q^n(t)$	The number of jobs in the job queue of stacker n at time t .
τ_l	The arrival time of l th order, O^l .
$RT^n(i, j)$	The retrieval time of item I_j^i
W	The set of orders pending for Order Tag assignment
$Busy^n$	Busy Time of stacker n
$EFT^n(i, j)$	Expected Finish Time of item I_j^i
$TempOT(i)$	Temporary Order Tag of O^i , a running variable
$OrderTag(i)$	Order Tag of O^i , the output of the Order Tag Calculation/Recalculation Procedure

6.4.1 Flow of DOB Scheduling Algorithm

With reference to Figure 6.4.1, the following is the flow of the DOB scheduling algorithm.

- i. The scheduler continuously detects whether there is a new order arriving.
- ii. Assume that there is a new order O^l arriving at time τ_l .

- iii. The items of O^i will be scanned and mapped to the corresponding stackers' job queues, and the scheduler updates the values of $V^n(\tau_i)$ and $q^n(\tau_i)$ accordingly.
- iv. The scheduler then initializes the running variables for the Order Tag Calculation/Recalculation Procedure as follows.

Procedure 1 Initialization of the Statuses of
Stacker Job Queues

1: $\mathbf{W} \leftarrow$ The set of unfinished orders
 2: For $n \leftarrow 0$ to N
 3: $\text{Busy}^n \leftarrow \text{Residue}^n(\tau_i)$

- v. The scheduler invokes the Order Tag calculation/recalculation as follows.

Procedure 2 Order-Tag Calculation/Recalculation

1: While $\mathbf{W} \neq \phi$ do
 2: For each i where $O^i \in \mathbf{W}$
 3: $\text{TempOT}(i) \leftarrow 0$; // Initialize the temporary order tag of O^i
 4: For $j \leftarrow 1$ to L_i // Update the temporary order tag of O^i
 5: If I_j^i has not been retrieved
 6: then $n \leftarrow$ The stacker of I_j^i ;
 7: $\text{EFT}^n(i, j) \leftarrow \text{RT}^n(i, j) + \text{Busy}^n$;
 8: $\text{TempOT}(i) \leftarrow \max_{n, j} \{\text{TempOT}(i), \text{EFT}^n(i, j)\}$
 9: End For
 10: End For
 11: $s \leftarrow 0$; $\text{TempOT}(s) \leftarrow \infty$ // Initialize the order with min TempOT
 12: For each i where $O^i \in \mathbf{W}$ // Identify the order with min TempOT
 13: If $\text{TempOT}(i) < \text{TempOT}(s)$
 14: then $s \leftarrow i$
 15: End For
 16: $\text{OrderTag}(s) \leftarrow \text{TempOT}(s)$ //Confirm the order tag of O^s
 17: For $j \leftarrow 0$ to L_s // Update queue statuses
 18: If I_j^s has not been retrieved
 19: then $n \leftarrow$ The stacker of I_j^s ;
 20: $\text{Busy}^n \leftarrow \text{Busy}^n + \text{RT}^n(s, j)$
 21: End For
 22: $\mathbf{W} \leftarrow \mathbf{W} - \{O^s\}$
 23: End While

- vi. Arrange each stacker to retrieve items according to the ascending order of Order Tags of the item.
- vii. Each stacker retrieves items according to the scheduled sequence from the Scheduler.
- viii. Every time when an item has been retrieved from stacker n , the scheduler will update $V^n(t)$ and $q^n(t)$. Furthermore, the scheduler will also check whether the corresponding order is completed. If yes, then remove the completed order from the set of unfinished orders.
- ix. If $q^n(t) \neq 0$, the stacker n continues the retrieval process.

6.4.2 Discussion of DOB

Assume that we have \hat{K} unfinished orders in the system when a new order arrives. Since we need to do \hat{K} rounds of Order Tag Calculation/Recalculation, and in each round the Scheduler needs to find the order with the smallest Order Tag among the \hat{K} orders, the time complexity of DOB algorithm is then $O(\hat{K}^2)$.

With the consideration of order integrality, DOB can significantly improve the overall retrieval delay of orders. This is verified by the simulation results presented in Chapter 7. However, an issue is that some of the orders may incur a huge retrieval delay. It is because DOB always sacrifices those orders that contain items with a large retrieval time, so much so that DOB will keep delaying their retrieval process by assigning them large Order Tags. In this case, starvation occurs. This is undesirable because if the delays of some orders are too large, it will extremely affect the experience of a small group of customers. In other words, DOB actually lacks the fairness among all orders.

6.5 Dynamic Order-Based with Threshold (DOBT) Scheduler

In order to resolve the starvation issue, we further propose an algorithm named Dynamic Order-Based with Threshold (DOBT) Scheduling algorithm. The flow of DOBT is quite similar to that of DOB. The only difference is that DOBT introduces a threshold limit in the Order Tag Recalculation Procedure to limit the maximum waiting time of so as to provide a certain degree of fairness among orders. Through the experiments shown in Chapter 7, we find that with an appropriate threshold value,

DOBT can solve the problem of large maximum delay very well. It should also be noted that the time complexities of DOB and DOBT are the same.

6.6 Methodology for Performance Study

In the simulation experiment, we will use the Poisson process to randomly generate a large number of orders and the corresponding arrival times, and this process is different from the “Static Order Arrival” case. For each order, we use uniform distributions to randomly generate the number of items and the shell location of each item. These serve as the inputs to different dynamic scheduling algorithms. We collect the retrieval delay value of each item and of each order under these algorithms to calculate Average Retrieval Delay and Maximum Retrieval Delay to compare the pros and cons of these algorithms. At the same time, we also collect downstream pressure data to study the possible downstream pressure caused by different algorithms from the statistical data, thus highlighting the advantages of our proposed algorithm. The pressure generated by each algorithm to the downstream process is measured by the downstream item pressure and order pressure. In all experiments, under the condition of the same order rate, the order information processed by each scheduler is completely consistent, so as to control the deviation of the experimental results.

CHAPTER 7

SIMULATION and DISCUSSION II

7.1 Simulation Models

In this section, we discuss the simulation models for our experiments to evaluate the performances of the scheduling algorithms discussed in Section V. To implement the discrete event simulation, we need to find a proper way to create new orders, simulating the scenario where customers place their orders at different times to the warehouse. Furthermore, each order may contain one or several items. On the other hand, each item is stored in a storage unit with a specific coordinate in the shelf.

7.1.1 Random Model for Generating Order Arrivals

Assume that the system time starts from $t = 0$. The arrival times of these orders to the ARS are denoted by τ_1, τ_2, \dots , respectively. We assume the arrival of orders is a Poisson process with an arrival rate of λ . Then the interval between two consecutive arrivals of orders is a random number with an exponential distribution with mean $1/\lambda$, and this can be used to generate random numbers to create stochastic order arrivals.

7.1.2 Random Model for Generating Item Quantity

Each order from a customer may contain various items. Let the number of

items in an order be modeled by a random variable, L , with a binomial distribution in the range of $[1, L_{max}]$, where L_{max} is the maximum number of items that an order may have. Then the probability that there are r items in an order is denoted by $P(L = r)$, where

$$P(L = r) = C_{r-1}^{L_{max}-1} p^{r-1} (1-p)^{L_{max}-r} \quad (7.1)$$

Note that the expected number of items in an order, $E[L]$, is equal to

$$E[L] = (L_{max} - 1) \times p \quad (7.2)$$

With (7.1) and (7.2), the value of p can be identified if the values of L_{max} and $E[L]$ are given. For example, if $L_{max} = 30$ and $E[L] = 10$, then p is equal to 0.345.

