Warehouse Layout Revolution: AI Designs for Maximum Efficiency

Vol.3 No.3 2022

Lakshmi Narasimha Raju Mudunuri,
Valero Energy Corporation,
Senior Business Systems Design Specialist-Refining systems
Information Services
rajumudunuri17@gmail.com
USA
Accepted/Published: June 2022

Abstract:

In typical e-commerce warehouses, once customer orders are received, the purchased items must be retrieved from shelves and packaged for delivery. To automate and expedite this retrieval process, a Smart Warehouse often utilizes an Automated Retrieval System (ARS) to manage and schedule retrieval tasks. The efficiency of the ARS is crucial as it significantly affects subsequent processes. If any item in an order faces a substantial retrieval delay compared to others, the entire order may experience unnecessary delays. Traditionally, the integrity of orders has been overlooked in the parallel retrieval processes of multiple stackers. To address this, this paper proposes using an Order Tag to label all items within the same order, which is then used for scheduling retrieval tasks. The method for calculating these Order Tags influences the scheduling discipline of the ARS. To minimize average delay and ensure fairness, two new algorithms are introduced: the Dynamic Order-Based (DOB) Scheduling Algorithm and the Dynamic Order-Based with Threshold (DOBT) Scheduling Algorithm. These algorithms ensure fairness in the retrieval process, instilling confidence in the system's integrity. Simulation results demonstrate that

DOB and DOBT algorithms reduce the average order retrieval delay by at least 30% compared to First-Come-First-Serve and other methods while alleviating backlog pressure on downstream operations.

I. Introduction

With the rise of e-commerce, many merchants have turned to public warehouses to boost operational efficiency and cut business costs [1][3]. In a typical warehouse setting, after receiving customer orders, items must be retrieved from shelves and packaged for delivery. Effective management of these processes is crucial to ensuring a positive customer experience [4][5].

Customer satisfaction is influenced by various factors, but "delivery delay" is a particularly significant one that can damage a merchant's reputation and affect customer preferences. Indeed, some customers are even willing to pay more for faster delivery [6]. Although it is possible to implement a differentiated handling policy for high-priority customers to minimize delivery delays [7], the overall efficiency of warehouse operations remains essential for satisfying the majority of customers [8][9]. Consequently, optimizing every aspect of warehouse operations is necessary for improving service quality for all customers.

Traditional warehouses rely on manual labor for item retrieval, which can be error-prone and dangerous, and is not scalable due to the need for a large workforce. However, advancements in automation technology have led to the development of new facilities aimed at improving warehouse operations [10][12]. One such advancement is the Automated Stacker, which automates the retrieval of items from shelves in a Smart Warehouse [13].

To maximize storage capacity, Smart Warehouses often feature multiple parallel shelves of significant size and utilize a management system known as the Automated Retrieval System (ARS) to coordinate and schedule retrieval tasks among these parallel shelves [14][15]. The design of the ARS is critical for Smart Warehouses, as it directly affects subsequent processes. Specifically, it is important for the ARS to consider the integrality of customer orders during item retrieval. Without this consideration, delays may occur if an order cannot be packaged until all items are retrieved.

Historically, the integrality of orders has been overlooked in the parallel retrieval processes of multiple stackers. To address this issue, this paper introduces the concept of an "Order Tag" to label items belonging to the same order. These Order Tags are used to schedule retrieval sequences for each stacker to optimize performance. Essentially, the design of the Order Tags dictates the scheduling discipline of the ARS.

This paper proposes two new scheduling algorithms based on the concept of order integrality: the "Dynamic Order-Based" (DOB) algorithm and the "Dynamic Order-Based with Threshold" (DOBT) algorithm. Through simulations, it is demonstrated that DOB and DOBT significantly outperform existing methods such as First-Come-First-Serve (FCFS), Last-Come-First-Serve (LCFS), and Shortest-Job-First (SJF). Specifically, DOB and DOBT reduce the average order retrieval delay by at least 30% and decrease backlog pressure on downstream operations. Additionally, the DOBT algorithm allows for adjustment of a threshold value to control the maximum delay of orders, thus balancing fairness among all orders.

II. Related work

Many studies have explored "d ways to Penh "since the performance of Smart Warehouses. For instance, Kung et al. investigated a warehouse system incorporating multiple stackers on a common rail to minimize stacker collisions and significantly improve work efficiency [16]. Leading research in Smart Warehouse retrieval processes focuses on optimizing stacker movement paths based on item storage locations to minimize overall order completion time [17][19].

Much of this research assumes that a stacker can traverse through shelves to retrieve multiple items in one trip. These studies often model stacker retrieval jobs as combinatorial optimization problems, such as the Travelling Salesman Problem (TSP) [20][27], which allows the application of various optimization algorithms like the HGA-VNS Algorithm [21], a Combination of Free Search and Amendment Circle Algorithm [22], Genetic Algorithms [23], Non-dominated Sequencing Genetic Algorithm with Elite Strategy [24], and Genetic Particle Swarm Algorithms [25]. Li et al. examined stacker utilization in the tobacco industry. They addressed the coordination challenges of Automated Storage and Retrieval Systems through engineering test cases, demonstrating the effectiveness of their proposed algorithm [26]. Additionally, some researchers have explored different stacker aisle types. Yu analyzed the ant colony system and parthenogenetic algorithms and proposed a parthenogenetic ant colony algorithm that significantly reduces order picking times and enhances warehouse efficiency [27].

However, as the number of orders increases, these optimization algorithms can require exponential time to produce acceptable results and often do not guarantee optimal solutions within a fixed number of iterations. Furthermore, designing an optimization algorithm to achieve optimal or near-optimal solutions can be complex, with parameters needing fine-

tuning. For example, inappropriate values for inspiration and heuristic factors in Ant Colony Algorithms can adversely affect solution speed and performance.

Another limitation of existing algorithms is their focus on the stacker's travelling path for item retrieval without addressing the integral stackers across multiple parallel stackers. In practice, an order is only complete when all its items are retrieved, so even if some items are picked earlier, delays in retrieving just one item can cause unnecessary waiting times for the entire order.

This study addresses the scenario where an Automated Retrieval System (ARS) is employed in a Smart Warehouse to manage the retrieval tasks of multiple stackers. We assume that each customer order may consist of various items spread across different shelves, with each stacker able to pick up only one item at a time but capable of parallel processing to complete retrieval tasks [28]. Additionally, during the wait for the last item to be retrieved, the remaining items of the order must occupy temporary storage at downstream operations, such as a packaging station, creating a backlog and increasing system pressure.

Our approach focuses on how the item retrieval sequence across different orders and stackers impacts overall performance [29][30]. A well-designed scheduling strategy can reduce average retrieval delays and backlog pressures at downstream operations. For example, Guo et al. [31] examined order integrality in scenarios where all orders are known in advance, aiming to resolve stacker resource contention and optimize retrieval sequences using the Ant Colony Algorithm. Their experiments demonstrated about a 10% efficiency improvement over traditional optimization methods. However, they needed to account for the dynamic nature of order arrivals in real-world warehouses.

This paper extends our previous work by introducing a label-based approach named Order Tag for stacker job scheduling. Our earlier research showed that Order Tag effectively reduces average retrieval delays under static order arrivals [32]. This paper further develops this approach to handle dynamic order arrivals, proposing two dynamic order-based scheduling algorithms: DOB and DOBT. The details of these algorithms will be discussed in the following sections.

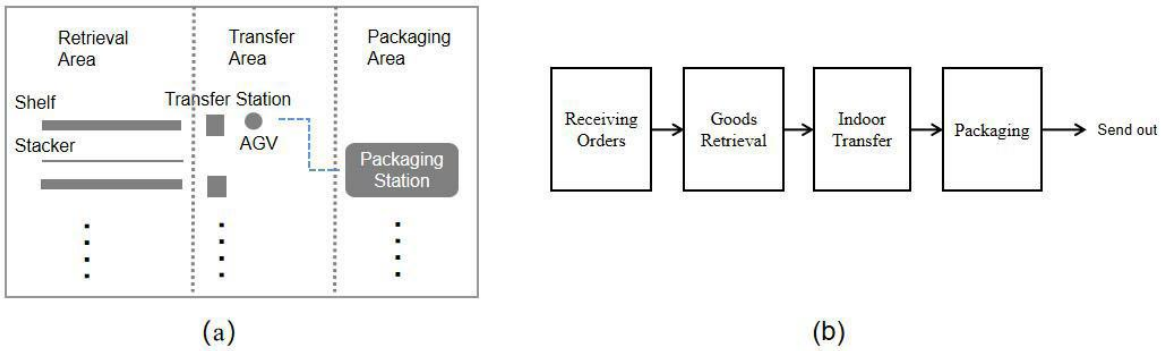

FIGURE 1. (a) Layout of the smart warehouse. (b) Workflow of the warehouse.

