Int. J. Adv. Eng. Pure Sci. 2024, 36(3): <235-246>
DOI: 10.7240/jeps.1475312

RESEARCH ARTICLE / ARAŞTIRMA MAKALESİ

Solving an Order Batching and Sequencing Problem with Reinforcement Learning

Begüm Canaslan [1], **Ayla Gülcü** [2]

[1] *Netaş Telecommunications, Innovation and R&D Strategies Department, 34912, Istanbul, Turkey*

[2] *Bahçeşehir University, Department of Software Engineering, 34349, Istanbul, Turkey*

Abstract
The purpose of this research is to determine whether a DRL solution would be a suitable solution for the OBSP problem and to compare it with traditional methods. For this purpose, models trained utilizing the PPO algorithm were tested in a complex and realistic warehouse environment, and an attempt was made to measure whether a strategy was developed to decrease the number of orders being late. A heuristic method was also applied and the results were compared on the same environment and data. The results showed that DRL approach that combines heuristics with the PPO algorithm outperforms the heuristics in minimizing the tardy order percentage in all tested scenarios.
Keywords: Reinforcement Learning, Order Batching and Sequencing, Proximal Policy Optimization, Warehouse Optimization

I. INTRODUCTION

Warehousing entails the act of storing physical goods or inventory within a warehouse or stock facility prior to their sale or distribution. It enables companies to meet the increasingly demanding customer requirements for fast and effective order processing, a necessity heightened in the digital age. Optimization of warehouse operations is an issue that needs to be addressed in e-commerce, and has gained more importance with the increasing e-commerce volume. Solutions in this area are necessary to enhance efficiency, reduce costs, improve order accuracy, maximize space utilization, streamline workflow, and increase overall productivity, thereby meeting customer demands effectively and maintaining competitive advantage.

Mainly two types of stock systems can be mentioned in today's warehouses, which are Person-to-Goods (PtG) and Goods-to-Person (GtP) systems. In PtG systems, order pickers follow designated paths, which can lead to inefficiencies when dealing with small orders due to extensive walking among the shelves and the central depot. GtP systems leverage automated solutions to bring items to pickers stationed at specific areas. GtP offers fast order picking and significantly decreases the reliance on human labor, but they come with a considerably higher initial investment compared to PtG systems.

Items must be collected from the warehouse stock systems before orders are prepared. The two types of picking methods are single-select and multi-select. In single-select method, just one order is picked in a tour, whereas in multi-select method multiple orders are selected. Single-select results in shorter lead times but carries the potential drawback of inefficient traveled distances, while batching orders leads to longer lead times but boosts the picker's productivity.

Optimizing stock systems, on the other hand, is crucial for improving the overall performance of a warehouse. It enhances efficiency, accuracy, and productivity while reducing costs and errors. The optimization of a specific warehousing system which combines PtG and GtP stock systems to improve efficiency is studied in the literature. Because of the limited capacity of warehouse resources and tight delivery schedules, an order picking plan is

Corresponding Author: BEGÜM CANASLAN, **Tel:** 05385873567, **E-mail:** begumc@netas.com.tr
Submitted: 29.04.2024, **Revised:** 22.06.2024, **Accepted:** 27.06.2024

required to reduce the quantity of late orders. For instance, a Deep Reinforcement Learning (DRL) based approach is proposed in [1] for the optimization of order picking process to cope with the tardiness in delivery in a specific warehouse environment. Heuristics, which are often employed in optimization problems where the search space is vast and exploring every possible solution is not feasible have also been applied by many researchers with a purpose of minimizing tardy orders or decreasing total processing time of the orders [2][3]. [4] proposes a strategy to optimize warehouse layout, aiming to minimize transport time and reduce injury risks by employing association rule analysis with the apriori and FP-growth algorithms.

This research aims to create a method that determines an order picking strategy to use, with the goal of minimizing order delays in an environment where resources are limited. To be more specific, it should be decided whether orders will be picked individually, or in batches, and if they will be picked in batches, which orders will be in the same batch. The problem is called Order Batching and Sequencing Problem (OBSP) in existing literature, and proposed solutions are mostly conventional methods such as heuristics.

In [1], a deep reinforcement learning (DRL) methodology that applies a Proximal Policy Optimization (PPO) algorithm for minimizing the number of tardy orders on their specified warehousing concept is proposed. The solution is improved by including a more dynamic warehouse environment, and optimizing the RL solution in [5].

The purpose of the study is to investigate whether DRL will be a good solution to OBSP of a warehouse system that has been already utilized in e-commerce. The main criteria is minimizing the number of tardy orders, which has become an issue of increasing importance because of time-limited delivery expectation and order uncertainties in today's e-commerce. This research is also built upon the same environment given in [1] since it fullfills the requirements for the topic and reflects the general. The generalizability of the DRL solution is investigated by testing the methods used in the previous literature with different order arrival data. In addition, how the performance of the solution could be improved with different parameters was tested.

This paper is structured as follows: Relevant literature in problem domain has been overviewed in Section 2. How the problem is formulated to apply for both RL solution and heuristic solutions are presented in related subsections of Section 3. In Section 4, the experimental findings are deliberated. Conclusion and some possible future imrovements are given in Section 5.

II. RELATED WORK

Traditional warehousing systems need to adapt to the requirements such as small orders, wide variety, strict delivery timelines, and fluctuating workload. Order picking in a batching, zoning and sorting environment is the most researched topic in this context, however it still needs further research to address the characteristics of e-commerce [6].

In logistics, there exists many machine learning solutions for various problems. For instance, [7] proposes a novel approach using multiple machine learning models to address Pallet Loading Problem, which involves maximizing the number of boxes loaded onto a pallet and a major issue in shipments.