It should also be noted that, combining the order arrival model and item quantity model, the item arrival rate is equal to $\lambda \cdot E[L]$. This value should not exceed the maximum throughput of the ARS, or the system will saturate, resulting in buffer overflow.

7.1.3 Random Models for Generating Shelf Locations and Retrieval Times

Before calculating the retrieval time of an item, we need to find out the shelf and the coordinate of the storage unit where the item is stored.

To simplify the simulation, we assume that each item is randomly distributed over any of the N shelves, and its coordinate is evenly and randomly distributed over all storage units of the shelf.

With reference to (3.2) shown in Chapter 3, the values of h , w , v_x , v_y , and T are

all constant in a simulation; and the values of x and y can be randomly generated for each item.

Let $E[F]$ be the average retrieval time of items. Then the maximum throughput of the ARS is given by $N/E[F]$. Thus, to avoid saturation, the order arrival rate must fulfil the following condition in any simulation experiment.

$$\lambda < \frac{N}{E[L] \cdot E[F]} \quad (9)$$

7.2 Simulation Settings

In our simulation, we assume that each shelf is of a fixed size of $40\text{m} \times 20\text{m}$. On the other hand, each storage unit has a size of $2\text{m} \times 1\text{m}$. In other words, each shelf contains 400 storage units.

We have two main experimental settings: one is in the warehouse with 6 shelves, and the other with 12 shelves. Note that, each shelf is associated with one and only one stacker. The experiments are conducted with these two settings in order to compare the performances of DOB and DOBT against FCFS, LCFS, and SJF algorithms. In the next Subsections, we comprehensively study these algorithms under various values of order rates, running times, and thresholds.

7.3 DOB VS. other Schedulers

In this subsection, we compare the performances of the DOB, FCFS and SJF algorithms. Since we need to simulate the dynamic arrival of orders at the warehouse,

different order arrival rates must be taken into consideration in our experiments. In the case of 6 stackers, the range of order rate is set between 20 orders/hour and 60 orders/hour; in the case of 12 stackers, the range is set between 90 orders/hour and 120 orders/hour. The different ranges of order arrival rates are considered for the two settings because we need to prevent the system overflow in the respective settings, while stress testing the algorithms. In addition, in our simulations, we consider the following performance metrics: (A) Average Retrieval Delay of Orders, (B) Maximum Retrieval Delay of Orders, and (C) Downstream Backlog Pressure.

Downstream Backlog Pressure is defined as the pressure generated from the started-but-unfinished orders to the packaging stations. It is because when the retrieval process of an order has begun, the corresponding packaging station needs to wait until the last item of the order is retrieved, only then the packaging station can start to do the packaging. Throughout this period of time, all other items than the last one will need to wait at the packaging station, which creates pressure on the packaging area's buffer zone.

To quantify the measure, we consider the Downstream Backlog Pressure from two aspects: (i) order quantity pressure and (ii) item quantity pressure. The order quantity pressure is referred to as the number of pending orders at any time in the packaging area, and the item quantity pressure is referred to as the number of pending items at any time in the packaging area. Both order and item quantity pressures are meaningful to reflect the performances of different algorithms on the pressure to the

packaging stations. Besides, we also study these two metrics from the perspectives of the average pressure value and the maximum pressure value.

7.3.1 Average Retrieval Delay of Orders

Figure 7.3.1.1 and Figure 7.3.1.2 compare the performances of DOB, FCFS, SJF and LCFS with different order rates.

Although the numbers of stackers are different in Figure 7.3.1.1 and Figure 7.3.1.2, they reflect the same phenomenon. It can be found that when there are only 6 stackers and the order rate reaches 60 orders/hour, the average delays of FCFS, SJF and LCFS are 764s, 1440s and 1514s respectively, whereas the DOB just needs 477 seconds, a decrease of 37% compared with FCFS. Similarly, when we have 12 stackers and the order rate reaches 120 orders/hour, the average delay of DOB also drops by about 39% compared with FCFS.

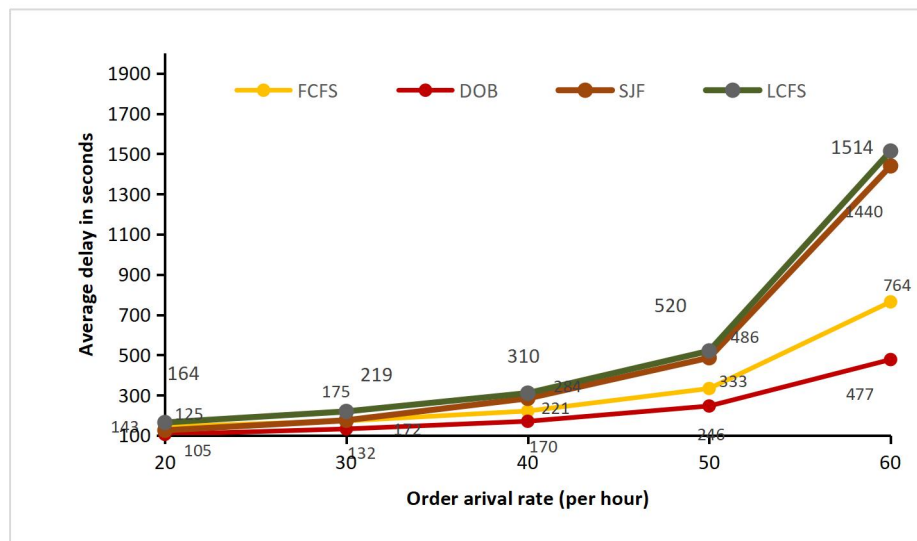


Figure 7.3.1.1 Comparison of average delays with 6 stackers

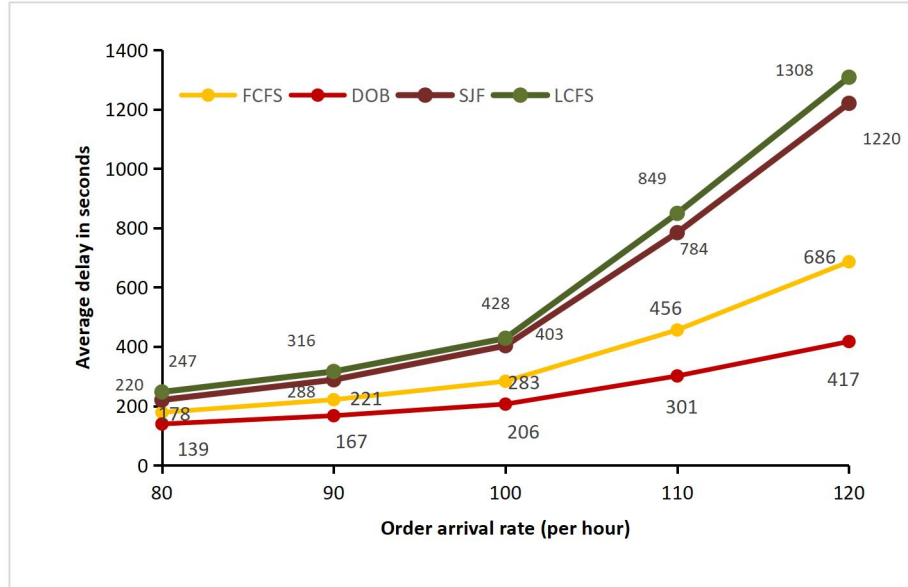


Figure 7.3.1.2 Comparison of average delays with 12 stackers

Regardless of 6 stackers or 12 stackers, when the order rate is relatively small, for example, in the 6-stacker setting, when the order rate is less than 40 orders/hour, there is no noticeable gap in average delays brought by the algorithms. When the order rate gets even smaller, the performance of DOB gets closer to FCFS, but it is always slightly better than FCFS. On the other hand, the higher the input pressure (order rate/stacker sum), the more pronounced the advantages of DOB algorithm over the FCFS. Moreover, the average delays of SJF and LCFS are always worse than FCFS and DOB.