III. WAREHOUSE ENVIRONMENT

Typically, a Smart Warehouse incorporates multiple types of built-in facilities that must work in coordination to meet customer demands, as illustrated in Fig. 1. These facilities handle various tasks, including picking and retrieving ordered items from shelves, transferring these items to packaging stations, and packaging them into parcels for shipment. These facilities' design and layout are crucial for optimizing the efficiency of the shelf-to-package operations. Two main automated shelf-to-package approaches are commonly used, which are described as follows:

The first approach involves using heavy-duty Automated Guided Vehicles (AGVs) to lift and transport entire shelf units to the pickers at the packaging stations. In this setup, pickers

manually select the ordered items from these shelf units and package them into parcels [33][34]. This method, however, faces several challenges, such as avoiding AGV collisions [35][36]. Additionally, because the AGVs must move entire shelf units, they operate at lower speeds, which can limit warehouse throughput. The need to move all items on a shelf unit, including those not required for the current order, can lead to inefficient energy use. Moreover, the high maintenance costs of heavy-duty AGVs and limitations on the size of shelf units are notable drawbacks of this approach.

In contrast, as depicted in Fig. 1a, the second approach involves three main sub-operations [37][38]. The first sub-operation uses automated stackers to retrieve the ordered items from shelves in the Retrieval Area and transfer them to the Transfer Area. In the second sub-operation, a fleet of lightweight AGVs or a conveyor belt transports the items from the Transfer Area to the Packaging Area. The final sub-operation involves several packaging stations in the Packaging Area, where the items are packaged into parcels. This approach ensures that only the ordered items are moved, optimizing energy use and automating the entire picking and retrieval process. The workflow for this approach is summarized in Fig. 1b. This paper specifically addresses the retrieval process within the Retrieval Area.

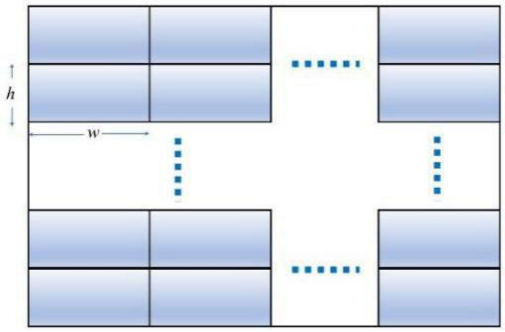

Figure 2. Structure of a shelf.

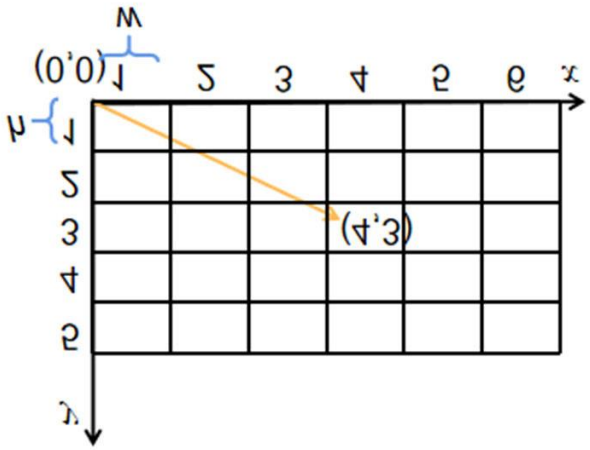

Figure 3. Coordinate map of the shelf.

Referring to Fig. 2, the Retrieval Area consists of shelves that are vertically divided into multiple levels, each with a height of h. Horizontally, each level is divided into several storage units, each with a width of w, where each storage unit holds only one type of product. Each shelf is equipped with an automated stacker system that includes a movable cargo platform and a robotic picker. The cargo platform can move to any storage unit on the shelf, allowing the robotic picker to retrieve one item at a time and transport it back to the transfer station. This setup forms the basis for the mathematical model of the retrieval process for stackers, which will be developed in the next section.

IV. MATHEMATICAL MODEL

A. RETRIEVAL TIME

Without loss of generality, let the position of a storage unit be denoted as (x, y), where x_{th} and y_{th} represent the ordinal numbers of the storage unit along the horizontal and vertical directions, respectively, on the shelf. We assume that the parking position of the stacker and the item transfer station are located at (0, 0). The stacker can move horizontally and

vertically to access any storage unit for item retrieval. For instance, as illustrated in Fig. 3, if the stacker needs to retrieve an item from the storage unit at (4, 3), it must first travel from (0, 0) to (4, 3). Once there, the stacker retrieves the item and transports it to the transfer station at (0, 0). The retrieval process is completed when the item is dropped at the transfer station. The horizontal unit distance is \(w\), and the vertical unit distance is *h*. We also assume that the stacker's horizontal and vertical travelling speeds are v_x and v_y, respectively. As shown in (1), the travel time for the stacker to move from (0, 0) to (x, y) is determined by the maximum time required to traverse along both directions.

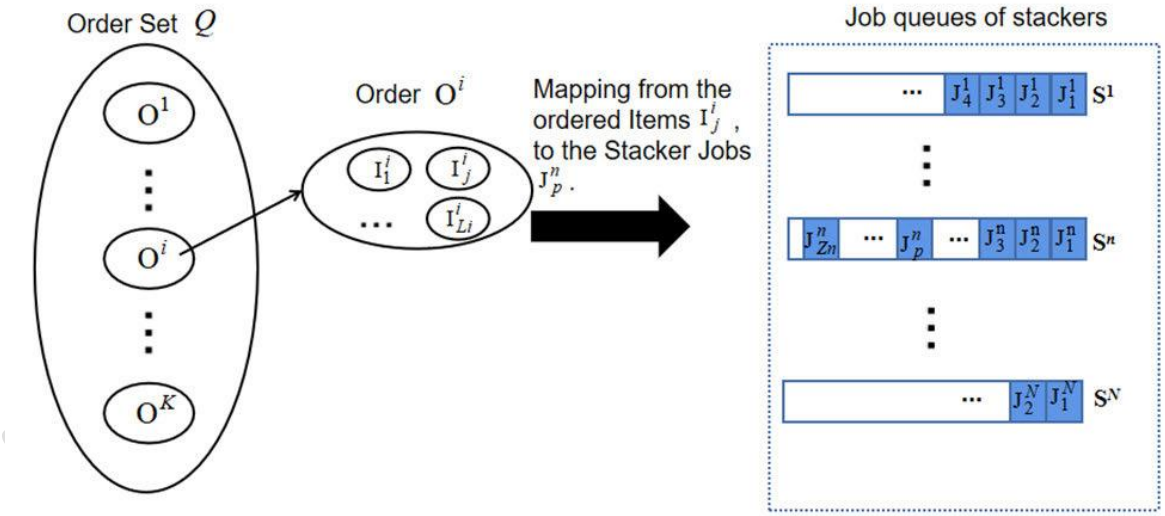

FIGURE 4. Mapping of the ordered items to the jobs of stackers.

V. Scheduling algorithms for dynamic order Arrival

With the objective function outlined in Section III, this section explores several existing scheduling algorithms, including FCFS, LCFS, and SJF. We then introduce two novel scheduling algorithms, DOB and DOBT, designed to minimize the average delay of orders more effectively.

A. FIRST-COME-FIRST-SERVE (FCFS) SCHEDULER

FCFS is the most straightforward algorithm for mapping items to jobs. Whenever the ARS receives new orders, the FCFS scheduler always processes the order that arrives first. The advantages and disadvantages of FCFS are evident. On the positive side, FCFS has a meagre time complexity. Assuming the maximum number of items in an order is L_{max}, the time complexity for scanning through the order and placing its items into the appropriate stacker queues is $O(L_{max})$. If L_{max} is a constant value, this scanning process has a time complexity of O(1)O(1)O(1). Additionally, since FCFS does not require rearrangement of the retrieval job sequence for new arrivals, the overall complexity of the FCFS algorithm remains O(1).

The FCFS scheduler does not account for order integrality and overall delay, which makes it inefficient in minimizing the total delay of all orders. For instance, if an order (O_i) contains an item (I_{ij}) with a long delay at a specific stacker (stacker n), the presence of other items (I_{ik}, where $k \neq j$) that could be retrieved earlier at different stackers (stacker n0, where $n0 \neq n$) will not reduce the total delay because all items (I_{ik}) must wait for the last item (I_{ij}) to be retrieved before packaging can occur. A more efficient scheduler would address this issue by scheduling jobs from other orders (O_i0) earlier rather than strictly adhering to FCFS, to reduce the total delay across all orders.

