IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received January 4, 2021, accepted February 3, 2021, date of publication February 9, 2021, date of current version February 18, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3058190

Adaptive Task Planning for Multi-Robot Smart Warehouse

ALI BOLU, (Member, IEEE), AND ÖMER KORÇAK, (Member, IEEE)
Department of Computer Engineering, Marmara University, 34722 Istanbul, Turkey

Corresponding author: Ali Bolu (alibolu@marun.edu.tr)

This work was supported by the Makhina Robotics, Istanbul, Turkey.

ABSTRACT Using autonomous mobile robots is now a necessity for today's large e-commerce warehouses to save time and energy, and to prevent human-based errors. Robotic Mobile Fulfillment System (RMFS) controls these robots as well as all other resources and tasks in a warehouse. There are challenges in the management of an RMFS-based smart warehouse because of the high dynamics in the system. Limited resources such as robots, stations, totes, and item spaces should be managed efficiently after tracking their status continuously. In this study, we propose a centralized task management approach that is adaptive to the system dynamics. We describe a novel task conversion algorithm that generates tasks from a batch of orders and provides a high pile-on value. Then we propose an adaptive heuristic approach to assign generated tasks to robots, considering system dynamics such as the location of robots and pods, utilization of totes, and age of the tasks. To evaluate the proposed algorithms, we perform an extensive set of simulations in a highly realistic environment including robot charging, replenishment process, and path planning algorithms. We show that the proposed task planning approach significantly reduces order completion time even for a high number of stock-keeping units (SKU). It also provides a balanced workload among robots. We analyze the optimal value of order batch size and the effect of important system parameters such as robot count, order count, and SKU. The obtained results shade light on how to design a smart warehouse system with high efficiency.

INDEX TERMS Autonomous mobile robots, resource management, robotic mobile fulfillment system, smart warehouse, task planning, warehouse execution system.

I. INTRODUCTION

Robotic Mobile Fulfillment System (RMFS) is a new trend warehouse system that employs autonomous mobile robots. Today, many e-commerce warehouses use this system, while many more are investigating for future use. RMFS-based warehouses in which robots lift pods and bring to pick stations were first introduced by KIVA Systems, Wurman *et al.* [1]. Amazon bought Kiva systems in 2012 and renamed it to Amazon Robotics [2]. Since then, many companies have entered the market with their robots such as Swisslog CarryPick, GreyOrange Butler, Fetch Robotics Freight (and Fetch), Scallog System, Hitachi Racrew, etc., [3]. Moreover, major retailers, such as Amazon and Alibaba, use RMFS [2]. However, investment in RMFS to warehouses will cost more than a million-dollar generally [4]. The most expensive part of an RMFS investment is the robots. Hence, optimization on RMFS to complete more order with less number of robots is a crucial point of efficiency.

The associate editor coordinating the review of this manuscript and approving it for publication was Zhenliang Zhang.

The order picking process is the most important part in a warehouse system. Traditional warehouses use picker-to-part method, i.e. workers travel in the warehouse and collect order lines. The new RMFS approach is part-to-picker, such that robots carry the mobile shelves (named as pod) that contain items to the workers who are waiting at the pick stations. An RMFS increase picking rate significantly compared to traditional methods especially for large e-commerce warehouses with many SKUs (Stock Keeping Units) [5]. It is crucial to optimize picking order, i.e. to collect more order items with less effort. The order batch method is used for picker-to-parts picking methods. In this method, a set of orders are splitted into several subsets of order items and assigned to pickers in an efficient way in order to reduce travel and picking time [6]. However, in an RMFS, it is not feasible for robots to carry all order items in a single pod. Therefore, the optimization objective is to collect more items per pod at the pick station.

The average number of order items picked from one pod, named as *pile-on*, is the most important metric to reduce order picking time [5], [7].

Another important task of an RMFS is the replenishment to pods, in other words to place stocks in the warehouse. There are two types of stations; pick stations and replenishment stations. Human workers work in these stations for collecting orders (picking) or placing items to the pods (replenishment). In all tasks, the pods are carried to and from the station by robots. The picking process consumes more energy and time than replenishment, it is up to 80% according to [5].

It is obvious that robots need energy and charging to complete their tasks. When a robot completes the assigned task and ready for another one, the RMFS will decide to assign one of the three tasks: order picking, replenishment, or charging. In this study, we mainly focus on order picking optimization, while we also consider and do not neglect replenishment and charging tasks. We propose novel order-to-task conversion and task selection methods aiming to increase pile-on and total efficiency of an RMFS by considering order throughput, time priority, and utilization of resources. We also test our methods and algorithm with different warehouse parameters such as number of robots, pick stations, totes, orders and SKU.

Management of an RMFS requires knowledge of all existing and moving objects in the warehouse. For this purpose, we develop a web-based central software called Warehouse Execution System (WES) to manage warehouse automation. WES manages all resources such as robots, stations, totes, pods etc. We simulate the RMFS on WES in a highly realistic environment (which can be considered as a digital twin of the real system) in order to evaluate the performance of the proposed approaches for various system parameters.

A. RELATED WORK

RMFS is a new approach with an increasing popularity in the last decade. Cooperation of multiple robots and coordination of many other resources such as pods, totes, stations etc. increase the complexity. Enright and Wurman (2011) present general concepts of optimization and coordination of an RMFS in [7]. They mention importance of pile-on optimization to reduce order picking time. Lamballais *et al.* [4] analyze performance of an RMFS under four models such as single-line and multi-line orders and with or without storage zones. They develop queuing network model that includes storage zoning and multi-line orders in order to estimate performance according to order throughput, average order cycle time, and utilization of robots and workstations.

The assignment of pick orders to pick stations (Pick Order Assignment - POA) and the pod selection in the pick process (Pick Pod Selection - PPS) are two important issues for order picking optimization in an RMFS. First POA, then PPS is applied [1], [5]. Moreover, [8] performs an approach for integrating POA and PPS. Merschformann *et al.* [5] analyze optimization performance by studying important decision rules such as POA, PPS, Replenishment Order Assignment (ROA), Replenishment Pod Selection (RPS) and Pod Storage Assignment (PSA). They compare multiple rules and find correlation between them to increase order item throughput. They point out importance of pile-on and robot travelling distance for RMFS performance. Boysen *et al.* [9] study on processing the orders at the pick station. They focus on batching and sequencing of picking orders in order to decrease robot needs. The decision problem they formulate is NP-Hard and they provide several heuristic algorithms to converge to the optimal. They show that their algorithms decrease the number of pod visits, but they study only for small instances (up to 100 orders). Their results imply that the provided heuristic approaches would take significant amount of time for larger instances and may not be feasible for real time applications. Reference [8] integrate assignment of pods to station and orders to station for order picking process, instead of calculating separately. They also propose to split orders in order to improve RMFS efficiency. In other words, parts of an order are allowed to be processed in different stations and combined later at the packaging stations. While splitting orders improve the efficiency, it cause additional processing load. Evaluation of extra effort for splitting orders and combining later requires real experiments beyond simulation and analytical models.

Zou *et al.* [10] focus on rule based robot to task assignment with handling speeds of workstations and propose a neighbourhood search algorithm to find a near optimal solution. Moreover, they analyze shelf block size effect to the RMFS throughput. Reference [11] identify the performance characteristics of an RMFS by providing a literature review. They mention the relation between performance and the design decisions of RMFS for operation as a preliminary finding from an ongoing study.

Other important optimization approach in order picking process is to use efficient replenishment and pod allocation methods. These methods proactively increase picking order throughput since they provide more efficient options to select pods. Reference [12] focus on efficient pod alignment on the warehouse storage area. Reference [13] prove that spreading inventory across many pods significantly decreases the time of collecting orders. They also analyze the optimization of variables such as the number of items per pod, replenishment level per pod, and picking station to replenishment station ratio.