7.3.2 Maximum Retrieval Delay of Orders

The DOB has a better performance than FCFS in terms of Average Retrieval Delay, because the DOB algorithm will first execute the items that belong to the orders with small Order Tags and delay the execution of other orders with big Order Tags. That means it will sacrifice a small group of orders in order to improve the

overall delay experienced by other orders. However, this is not actually fair because the Order Tags are calculated based on the locations of the items of these orders in the shelf storage from the ground point.

To observe how DOB may sacrifice this small group of orders, we also take the Maximum Retrieval Delay into consideration in our simulation. In Figure 7.3.2.1. Under the condition of 12 stackers, we found that when the order rate is low, the maximum delay of the DOB algorithm is a little higher than the maximum delay of the FCFS algorithm. However, as the order rate grows, the gap between FCFS's and DOB's curves has multiplied. When the order rate reaches 120 orders/hour, the maximum delay of DOB is 27738s, but the maximum delay of FCFS is only 2718s. Although DOB reduces the average delay, it also terribly increases the maximum waiting time. It should be noted that the maximum delays of LCFS and SJF are even worse than DOB.

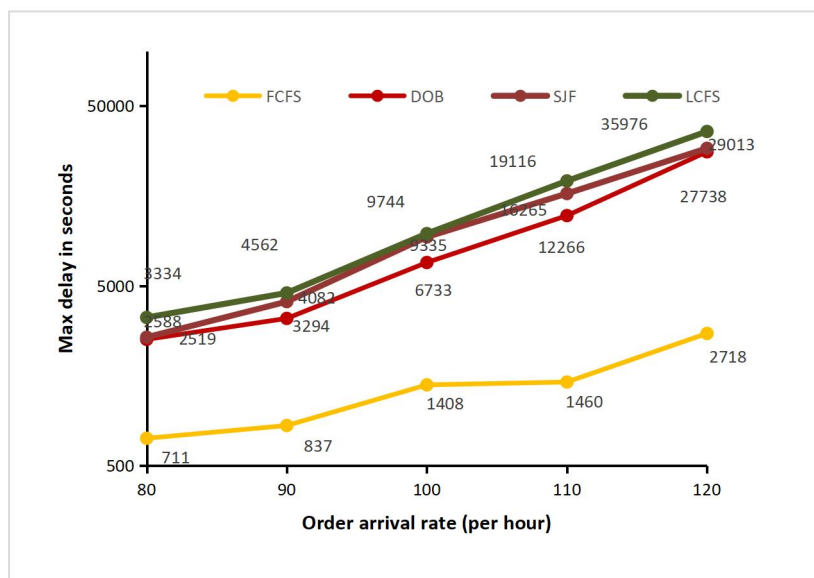


Figure 7.3.2.1 Comparison of max delays with different order rates

7.3.3 Downstream Backlog Pressure

One of the reasons that the retrieval process plays an important role in warehouses is that it affects subsequent processes, such as the Packaging Area. The downstream backlog pressure actually comes from the integrality of order. Because only when all of the items belonging to the same order arrive at the Packaging station, they can be packaged and sent to the next place.

In the following figures, the quantity of the pending orders and the quantity of the pending items in the Packaging Area will be used to measure the downstream backlog pressure. Note that we show only the curves for DOB, FIFO, and SJF, but not LCFS, because the downstream backlog pressures of LCFS is significantly larger than the others, and thus it is not suitable to be put in the same figure for comparison.

As shown in Figure 7.3.3.1 and Figure 7.3.3.2, we can find that as the order rate increases, the FCFS algorithm's average order pressure and maximum order pressure are getting higher and higher than the DOB algorithm. This also shows that the DOB algorithm can not only reduce the average delay, but also effectively relieve the downstream backlog pressure. When it comes to the item quantity, shown in Figure 7.3.3.3 and Figure 7.3.3.4, it more clearly reflects that the DOB algorithm can effectively relieve downstream backlog pressure. It should also be noted that SJF always has worse downstream backlog pressure than FCFS and DOB do.