B. LAST-COME-FIRST-SERVE (LCFS) SCHEDULER

LCFS (Last Come, First Served) is the algorithm that is opposite FCLS (First Come, Last Served). When a new order arrives, stackers always execute the items from this new order first. In other words, LCFS schedules the job retrieval sequence based on the waiting times

of the pending orders. The longer an order has waited, the lower its priority to be serviced. This service discipline has an obvious disadvantage: the service to the pending items will be further delayed whenever a new order arrives. Consequently, LCFS is expected to perform poorly regarding overall delay and fairness.

C. Shortest-Job-First (SJF) Scheduler

When an item has a long service (retrieval) time at a stacker, the rest of the queue items will have to wait and experience the same large amount of waiting time. SJF is a simple greedy algorithm that addresses this by scheduling the shortest job in each queue (stacker) to be serviced first. This approach aims to minimize the overall waiting time for all queue items.

When a new retrieval job arrives in a sorted queue, SJF inserts the job into the queue based on its retrieval time. If a 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 integrity of orders across multiple queues. As a result, even if some items of an order have been retrieved, they will still need to wait for the last item of the order to be retrieved before they can be packaged. Consequently, the overall order delay may need to be improved with SJF.

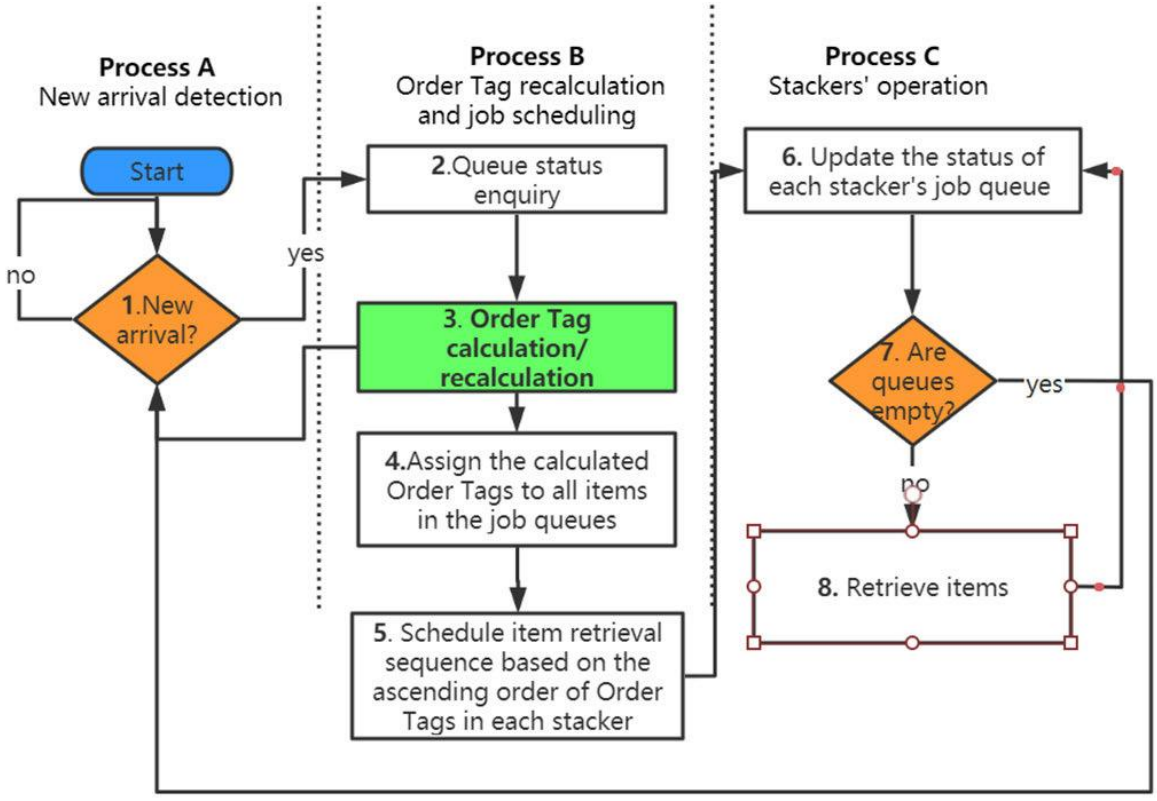

FIGURE 7. Flow of dynamic order-based algorithm.

D. Dynamic Order-Based (DOB) Scheduler

We have developed a dynamic algorithm called the Dynamic Order-Based (DOB) Scheduling Algorithm, which considers both order integrality and overall delay. In our previous work, we demonstrated that the Order-Based scheduling algorithm significantly reduces the total delay of orders in scenarios with static order arrivals. This paper extends the algorithm to handle dynamic order arrivals for stackers.

The key concept introduced in this algorithm is the highly adaptable Order Tag. When a new order arrives, it is assigned an Order Tag, which is then applied to all items within that order. Stackers execute the item retrieval process based on the ascending order of the items' Order Tags. Once an Order Tag is assigned to an item, it remains until another new order

arrives, at which point the Order Tags may need to be recalculated according to the scheduler's criteria, showcasing its flexibility.

The Order-Based Algorithm follows a systematic approach involving three main parallel processes, as illustrated in Fig. 7. The first process detects new orders arriving in the system. If a new order is detected, the second process calculates or recalculates the Order Tags for all items in the queues and schedules the retrieval sequences of stackers accordingly. Meanwhile, the stackers continue retrieving items in the third process, provided the job queues are not empty, ensuring a reliable method.

The concept of the Order Tag can be implemented with various schedulers, such as FCFS (First Come, First Served). In FCFS, the Order Tag for each order is set to the order's arrival time. Once assigned, the Order Tag does not change throughout the retrieval process. However, this approach can lead to inefficiencies. New items with shorter retrieval times can only be placed in the queue after the existing items, potentially resulting in significant delays. Thus, the performance of FCFS can be adversely affected by earlier-arriving items with long retrieval times.

To resolve the issue in FCFS, we propose that when a new order arrives, not only should its Order Tag be calculated, but the Order Tags of the existing items in the stackers' job queues may also need recalculating. Order Tag calculation/recalculation aims to prioritize orders that can complete retrieval jobs sooner, assigning them lower Order Tag values.

Variables	Definition

Residue"(t)	The residual retrieval time of the job being executed by the stacker n at time t.
V"(t)	The set of jobs waiting to be retrieved by the stacker n at time t.
q"(t)	The number of jobs in the job queue of stacker n at time t.
T1	The arrival time of /th order, 01.
RT"(i,j)	The retrieval time of item I
W	The set of orders pending for Order Tag assignment
Busy"	Busy Time of stacker n
EFT"(i,j)	Expected Finish Time of item I}
TempOT (i)	Temporary Order Tag of O', a running variable
OrderTag(i)	Order Tag of O, the output of the Order Tag Calculation/Recalculation Procedure

The Order Tag calculation/recalculation should consider the following factors: orders' arrival times, item retrieval times, and expected delays. Order Tag calculation/recalc calculation/recalculation of the service disciplines and the complexities of the respective schedulers.

1) FLOW OF DOB SCHEDULINE ALGORITHM

With reference to Fig. 7, the following is the flow of the DOB scheduling algorithm.

Procedure 1 Initialization of the Statuses of Stacker Job Queues

1: W <-The set of unfinished orders

2: For n <- 0 to N

3: $Busy^n$ $Residue^n$

4: End For

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 T_l .

iii. The items of O^l will be scanned and mapped to the corresponding stackers' job queues, and the scheduler updates the values of V^n (T_i) and $q^n(T_t)$ accordingly.

iv. The scheduler then initializes the running variables for the Order Tag Calculation/Recalculation Procedure as follows.

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

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 $q^n(t)$!=0, the stacker n continues the retrieval process.

E. DYNAMIC ORDER-BASED WITH THRESHOLD (DOBT) SCHEDULER

To address the issue of starvation, we propose an algorithm named Dynamic Order-Based with Threshold (DOBT) Scheduling Algorithm. The flow of DOBT is quite similar to that of DOB, with the critical difference being introducing a threshold limit in the Order Tag Recalculation Procedure. This threshold limits the maximum waiting time, ensuring a certain degree of fairness among orders.

Experiments detailed in Sections VI-VII demonstrate that with an appropriate threshold value, DOBT effectively solves the problem of considerable maximum delays. It is also noteworthy that the time complexities of DOB and DOBT are identical.

VI. SIMULATION MODELS

We must devise an effective method for generating new orders to implement the discrete event simulation. This involves simulating the scenario where customers place their orders at varying times to the warehouse. Additionally, each order can contain one or more items, with each item stored in a storage unit at a specific coordinate on a particular shelf.

A. ORDER ARRIVAL MODEL

Assume that the system time starts at t = 0. The arrival times of these orders to the Automated Retrieval System (ARS) are denoted by t_1, t_2…. . We assume that the arrival of orders follows a Poisson process with an arrival rate of λ Consequently, the interval

between two consecutive arrivals is a random variable with an exponential distribution having a mean of $1/\lambda$. This can be used to generate random numbers for creating stochastic order arrivals.