Interest on Reinforcement learning (RL) applications especially in logistics increased a lot in the last few years. Q-learning, both tabular form and Deep Q-Network (DQN) is the most favored RL method researched, followed by policy gradient and actor-critic methods. Among all the RL agents used, multilayer perceptrons and regression are the most popular ones. Heuristics such as genetic algorithms, greedy algorithms and dynamic programming are used as benchmarking methods in many studies [8].

RL is advantageous over heuristics because it learns parameters to determine actions based on the current state whereas in heuristics predefined rules are utilized, it may learn from historical data and integrate forecasting and optimization whereas heuristics only uses forecasting on prediction and it can rely on simulation environments. The limitations of RL are the challenge in handling complex multi-agent systems and coming up with generalized solutions, the cost and complexity of the solutions, and partially observable states [8].

In following sections, some heuristics and RL solutions in the literature are reviewed.

2.1. Heuristic Approaches for OBSP

Metaheuristic algorithms utilizing Iterated Local Search, Attribute-Based Hill Climbing, and a basic tabu search principle have been employed in [9] to reduce the overall tardiness for a specific collection of customer orders. The results of the proposed methods are evaluated against the typical constructive heuristics such as the Earliest Due Date rule on various categories of problems. It has been demonstrated that the results are highly improved and the proposed solutions can be used for a more efficient order picking system. Later, Variable Neighbourhood Descent and Variable Neighbourhood Search have been implemented within [10] on the same problem with the same objective, the findings indicated that the suggested approaches may improve the order picking effectiveness.

A heuristic solution grounded on the Variable Neighbourhood Search method to achieve a minimum tardiness in the context of the OBSP has also been proposed in [2], and it has been shown that the proposed algorithm is better than the state of the art in aspect of quality and speed.

The modified seed algorithm proposed in [3] aims to reduce the overall picking, sorting and packing duration in an OBSP with limited buffers. The initial seed algorithm attempts to consolidate orders with nearby stock locations into the same batch to minimize picking time. However, it fails to achieve coordinated production of the order picking process and sorting-packing process with limited buffers. Therefore, the modified algorithm takes into account the relationship between the current batch and the previous batch in terms of processing time.

Heuristics have also been utilized in many related problems in operations research domain. For instance, the order picking problem is handled in [11]. A method involving multiple Genetic Algorithms (GAs) is introduced for optimizing batch picking, taking into account travel costs and order due times. The results showed that batch picking achieves better results than single order picking in general, and the proposed model has a solution quality better than all benchmark models in all datasets.

The joint order batching and picker routing problem in warehouses that is seldom studied together and not studied at all for warehouses consisting of multiple blocks is investigated in [12]. They formularized the issue revolving around an exponential quantity of connectivity constraints and presented several inequalities based on the conventional layout of warehouses. They showed the applicability of the proposed method by presenting results for problems with up to 5000 orders.

2.2.Reinforcement Learning Applications for OBSP

Reinforcement Learning (RL) is a subset of machine learning focused on teaching agents to make sequences of decisions in dynamic environments to maximize cumulative rewards. Unlike supervised learning, where models are trained on labelled data, or unsupervised learning, which seeks to identify patterns in unlabeled data, RL relies on trial-and-error interactions to discover optimal strategies. Deep Reinforcement Learning (DRL) integrates RL with deep learning techniques to solve problems in complex environments.

There exist a few studies with RL on the spesific problem of OBSP. In [1], it is questioned how can a DRL solution contribute to minimizing the quantity of late orders within a particular warehousing concept. The problem is formulated as a Semi-Markov Decision Process and solved with a Proximal Policy Optimizaiton (PPO) algorithm. To benchmark the algorithm, several heuristic solutions are also developed. A simulation model built in a 3D simulation program to assess the effectiveness of the algorithms and an order dataset belong to an e-commerce company are used. It is concluded that DRL is a preferred method because it generalized across various warehouse setups well, eliminating the need to train a new agent. The importance of the study is that there was no literature about solving order batching problems with DRL at its time. A paper based on [1] has been published later [13].

An RL algorithm for a different kind of batching problem is studied in [14]. The problem was minimizing the difference between the target weight and the real weight of a product batch. The difference which is called as giveaway can cause customers get an extra amount or lose some of the product they ordered. There are regulations called e-weighting regulations applied by European Union Directive to prevent customers getting a less amount. They converted the environment to a model that changes in episodes to implement RL. The complexity of the algorithm increased in several iterations to close to a real production setup, and the results was passing the e-weighting regulations after the third iteration.

Cals, Zhang, Dijkman, and van Dorst (2021) later published a paper based on Cals (2019). They applied PPO in conjunction with a heuristic rule to adress OBSP. The agent used heuristics for sequencing decisions and DRL for batching decisions, and the resulting performance was compared to several heuristic approaches. Results showed that the agent utilizing DRL for order batching surpasses the heuristics accross most warehouse settings examined, and the results are more robust and generalizable.

[5] and [15] worked on the same problem and addressed the problem in a more realistic way by involving larger instances of hourly orders. They also included a second objective of reducing order picking costs in addition to the first objective of reducing tardy order percentage. All these improvements added more complexity to the problem, so he improved the solution by using a method utilizing Bayesian optimization for shaping rewards. Additionally, they involved approximating DRL policies using decision trees, which can then be used to deduce logic and generate understandable decision rules, thus enhancing the explainability of the learned policies.