Robots can be dedicated specifically to order picking task or replenishment task. This approach is called dedicated robot assignment. On the other hand, in pooled robot assignment, both tasks are performed by a single pool of robots. Merschformann *et al.* [5] adopts dedicated robot assignment and assigns two-third of the robots for order picking tasks, and others to replenishment tasks. Roy *et al.* [14] analyze RMFS for single and multiple zone with both dedicated and pooled robot assignment. They notice that using pooled robot assignment reduces order picking time, while it increases the replenishment time. Yuan and Gong [15] perform their order picking analysis with the pooled and dedicated robot system. They also analyze optimal robot count and speed for their test environment. Zhou *et al.* [16] focus on balancing robot workload while optimizing total robot travel time. They propose a heuristic balance mechanism to assign tasks to robots. In other words they select robots for tasks with an aim of minimizing total travel cost and balancing robot workload. However, this approach will reduce robot utilization since robots will need to wait for the others to become free. In a cost-efficient RMFS, number of robots are optimized and they are busy most of the time. In a warehouse with several hundreds of robots, 3 to 4 robots may become idle concurrently [7] and it can be suitable to choose only among those robots.

B. MAIN CONTRIBUTIONS

In this study we propose a novel Order Batch to Robot Task Conversion (OBRTC) algorithm which tries to find minimum number of pods that includes items of multiple orders. In other words, contrary to existing studies, OBRTC algorithm handles a number of orders as a single order instead of handling orders one by one. The number of orders to be handled as a batch has a crucial role to increase pile-on value even for high SKU. After OBRTC is executed, WES selects available totes at the pick stations for items in pod when the order task is assigned to the robot. The pod of order task will consist of multiple order's items, and it can be assigned to different stations if necessary. Furthermore, we propose Adaptive Robot Task Selection (ARTS) method to select a new task for a robot that become available. New task selection is performed according to a novel priority based heuristic model which depends on various criteria such as distance to robot, totes usage and time. In brief, we first select pods for multiple orders without assigning stations, totes, or robots to reach maximal pile-on value. Then we adaptively manage other resources considering the system dynamics.

Contrary to the most of the studies in the literature, we develop and use a fully realistic simulation environment that includes robot collisions, waiting time on path and stations, and effects of the loaded pod weight. Moreover, our simulation includes charging process by considering a realistic energy consumption model (depending on movements and loads) and real charging times.

Some e-commerce companies may sell items even if they are not already stored in their warehouses and these items are replenished on demand. Therefore there could be many already ordered items on the pods loaded at the replenishment stations. Management of this case is mostly not addressed in existing studies. In our study, we also test such a scenario and analyze its effects on the system performance.

In summary, our work has three main contributions;

1) We provide a task planning approach with high pile-on value even for high number of SKUs.
2) We propose a novel parametric heuristic model for order task selection process.
3) We develop a highly realistic simulation environment and provide test results under various decision rules which shade light on how to design a smart warehouse system with high efficiency.

Rest of the paper is organized as follows. We describe the system model and our methods in Section 2. We explain our simulation model and test parameters in Section 3. Test results are shown in Section 4. We conclude the paper in Section 5.

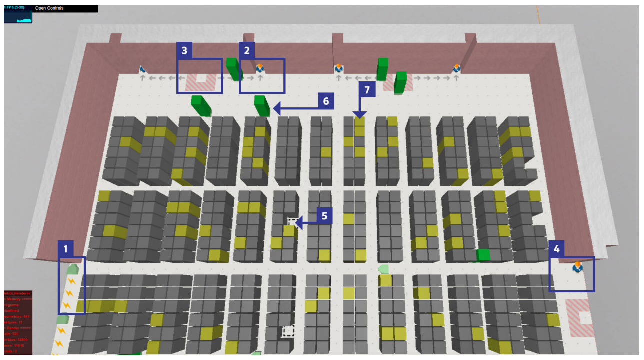


FIGURE 1. Warehouse execution area.

II. MODEL AND METHODOLOGY

A. SYSTEM MODEL

Figure 1 illustrates an RMFS based warehouse system. The components of this system are the following:

- **Pods:** All the warehouse items are stored in the pods. When an order or replenishment arrives to the system, it should be converted to the task by assigning appropriate pod or pods. These pods that are reserved by the system are shown in yellow color in Figure 1 (such as 7). When a pod is lifted by a robot, its cell becomes free (5 in Figure 1).
- **Robots:** Robots carry pods to the stations for order collection or replenishment (6 in Figure 1).
- **Pick station** (2 in Figure 1): There can be multiple pick stations in a warehouse, where human workers collect the order items carried by the robots. Figure 2 shows a pick station. There exists limited number of totes, and each tote is assigned for a single order. When an order is collected, that tote is replaced with an empty one.
- **Replenishment station** (4 in Figure 1): Human workers place replenishment items to the shelves of pods that are carried by the robots. In a the warehouse, human workers work only in pick and replenishment stations, and the rest is no-human zone.
- **Turning cells** (3 in Figure 1): A pod may have multiple faces and an item can be reached only through a single face [17]. There are two types of pods according to the number of faces: two-directional and four-directional (pods that can be seen in Figure 2 are two-directional). Robots can rotate pods only on turning cells.

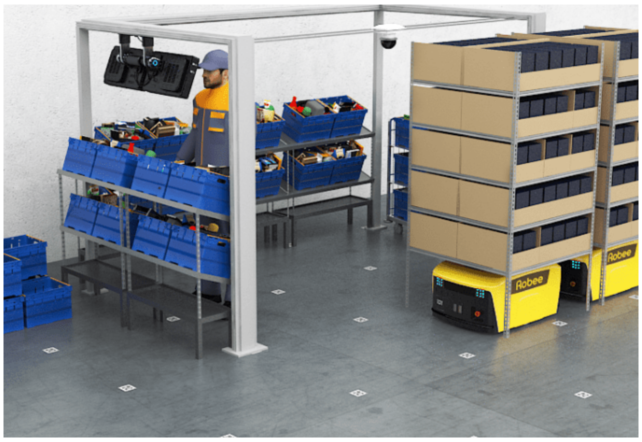


FIGURE 2. Pick station [18].

- **Charging cells** (1 in Figure 1): Robots go to the charging cells when their batteries are low.

B. METHODOLOGY

There are three main factors that should be handled in a best way to succeed an efficient life-long task planning in an RMFS based warehouse: incoming of stock (replenishment), outgoing of stock (collecting orders) and necessity (such as charging of robots). In this section, we explain our methodology, algorithms and design decisions to handle these factors in an efficient way.

1) ORDER-TO-TASK CONVERSION

Order task is the process of collecting orders. We consider multi-line orders as a usual behavior of e-commerce customers. In the basic approach, each order is converted to order task(s) individually. Let assume that we have an order with three items. WES tries to find pods that includes these items. This order will be completed with at least one pod and at most three pods. In the basic approach, most of the time robots carry one pod for just one item of an order. Collecting a multi-line order will mostly need more than one order task.

In order to increase the pile-on value, we handle order batches, instead of handling orders one by one. Order batch method is usually applied in traditional warehouses which do not employ robots [6]. Human workers collect orders after applying some order batch optimization. In this study, we aim to adapt "order batch" concept to RMFS-based smart warehouse in most effective manner. Figure 3 shows high level flow of order task process. When an order arrives to the WES, it is stored in an Order Pool. WES chooses a batch of orders from this pool and converts these orders to order tasks (Step 1 in Figure 3). Then these tasks are assigned to the available robots (Step 2). Then the assigned robot goes under the pod, lifts it, and carries to the pick station (step 3). Picker puts order items to the totes. Each order is collected in the tote that is reserved for that order. When an order is completed, its tote is departed from the pick station and a new tote is put in its place.