B. ITEM QUANTITY MODEL

Each order from a customer may contain a varying number of items. Let the number of items in an order be represented by a random variable L, which follows a binomial distribution within the range $[1, L_{max}]$, where L_{max} is the maximum number of items an order can have. The probability that an order contains r items is denoted by $P(L = r)$.

C. Shelf Location and Retrieval Time Models

Before calculating the retrieval time of an item, we need to determine 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 across any of the N shelves, and its coordinate is evenly and randomly distributed among all storage units on the shelf.

VII. SIMULATION RESULTS AND DISCUSSIONS

In our simulation, we assume that each shelf has a fixed size of 40m by 20m. Each storage unit measures 2m by 1m, meaning each shelf contains 400 storage units.

We have two main experimental settings: one with six shelves and the other with 12 shelves. Each shelf is associated with a single stacker. The experiments are conducted with these two settings to compare the performances of DOB and DOBT against the FCFS,

LCFS, and SJF algorithms. In the following subsections, we comprehensively study these algorithms under various order rates, running times, and thresholds.

A. DOB vs. Other Schedulers

In this subsection, we compare the performances of the DOB, FCFS, and SJF algorithms. We consider different order arrival rates in our experiments to simulate the dynamic arrival of orders at the warehouse. For the scenario with six stackers, the order rate ranges from 20 to 60 orders per hour. In the scenario with 12 stackers, the order rate ranges from 40 to 120 orders per hour.

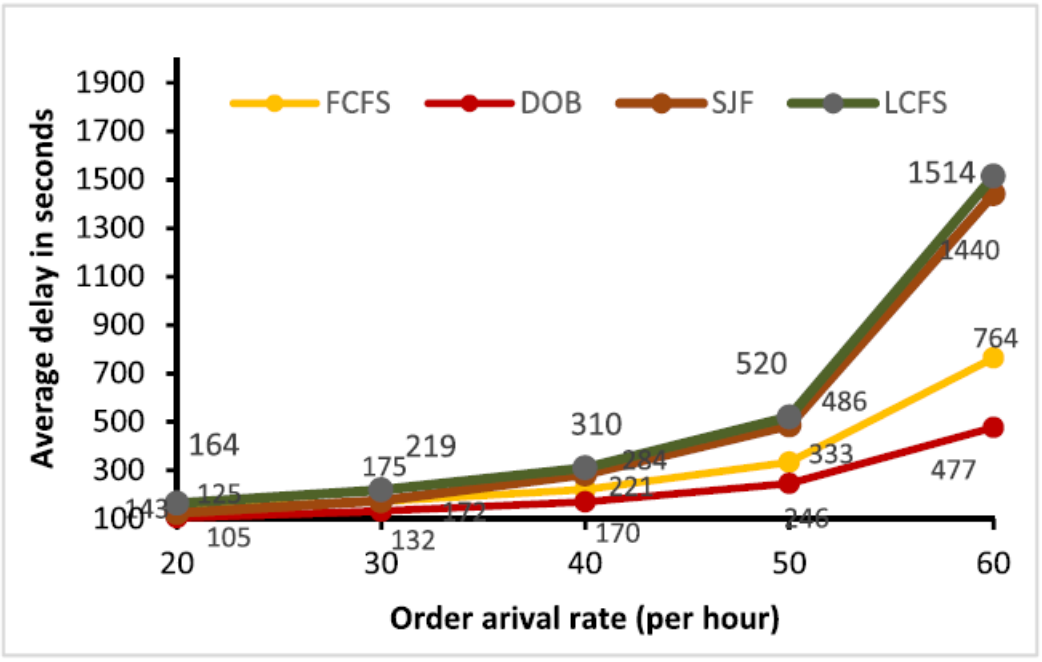

FIGURE 8. Comparison of average delays with 6 stackers.

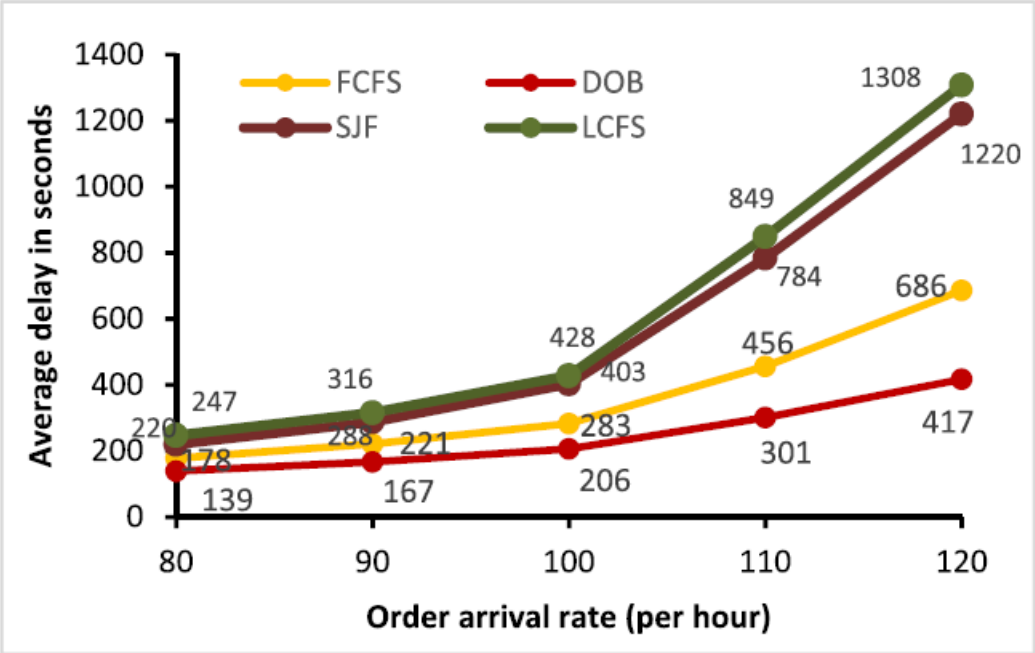

FIGURE 9. Comparison of average delays with 12 stackers.

With 12 stackers, the order arrival rates are between 90 and 120 per hour. These different ranges are considered to prevent system overflow in their respective settings while also stress-testing the algorithms. In our simulations, we evaluate the following performance metrics:

(A) Average Retrieval Delay of Orders

(B) Maximum Retrieval Delay of Orders

(C) Downstream Backlog Pressure

Downstream Backlog Pressure is defined as the pressure generated by started-but-unfinished orders waiting at the packaging stations. Once an order has begun being retrieved, the corresponding packaging station must wait until the last item of the order is retrieved before it can start packaging. All items except the previous ones remain at the

packaging station during this waiting period, creating pressure on the packaging area's buffer zone.

Our evaluation of Downstream Backlog Pressure considers two crucial aspects: order quantity pressure and item quantity pressure. These aspects are important as they reflect the number of pending orders and items in the packaging area at any given time. By analyzing these pressures, we can effectively evaluate the performance of different algorithms in terms of their impact on the packaging stations. We also consider the average and maximum pressure values for a comprehensive analysis.

1) AVERAGE RETRIEVAL DELAY OF ORDERS

Figures 8 and 9 compare the performance of DOB, FCFS, SJF, and LCFS at different order rates. The same phenomenon is observed despite the other number of stackers in Figures 8 and 9. When there are only six stackers and the order rate reaches 60 orders per hour, the average delays for FCFS, SJF, and LCFS are 764s, 1440s, and 1514s, respectively, whereas DOB only requires 477 seconds, a 37% decrease compared to FCFS. Similarly, with 12 stackers and an order rate of 120 orders per hour, the average delay for DOB also drops by about 39% compared to FCFS.

It's crucial to note that the order rate, which refers to the number of orders processed per hour, significantly influences the performance of the algorithms, regardless of the number of stackers. When the order rate is relatively low (e.g., in the 6-stacker setting, when the order rate is less than 40 orders per hour), there is no noticeable difference in average delays among the algorithms. However, as the order rate decreases further, the performance of DOB becomes closer to that of FCFS, although DOB is always slightly better. Conversely,

as the input pressure (order rate divided by the number of stackers) increases, the advantages of the DOB algorithm over FCFS become more pronounced. Moreover, the average delays of SJF and LCFS are consistently worse than those of FCFS and DOB.

2) MAXIMUM RETRIEVAL DELAY OF ORDERS

DOB outperforms FCFS in terms of average retrieval delay by prioritizing items from orders with smaller Order Tags and delaying those with larger Order Tags. This strategy, however, may be perceived as unfair, as Order Tags are determined by the items' locations in shelf storage from the ground point. This fairness issue underscores the need for a balanced algorithm that considers all orders equally.