III. METHODOLOGY

3.1. Problem Description

The scope of this research is finding a solution to schedule and fulfill orders for a warehouse system. This research is built upon the same warehousing system mentioned in [1] due to the difficulty of accessing real data. Also, the concept applies for modern warehouses and suitable to study the research topic. The warehousing system is developed by an anonymous

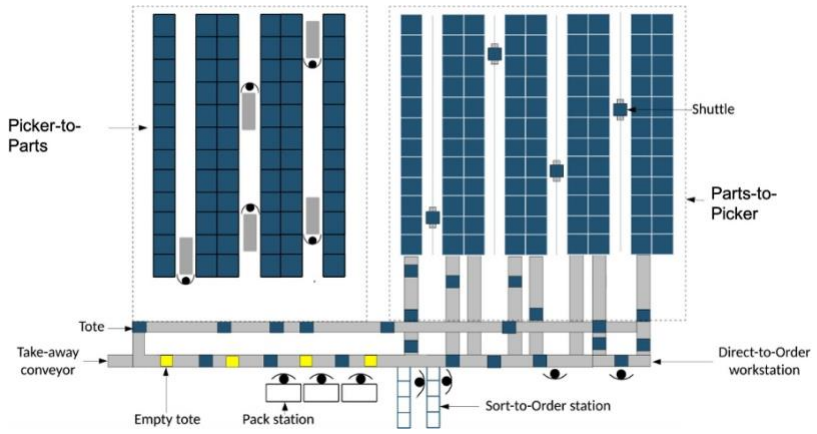

Figure 1. PtG and GtP order retrieval system incorporating a pack zone, StO and DtO zones [1]

logistics company to provide solutions to the problems in e-commerce market such as tight delivery schedules, orders with small number of items, large assortment, fluctuations in workload, labour scarcity and growing e-commerce market.

The proposed system combines PtG and GtP to take advantage of both, and also applies a unique batching method by using single-select and multi-select simultaneously. When using this warehousing setup, two choices must be rendered:

1. When should single-select or multi-select be chosen
2. If multi-select is chosen which orders ought to be combined together in the same batch

The first issue is an order batching problem, whereas the second one is a sequencing problem. Therefore, the problem is an Order Batching and Sequencing Problem (OBSP). The issue is complex because there are many inputs that will affect the decisions. The decisions are influenced by the attributes of the orders, various processing stages, limitations in capacity, and uncertainties associated with the orders.

There exist two kind of stock systems, PtG and GtP, and three kind of operating areas in the proposed warehouse system which is shown in Figure 1. Items are stored on shelves or palettes in PtG systems, and picker stops at the locations where certain items are stored and picks the quantity needed. The GtP system operates as an Automated Stock/Retrieval System (AS/RS), retrieving items from shelves and transporting to a area at which a picker is stationed. The picker then retrieves the requested items from the tote. In the PtG approach, where single-select is implemented, the picker gathers the items into a carton box, preparing them for shipping without stopping at areas. When multi-select, totes are used in the PtG system. Two kinds of totes exist which are product totes and batch totes. Product totes contains items of same SKU and are stored in the GtP whereas batch totes are created in the PtG system while multi-select. Batch totes are subsequently moved to either the GtP system or a packing area.

Totes have the option to be conveyed to three distinct areas: direct-to-order (DtO) zones, sort-to-order (StO) zones, and packing zones. At a DtO zone, totes of products arrive sequentially, and the required items for each order are picked and deposited into a carton box. Alternatively, batches can be formed by gathering items one by one into a batch tote instead of a carton box. These batch totes are stored in GtP system, later dispatched to either a StO or packing zone. At a StO zone, orders are sorted, buffered and packed. First, a picker extracts products from a batch tote and positions them in a put wall, where shelves are designated to a unique order one at a time. After the put wall is occupied and the needed products are gathered for every individual order, they are sent to the buffering area. Then, the operator asks for a put wall, places it into the packing area, packages each order in a carton box and sends it to shipping. Product totes can also be used instead of batch totes, but it may cause long queues since many of them will be required. When all items does not exist in PtG system, A hybrid approach involving both product totes and batch totes is also feasible for completing the missing items from GtP system. At a pack zone, only batch totes arrive and sorting is not required since each item within the batch corresponds to a single order. Here, the processed orders consist of one type of product.

An order consisting of only one item and one SKU is referred to as a single-item order (SIO). An order that contains multiple items and multiple SKUs is called multiple item order (MIO). Order type is an important factor on order picking process.

This research concentrates on aforementioned stock units and areas. Other aspects of the warehouse system are disregarded to prevent further complexity in the problem. Additionally, certain assumptions are made, such as consistently having adequate capacity to buffer

orders and always being able to transport totes between processes via conveyors. Although the proposed warehouse system is designed for much higher amount of orders, research is limited by 360 to 500 orders per hour. The algorithm takes inputs including a warehouse configuration with pickers, shuttles, and areas, as well as order arrival events. It sequentially allocates orders to the elements of the warehouse, ensuring that the limitation on capacity is respected and the number of late orders is minimized.

3.2. Solution Methods

A DRL solution is applied on the aforementioned OBSP. Also a heuristic solution is applied to use as a benchmark. The details of the applied methods are given in the following sections.

3.2.1. Deep reinforcement learning basics

RL is a type of machine learning where an agent learns to make a sequence of decisions (actions) within an environment to maximize a cumulative reward signal. Upon taking an action, the environment responds with a reward and possibly a new state. The aim of the agent is to acquire an optimal policy which makes its long-term reward as large as possible. The conceptualized model of RL can be viewed in Figure 2.

DRL is a subfield of machine learning that merges principles from RL with deep learning techniques, to enable agents - the learner or decision maker that interacts with the environment - to learn how to make decisions.