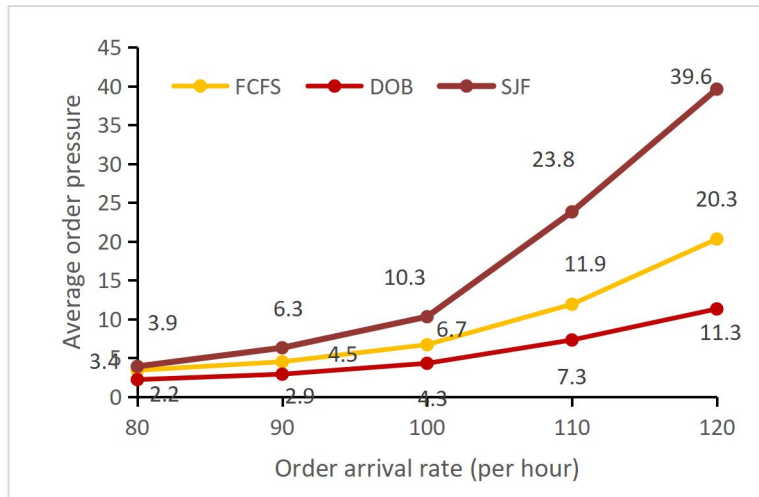


Figure 7.3.3.1 Comparison of average order pressures with different order rates

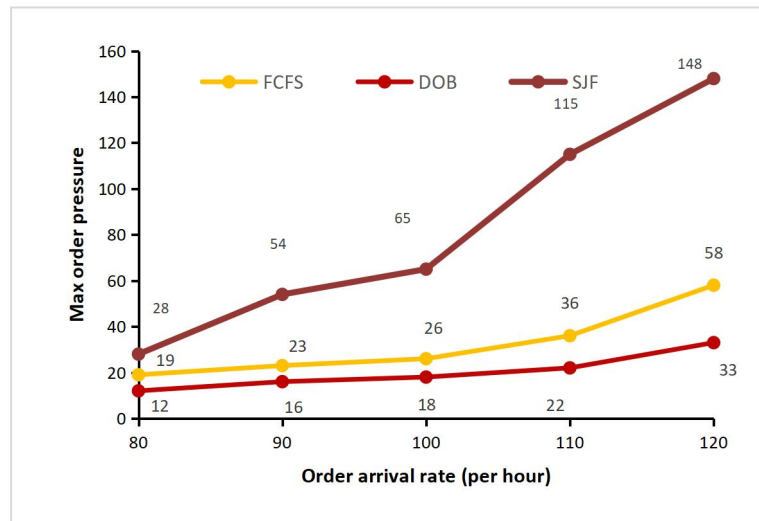


Figure 7.3.3.2 Comparison of max order pressures with different order rates

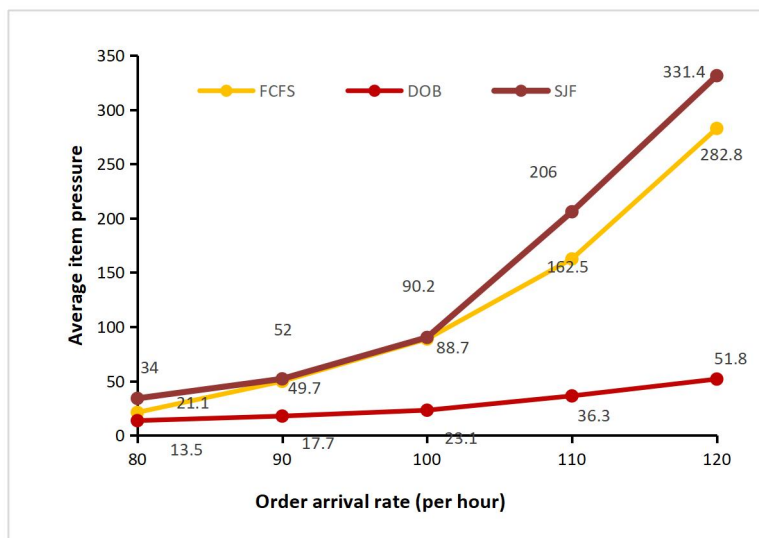


Figure 7.3.3.3 Comparison of average item pressures with different order rates

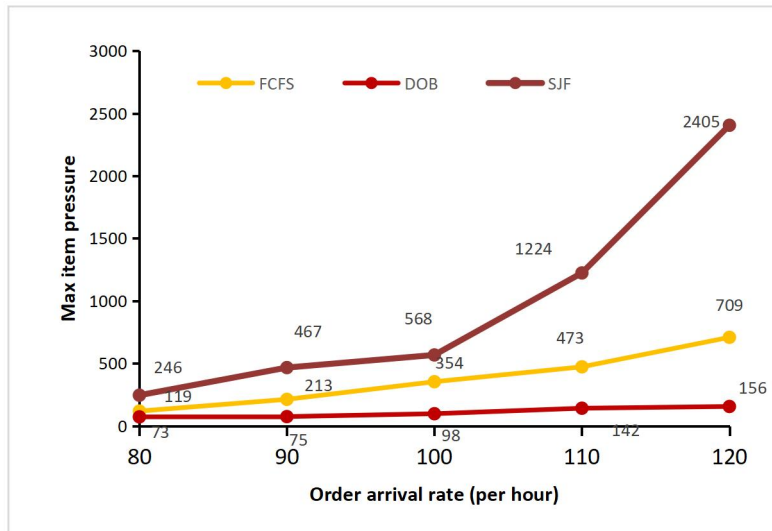


Figure 7.3.3.4 Comparison of max item pressures with different order rates

7.4 DOBT vs. DOB and FCFS

As the DOB algorithm improves the average delay and reduces the downstream backlog pressure, it also brings a huge maximum delay at the same time. That means, if the system schedules the orders based purely on DOB, there will be some customers who may need to wait 5 times longer than the waiting time incurred in FCFS. This will cause a horrible customer experience. To fix this problem, we further introduce the “threshold” to limit the waiting time of orders. When some orders that have waited longer than the threshold, the DOBT algorithm will arrange these orders to be executed first. This will also provide a certain degree of fairness among the orders.

To simplify the notation, we substitute threshold with “th”. That is, “th-400” means that we set a threshold of 400 seconds to all orders.

Table 7.4.1 records the experiment data, and Figure 7.4.1 shows the corresponding curves of the average delays of the algorithms. With the order rate

increasing, all the curves are in an upward trend. The curve of FCFS is always higher than the others', whereas the curve of DOB is at the bottom all the time. However, the curve of th-100 almost coincides with the curve of FCFS, because when the threshold is set to be very small, the Order Tag will not be effective. Besides, when the threshold is larger, the algorithm performance is closer to DOB. When the order rate reaches 120 orders/hour, the curve of th-2000 is much closer to the curve of DOB than the curve of th-400. The curve of FCFS and the curve of DOB are the upper and lower bounds of the cures with thresholds respectively.

Table 7.4.1 Average Delays of Different Algorithms with Different Thresholds

rate	FCFS	DOB	th-100	th-400	th-1000	th-2000
80	177	137	156	140	138	137
90	221	167	203	178	169	167
100	272	199	257	224	204	199
110	442	295	433	326	326	303
120	715	426	710	674	574	489

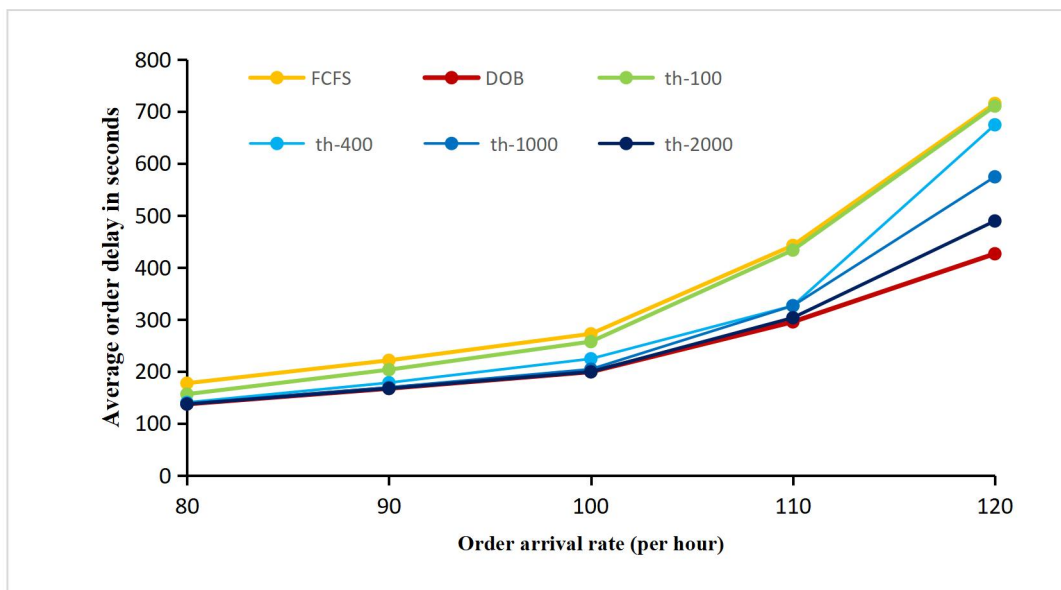


Figure 7.4.1 Average Delays of DOBT with different threshold values.

As shown in Figure 7.4.2, to compare the maximum delay of DOBT with different threshold values, we set the order rate to be 110 orders/hour. We experiment it with the threshold values from 400 seconds to 4000 seconds. It is not difficult to find that with a small threshold value, DOBT can well control the max delay of orders, approaching the performance of FCFS. In other words, DOBT is good at reducing the maximum waiting time with an appropriate setting of the threshold value, yet providing an acceptable average delay performance.