We also consider the Maximum Retrieval Delay in our simulation to observe how DOB may sacrifice this small group of orders. As shown in Figure 10, under the condition of 12 stackers, we found that when the order rate is low, the maximum delay of the DOB algorithm is slightly higher than that of the FCFS algorithm. However, as the order rate increases, the gap between the maximum delays of FCFS and DOB widens significantly. When the order rate reaches 120 orders per hour, the maximum delay of DOB is 27,738 seconds, while that of FCFS is only 2,718 seconds. Although DOB reduces the average delay, it significantly increases the maximum waiting time. It should be noted that the maximum delays of LCFS and SJF are even worse than those of DOB.

3) DOWNSTREAM BACKLOG PRESSURE

The retrieval process is a pivotal part of warehouse operations, as it directly impacts subsequent processes, such as the packaging area. The downstream backlog pressure, which arises from the need for order completeness, underscores the importance of a well-

organized and efficient retrieval system. Only when all items belonging to the same order arrive at the packaging station can they be packaged and sent to the next destination. This highlights the critical role of the retrieval process and the importance of the audience's role in warehouse management.

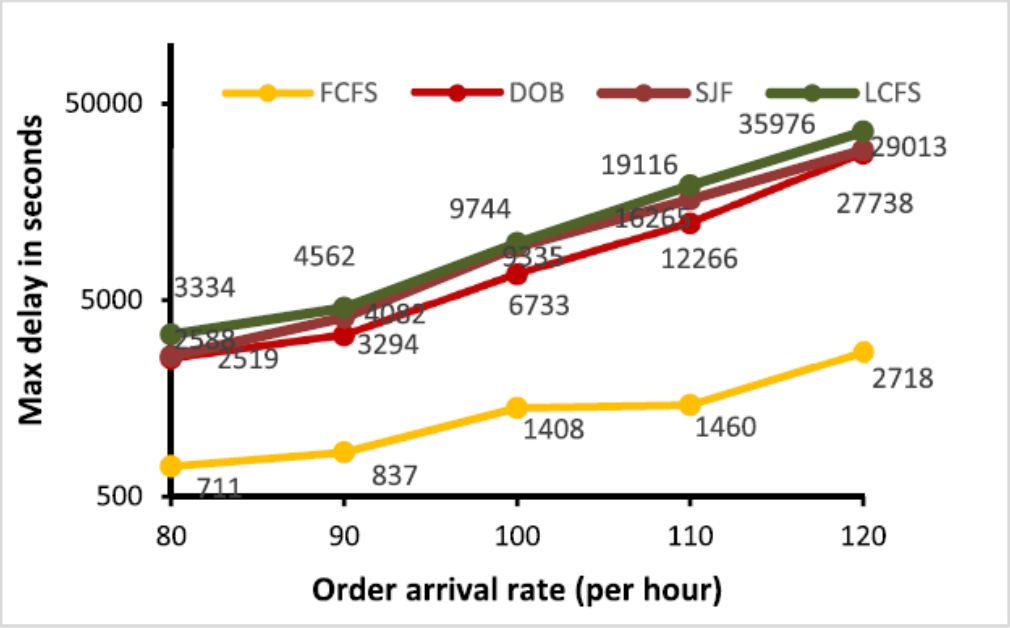

FIGURE 10. Comparison of max delays with different order rates.

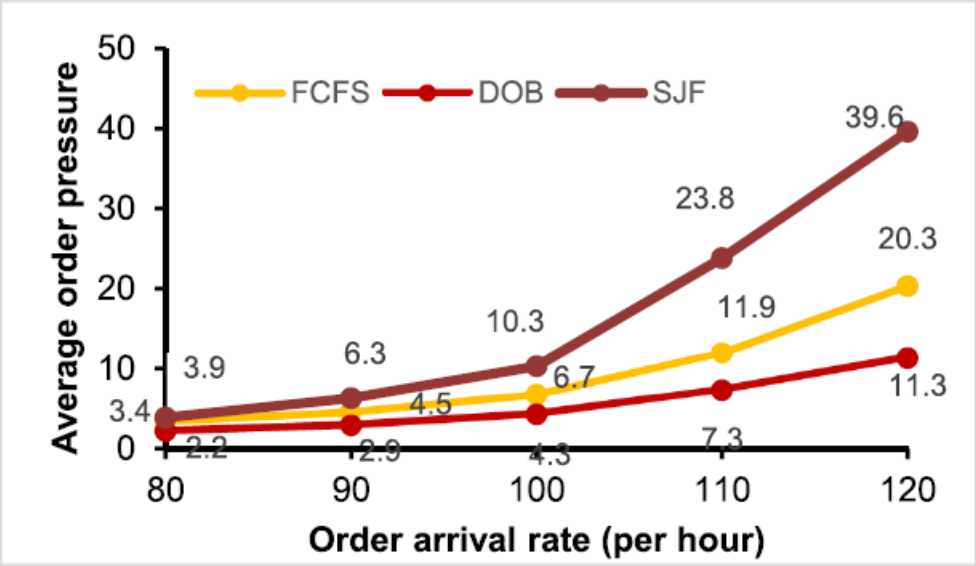

FIGURE 11. Comparison of average order pressures with different order rates.

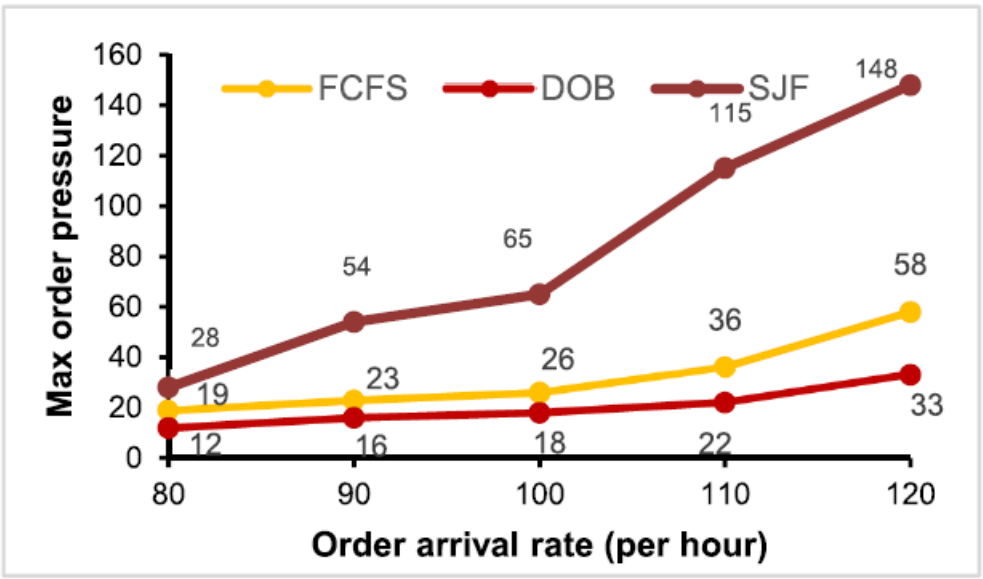

FIGURE 12. Comparison of max order pressures with different order rates.

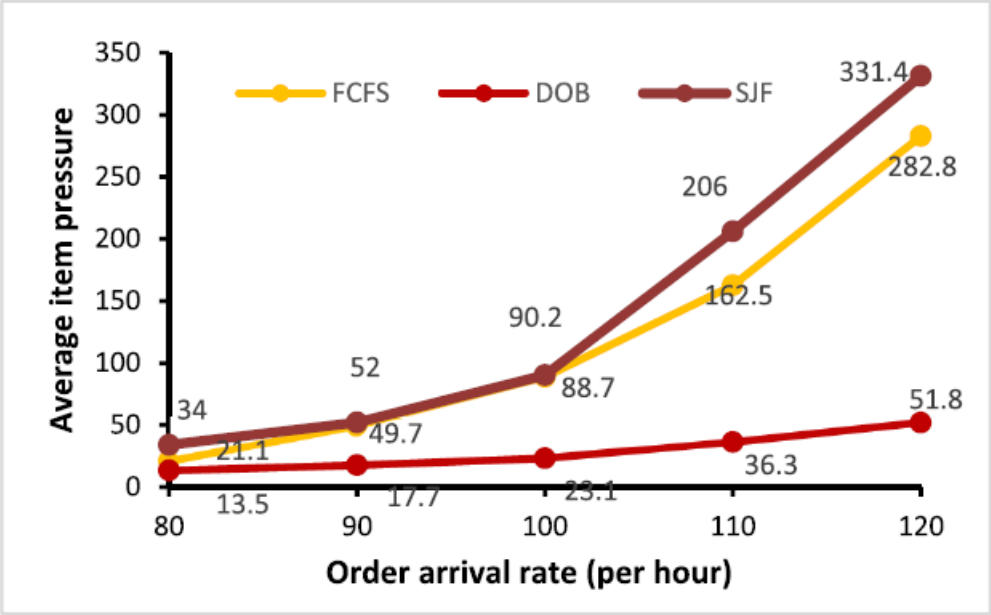

FIGURE 13. Comparison of average item pressures with different order rates.