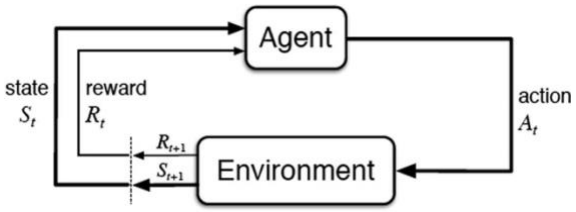

Figure 2. A conceptual model for reinforcement learning [17]

DRL has shown remarkable achievements in numerous fields such as robotics, gaming, autonomous driving, finance, and healthcare. It poses significant challenges, including sample inefficiency, instability during training, and the need for extensive computational resources. However, recent advances in algorithms (e.g., deep Q-networks, policy gradient methods), hardware (e.g., GPUs, TPUs), and environments (e.g., simulation platforms like OpenAI Gym) have accelerated progress in DRL research and applications.

Considering that the OBSP involves making consecutive decisions in presence of uncertain demand and postponed rewards (order batching might decrease the capacity for other orders usage, possibly causing delays), Deep Reinforcement Learning (DRL) may be an appropriate method for a solution.

3.2.2. Simulation model

Simulation environments are necessary develop, test, and refine RL agents in a safe, efficient, and cost-effective manner. The simulation environment for the DRL agent for our spesific OBSP has been developed using OpenAI Gym [16]. There exist seven entities which are shuttle, picker, order/batch, product tote, batch tote, carton box and queues in the simulation model. Entities can carry data by means of its attributes. Whenever an entity moves between the parts of the warehouse, an event is triggered. There exist four main events which are arriving, picking, order consolidation and shipping.

The simulation model's environment interacts with the DRL agent to facilitate learning. It is structured as a semi-Markov decision process (SMDP), comprising transition times, a finite state space, a set of actions, and a reward function. The DRL agent tries to solve instances within each episode, that is comprised of a sequence of states, actions, and rewards, terminating when all orders are fulfilled or when an excessive number of late orders accumulated.

State space is represented with three main components: the number of orders that are not processed yet, available capacity and extra information such as the count of late orders, the count of fulfilled orders, and the present simulation time. Action space is formulated so that there are two actions - single-select and multi-select - for each order category plus the “do nothing” action when there is no available capacity, there are 11 actions totally. Taking an action in case of no order or choosing an order when there is no capacity is considered to be an infeasible action. In such instances, the state remains unchanged, and orders remain unprocessed. If wait action is chosen when there are orders and capacities, it is also an infeasible action and not performed. The action and state formulas are taken from [5].

The reward function similar to the reward function in [1] is shown in Equation 1, a reward proportional to the finished order percentage by the conclusion of the episode, with penalties imposed for late orders and infeasible actions. w is the sum of the count of late orders and count of non-processed orders. N is the total count of orders. Then, 1 – w/N is the ratio of the processed orders before their cut-off time, and the reward exponentially depends on this value. An environment is simulated for the agent to know how the state changes as a result of its actions. The engagement between the agent and the simulation model is shown in Figure 3. After action a is performed, the simulation model simulates the action till state s changes to s’. At the time that orders arrive or available capacity increase or decrease, the state changes. When orders are ready for shipping, tardiness information is received.

$$r(s, a, s') = \begin{cases} -0.5 & \text{if infeasible action} \\ -1.5 & \text{if late order} \\ (1 - w/N)^2 & \text{if episode finishes} \\ 0 & \text{otherwise} \end{cases} \quad \textbf{(1)}$$

Transition time τ is minimal when capacities are available, but it increases significantly when the wait action is selected. The agent needs to learn the actions for each $O_{c_i l j e k}$, and the capacities needed for each of the actions. This is encouraged by giving a penalty for infeasible actions.

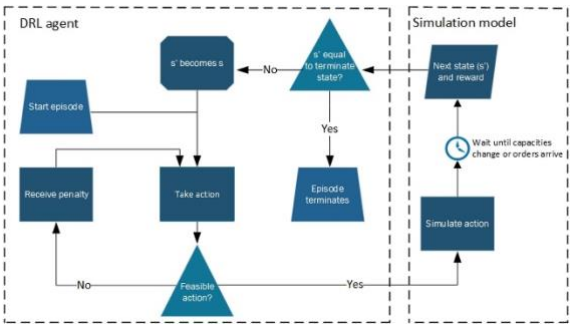

Figure 3. Interaction of the DRL agent with the simulation model [1]

3.2.3. Proximal policy optimization

Proximal Policy Optimization (PPO) introduced by [18] is a cutting-edge RL method designed to optimize policies for decision-making tasks in environments with complex and continuous action spaces. PPO is a member of the policy gradient methods family that directly learn the optimal policy through gradient ascent on the objective function. PPO finds applications in various industries including gaming, robotics, finance, healthcare, and autonomous systems and it has been proved that PPO achieves good stability, sample efficiency and robustness in various environments [19][20].

PPO algorithm is chosen to be applied to OBSP. One of the reasons is that it is an extension of Trust Region Policy Optimization (TRPO) and Actor Critic with Experience Replay (ACER) algorithms which are both extensions of DQN -the first developed DRL algorithm- with the recent improvements. Another reason is reducing computational expenses. Since PPO only updates policies instead of individual states as in DQN, training time is significantly reduced. The third reason is the success of PPO algorithm in coming by a more general strategy so that it can be applied to another warehouse settings later on.

Self-dependence of training data on the policy creates instabilities in the process of training in RL. Besides, parameter tuning is sensitive in a considerable amount. PPO solves these problems of RL.