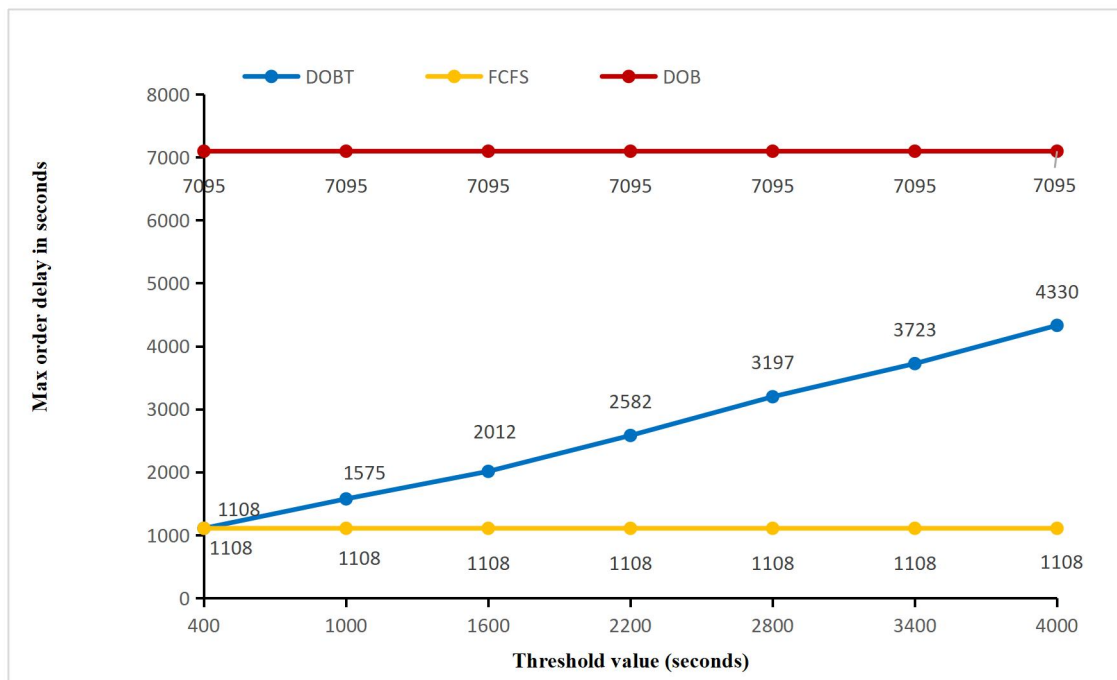


Figure 7.4.2 Max order delays of DOBT under different threshold values ($\lambda=110$ orders/hour).

Combining the observations drawn from Figure 7.4.1 and Figure 7.4.2, we can conclude that with the increasing of threshold value, the average order delay of the DOBT algorithm is getting closer to the performance of the DOB algorithm. Besides, the maximum order delay of DOBT is close to the maximum order delay of FCFS when the threshold is small. In other words, maximum order delay and average order

delay are at the opposite sides of the balance, and the threshold value a parameter that one can adjust in order to get the expected performance for an optimal warehouse operational plan.

Figure 7.4.3 – Figure 7.4.6 shows the downstream backlog pressure of DOBT, FCFS, and DOB. The similar phenomenon can be observed. The higher threshold value of DOBT, the closer performance to the DOB; and when the threshold value is small, the performance of DOBT is close to the FCFS's.

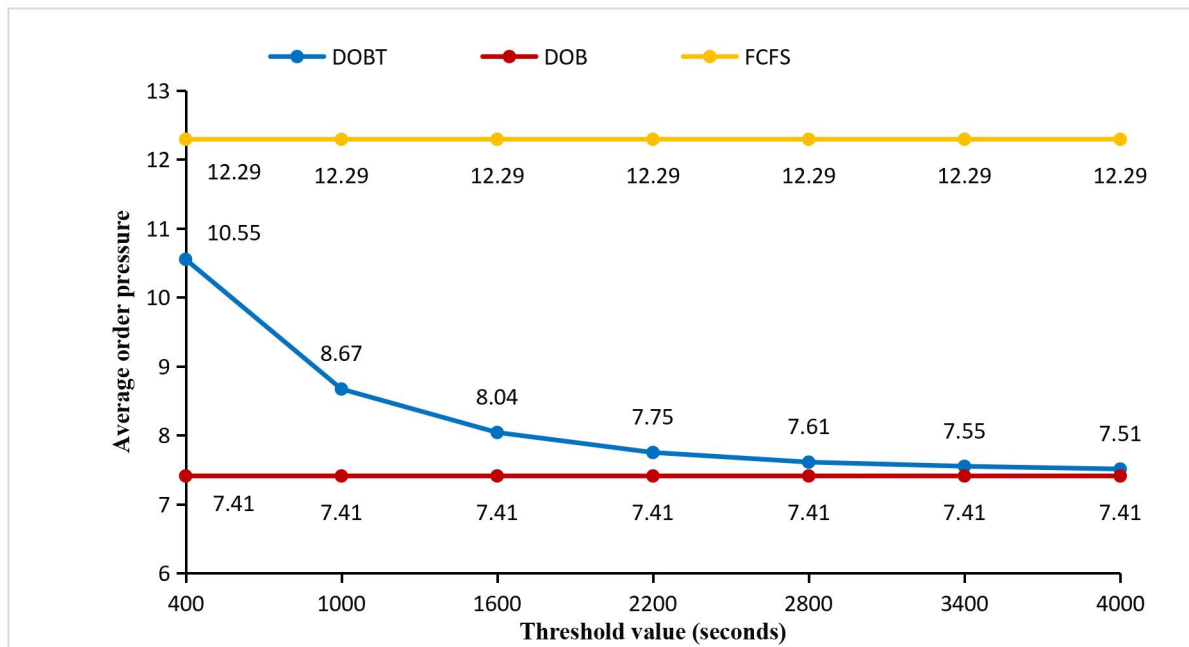


Figure 7.4.3 Average order pressures of DOBT under different threshold values ($\lambda=110$ orders/hour).

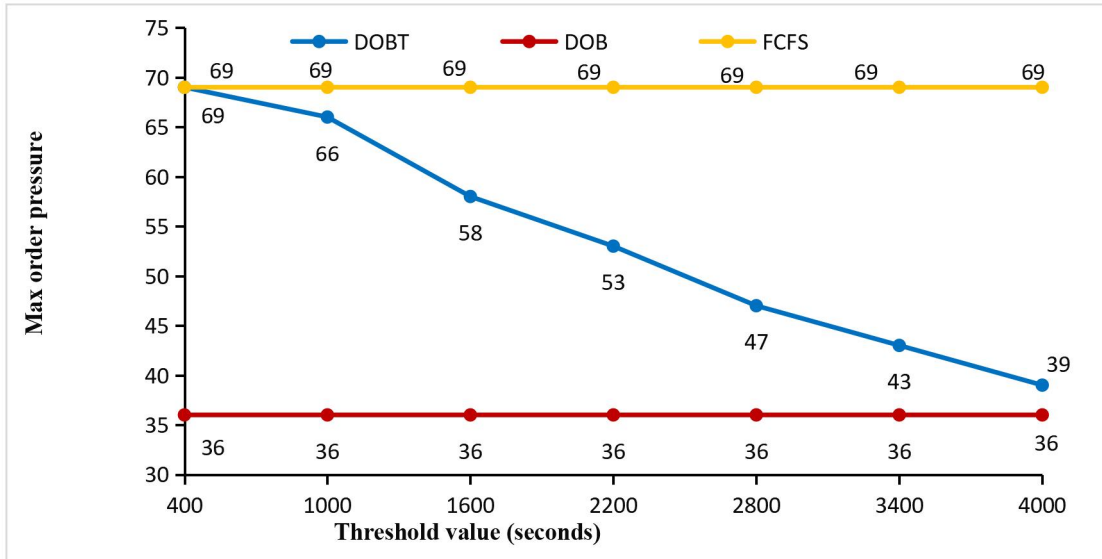


Figure 7.4.4 Max order pressures of DOBT under different threshold values ($\lambda=110$ orders/hour).