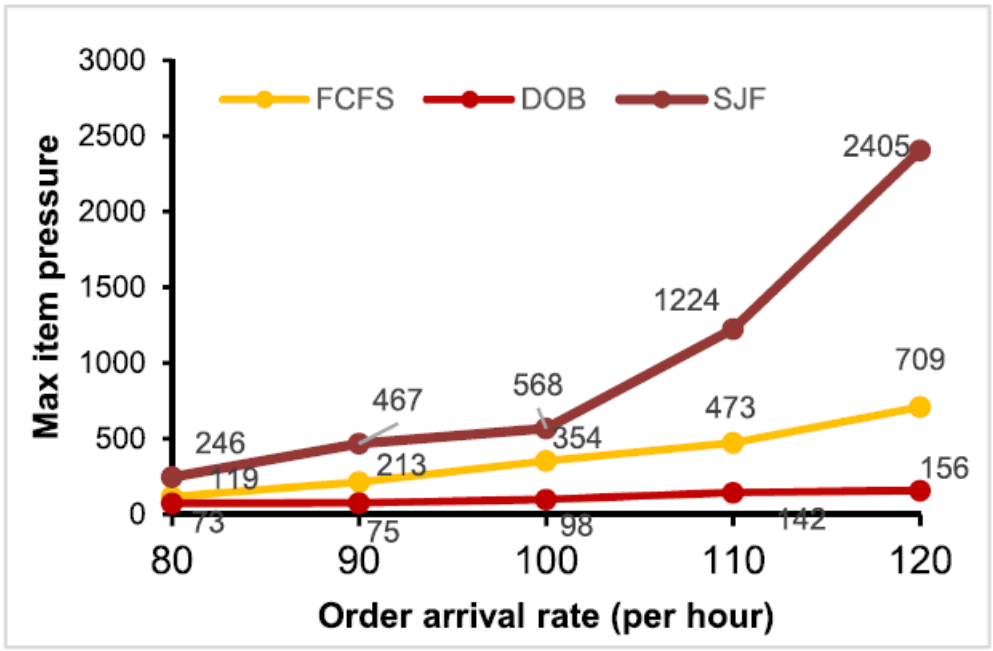

FIGURE 14. Comparison of max item pressures with different order rates.

In the following figures, the number of pending orders and pending items in the Packaging Area will be used to measure downstream backlog pressure. Note that we only show the

curves for DOB, FIFO, and SJF, but not LCFS, because the downstream backlog pressures of LCFS are significantly more significant than the others, making it unsuitable for comparison in the exact figure.

As shown in Figures 11 and 12, as the order rate increases, the average and maximum order pressure of FCFS become significantly higher than those of DOB. This demonstrates that DOB can reduce the average delay and effectively relieve downstream backlog pressure.

When the quantity of backlogged items is used to measure backlog pressure, as shown in Figures 13 and 14, it more clearly reflects that DOB can effectively relieve downstream backlog pressure. It should also be noted that SJF consistently has the worst downstream backlog pressure compared to FCFS and DOB.

B. DOBT vs. DOB and FCFS

While the DOB algorithm improves average delay and reduces downstream backlog pressure, it also significantly increases maximum delay, as shown in Figure 10. This can lead to a poor experience for specific customers. To address this issue, we introduce a "threshold" to limit the waiting time for orders. When orders have waited longer than the threshold, the DOBT algorithm prioritizes these orders for execution. This approach also provides a certain degree of fairness among orders.

To simplify notation, we use "th" to represent the threshold. For example, "the-400" means that we set a threshold of 400 seconds for all orders.

TABLE 3. Average delays of different algorithms with different thresholds.

rate	**FCFS**	**DOB**	**th-1000**	**th-4000**	**th-10000**	**th-20000**
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

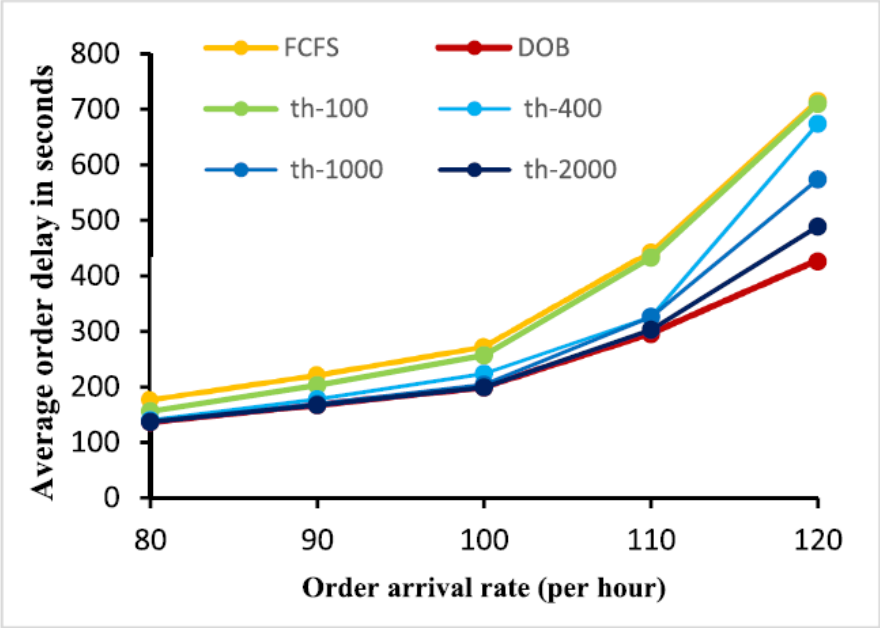

FIGURE 15. Average delays of DOBT with different threshold values.

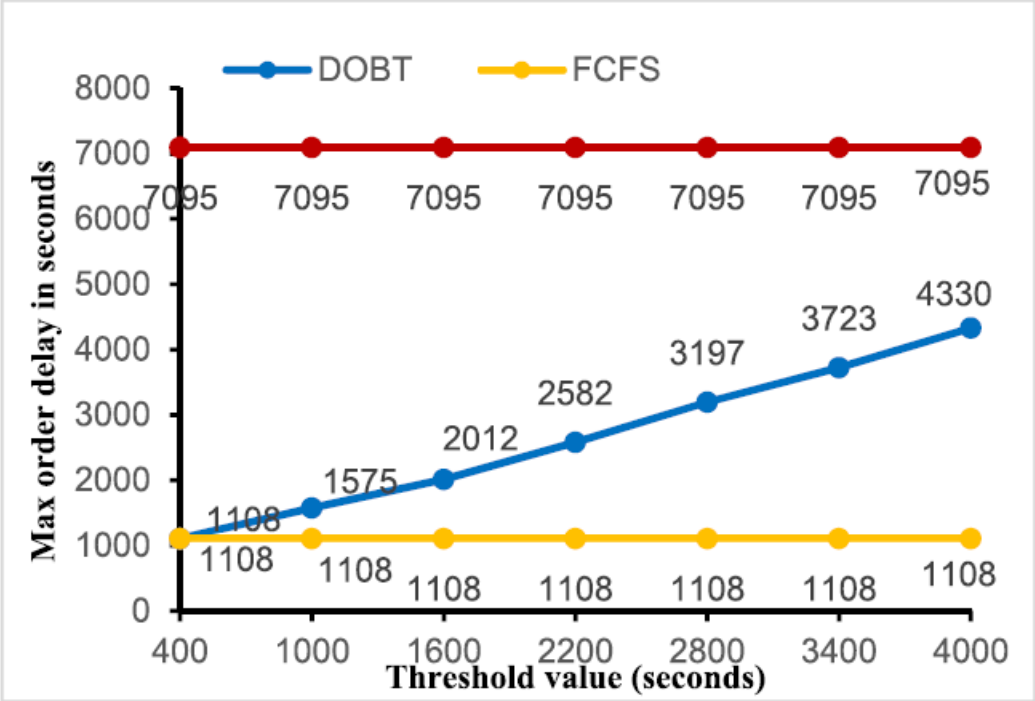

FIGURE 16. Max order delays of DOBT under different threshold values

Table 3 records the experimental data, and Figure 15 shows the corresponding curves for the average delays of the algorithms. As the order rate increases, all curves exhibit an upward trend. The FCFS curve is consistently higher than the others, while the DOB curve remains at the bottom. However, the th-100 curve almost coincides with the FCFS curve because when the threshold is set very low, the Order Tag becomes ineffective. Additionally, as the threshold increases, the algorithm's performance approaches that of DOB. When the order rate reaches 120 orders per hour, the th-2000 curve is much closer to the DOB curve than the th-400 curve. The FCFS and DOB curves effectively represent the upper and lower bounds of the DOBT curves with different thresholds.