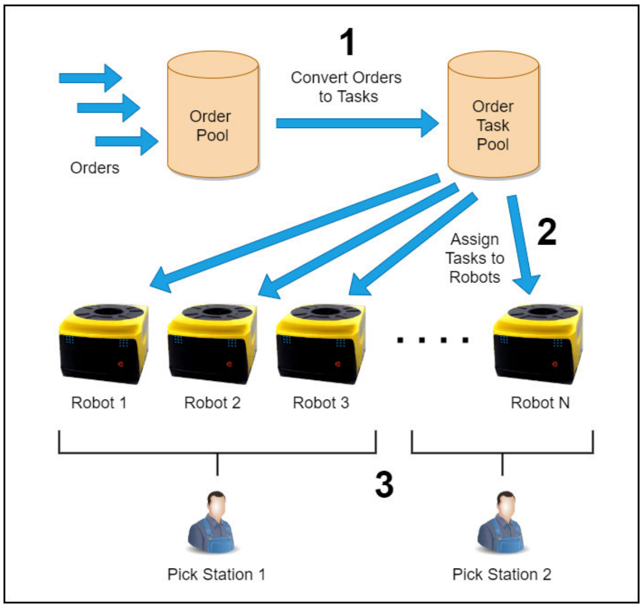


FIGURE 3. Order task process.

Now we will describe Step 1 of Figure 3, i.e. the order to task conversion method in more detail. To accomplish high pile-on value and provide high utilization of robots, we propose a novel Order Batch to Robot Task Conversion (OBRTC) algorithm as shown in Algorithm 1.

Algorithm 1 Order Batch to Robot Task Conversion

```
OrderList ← $n_{ob}$ orders from the set of all orders;
OrderItemList ← all items of orders in OrderList;
PodList ← set of all pods in the warehouse;
while $OrderItemList \neq \emptyset$ do
    1. $Pod_{max}$ ← Pod that contains highest number of
       items in OrderItemList;
    2. Create OrderTask with $Pod_{max}$;
    3. OrderItems ← All the items of OrderItemList
       included in Pod1;
    4. Add OrderItems to the OrderTask;
    5. Remove OrderItems from OrderItemList;
    6. Remove $Pod_{max}$ from PodList;
end
```

In OBRTC algorithm, firstly we select n_{ob} orders from order list for order to task conversion. This method considers n_{ob} orders as a single order to increase item pod matching possibilities. Thus, more order items can be collected with less number of pods. All the items of an order should exist in the warehouse stock. If there is a missing item for an order, this order is marked as "stock-waiting" order and not included in the task conversion process until the required stock is provided through replenishment process.

At each iteration, OBRTC algorithm creates an order task with the pod that includes maximum number of items. Note that if an order task already exists for this pod (as a result of previous run of OBRTC algorithm), then new items would be included in this order task, instead of creating a new one. Then this pod and the items it includes are removed from the list and the algorithm continues with the next iteration for converting remaining items to tasks with the remaining pods. The algorithm terminates when all the items in these n_{ob} orders are converted to tasks.

In the OBRTC algorithm, there are some important design parameters and restrictions to be considered. The value of n_{ob} has an important role for optimization. Increasing the value of n_{ob} will increase the number of items per task. In other words, more items will be collected at the pick station when the robot brings the pod. On the other hand tote count at the pick station is the important restriction for executing n_{ob} orders during the same time interval. For single order to task conversion, it is simple to handle tote reservation. Because all the items of an order task will be collected consecutively in a short time. However, when the orders are handled as a batch, items of an order may be collected in an extended time period. Because a number of order tasks are created by OBRTC simultaneously and items of an order may be spread over these order tasks and there could be some gaps between their collection times. This results in high number of reserved totes that include a portion of order items. We analyzed the best value of n_{ob} in different scenarios in Section III.

OBRTC algorithm should be executed and the order task pool should be refilled in appropriate intervals. This is important to avoid idle waiting of robots (due to lack of order tasks). On the other hand, if task pool includes too many tasks, then this can increase the execution time of the task selection process described in the next subsection. Therefore, OBRTC algorithm is executed when the number of order tasks in the order task pool becomes less than the number of working robots.

2) ORDER TASK SELECTION METHOD

After order tasks are created, an order task selection method is applied to manage resources efficiently and adaptively. For an available robot, WES assigns a task from the order task pool. In this study, we propose a heuristic model for task selection, namely Adaptive Robot Task Selection (ARTS) which adapts to the dynamics of the resources. WES selects a task for an idle robot from the task pool according to three important parameters: 1. Distance of the order task's pod to the robot; 2. Time elapsed after the creation of the order task; 3. Completion rate of the tote(s) assigned for the order task. When a robot finishes its previous task and becomes available, ARTS calculates a priority value for all order tasks in the pool, and assign the task with the highest priority.

The priority value of an order task i for a robot r (pr_{ir}) consists of three components (that correspond to three parameters mentioned above): Spatial priority pr^s_{ir}, temporal priority pr^t_i and tote priority pr^τ_i. Each of these components are calculated as follows.

i. Spatial Priority:

$$pr^s_{ir} = \frac{\alpha}{1 + \log_2(dist_{ir} + 1)} \tag{1}$$

where α is a normalization constant and $dist_{ri}$ is the Manhattan distance between robot r and pod p_i of task i, which is calculated as in (2).

$$dist_{ir} = |x(r) - x(p_i)| + |y(r) - y(p_i)| \tag{2}$$

Note that $x(\cdot)$ and $y(\cdot)$ are x and y coordinates of the cells that robots and pods stay on. Spatial priority is modeled using a logarithmic function. This indicates that if the distance is too low, high priority is given for that robot in order to decrease path traveling distance, and consequently to reduce the traffic. As the distance increases, there would be diminishing marginal effect of the distance to the task priority. The value of α describes maximum possible value of pr^s_{ir}, which is chosen to be 10 in our warehouse.

ii. Temporal Priority:

$$pr^t_i = \beta \cdot age_i \tag{3}$$

where β is a normalization constant and age_i is the elapsed time (in terms of minutes) after the creation of task i. Temporal priority is modeled using a linear function that is not bounded above. This is to guarantee that all the tasks are handled before a maximum waiting time. The value of β is important to define maximum task age. In our study, we chose $\beta = 0, 2$.

iii. Tote priority:

Let us define $\mathcal{O}$ as set of all orders and $\mathcal{O}_i \in \mathcal{O}$ as set of all orders that has some items to be handled by task i. Tote priority of task i is defined as

$$pr^\tau_i = \sum_{o \in \mathcal{O}_i} (B_{oi})^{1+\rho} \tag{4}$$

where $\rho \in [0, 1]$ is the utilization ratio of the totes. In other words, it is the ratio of the number of reserved totes to the number of all totes. B_{oi} is the base value for order-task pair (o, i), which is related to the effect of task i on completing order o. We use the following values for B_{oi}:

- (Reserve) $B_{oi} = 1$ if task i reserves new tote for order o, but not includes all items of the order.
- (ReserveAndComplete) $B_{oi} = 2$ if task i reserves new tote for order o, and includes all items of the order (hence completes the order).
- (Fill) $B_{oi} = 3$ if there already exists a partially filled tote for order o, and task i puts some more item(s) to the tote without completing the order.
- (Complete) $B_{oi} = 4$ if there already exists a partially filled tote for order o, and task i puts some more item(s) to the tote and completes the order.

Let us describe tote priority by an example scenario illustrated in Figure 4. We consider a pick station with five totes (T1 - T5) and consider collection of four orders (O1 - O4). The items of the orders are shown in the figure. In a specific time, tote 3 (T3) is reserved for order 2 (O2) and tote 4 is

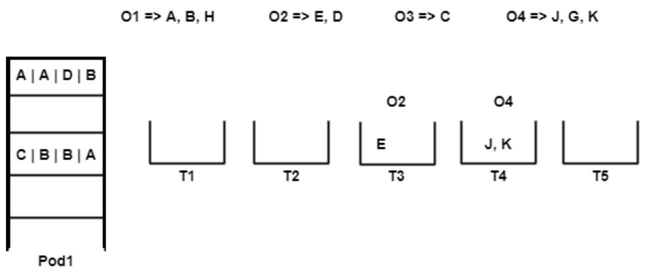


FIGURE 4. Tote reservation scenario.

reserved for order 4, and these totes are partially filled. Now we will calculate tote priority of the order task 1 (Pod1). Pod1 includes A and B items of order 1, but doesn't include H. Therefore this task 1 reserves a tote for order 1 without completing the order, so $B_{11} = 1$. On the other hand, order 2 has two items, E and D. E is already put in tote 3, and D is included in Pod1. Therefore task 1 completes order 2, and $B_{21} = 4$. Moreover, order 3 includes single item (C) which is included in Pod1. So task 1 reserves and completes order 3, i.e. $B_{31} = 2$. Task1 doesn't include any missing item of order 4, so order 4 will not be included in the calculation of tote priority. Since two of the five totes are already reserved, $\rho = \frac{2}{5} = 0.4$. Tote priority for task 1 is calculated as $1^{1.4} + 4^{1.4} + 2^{1.4} = 10.6$.