PPO is an online learning algorithm as opposed to DQN, which means it learns directly from encountered experiences rather than storing and replaying. This method employs a policy gradient approach, necessitating the computation of an estimator for the policy gradient and its utilization within a stochastic gradient ascent algorithm. Gradient estimators has a common form in Equation 2. In neural network implementations, gradient estimator can be obtained by differentiating objective function in Equation 3. The policy π_θ takes states as input and proposes actions as log probabilities. $\hat{A}_t$ is the advantage function which is the comparative value of chosen action for current state. It is the difference between cumulated and discounted sum of rewards G_t and the baseline estimate.

$$\hat{g} = \hat{E}_t\left[\nabla_\theta log\pi_\theta(a_t \vee s_t)\hat{A}_t\right] \quad \textbf{(2)}$$

$$L^{PG}(\theta) = \hat{E}_t\left[log\pi_\theta(a_t \vee s_t)\hat{A}_t\right] \quad \textbf{(3)}$$

G_t is formulated in Equation 4. γ indicates the discount factor which is a value in [0,1], so the importance of future rewards are less than the close ones. $\hat{A}_t$ is calculated after all rewards in the episode is collected. Baseline estimate is the value function in state s_t, which is produced by the neural network and has some variance. $\hat{A}_t$ indicates the comparative value of the chosen action over the expectation of the state.

$$G_t = R_t + \gamma R_{t+1} + .. = \sum_{k=0}^{\infty} \gamma^k R_{t+k} \quad \textbf{(4)}$$

Due to the parameter updates of the neural network extending beyond the range of collected data, advantage function predicts the wrong estimate. This problem is solved in TRPO formulated which is also the basis for PPO. A limit on the policy update magnitude is applied as in Equation 5.

$$\max_\theta \hat{E}_t\left[\frac{\pi_\theta(a_t \vee s_t)}{\pi_{\theta old}(a_t \vee s_t)}\hat{A}_t\right], \text{ subject to } KL_{\pi_{\theta old}}(\pi_\theta) \le \delta \quad \textbf{(5)}$$

However, this additional overhead of KL constraint in optimization process can cause problems in training. This is solved by the clipping operation in the objective function of PPO as formulated in Equation 6. $\frac{\pi_\theta(a_t \vee s_t)}{\pi_{\theta old}(a_t \vee s_t)}$ is denoted by $r_t(\theta)$. Expectation over the minimum of two terms is calculated. First term is the default objective, and the second term is obtained by applying a clipping operation on the first term where ϵ is between 0 and 0.2.

Algorithm 1 PPO for DeepRele

PPO Initialization
Simulation model initialization create interaction between agent and model
for *episode = 1 to T-steps* **do**
  Load dataset set and initialize problem instance
  sample action a_t using current policy π_θ
  if *Action == Feasible* **then**
    Let environment simulate action a_t and receive reward r_t and the next state s_{t+1}
    else
      Receive penalty r_t for infeasible action
      $s_{t+1} = s_t$
    end
  end
  Compute advantage estimates $\hat{A}_t$,...., $\hat{A}_T$
  Optimize L_t, via minibatch gradient descent
  $\pi_\theta = \theta_{\text{old}}$
  if *end of episode* **then**
    break
  end

Figure 4. PPO Algorithm for the proposed warehouse system [1]

$$L^{CLIP}(\theta) = \hat{E}_t\left[min\left(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right] \quad \textbf{(6)}$$

The PPO objective function is created by adding extra terms to $L^{CLIP}(\theta)$ as in Equation 7. ${L_1}^{VF}(\theta)$ updates the baseline network. $S[\pi_\theta](s_t)$ encourages exploration. c1 and c2 weights are hyperparameters to be adjusted.

$$L_t{}^{CLIP+VF+S}(\theta) = \hat{E}_t\left[{L_t}^{CLIP}(\theta) - c_1{L_1}^{VF}(\theta) + c_2 S[\pi_\theta](s_t)\right] \quad \textbf{(7)}$$

PPO training algorithm for the proposed warehouse system is shown in Figure 4.

3.2.4. Heuristic algorithm

Least Slack Time (LST) rule which is a heuristic method that assigns a priority to the orders with least slack times is applied as a benchmark. Slack time is defined as the difference between the amount of time until the due date of the order and the total processing time. When slack time turns negative, single-select is applied. Our algorithm for LST batching and sequencing rule is as follows:

1. Orders are ordered by ascending cut-off times
2. Orders are grouped into batches or single orders according to greedy rule
3. Slack time of each order/batch is calculated
4. If there are batches that have a negative slack time, those are dismantled into single orders
5. Order/batches are ordered by ascending slack time
6. First order/batch in the list is selected to be processed

3.3. Experiments

Order arrival events dataset shared by [21] is utilized in this research. It is a public dataset that includes purchase data from April 2020 to November 2020 from a major online retailer of home appliances and electronics. The dataset includes the information of order arrival times and products in each order, which are interested for OBSP. The other required information for our algorithm, such as the stock location of items and the composition of orders are generated in the scope of this work. The dataset is chosen because it is e-commerce purchase history data which is suitable for the proposed warehouse system. The dataset includes 1.4M orders and 2.6M items in total. It is observed that order arrivals are concentrated on morning hours, see Figure 5 for order arrival distribution.

Some preprocessing steps are done to simplify the problem, such as removing the outliers which are orders consisting of 10 or more items and orders containing more than one from the same SKU. Only hours between 06:00 and 12:00 is taken into consideration since these are the busiest hours.The cut-off times are generated since they do not exist in the original dataset, however cut-off times will be applied to decide the tardiness of the order. Two cut-off time settings with every hour or 15 minutes between 07:00 and 13:00 are examined. Items that are purchased before 12:00 will be delivered same day, so orders before 12:00 are processed same day and have cut-off times before 13:00. A pareto analysis has been done, and the most frequently ordered items are placed in PtG stock whereas the others are placed in GtP.