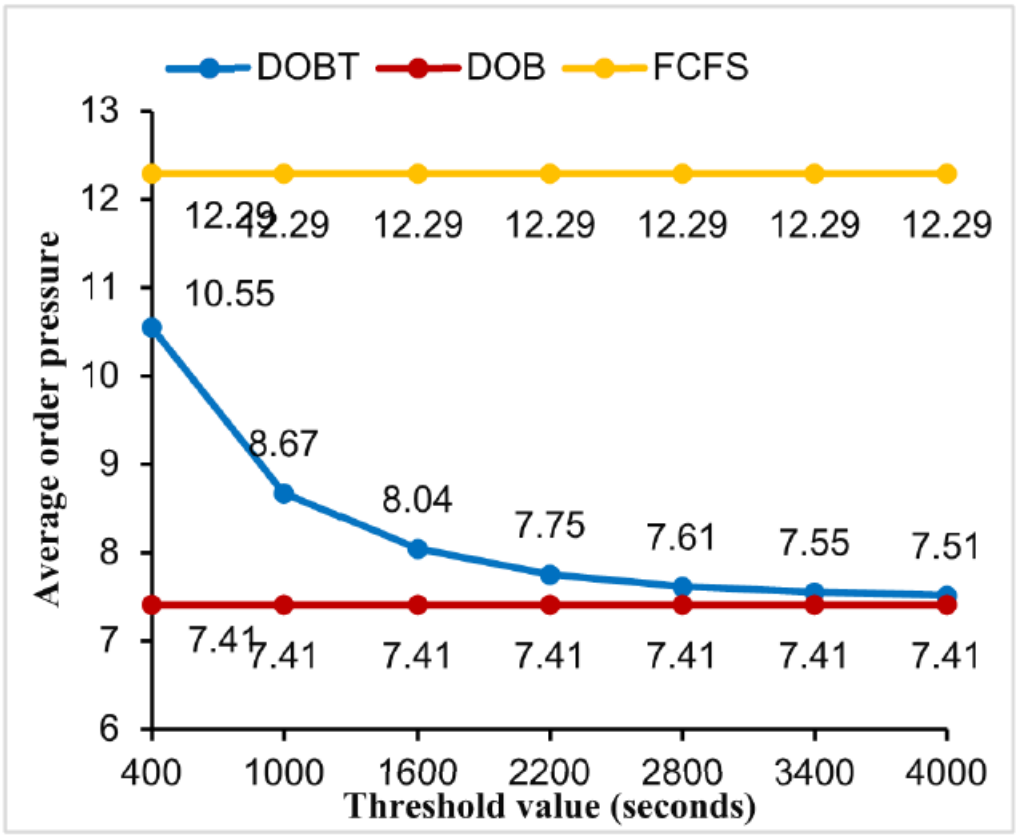

FIGURE 17. Average order pressures of DOBT under different threshold values

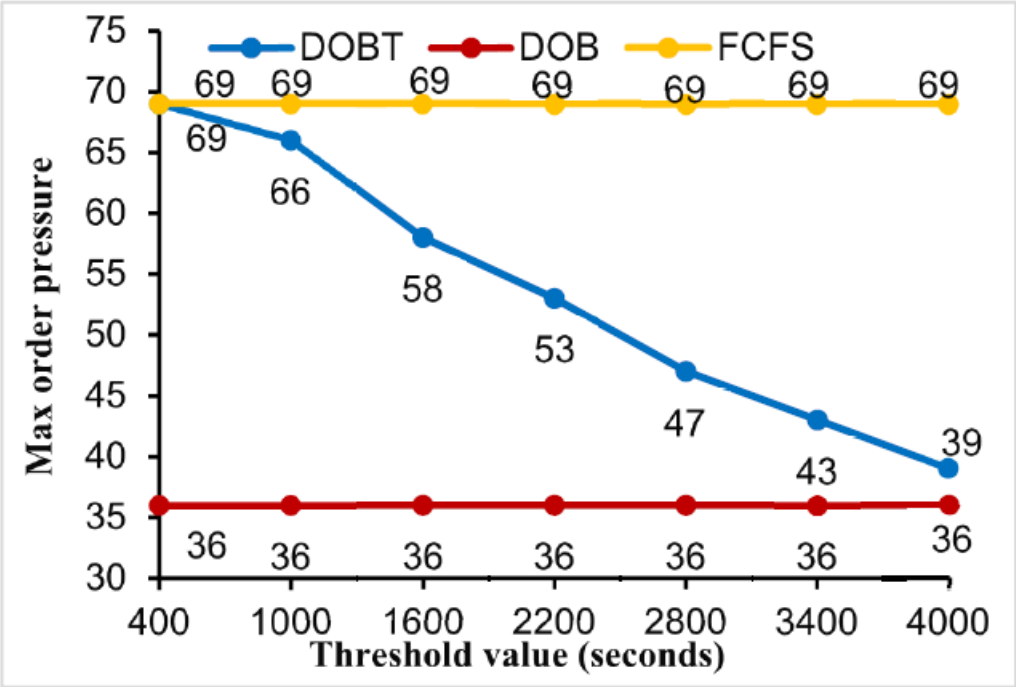

FIGURE 18. Max order pressures of DOBT under different threshold values

As shown in Figure 16, to compare the maximum delay of DOBT with different threshold values, we set the order rate to 110 orders per hour and experimented with threshold values ranging from 400 to 4000 seconds. It is evident that with a small threshold value, DOBT effectively controls the maximum delay of orders, approaching the performance of FCFS. In other words, DOBT excels at reducing maximum waiting time with an appropriately set threshold while providing an acceptable average delay performance.

Combining the observations from Figures 15 and 16, we can conclude that as the threshold value increases, the average order delay of the DOBT algorithm approaches the performance of the DOB algorithm. Additionally, the maximum order delay of DOBT is close to that of FCFS when the threshold is small. In other words, maximum order delay and average order delay are on opposite sides of the balance, and the threshold value is a

parameter that can be adjusted to achieve the desired performance for an optimal warehouse operational plan.

Figures 17 to 20 show the downstream backlog pressure of DOBT, FCFS, and DOB. Similar phenomena can be observed: the higher the threshold value for DOBT, the closer its performance is to DOB. Conversely, when the threshold value is small, DOBT's performance is closer to FCFS's.