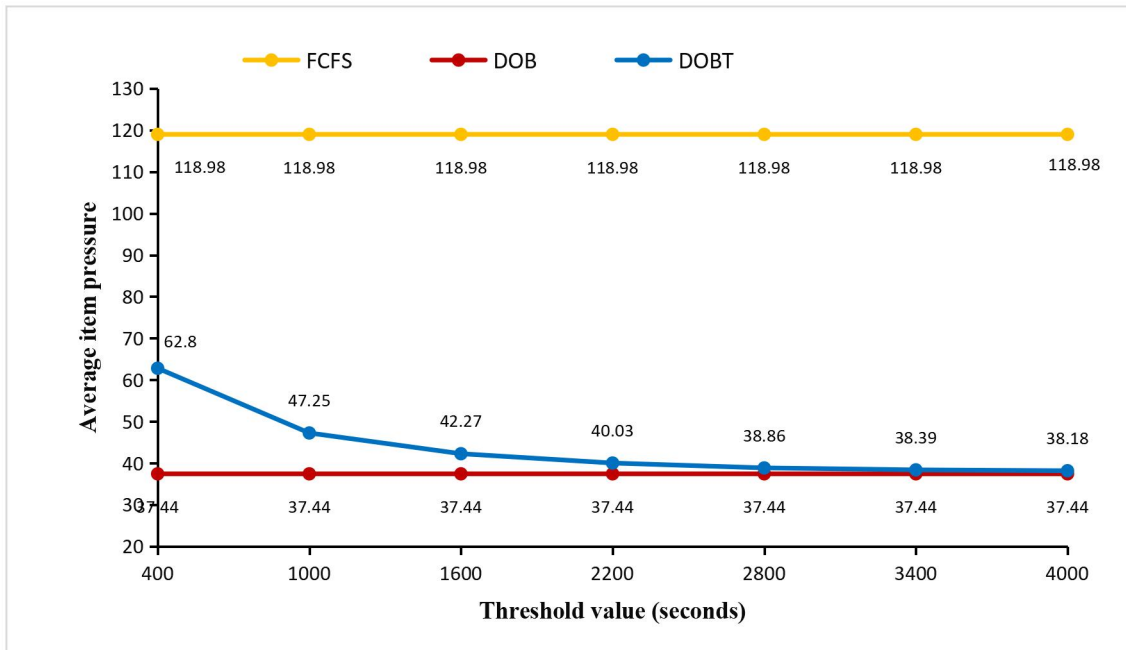


Figure 7.4.5 Average item pressures of DOBT under different threshold values ($\lambda=110$ orders/hour).

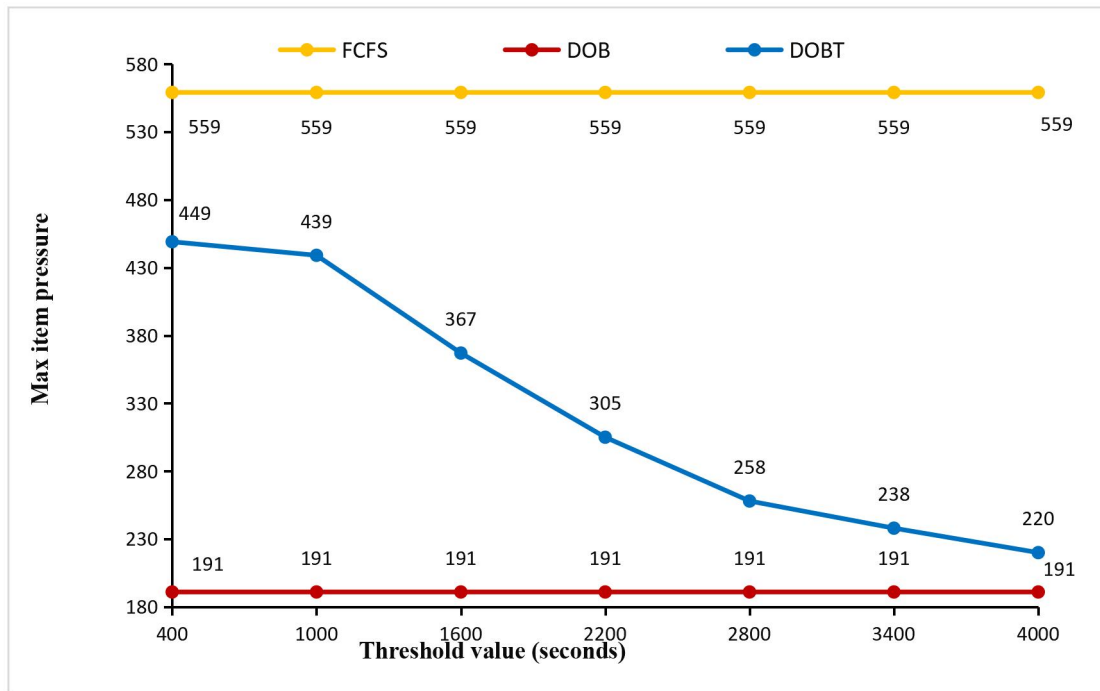


Figure 7.4.6 Max item pressures of DOBT under different threshold values ($\lambda=110$ orders/hour).

7.5 Discussion

In this Chapter, we have considered Dynamic Order Arrival and studied the corresponding scheduling algorithms of ARS to schedule the retrieval jobs of stackers in smart warehouses. In particular, we proposed using “Order Tag” to enable the ARS to consider the integrality of orders, while scheduling the retrieval jobs dynamically. Through our study, we found that two of the performance metrics, average order retrieval delay and max order retrieval delay are difficult to be co-optimized at the same time. Thus, we introduced a threshold to the scheduling algorithm to balance the two performance metrics. The simulation results shows that the Dynamic Order-Based with Threshold Algorithm (DOBT) can narrow the maximum order delay significantly through a small sacrifice of average order delay. Other than the delay performance, one of the important contributions of our work is to alleviate the

downstream backlog pressure in the Packaging Area, because both DOB and DOBT can reduce the number of pending orders and items significantly compared with FCFS and other algorithms.

CHAPTER 8

CONCLUSION

This chapter summarizes the algorithms proposed in this dissertation, highlights the contributions and discusses the possible future works.

8.1 Proposed Approaches and Achievement

In the past, the integrality of order has not received much attention for the parallel retrieval process of multiple stackers. To take this into account, this dissertation proposes using an Order Tag to label all the items that belong to the same order for retrieval job scheduling. The way of calculating the Order Tags will then determine the scheduling discipline of the ARS.

In order to study how the Order Tags can be incorporated into the scheduling process of ARS, this dissertation investigates the implementation of Order Tags under two scenarios. First, we verify whether the Order Tag strategy can reduce the overall delay in the case of "Static Order Arrival". "Static Order Arrival" means that the smart warehouse has received all the orders before the scheduling starts, and no new orders will be added during the processing of orders. For example, some warehouses set a time node so that orders received before this point are processed simultaneously.

We propose two static algorithms, namely Static Order-Based Scheduling Algorithm – I (SOB-I) and Static Order-Based Scheduling Algorithm II (SOB-II). Simulation results demonstrate that these two strategies can reduce the total retrieval delay by approximately 30% compared to the existing algorithms, such as Order-Based

Random Out Algorithm (OBRO), Item-Based Shortest-Job-First Algorithm (IB-SJF). Next, and more importantly, we study the case of "Dynamic Order Arrival". Instead of waiting for all orders to arrive before processing, the algorithm will process each new order once received, which makes the warehouse more flexible and efficient, and also has higher requirements on the scheduling algorithms. To minimize the average delay and ensuring the fairness, two algorithms are proposed. They are named as Dynamic Order-Based (DOB) and Dynamic Order-Based with Threshold (DOBT) Scheduling Algorithms, respectively. Compared with the First-Come-First-Serve and other approaches, the simulation results show that DOB and DOBT are able to reduce the average order retrieval delay by at least 30%, and generate less backlog pressure to the downstream operations. Overall, the contributions of this dissertation are meaningful to decrease the average delay of all the orders and improve the production efficiency of the Smart Warehouse, thus enhance the shopping experience of all the customers.