Scenarios are created from the order arrivals dataset processed according to these assumptions. Each scenario is analysed for different order throughput and resource settings and with settings in Table 1 between 06:00 and 12:00. Orders before 06:00 are not considered. All orders that are not processed between

release and next cut-off moment are considered to be tardy.

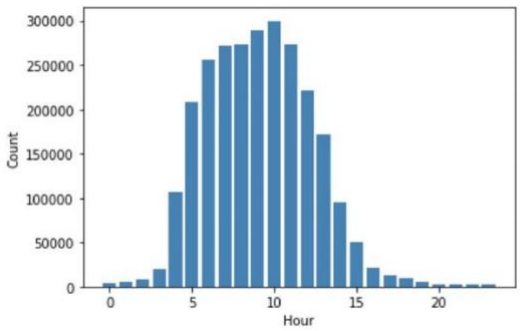

Figure 5. The distribution of order arrivals aggregated over a span of 30 days

Order arrivals have been released to the simulation environment at the start of the hour or once in every 15 minutes depending on the scenario. Then, it is decided which orders will be picked and whether it will be a batch or a single order. The decision is a result of the learned policy of the agent in DRL experiments, and depends on a rule in heuristic experiments. After the items for the order/batch is picked, it follows the designated path in our spesific warehouse environment. The orders are accepted to become tardy when their shipping time exceed the cut-off time specified for each scenario.

Table 1. Different scenarios applied for order arrivals dataset

Scenario	A	B
Length of run	1 hour	1 hour
SKU distribution	PtG - 70% GtP - 30%	PtG - 70% GtP - 30%
Order releasing times	Every hour	Every 15 minutes
Cut-off times	Every hour	Every 15 minutes

Same experiment setup has been applied for both heuristics and DRL algorithm to be able to compare them. Two different order throughput setting has been applied for each scenario, which are 360 and 500 orders per hour. Resources are 9 pickers, 15 shuttles, 1 DtO zone, 1 StO zone and 1 packing zone for all experiments. The number of warehouse resources and order throughput values have been determined by running heuristic experiments several times beforehand, and trying to achieve a low percentage of tardy orders. Four experiment have been done with LST, symbolized with lst360_3600, lst360_900, lst500_3600 and lst500_900. The first number in the model name symbolizes the order throughput and the second number symbolizes the order arrival interval in terms of seconds.

For the DRL algorithm experimentation, a training step is also required. Train and test datasets are obtained by splitting the original dataset randomly, and getting 70% for training, and the remaining 30% for testing. PPO algorithm is utilized for training, the model is trained for 1000000 steps. The step size, the number of steps per update is 1024. The discount factor is 0.9999. MlpPolicy is used as policy model. The clipping parameter is 0.2. Learning rate is 0.0003. The other parameters are all default values. Four models are trained: a360_3600, a360_900, a500_3600 and a500_900. The first number in the model name symbolizes the order throughput and the second number symbolizes the order arrival interval in terms of seconds.

The results are examined and compared in terms of tardy order percentage. Also, robustness and action strategy of the DRL solution are investigated.

All experiments are conducted on the same virtual machine on Microsoft Azure. The virtual machine is Standard D16as v5 size (16 vcpus, 64 GiB memory) and has a Linux (ubuntu 20.04) operating system. For training and testing the PPO model, stable-baselines3 package [22] and PyTorch [23] is used, and Tensorboard is used for monitoring the behaviour of the agent.

IV. RESULTS AND DISCUSSION

4.1. Tardy Order Percentages

To adjust the reward formula, different weights for penalties are tested for 1 hour order arrival interval and for 360 and 500 order throughputs. The effect of different reward formulas in terms of tardy order percentages can be viewed in Table 2. First we had applied smaller penalties, and after some trial and error, we decided on larger penalties. The weights for small penalties are -0.005 for infeasible action and -0.0075 for tardy order which are the weights used in [1], whereas for large penalties they are -0.5 and -1.5 respectively. Larger penalties resulted in more difference for higher order throughputs.

The reward formula is a fundamental component in RL that directly impacts how the agent learns and behaves. Its design requires careful consideration to ensure that it effectively guides the agent towards the intended goals while avoiding pitfalls and promoting efficient learning. The choice between large and small penalties in RL depends on the specific requirements and constraints of the task. By providing a strong deterrent against suboptimal actions, large penalties can help the agent converge to a better policy more quickly, as it more decisively learns which actions to avoid. However, they can cause instability. Small penalties, on the other hand, allow the agent to explore the environment more freely, understanding a broader range of actions and their outcomes without being overly discouraged by mistakes. However, they lead to

a slower convergence. On our spesific problem, smaller weights might not have been sufficient to strongly discourage tardy orders.

Table 2. Tardy Order Percentage for Different Reward Formulas with Standard Deviation Given in Paranthesis

Trained Model	Penalty Weights	Mean (%)
a360_3600	(-0.005, -0.0075)	1.91(1.79)
a360_3600	(-0.5, -1.5)	1.6 (1.74)
a500_3600	(-0.005, -0.0075)	12.15 (10.15)
a500_3600	(-0.5, -1.5)	8.51 (2.24)

After deciding on the reward formula, 4 different models are trained with the PPO algorithm for 360 and 500 order throughputs, and 1 hour and 15 minutes order arrival intervals. The models are tested for 20 episodes. LST rule has been applied on the same settings for same number of episodes and the results are compared for each setting. Table 3 shows the resulting average tardy order percentages and standard deviations.