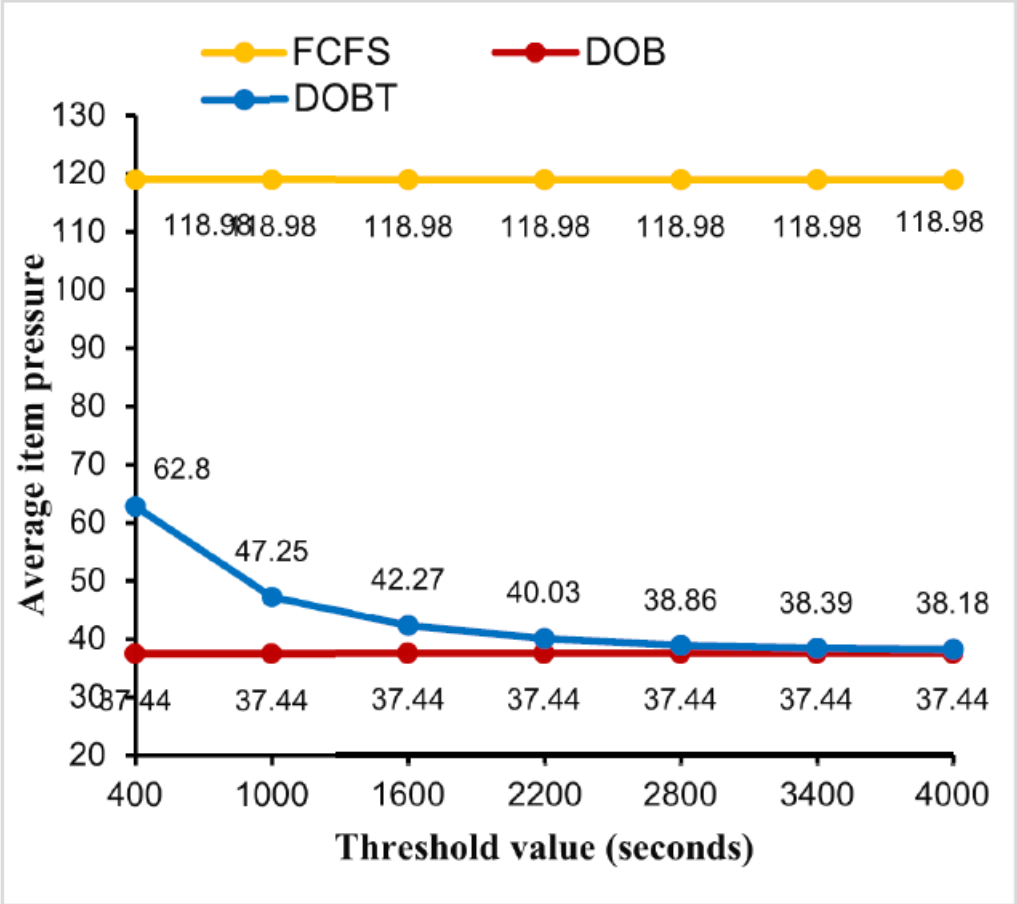

FIGURE 19. Average item pressures of DOBT under different threshold values

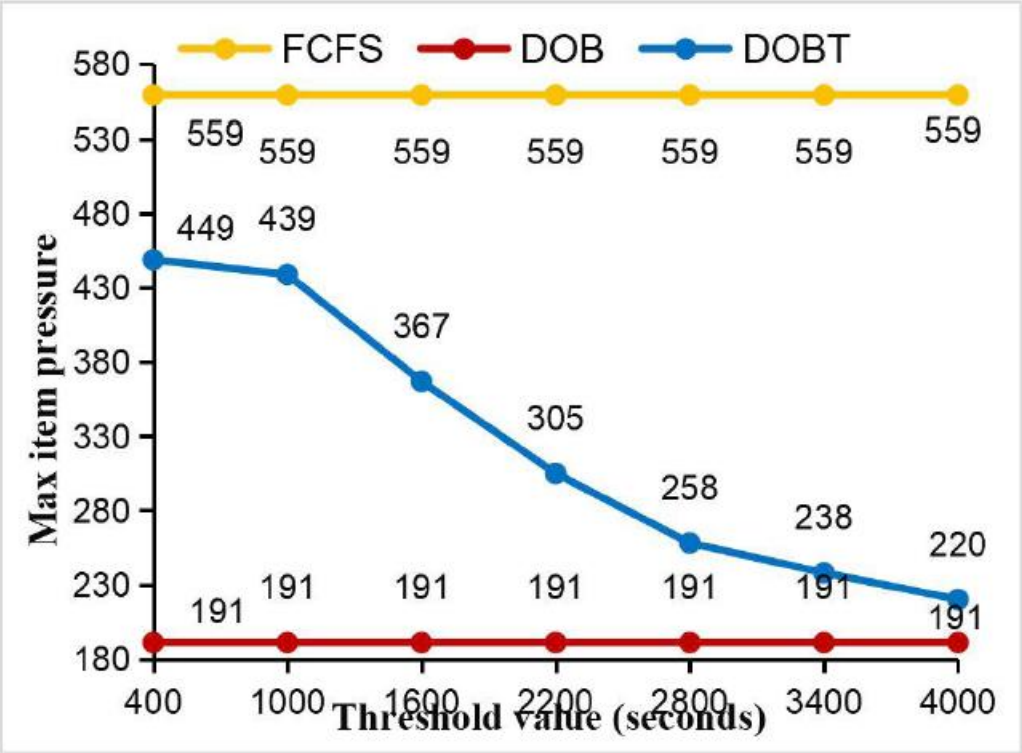

FIGURE 20. Max item pressures of DOBT under different threshold values

VIII. CONCLUSION

In this paper, we have considered dynamic order arrivals and studied the corresponding scheduling algorithms of Automated Retrieval Systems (ARS) to manage the retrieval jobs of stackers in intelligent warehouses. Specifically, we proposed using "Order Tag" to enable ARS to consider the integrity of orders while dynamically scheduling retrieval jobs. Our study revealed that it is challenging to simultaneously optimize two performance metrics: average order retrieval delay and maximum order retrieval delay. To address this, we introduced a threshold to the scheduling algorithm to balance these two metrics.

The simulation results demonstrate that the Dynamic Order-Based Threshold Algorithm (DOBT) can significantly reduce the maximum order delay with a minimal sacrifice in

average order delay. Additionally, an essential contribution of our work is alleviating downstream backlog pressure in the Packaging Area. DOB and DOBT significantly reduce the number of pending orders and items compared to FCFS and other algorithms, enhancing overall warehouse efficiency.

REFERENCES

[1] N. Boysen, R. de Koster, and F. Weidinger, ``Warehousing in the ecommerce era: A survey," Eur. J. Oper. Res., vol. 277, no. 2, pp. 396411, Sep. 2019.

[2] Y. Yu, X. Wang, R. Y. Zhong, and G. Q. Huang, ``E-commerce logistics in supply chain management: Practice perspective," Proc. CIRP, vol. 52, pp. 179185, Jan. 2016.

[3] A. D. T. Larsson, ``Selection of automated order picking systems," M.S. thesis, Dept. Technol. Manage. Econ., Chalmers Univ. Technol., Gothenburg, Sweden, 2016.

[4] V. Aggarwal, Y.-F. R. Chen, T. Lan, and Y. Xiang, ``Sprout: A functional caching approach to minimize service latency in erasure-coded storage," IEEE/ACM Trans. Netw., vol. 25, no. 6, pp. 36833694, Dec. 2017.

[5] D. Culler and J. Long, ``A prototype smart materials warehouse application implemented using custom mobile robots and open source vision technology developed using EmguCV," Proc. Manuf., vol. 5, pp. 10921106, Jan. 2016.

[6] K. N. Lemon and P. C. Verhoef, ``Understanding customer experience throughout the customer journey," J. Marketing, vol. 80, no. 6, pp. 6996, Nov. 2016.

[7] Z. He, V. Aggarwal, and S. Y. Nof, ``Differentiated service policy in smart warehouse automation," Int. J. Prod. Res., vol. 56, no. 22, pp. 69566970, 2018.

[8] R. B. De Koster, A. L. Johnson, and D. Roy, ``Warehouse design and management,'' Int. J. Prod. Res., vol. 55, no. 21, pp. 63276330, 2017.

[9] H. Zhang, Z. Guo, W. Zhang, H. Cai, C. Wang, Y. Yu, W. Li, and J. Wang, ``Layout design for intelligent warehouse by evolution with fitness approximation,'' IEEE Access, vol. 7, pp. 166310166317, 2019.

[10] B. S. S. Tejesh and S. Neeraja, ``Warehouse inventory management system using IoT and open source framework,'' Alexandria Eng. J., vol. 57, no. 4, pp. 38173823, Dec. 2018.

[11] A. Gilya-Zetinov, D. Demianova, and A. Khelvas, ``Palletizing for fullautomated warehouses on the genetics algorithm base,'' in Proc. Int. Conf. Eng. Telecommun. (EnT), Nov. 2019, pp. 15.

[12] X. Jiang, D. Zhao, and H. Xu, ``Analysis and reconstruction of pharmaceutical warehouse logistics delivery system,'' in Proc. IEEE Int. Conf. Smart Manuf., Ind. Logistics Eng. (SMILE), Apr. 2019, pp. 226229.

[13] X. Sun, Z. Ma, Z. Wang, and C. Ai, ``The development of stereoscopic warehouse stacker control system based on motion controller,'' in Proc. MATEC Web Conf., Jinan, China, vol. 139, 2017, pp. 3843.

[14] S. Li, N. Qin, D. Huang, D. Huang, and L. Ke, ``Damage localization of stacker's track based on EEMD-EMD and DBSCAN cluster algorithms,'' IEEE Trans. Instrum. Meas., vol. 69, no. 5, pp. 19811992, May 2020.

[15] H. Rams, M. Schöberl, and K. Schlacher, ``Optimal motion planning and energy-based control of a single mast stacker crane," IEEE Trans. Control Syst. Technol., vol. 26, no. 4, pp. 14491457, Jul. 2018.

[16] Y. Kung, Y. Kobayashi, T. Higashi, M. Sugi, and J. Ota, ``Order scheduling of multiple stacker cranes on common rails in an automated storage/retrieval system," Int. J. Prod. Res., vol. 52, no. 4, pp. 11711187, Feb. 2014.

[17] Y. Khojasteh and J.-D. Son, ``A travel time model for order picking systems in automated warehouses," Int. J. Adv. Manuf. Technol., vol. 86, nos. 58, pp. 22192229, Sep. 2016.

[18] X. Yang, Z. Xu, W. Jin, and F. Shu, ``Optimization of automatic stacker picking sequence under the mixed picking strategy," Comput. Integr. Manuf. Syst., vol. 27, no. 3, pp. 933942, 2021.

[19] K. W. Pang and H. L. Chan, ``Data mining-based algorithm for storage location assignment in a randomised warehouse," Int. J. Prod. Res., vol. 55, no. 14, pp. 40354052, 2017.

[20] Y. Wang, S. Mou, and Y. Wu, ``Task scheduling for multi-tier shuttle warehousing systems," Int. J. Prod. Res., vol. 53, no. 19, pp. 58845895, Oct. 2015.