The rationale behind the proposed tote priority model can be described as follows. In order to efficiently utilize the totes, partially filled totes should be completed as soon as possible. Therefore, if a task fills and completes a tote, it would have highest priority and the base value is set to 4. If it fills a tote without completing, again it has a high priority and the base value is 3. Tasks that reserve new totes are given lower priority, especially if they only partially fill a tote. Tote priority becomes much more critical when only a few unreserved totes remain. Therefore the base values are powered by $1+\rho$, which converges to 2 as the number of idle totes gets closer to zero.

After finding spatial, temporal and tote priority values, overall priority of task i with respect to robot r can be simply defined as the sum of these values.

$$Pr(i, r) = pr^s_{ir} + pr^t_i + pr^\tau_i \qquad (5)$$

While priority of a task is effected by all three parameters, it is dominated by temporal priority (pr^t_i) if the task is too old, and dominated by tote priority (pr^τ_i) if the tote utilization is too high. All the orders created from the same run of the OBRTC algorithm have almost same temporal priority. For these order tasks, other priority metrics effect the order of selection. But if there exists tasks that are generated from an older order-to-task conversion process, a newer task is favored only if its pod is too close to the robot. This is provided by the logaritmic function used in pr^s_{ir}.

The proposed heuristic model is designed to dynamically achieve several performance goals such as reducing robot travel distance and order completion time, and increasing system utilization. Here it should be noted that although the task selection method provides efficient utilization of totes, it is still possible to not have an available tote for any of the items in a selected task. In that case this task is not assigned to the robot, and WES tries to select the next task in the calculated priority order.

3) ORDER TASK EXECUTION

Order task execution starts just after WES assigns an order task to the robot by the order task selection method described above. Figure 5 illustrates the order task execution flow. The robot goes to the pod and lifts it. Since existence of tote is checked before task assignment (as described in the previous part), it is guaranteed that there is at least one available pick station. If tote assignment is required for any items in the task, WES tries to assign all totes in a single pick station in order to decrease robot moving time. WES also tries to balance workload at the stations while selecting the totes.

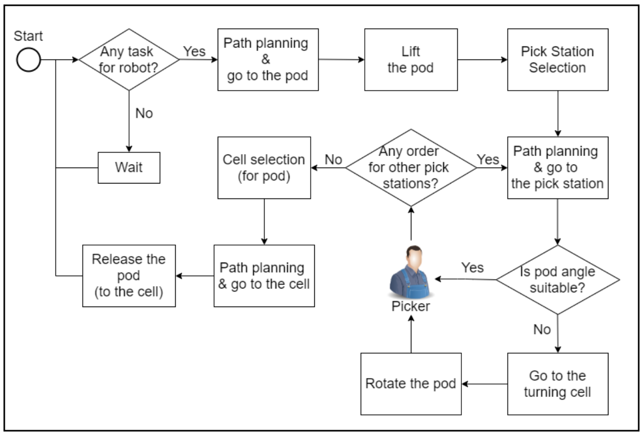


FIGURE 5. Order task execution flowchart.

The robot carries the pod to the selected pick station. When the robot is close to the pick station, WES checks whether the pod's current angle is appropriate for the pick station. If it is not as desired, the robot goes to the turning cell and the pod is rotated. After desired angle of the pod is provided, the robot takes the pod to the picker. Picker collects items and insert into the tote(s). After the last item of the current face of the pod is taken, WES checks whether another face of the pod is also required for this station. When the picking task is finished for all pod faces in this pick station, WES checks whether another pick station is waiting for that pod. If so, the same processes are handled until all required items at that pod are collected. Afterwards, WES selects a free cell for the pod to locate. When the robot goes to the pod cell and release it, order task is completed. Now the robot is ready for another order task, and task selection process is executed again. If there is no available task at that moment, the robot waits until a new task is created.

4) REPLENISHMENT TASK

Replenishment of goods is the other important task of WES that is performed by autonomous robots. Figure 6 illustrates overall replenishment task process and Figure 7 illustrates

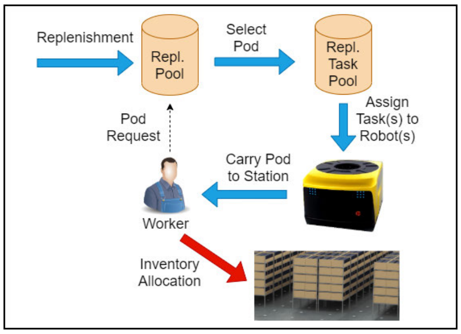


FIGURE 6. Replenishment task process.

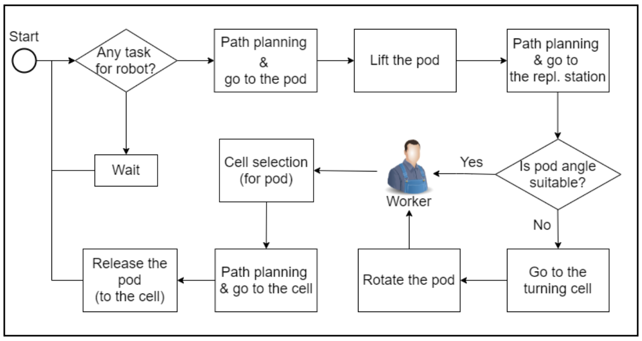


FIGURE 7. Replenishment task execution flowchart.

execution of replenishment task. Replenishment to task conversion is simpler than order to task conversion. It starts with the request of the worker in the replenishment station. Each station has a single worker. The worker requests a pod to fill with the replenishment goods. WES selects a pod according to the free space, i.e. the pods with more free space are prioritized. After selecting a pod, the system waits for an available robot if all the robots are busy. Replenishment task is assigned to the first robot that becomes available. The robot goes to the pod, lifts it and carries it to the replenishment station. If the pod's access direction is not appropriate, the robot first goes to the turning cell and rotates the pod as desired. Then the robot carries the pod to the station. A replenishment station is shown in Figure 8. The worker fills the pod with the items. Each item's barcode is first scanned with a barcode reader, and then the item is inserted in a shelf of the pod. The worker should also scan the shelf code in order to assign replenished items to the shelf. After the worker finishes these jobs, he/she informs the system by pressing a button. After that, the robot relocates the pod back to its place.

Several parts of the replenishment task execution directly depend on the decision of the worker. Replenishment task starts and ends upon request of the worker. Moreover, the worker decides which items to locate in the pod, and also in which shelf. The system may limit number of replenishment items even though the pod has enough space. Several studies (such as [12], [13]) analyze effects of pod selection and inventory allocation in the replenishment process. These works show that classification of items and spreading the

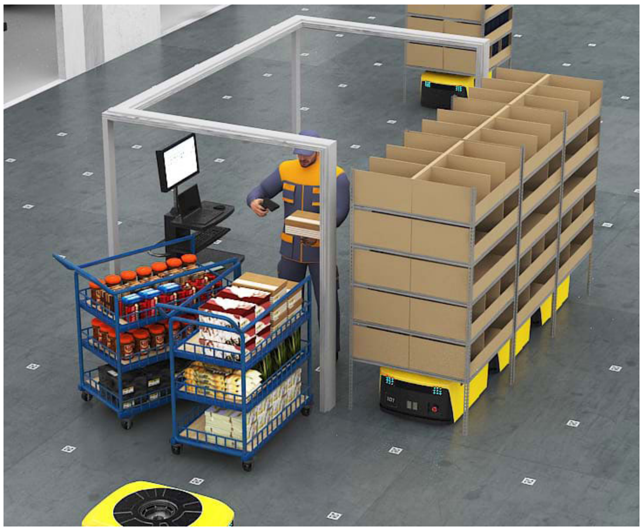


FIGURE 8. Replenishment station [18].

inventory across multiple pods will increase the pod selection options for order tasks and improve the order throughput performance. However, inventory classification significantly increases the complexity of the replenishment task. Inventory items should be known, classified and ordered. Although these additional efforts are mentioned in the related work, cost analysis is not performed and the trade-off between the extra effort and performance gain is not clear. In addition, the extra effort also requires extra resources in terms of equipment and workers, and extra rules.