Table 3. Comparison of Tardy Order Percentages of LST and PPO Algorithms

Experiment	Mean (%)	Standard Deviation(%)
lst360_3600	3.73	2.68
a360_3600	1.91	1.79
lst360_900	0.0	0.0
a360_900	0.0	0.0
lst500_3600	24.83	3.17
a500_3600	8.51	2.24
lst500_900	23.2	12.01
a500_900	8.3	9.1

It is observed that the agent trained with DRL achieves around a one third lower tardy order percentage for all settings when compared with the benchmark of applied heuristic. The higher order throughput values result in higher tardy order percentages as expected. The mean values are slightly lower when an arrival interval of 15 minutes applied, however the standard deviation is higher in that case. We observed that the amount of decrease in tardiness when DRL applied is significant especially when the amount of orders to be processed is higher.

The overall performance of the DRL agent in terms of tardy order percentages is higher than the heuristics, which is compatible with the previous studies [1][5]. It shows that RL can be an effective solution for achieving a lower tardy order percentage in logistics due to its ability to learn optimal strategies through interactions with the environment. It may be a promising method for the tight delivery schedules and order uncertanties frequently encountered in today's logistics solutions.

4.2. Action Strategy

The actions taken by the agents on each episode for different order throughput values and order arrival interval of 1 hour are logged during training and visualized in Tensorboard as shown in Figure 6. The last chart shows do nothing action which is selected when there are no capacity or available orders. Action1, action3, action5, action7 and action10 represent single-select actions whereas the others represent multi-select actions. First 4 actions are for picking single item orders, whereas the others are for multiple item orders. Action1, action2, action5 and action6 is for picking items that are stored in PtG stock. Action3, action4, action7 and action8 is for picking items that are stored in GtP stock. Action9 and action10 is for picking items from both PtG and GtP stocks.

The count of average performed actions per episode is higher in case of a high order throughput almost for all actions. It makes sense because more actions will be required to pick more orders. The results are only different for action1 and action5. Action1 and action5 was for picking items by order from PtG stock area.

Our agent may take a multi-select decision instead of single-select for a high order throughput value for PtG stocks only because total picking time for PtG decreases more than GtP when orders are batched. To explain it with an example; suppose that they will be 3 items in the batch. Then, total picking time will be roughly 3x30+100=190 to pick them from PtG, and roughly 3x15=45 for picking them from GtP. When not batched, the time will be (30+100)x3=390 for PtG stock and will not change for GtP. There is a constant value symbolizing the time for picker to arrive to the location of the item in PtG stock, which explains the decision of the agent.

The distribution of numbers of actions diverses mostly in action2, action6 and action8. Action2 increases for the low order throughput value, whereas it decreases for the high order throughput value. For action6 and action8, the situation is the vice versa. All of them are multi-select actions. The difference between the categories these actions address is that action 2 is for single item orders, whereas action 6 and 8 is for multiple item orders. We can make an inference that our agent improves a strategy to select multi-select actions more for single item orders and less for multiple

Figure 6. The mean number of actions executed per episode for different order throughputs during training for 1 hour arrival interval

item orders when total number of orders is high. It may be because of that batching single item orders requires less time to process orders at areas. For the case of smaller arrival intervals, the same strategy development is not observed.

Generally, the number of do-nothing actions is smaller for a low order throughput value. It may be because of resource deficiency met when high number of orders started to be processed in our warehouse environment.

Developing an action strategy is essential for an RL agent's success. It provides the framework within which the agent learns and makes decisions, ensuring that the agent can efficiently and effectively navigate its environment to achieve the desired goals. By monitoring the actions taken by our RL agent during training, it is observed that it develops an action strategy to achieve a lower number of tardy orders in a capacity constraint environment.

4.3. Robustness Analysis

The agents trained on 360 and 500 order throughputs are also tested on 500 and 360 orders respectively for different order arrival intervals, to observe how will they behave when there is a lower or higher order throughput is experienced than they are trained on.

Resulting tardy order percentages can be viewed on Table 4. It is observed that the agent trained on 500 hourly orders processes all orders in time when it met 360 hourly orders. The other agent which is trained on 360 hourly orders, does not perform as well as the one trained on 500 orders as expected. However, it can still achieve better than the heuristic approach.

Table 4. Tardy Order Percentage for Models Tested on Different Order Throughput Values Than They Trained

Trained Model	Test Order Throughput	Mean (%)
a360_3600	500	18.97 (2.73)
a360_900	500	0.57 (1.55)
a500_3600	360	0.0 (0.0)
a500_900	360	0.0 (0.0)

The behaviour of the RL agent in case of order uncertainty and variability is important since it is a case that frequently occurs in today's warehouses, especially in e-commerce. It is shown that the the proposed RL solution may cope with the order peeks, and exhibit an acceptable performance.

V. CONCLUSION

In this research, the performance of a DRL solution on an OBSP problem with an objective of minimizing the tardy order percentage is investigated. The results are compared to a heuristic solution which is applied on the same warehouse environment and order arrival dataset. The following conclusions are obtained:

1. DRL with an LST applied for sequencing gives better results than LST batching and sequencing on our spesific problem for all of the applied scenarios.
2. The proposed solution is robust to changes in order throughput.
3. It is observed that our agent develops an action strategy that will decrease the time for an order to be prepared.