8.2 Future Work

One limitation of Dynamic Order-Based with Threshold (DOBT) comes with that the determination of the threshold needs to be combined with the actual situation of the warehouse. Different warehouses, in different periods, have different order rates. In the future, we can combine the actual operation data of some warehouses to find a more suitable threshold. At the same time, we only considered the retrieval stage, not the storage stage, the process of putting the goods on the shelves. The strategy to

choose an appropriate way to combine these two stages, for example, before the stacker retrieve the goods from the transfer station, take some goods to the corresponding shelves on the way, which can greatly improve the overall efficiency of the warehouse.

REFERENCES

- [1] N. Boysen, R. De Koster, and F. Weidinger, 2019. Warehousing in the e-commerce era: A survey. *European Journal of Operational Research*, 277(2), pp.396-411.
- [2] Y. Yu, X. Wang, R.Y. Zhong and G.Q. Huang, 2016. E-commerce logistics in supply chain management: Practice perspective. *Procedia Cirp*, 52, pp.179-185.
- [3] A. LARSSON, 2016, Selection of Automated Order Picking Systems, M.S. thesis, Department of Technology Management and Economics, CHALMERS UNIVERSITY OF TECHNOLOGY.
- [4] V. Aggarwal, Y.F.R. Chen, T. Lan and Y. Xiang, 2017. Sprout: A functional caching approach to minimize service latency in erasure-coded storage. *IEEE/ACM Transactions on Networking*, 25(6), pp.3683-3694.
- [5] D. Culler and J. Long, 2016. A prototype smart materials warehouse application implemented using custom mobile robots and open source vision technology developed using emgucv. *Procedia Manufacturing*, 5, pp.1092-1106.
- [6] K.N. Lemon and P.C. Verhoef, 2016. Understanding customer experience throughout the customer journey. *Journal of marketing*, 80(6), pp.69-96.
- [7] Z. He, V. Aggarwal and S.Y. Nof, 2018. Differentiated service policy in smart warehouse automation. *International Journal of Production Research*, 56(22), pp.6956-6970.
- [8] H. Zhang, Z. Guo, W. Zhang, H. Cai, C. Wang, Y. Yu, W. Li and J. Wang, 2019. Layout design for intelligent warehouse by evolution with fitness approximation. *IEEE Access*, 7, pp.166310-166317.
- [9] B. Sainathuni, P. J. Parikh, X. Zhang and K. Nan. (2014). The warehouse-inventory-transportation problem for supply chains. *European Journal of Operational Research*, 237(2), 690-700.
- [10] Z. Zhang, Q. Guo, J. Chen and P. Yuan, 2018. Collision-free route planning for multiple AGVs in an automated warehouse based on collision classification. *IEEE Access*, 6,

pp.26022-26035.

- [11] M. Saska, M. Macas, L. Preucil and L. Lhotska, 2006, September. Robot path planning using particle swarm optimization of Ferguson splines. In *2006 IEEE Conference on Emerging Technologies and Factory Automation* (pp. 833-839). IEEE.
- [12] X. Fan, X. Luo, S. Yi, S. Yang and H. Zhang, 2003, October. Optimal path planning for mobile robots based on intensified ant colony optimization algorithm. In *IEEE International Conference on Robotics, Intelligent Systems and Signal Processing, 2003. Proceedings. 2003* (Vol. 1, pp. 131-136). IEEE.
- [13] G. Sharon, R. Stern, A. Felner and N.R. Sturtevant, 2015. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 219, pp.40-66.
- [14] T. Le-Anh and M.B.M. De Koster, 2006. A review of design and control of automated guided vehicle systems. *European Journal of Operational Research*, 171(1), pp.1-23.
- [15] G. Qing, Z. Zheng and X. Yue, 2017, May. Path-planning of automated guided vehicle based on improved Dijkstra algorithm. In *2017 29th Chinese control and decision conference (CCDC)* (pp. 7138-7143). IEEE.
- [16] Y. Guo, H. Huang, F. Gao and X. Qiao, 2017, October. Improving the Performance of Multi-AGV Systems with a Dynamic Unlock Algorithm. In *2017 10th International Conference on Intelligent Computation Technology and Automation (ICICTA)* (pp. 39-43). IEEE.
- [17] Y. Huang, 2015, The Research of Automated Warehouse Order Scheduling Based on Imperialist Competitive Algorithm. M.S. thesis, South China University of Technology.
- [18] B.S.S. Tejesh and S.J.A.E.J. Neeraja, 2018. Warehouse inventory management system using IoT and open source framework. *Alexandria engineering journal*, 57(4), pp.3817-3823.
- [19] A. Gilya-Zetinov, D. Demianova and A. Khelvas, 2019, November. Palletizing for Full-automated Warehouses on the Genetics Algorithm Base. In *2019 International Conference on Engineering and Telecommunication (EnT)* (pp. 1-5). IEEE.
- [20] X. Jiang, D. Zhao and H. Xu, 2019, April. Analysis and reconstruction of Pharmaceutical Warehouse logistics delivery system. In *2019 IEEE International Conference on Smart Manufacturing, Industrial & Logistics Engineering (SMILE)* (pp. 226-229). IEEE.
- [21] X. Sun, Z. Ma, Z. Wang and C. Ai, 2017. The development of stereoscopic warehouse stacker control system based on motion controller. In *MATEC Web of Conferences* (Vol. 139, p.

00038). EDP Sciences.

- [22] S. Li, N. Qin, D. Huang, D. Huang and L. Ke, 2019. Damage localization of stacker's track based on EEMD-EMD and DBSCAN cluster algorithms. *IEEE Transactions on Instrumentation and Measurement*, 69(5), pp.1981-1992.
- [23] H. Rams, M. Schöberl and K. Schlacher, 2017. Optimal motion planning and energy-based control of a single mast stacker crane. *IEEE Transactions on Control Systems Technology*, 26(4), pp.1449-1457.
- [24] Z. Xu, 2018, Research on Control Technology of the Style of Clamp Box Stacker Crane with Light and High-Speed. M.S. thesis, Wuhan University of Technology.
- [25] C. Hang, 2018, Design and Study of A Single Colum Stacker in A Three-dimensional Warehouse. Tianjin University of Science and Technology.
- [26] Y. Lin, 2014, Research on Order Picking System Improvement for Double Fast-pick Areas. Shandong University.
- [27] J. Cheng, 2019, The Analysis and Design of Automated Stereo Warehouse System of MT Company. Southeast University.
- [28] Y. Kung, Y. Kobayashi, T. Higashi, M. Sugi and J. Ota, 2014. Order scheduling of multiple stacker cranes on common rails in an automated storage/retrieval system. *International Journal of Production Research*, 52(4), pp.1171-1187.
- [29] Y. Khojasteh and J.D. Son, 2016. A travel time model for order picking systems in automated warehouses. *The International Journal of Advanced Manufacturing Technology*, 86(5), pp.2219-2229.
- [30] X. Yang, Z. Xu, W. Jin, and F. Shu, 2021, Optimization of automatic stacker picking sequence under the mixed picking strategy. *Computer Integrated Manufacturing Systems*, 27(03),933-942.
- [31] K.W. Pang and H.L. Chan, 2017. Data mining-based algorithm for storage location assignment in a randomised warehouse. *International Journal of Production Research*, 55(14), pp.4035-4052.
- [32] Y. Wang, S. Mou and Y. Wu, 2015. Task scheduling for multi-tier shuttle warehousing systems. *International Journal of Production Research*, 53(19), pp.5884-5895.