Optimization of the replenishment task by inventory allocation decisions is not in the main scope of this study. In order to increase algorithm efficiency in the picking process, we adopt replenishment rules that are easy to install with low cost of pre-processing. In our model, WES does not have to know replenishment items and their counts. Items on the replenishment station may be held on pallets or anywhere. The worker can choose any item that is easy to take. Pod selection rule is emptiest-first as mentioned in [5]. Item selection and shelf selection decisions are left to the worker.

Most e-commerce companies sell some items although they are not stored in their warehouse. They should supply such items in a short time. In such warehouses, the replenished items typically include lots of already ordered items (of ''stock-waiting'' orders). This fact causes some challenges. Let us assume that there exists items of 100 stock-waiting orders in a replenishment task. The first question is how to locate these items to the pod. The second question is when to start collecting these orders.

As mentioned above, WES chooses the pod with most available free space for replenishment task. However, allocating many items of stock-waiting orders in a single pod will cause some unwanted situations. WES will want to reserve 100 totes for 100 stock-waiting orders' items stored in this pod. Nevertheless, pick stations have limited totes to collect orders simultaneously. Therefore managing such a highly desired pod will cause moving the pod to the pick station several times. In order to avoid this situation and increase manageability of the pods, we limit replenishment count for each pod. This way, the already ordered items will be allocated to more pods and pod selection options will be increased during the order task selection process.

At the first glance, it seems to be an effective solution to take the incoming pod directly to the pick station without placing it to the warehouse. If this pod contains all items of an order, this order will be completed quickly. However, in most cases this pod will only partially contain items of stock-waiting orders and it will cause creation of lots of order tasks. Managing large number of order tasks with a small number of totes is very difficult. Consequently, it is observed that taking replenishment items directly to the pick station reduces global system efficiency, although it may bring a local gain.

To sum up, we limit number of items to be stored in a single pod in the replenishment stations with n_p items. Furthermore, we relocate the pod to the warehouse without visiting the pick station even if it includes already ordered items. This pod will be converted to order task in the next OBRTC execution in the usual way.

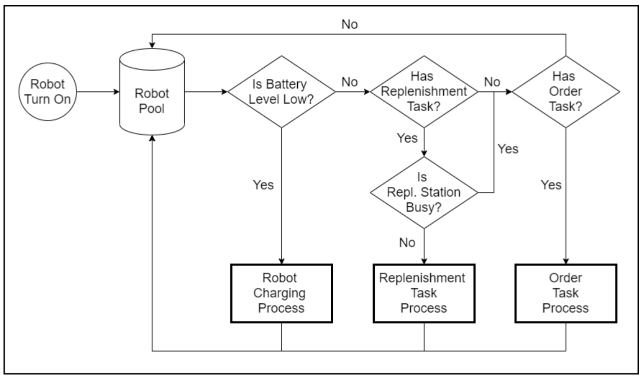


FIGURE 9. Robot life-cycle.

5) ROBOT LIFE-CYCLE

Automated mobile robots are physical components of WES. When a configured robot turns on, it sends a request to the WES by introducing itself. After handshaking, WES creates an entry for the robot in its database, if not exist. Robot life-cycle is shown in Figure 9. Robots wait in the robot pool until a new task is assigned to them. There are mainly three types of tasks: order, replenishment and charging tasks. WES will assign one of these three tasks to the robot according to the real-time dynamics of the warehouse. WES checks the battery level of the robot before assigning a task, If it is low, robot will go to the charging cell. It is charged up to a saturation level (which is defined as 85% of the full capacity in this study) if there is a waiting task in the system. Otherwise, it can wait until fully charging its battery.

The replenishment task starts with the pod request of the worker in the station. However, there should be a limit for concurrent pod requests of a worker to avoid needless waiting of robots in the replenishment station. In this study, we have limited the maximum number of concurrently running replenishment tasks with the replenishment station count. If there are replenishment items but there is no robot in the replenishment station (i.e. replenishment station is free), then WES creates a replenishment task and assigns it to the robot. Otherwise, WES assigns an order task to the robot with the order task selection method. The robot executes the order task as mentioned before. When the robot finishes its task, it informs the WES and asks for a new task. There is a separate thread running asynchronously for each robot. This avoids needless waiting of robots. A robot thread only waits if task selection process is running for another robot in order to avoid race conditions. Task assignment to a robot is done in less than a second, including all waiting time.

Task assignment is one directional. WES assigns tasks, but a robot cannot select its task, it can just request from the system. WES evaluates this request and handles task selection according to real time dynamics. If there is no task in the pool, it sends the robots to charging even if their battery levels are not low. WES checks new orders and replenishment for new task every minute. This system guarantees completion of order and replenishment processes with high utilization of resources.

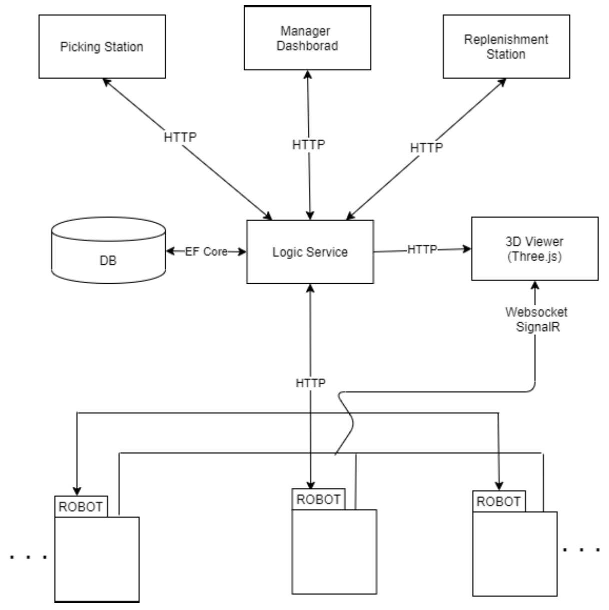


FIGURE 10. WES architecture.

III. SIMULATION STUDY AND NUMERICAL RESULTS

A. SIMULATION ENVIRONMENT

Figure 10 illustrates the main components of WES. The database functions (insert, update, delete, select) and business logic (task planning, resource allocation, etc.) are performed by the Logic (Core) Service. It is a RESTful service so other components connect with it via HTTP. Logistic Service, Pick Station and Replenishment station screens, Manager Dashboard and Layer 3 Robot API are. Net Core C# web applications. Pick Station screen is used by pick worker. When the robot arrives at the station with the pod, this screen shows the commands to the pick worker for collecting the order items. Replenishment station screen is used by the replenishing worker for inventory allocation. The Manager Dashboard is for management of warehouse process and components. Reports and results are also shown in the Manager Dashboard. The 3D Viewer shows robot movements in the warehouse in 3 dimensions and it is coded with Tree.js 3D Javascript library and Vue.js.

For each robot, we run a separate simulation application on the server as Robot Layer 3 API. Low level communication and process on the robot is simulated with realistic parameters. All the delays encountered while processing commands such as "go", "turn" or "lift" are considered in the simulations according to real measurements. Robot cannot access new command until finishing its last job. Robot charge management is also simulated. Robot charge decreases according to its state. For example, a robot will consume more energy when it picks a pod. All the parameters are set according to real measurements obtained from Robee of Makhina Robotics [18] shown in Figure 11.

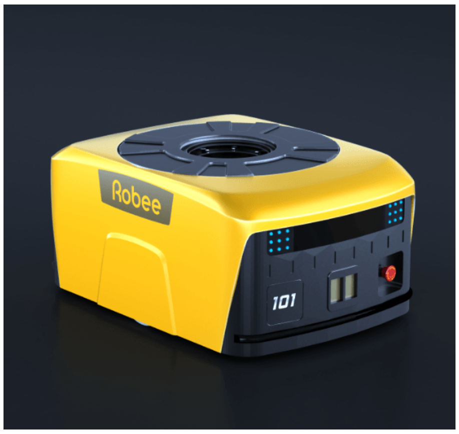


FIGURE 11. The physical robot (Robee of Makhina Robotics [18]) used in parameter setting.

Robot path planning is implemented according to the modified collision free A* algorithm proposed in our previous work [17]. However in this study, we implement some modifications as follows. In the previous algorithm, there exists some directed paths such that robots are allowed to go only in single direction. We relaxed this restriction and allow robots to go in opposite direction by assigning 5 times more cost for this movement. Increasing the cost per one cell movement will result in the selection of such paths very rarely, only if it has significant gain. Allowing the opposite direction will increase possibility of head to head encountering but it decreases the total robot traveling time by providing shorter options. Another modification is that robots are also allowed to go under the pods. Moving under the pods is not preferred because there is a restricted movement area and small deviations may cause stopping of the robots. Therefore we set the same cost as the opposite direction movement for moving under the pods. Path planning is out of the scope of this paper so we skip the details to [17].

We analyzed orders of a book e-commerce warehouse in a two months span. The mean of order line count for an order was 3,28. Approximately half of the daily orders included only a single item. We calculate the selling rate of the items. We create our simulation orders randomly using statistical characteristics of these orders, so that we accomplish realistic order line counts and items selling rate. Orders arrival to the WES is simulated with appropriate time intervals to analyze utilization of robots and picking stations with changing order density in time. We track and record state of all dynamic WES processes including robots, pods, shelves, items, orders, stations, and totes while simulation tests are running. Main assumptions of our simulation model are the following.

1) A pod can include up to 500 items. In general, we used approximately half of this capacity.
2) Orders can be single-line or multi-line.
3) Robot's velocity, acceleration/deceleration, and energy consumption will change according to the weight of the robot load (picked pod). We have simulated the weight effect according to whether the robot is loaded a pod or not. Loaded pod weight is set to 300 kg which is the half of the robot's maximum load capacity.
4) After an order completed in the tote at the pick station, a new free tote is located instead of the completed one. This process can be done separately without blocking picking processes. Hence, time of tote relocating is ignored.
5) Charging cell count is equal to the robot count so there is no queue for charging.
6) Each picking and replenishment station has a single worker.
7) 80% of our pods are two-sided and the rest are four-sided. The robot should rotate the pod to the desired side before the picking or replenishment process.
8) Time elapsed for picking one item at the pick station is set to 8 seconds. Time elapsed for locating an SKU to a pod at the replenishment station is set to 6 seconds. Process time per single item for worker at the replenishment station will be less than the pick station because typically replenishment task is easier. Replenishment workers mostly allocate same SKU items at once with less effort and time. These times may change according to the physical environment, worker capabilities, etc.
9) Totes are of the same size and they have enough space to contain items.
10) 10% of the orders arrive the system at the beginning. Remaining orders arrive the WES gradually in different time periods. We want to realize e-commerce orders such that there are different number of orders in the pool at different time periods. However, there will be

always enough orders in the pool to keep robots busy until all orders are completed.

11) Order arrival duration is divided into four time periods (for 2000 orders) and replenishment arrives as a bulk at the end of each period, if not otherwise stated. For more orders, number of time periods increase proportionally.

B. NUMERICAL RESULTS

We simulate our algorithms on WES under various design decisions and parameters. We have analyzed our tests under 6 subtitles: (1) Effect of order batch size (n_{ob}); (2) Effect of the number of robots; (3) Robots' charging time and workload balance; (4) Effect of the order count; (5) SKU effects on pile-on and time; (6) Effect of stock-waiting orders. All tests are performed for a warehouse where 624 pods are located in an area of $36 \times 43 = 1548m^2$. If not otherwise stated, there are 48272 SKU and 169343 items stored in the warehouse. The mean number of items per SKU is 3,5.

1) ORDER BATCH SIZE

In the first set of experiments, we analyze effect of the proposed algorithm for various values of n_{ob}, which corresponds to order batch size. Tests are performed for 12 robots, 2000 orders, 6560 order items and 6204 replenished items. There exists 2 or 3 pick stations, and each pick station has 20 totes.

We first simulated the system without applying OBRTC and ARTS methods. We set n_{ob} to one, in other words, each order is converted to a task individually and WES assigns these tasks to the robots randomly. In a warehouse with 2 pick stations (and 40 totes), 12 robots completed orders with 6560 items in 1637 minutes. The pile-on number is 1,3. For 3 pick stations (with 60 totes), same number of orders are completed in 1677 minutes. Then we applied the proposed OBRTC and ARTS methods by handling orders in batches. Initially we set n_{ob} to 10, and we increase it ten by ten in each new test in order to find the best value of n_{ob}. The results of tests can be seen in Figure 12. For 2 pick stations, the minimum time for collecting items (981 minutes) is obtained when n_{ob} is 70 for 40 totes. The pile-on values is 2,6. As mentioned before, achieving high pile-on value is very important to reduce total time for collecting orders. The results show that the proposed methodology (with an optimal n_{ob} selection) reduces the process time by 40% for a warehouse system with 2 pick stations. For a warehouse with 3 pick stations, the minimum completion time is found as 910 minutes when n_{ob} is set to 120. The pile-on value is 2,99. For this case, reduction in total time is 45,7%, which is even more than the case with 2 pick stations.

The results shown in Figure 12 indicate that after an optimal value, further increasing the n_{ob} value would start increasing the order completion time. This is mainly because of the tote limitation. When there is no available tote in the pick station, pods will be returned back to the warehouse without all the items collected. Then they will be carried to the pick station again after totes become available.

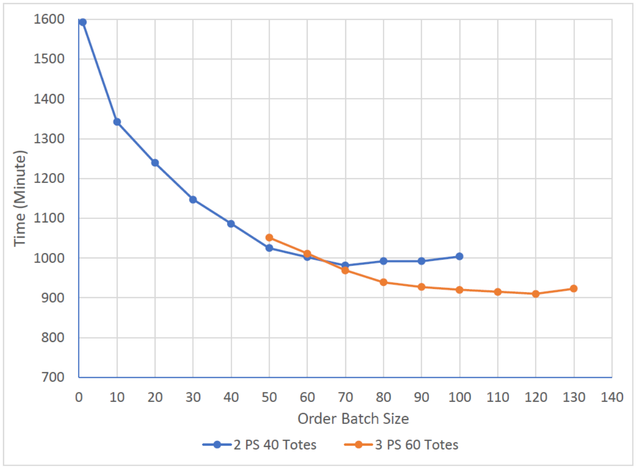


FIGURE 12. Order batch size (n_{ob}) vs orders completion time.

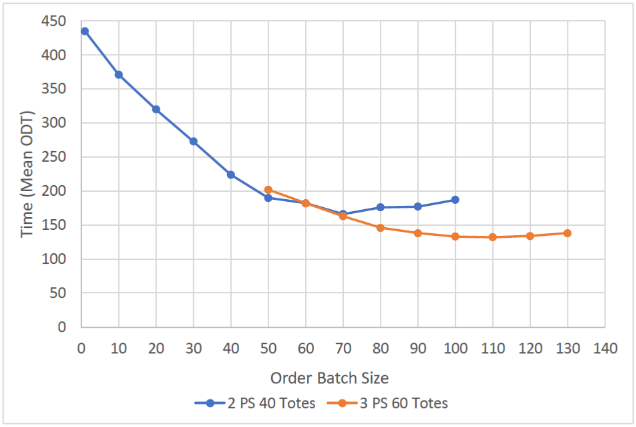


FIGURE 13. Order batch size (n_{ob}) vs average order departure time.

n_{ob} value has also an important effect on the order departure time (ODT), i.e. the waiting time of an order until departing from the warehouse. The relation between n_{ob} and mean ODT is shown in Figure 13. It is observed that the best mean ODT value is obtained at the optimal n_{ob} value. Although the ROB algorithm changes FIFO order and items of an order may be spread over multiple order tasks, mean ODT value decreases with the batch size until the optimal n_{ob} value. Temporal priority in the ARTS method provides a balance to avoid long waiting for an order in order task pool. The maximum ODT, in other words, the longest waiting time of an order in the pool, is approximately double the mean ODT value. The mean and maximum values of ODT are 166 minutes and 305 minutes respectively when n_{ob} is 70 and tote count is 40. These values are 134 minutes (mean) and 260 minutes (max) for 60 totes.