- [33] X.T. Kong, X. Yang, K.L. Peng and C.Z. Li, 2020. Cyber physical system-enabled synchronization mechanism for pick-and-sort ecommerce order fulfilment. *Computers in Industry*, 118, p.103220.
- [34] H. Hu, L. Li and Z. Lv, 2018, July. A Novel Hybrid Algorithm for Order Picking Optimization in Automated Warehouse. In *2018 37TH Chinese Control Conference (CCC)* (pp. 3216-3220). IEEE.
- [35] G. Nastasi, V. Colla, S. Cateni and S. Campigli, 2018. Implementation and comparison of algorithms for multi-objective optimization based on genetic algorithms applied to the management of an automated warehouse. *Journal of Intelligent Manufacturing*, 29(7), pp.1545-1557.
- [36] X. Yang, Z. Xu, W. Jin, and F. Shu, 2021, The optimization problem of stacker operation sequence under compound picking strategy. *Computer Integrated Manufacturing System*, 27, pp. 933-942.
- [37] K. Liu, J. Niu, Y. Shen and S. Li, 2016, Stacker Job Path Optimization Based on Genetic Particle Swarm," (in Chinese), *Journal of Shijiazhuang Tiedao University (Natural Science Edition)*, 29(2), pp. 67-71.
- [38] L. Wencan, H. Xuli, X. Yanggao and L. Yu, 2021, March. Research on Strategies and Algorithms for Intelligent Scheduling of Tobacco Industry Stackers. In *2021 4th International Conference on Electron Device and Mechanical Engineering (ICEDME)* (pp. 279-286). IEEE.
- [39] X. Yu, X. Liao, W. Li, X. Liu and Z. Tao, 2019. Logistics automation control based on machine learning algorithm. *Cluster Computing*, 22(6), pp.14003-14011.
- [40] F. Chen, Y. Wei and H. Wang, 2018. A heuristic based batching and assigning method for online customer orders. *Flexible Services and Manufacturing Journal*, 30(4), pp.640-685.
- [41] J. Zhang, X. Wang and K. Huang, 2016. Integrated on-line scheduling of order batching and delivery under B2C e-commerce. *Computers & Industrial Engineering*, 94, pp.280-289.
- [42] C.N. Guptha, M.G. Bhaskar and V. Meghasree, 2018, December. Design of IoT Architecture for order picking in a typical warehouse. In *2018 3rd International Conference on Computational Systems and Information Technology for Sustainable Solutions (CSITSS)* (pp. 50-53). IEEE.
- [43] J. Guo, Z.B. Jiang, Y.B. Chu and N. Geng, 2012. Study of Scheduling of Parallel Multi-task

Picking Based on Limited Working Area. *Industrial Engineering and Management*, 3, pp.102-107.

[44] M. Çelk and H. Süral, 2014. Order picking under random and turnover-based storage policies in fishbone aisle warehouses. *IIE transactions*, 46(3), pp.283-300.

[45] Y.C. Ho, T.S. Su and Z.B. Shi, 2008. Order-batching methods for an order-picking warehouse with two cross aisles. *Computers & Industrial Engineering*, 55(2), pp.321-347.

[46] H. Tang, X. Cheng, W. Jiang and S. Chen (2021). Research on equipment configuration optimization of AGV unmanned warehouse. *IEEE Access*, 9, 47946-47959.

[47] J. Santos, P. Costa, L. Rocha, K. Vivaldini, A. P. Moreira and G. Veiga, (2016). Validation of a time based routing algorithm using a realistic automatic warehouse scenario. In *Robot 2015: Second Iberian Robotics Conference* (pp. 81-92). Springer, Cham.

[48] N.C. Truong, T.G. Dang and D.A. Nguyen, 2017, December. Building management algorithms in automated warehouse using continuous cluster analysis method. In *International Conference on Advanced Engineering Theory and Applications* (pp. 1068-1077). Springer, Cham.

[49] P. Gharat, C. Gori, A. Kothawade and D.K. Chitre, WEB BASED AUTOMATED WAREHOUSE MANAGEMENT SYSTEM.

[50] D. Roy, A. Krishnamurthy, S.S. Heragu and C.J. Malmborg, 2013. Blocking effects in warehouse systems with autonomous vehicles. *IEEE Transactions on Automation Science and Engineering*, 11(2), pp.439-451.

[51] D. Roy, A. Krishnamurthy, S. Heragu and C. Malmborg, 2015. Stochastic models for unit-load operations in warehouse systems with autonomous vehicles. *Annals of Operations Research*, 231(1), pp.129-155.

[52] H. Yoshitake, R. Kamoshida and Y. Nagashima, 2019. New automated guided vehicle system using real-time holonic scheduling for warehouse picking. *IEEE Robotics and Automation Letters*, 4(2), pp.1045-1052.

[53] H. Tang, X. Cheng, W. Jiang and S. Chen, 2021. Research on Equipment Configuration Optimization of AGV Unmanned Warehouse. *IEEE Access*, 9, pp.47946-47959.

[54] K. Guo, J. Zhu and L. Shen, 2020. An Improved Acceleration Method Based on Multi-Agent System for AGVs Conflict-Free Path Planning in Automated Terminals. *IEEE Access*, 9,

pp.3326-3338.

- [55] S. Liu, 2018, October. Research on Scheduling Policy of Automated Warehouse System. In *Proceedings of the 2nd International Conference on Computer Science and Application Engineering* (pp. 1-5).
- [56] B. Sun, X. Zhang, H. Qiao, G. Li and Y. Chen, 2020. Multi-type resources collaborative scheduling in automated warehouse with fuzzy processing time. *Journal of Intelligent & Fuzzy Systems*, 39(1), pp.899-910.

APPENDIX A: EXPERIMENT DATA

Total Delay in seconds of all orders with five stackers.

Order Sum	SOB-I	SOB-II	OBRO	IB-SJF
100	55449	53998	80688	81254
200	217845	210814	312561	322654
300	484411	466676	681656	718193
400	858155	825373	1209726	1272753
500	1334691	1291989	1888358	1993485
600	1915224	1849380	2699063	2852805
700	2605367	2515651	3663885	3887056
800	3395639	3280787	4765562	5072021
900	4288132	4159940	6014099	6426418
1000	5311749	5148556	7417422	7946654

Total Delay in Seconds of all orders with ten stackers

OrderSum	SOB-I	SOB-II	OBRO	IB-SJF
100	5930	5849	8712	9434
200	23185	22135	33082	36863
300	50794	48059	71562	81090
400	89868	84849	126184	144092
500	139480	131613	194601	224470
600	202516	188948	278866	323152
700	270843	254792	376218	436515
800	354100	331428	487469	567899
900	446158	418487	614874	717969
1000	551547	517069	757867	887304

Total Delay of in Seconds all orders with fifteen stackers.

Order Sum	SOB-I	SOB-II	OBRO	IB-SJF
100	4113	4106	6141	6705
200	15608	15093	22677	25834
300	34534	32872	49551	57187
400	61541	57950	86962	101882
500	95612	88959	132829	157859
600	136755	127471	189964	226378
700	185210	171443	255137	305313
800	241106	223271	332809	398594
900	306186	282098	417234	503647
1000	376772	347400	515235	622030