2) ROBOT COUNT

We analyze the effect of robot count on the order completion time and pile-on. We test for 2 pick stations (with 8-16 robots), 3 pick stations (with 8-20 robots) and 4 pick

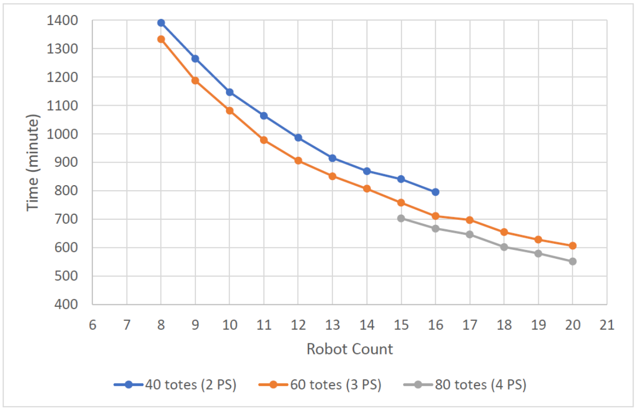


FIGURE 14. Robot count vs orders completion time.

stations (with 15-20 robots). n_{ob} is set to 70 for 2 pick stations and 120 for 3 and 4 pick stations. Figure 14 illustrates the effect of the robot count on the order completion time. Firstly, it is observed that completion time does not decrease linearly with the number of robots. The contribution of extra robot decreases as number of robots increase. This is because of extra waiting of robots on path and picking line. As an example result, a robot can carry 37 items per hour (on average) if 8 robots are employed, while 32 items can be carried per hour when 20 robots are employed (for 3 pick stations).

Another result is related to the increment in the number of pick stations. Increasing number of totes increase optimization effect as mentioned before. However, there is a limit for number of totes in a pick station. Furthermore, opening new pick station has a cost of employing new worker. Hence, pick station count should be well optimized. Number of pick stations should be proportional to number of robots in order to increase utilization of pick workers. For example, using 3 or 4 pick stations is unnecessary for 8 robots, because robots will not be able to provide enough items to the pickers. Figure 14 shows that the difference in completion times for 40 and 60 totes (for the same number of robots) increases with the robot count. When 8 robots are employed, adding new 20 totes yields a 4% decrease in the completion time, while this ratio is (10%) when 16 robots are employed. Similar result is obtained when we compare the results for 60 and 80 totes. Decrease in the completion time is 6% for 16 robots and 9% for 20 robots.

The effect of the robot count on the pile-on value is shown in Figure 15. It is observed that while the number of robots is increased, the pile-on value has been preserved with minor changes.

3) ROBOTS' CHARGING AND WORKLOAD BALANCE

We analyze the balance in the workload and charging time of robots in the warehouse setting with 10 robots and 2 pick stations. All robots work for 42 hours (including charging times) and collect 13073 items. The results are shown in Table 1. The total covered distances and total charging times for all robots are very close to each other. Any robot spends approximately

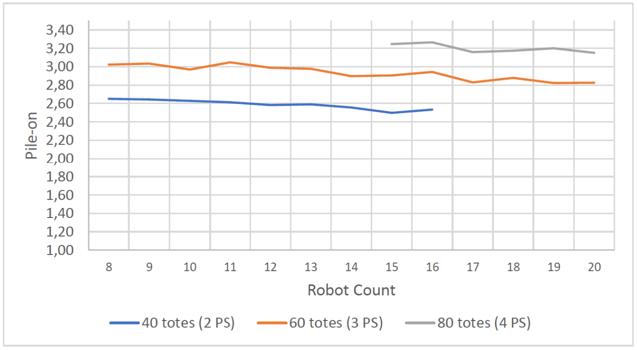


FIGURE 15. Robot count vs pile-on.

TABLE 1. Robots' charging and workload balance.

Robot No	Distance Covered (km)	Charging Time / Total Time
Robot 1	56,61	0,1459
Robot 2	56,38	0,1432
Robot 3	56,98	0,1472
Robot 4	56,56	0,1450
Robot 5	56,70	0,1455
Robot 6	56,10	0,1437
Robot 7	56,53	0,1453
Robot 8	56,48	0,1460
Robot 9	56,84	0,1456
Robot 10	55,96	0,1448
Average	**56,51**	**0,1452**

14.5% of the time on charging. This indicates that all the robots consume almost same amount of energy. It can be realized that the balance in the workload and energy consumption of the robots comes with the nature of RMFS, where robots are employed anonymously and are busy most of the time. This result suggests that there is no need for extra load balancing effort (such as [16]) in the proposed system.

TABLE 2. Effect of order count.

Order Count	Order Items	Task Count	Repl. Items	Time (min)	Dist. (km)	Pile-on	Items/ Robot (hourly)
2000	6560	2082	6204	551	259	3,15	35,72
3000	9757	3059	9491	837	390	3,19	34,97
4000	13073	4100	12511	1116	519	3,19	35,14
5000	16305	5144	15552	1402	650	3,17	34,89

4) ORDER COUNT

We increase number of orders up to 5000, and the number of replenished items in parallel. We test for 20 robots and 4 pick stations (80 totes). Table 2 illustrates the obtained results. It is observed that there is an approximately linear relation between order count and completion time. Furthermore, the pile-on value is preserved while the order count increases. For each case, a robot carries around 35 items per hour.

5) SKU EFFECTS ON PILE-ON AND TIME

SKU and stock count have important effect on pile-on value. Increasing SKU decreases the pile-on value. Merschformann *et al.* [5] describe the negative effect of high SKU by performing tests for up to 10000 SKU. However, in the above tests, we analyze our algorithms for 48272 SKU which is approximately 5 times more than the maximum SKU value considered in [5]. We perform another set of tests after decreasing SKU to 22603 for approximately same amount of stock, and increasing stock per SKU from 3,51 to 7,53. The tests are performed for 2000 and 4000 orders, 12 robots and 3 pick stations. The results are shown in Table 3. When SKU is decreased, the pile-on value increases from 2,99 to 3,7 (for 2000 orders) and 3,02 to 3,78 (for 4000 orders). This corresponds to an increase of approximately 25%. As a result, 4000 orders (12868) are collected in $1815 - 1601 = 214$ minutes less time after decreasing the SKU value.

TABLE 3. Effect of SKU.

Stock/ SKU	Order Count	Order Items	Task Count	Repl. Items	Time (min)	Dist. (km)	Pile-on	Items (hourly)
3,51	2000	6560	2197	6204	913	267	2,99	431,11
7,53	2000	6434	1741	5510	791	222	3,70	488,04
3,51	4000	13073	4330	12511	1815	526	3,02	432,17
7,53	4000	12868	3404	11059	1601	446	3,78	482,25

6) EFFECT OF STOCK-WAITING ORDERS

In the replenishment station, the maximum number of items that can be located in a pod is limited with n_p in order to manage stock-waiting orders, as mentioned in Section II-B4. In all our tests, we set n_p to 40. In order to test the system with more stock-waiting orders, we remove some items arbitrarily from the stock at the beginning. Tests are performed for 12 robots, 3 pick stations and 2000 orders (6560 items). We consider 3 different replenishment scenarios which result in different number of stock-waiting orders. Although all 3 scenarios have the same number of items to replenish as 6204, they differ in the time intervals that the replenishment arrives to the warehouse. In the first scenario, all the replenishment arrives as a bulk after arrival of all orders. In the second scenario, replenishment is partitioned into two parts. The first part arrives to the system in the middle and the second part arrives at the end. In the third scenario, replenishment is partitioned into four parts and each part arrives in equal time intervals.

TABLE 4. Effect of stock waiting orders.

Waiting Orders	Waiting Items	Repl. Part.	Repl. Items	Time (min.)	Pile-on	Distance (km)
0	0	4	6204	913	2,99	267
208	410	4	6204	983	2,64	290
281	619	2	6204	992	2,57	294
346	862	1	6204	1050	2,50	300

The effect of stock-waiting orders can be observed in Table 4. When stock-waiting orders increase, total collection time of all orders also increase. This increase is aligned with expectations, because it causes lower utilization of the system while orders are waiting for stock, compared to the case where these order tasks are distributed throughout the day. Hence, this fact causes a decrease in the pile-on value.

IV. CONCLUSION

In this work, we propose a task planning approach for RMFS-based smart warehouses, with the aim of utilizing system resources efficiently and adaptively. We describe a novel algorithm for conversion of order batches to tasks. We also propose a heuristic model for assignment of tasks to robots. We consider various system dynamics such as location and battery level of robots, replenishment of new stock in various pods, utilization level of totes and time elapsed after generation of order tasks. We develop a highly realistic simulation environment which can be considered as a digital twin of the real system. We take into account all realistic considerations such as routing, queuing, turning, energy consumption and charging of the robots. We analyzed the total elapsed time, pile-on value, mean order waiting time and total distance covered by the robots. We realize crucial relations between all of these outputs, especially between pile-on value and total elapsed time. It is shown that the proposed algorithms preserve the pile-on value for various number of robots and orders, and work well under different situations even in high SKU. The obtained results show that the reduction in total completion time of orders is about 40% in a warehouse with 2 pick stations and 40 totes, and about 46% in a warehouse with 3 pick stations and 60 totes.

Although this work gives significant insight into how to design an efficient RMFS-based warehouse system, there are always open directions for research. In this study, in the order-to-task conversion step, orders are handled with the order they arrive. As a future work, an additional optimization study can be performed while selecting orders into the order batch. Another idea is to allow allocation of temporary totes in pick stations when there is no available tote for some of the order items in a coming pod. This may increase efficiency but also increase complexity due to additional process for workers. Performance effects and induced costs of these further improvements remain to be discovered.

ACKNOWLEDGMENT

Ali Bolu worked as a Robotics Engineer for developing RMFS in Makhina Robotics.

REFERENCES

[1] P. R. Wurman, R. D'Andrea, and M. Mountz, "Coordinating hundreds of cooperative, autonomous vehicles in warehouses," *AI Mag.*, vol. 29, no. 1, p. 9, 2008.
[2] N. Boysen, R. de Koster, and F. Weidinger, "Warehousing in the E-commerce era: A survey," *Eur. J. Oper. Res.*, vol. 277, no. 2, pp. 396–411, Sep. 2019.
[3] Steve Banker. (2016). *Robots in the Warehouse: It's Not Just Amazon*. Accessed: Sep. 7, 2020. [Online]. Available: https://www.forbes.com/sites/stevebanker/2016/01/11/robots-in-the-warehouse-its-not-just-amazon/?sh=d7f4dcc40b8f
[4] T. Lamballais, D. Roy, and M. B. M. De Koster, "Estimating performance in a robotic mobile fulfillment system," *Eur. J. Oper. Res.*, vol. 256, no. 3, pp. 976–990, Feb. 2017.
[5] M. Merschformann, T. Lamballais, M. B. M. de Koster, and L. Suhl, "Decision rules for robotic mobile fulfillment systems," *Oper. Res. Perspect.*, vol. 6, 2019, Art. no. 100128.

[6] J. Zhang, X. Wang, and K. Huang, "Integrated on-line scheduling of order batching and delivery under B2C E-commerce," *Comput. Ind. Eng.*, vol. 94, pp. 280–289, Apr. 2016.

[7] J. J. Enright and P. R. Wurman, "Optimization and coordinated autonomy in mobile fulfillment systems," in *Proc. 25th AAAI Conf. Artif. Intell.*, 2011, pp. 33–38.

[8] L. Xie, N. Thieme, R. Krenzler, and H. Li, "Efficient order picking methods in robotic mobile fulfillment systems," 2019, *arXiv:1902.03092*. [Online]. Available: http://arxiv.org/abs/1902.03092

[9] N. Boysen, D. Briskorn, and S. Emde, "Parts-to-picker based order processing in a rack-moving mobile robots environment," *Eur. J. Oper. Res.*, vol. 262, no. 2, pp. 550–562, Oct. 2017.

[10] B. Zou, Y. Gong, X. Xu, and Z. Yuan, "Assignment rules in robotic mobile fulfilment systems for online retailers," *Int. J. Prod. Res.*, vol. 55, no. 20, pp. 6175–6192, Oct. 2017.

[11] R. Hanson, L. Medbo, and M. I. Johansson, "Performance characteristics of robotic mobile fulfilment systems in order picking applications," *IFAC-PapersOnLine*, vol. 51, no. 11, pp. 1493–1498, 2018.

[12] F. Weidinger, N. Boysen, and D. Briskorn, "Storage assignment with rack-moving mobile robots in KIVA warehouses," *Transp. Sci.*, vol. 52, no. 6, pp. 1479–1495, Dec. 2018.

[13] T. Lamballais Tessensohn, D. Roy, and R. B. M. De Koster, "Inventory allocation in robotic mobile fulfillment systems," *IISE Trans.*, vol. 52, no. 1, pp. 1–17, Jan. 2020.

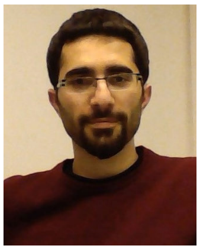
[14] D. Roy, S. Nigam, R. de Koster, I. Adan, and J. Resing, "Robot-storage zone assignment strategies in mobile fulfillment systems," *Transp. Res. E, Logistics Transp. Rev.*, vol. 122, pp. 119–142, Feb. 2019.

[15] Z. Yuan and Y. Y. Gong, "Bot-in-time delivery for robotic mobile fulfillment systems," *IEEE Trans. Eng. Manag.*, vol. 64, no. 1, pp. 83–93, Feb. 2017.

[16] L. Zhou, Y. Shi, J. Wang, and P. Yang, "A balanced heuristic mechanism for multirobot task allocation of intelligent warehouses," *Math. Problems Eng.*, vol. 2014, Nov. 2014, Art. no. 380480.

[17] A. Bolu and O. Korcak, "Path planning for multiple mobile robots in smart warehouse," in *Proc. 7th Int. Conf. Control, Mechatronics Autom. (ICCMA)*, Nov. 2019, pp. 144–150.

[18] *Boost Your Warehouse With Robee*. Accessed: Dec. 25, 2020. [Online]. Available: https://www.makhina.com/product



ALI BOLU (Member, IEEE) received the B.Sc. degree in telecommunication engineering from Istanbul Technical University, Istanbul, Turkey, in 2015. He is currently pursuing the M.Sc. degree in computer engineering with Marmara University, Istanbul, Turkey.

He worked as a Robotics Engineer for developing RMFS with Makhina Robotics. He has been working on robotic software technologies, since the beginning of 2018, coping with math, optimization, and algorithmic challenges. He currently works in Delivers.ai as a Robotic Software Engineer. His research interests include robotics systems and their optimizations.

ÖMER KORÇAK (Member, IEEE) received the B.S. (Hons.), M.S., and Ph.D. degrees in computer engineering from Bogazici University, in 2002, 2004, and 2009, respectively. In 2011, he worked as a Visiting Researcher with Deutsche Telekom Laboratories, Berlin. He is currently an Assistant Professor with Marmara University, Istanbul. His main research interests include in the general area of network optimization, network economics, game theory and resource management, and with applications to wireless networks and multi-agent systems.