The DRL model may be improved by including different techniques for optimizing reward function, further investigating different model parameters. There exists studies with the methods based on PPO and proved to have a better sample efficiency [24][25]. Following up recent improvements in RL algorithms and including in our solution may improve the results. For a better simulation of a real life problem, higher order throughputs may be experimented and a more detailed warehouse simulation may be used. Other performance metrics such as maximization of the utilization of resources may be measured to investigate the versatility of the solution.

ACKNOWLEDGEMENT

This research was prepared within the scope of Bahçeşehir University postgraduate thesis study. I would like to express my gratitude to my supervisor Assos. Prof. Ayla Gülcü for her valuable guidance and advice which makes this study possible.

REFERENCES

[1] Cals, B. J. H. C. (2019). The order batching problem: a deep reinforcement learning approach. (Master Thesis, Eindhoven University of Technology, Eindhoven, Holland). Retrieved from https://research.tue.nl/en/studentTheses/the-order-batching-problem

[2] Menéndez, B., Bustillo, M., G. Pardo, E., & Duarte, A. (2017). General Variable Neighborhood Search for the Order Batching and Sequencing Problem. European Journal of Operational Research. 263. 10.1016/j.ejor.2017.05.001.

[3] Xiaowei, J., Zhou, Y., Zhang, Y., Sun, L., & Hu, X. (2018). Order batching and sequencing problem under the pick-and-sort strategy in online supermarkets. Procedia Computer Science. 126. 1985-1993. 10.1016/j.procs.2018.07.254.

[4] Aylak, B. L. (2022). WAREHOUSE LAYOUT OPTIMIZATION USING ASSOCIATION RULES. FRESENIUS ENVIRONMENTAL BULLETIN, 31(3 A), 3828-3840.

[5] Beeks, M. S. (2021). Deep reinforcement learning for solving a multi-objective online order batching problem. (Master Thesis, Eindhoven University of Technology, Eindhoven, Holland). Retrieved from https://research.tue.nl/en/studentTheses/deep-reinforcement-learning-for-solving-a-multi-objective-online-

[6] Boysen, N., De Koster, R.B.M, & Weidinger, F. (2018). Warehousing in the e-commerce era: A survey. European Journal of Operational Research. 277. 10.1016/j.ejor.2018.08.023.

[7] Aylak, B. L., İnce, M., Oral, O., Süer, G., Almasarwah, N., Singh, M., & Salah, B. (2021). Application of machine learning methods for pallet loading problem. Applied Sciences, 11(18), 8304.

[8] Yan, Y., Chow, A.H.F., Ho, C.P., Kuo, Y.H., Wu, Q., & Ying, C. (2021). Reinforcement Learning for Logistics and Supply Chain Management: Methodologies, State of the Art, and Future Opportunities. Retrieved from SSRN: https://ssrn.com/abstract=3935816

[9] Henn, S. & Schmid, V. (2011). Metaheuristics for Order Batching and Sequencing in Manual Order Picking Systems. Computers and Industrial Engineering. 66. 10.1016/j.cie.2013.07.003.

[10] Henn, S. (2012). Order batching and sequencing for the minimization of the total tardiness in picker-to-part warehouses. Flexible Services and Manufacturing Journal. 27. 10.1007/s10696-012-9164-1.

[11] Tsai, C.-Y., Liou, J. J. H., & Huang, T.-M. (2008). Using a multiple-GA method to solve the batch picking problem: considering travel distance and order due time. International Journal of Production Research, 46:22, 6533-6555. DOI: 10.1080/00207540701441947

[12] Valle, C.A., Beasley, J.E., & Cunha, A.S. (2017). Optimally solving the joint order batching and picker routing problem. European Journal of Operational Research. 10.1016/j.ejor.2017.03.069.

[13] Cals, B., Zhang, Y., Dijkman, R. M., & van Dorst, C. (2021). Solving the Online Batching Problem using Deep Reinforcement Learning. Computers & Industrial Engineering, 156, [107221]. https://doi.org/10.1016/j.cie.2021.107221

[14] Hildebrand, M., Frendrup, J., & Sarivan, M. (2019). Batching using reinforcement learning. The 7th Student Symposium on Mechanical and Manufacturing Engineering. Department of Materials and Production, Aalborg University.

[15] Beeks, M., Refaei Afshar, R., Zhang, Y., Dijkman, R., Dorst, C. & Looijer, S. (2022). Deep Reinforcement Learning for a Multi-Objective Online Order Batching Problem. In Proceedings of the International Conference on Automated Planning and Scheduling. 32. 435-443. 10.1609/icaps.v32i1.19829.

[16] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J. & Zaremba, W. (2016).

[17] Sutton, R. S., & Barto, A. G. (2018). Reinforcement Learning: An Introduction. A Bradford Book, Cambridge, MA, USA.

[18] Schulman, J., Wolski, F., Dhariwal, P., Radford, A. & Klimov, O. (2017). Proximal Policy Optimization Algorithms. https://doi.org/10.48550/arXiv.1707.06347

[19] Lopes, G.C., Ferreira, M., da Silva Simões, A., & Colombini, E. L. (2018) Intelligent Control of a Quadrotor with Proximal Policy Optimization Reinforcement Learning. 2018 Latin American Robotic Symposium, 2018 Brazilian Symposium on Robotics (SBR) and 2018 Workshop on Robotics in Education (WRE), João Pessoa, Brazil, 2018, pp. 503-508, doi: 10.1109/LARS/SBR/WRE.2018.00094.

[20] Funika, W., Koperek, P., & Kitowski, J. (2020). Automatic Management of Cloud Applications with Use of Proximal Policy Optimization. In: Krzhizhanovskaya, V., et al. Computational Science – ICCS 2020. ICCS 2020. Lecture Notes in Computer Science, vol 12137. Springer, Cham. https://doi.org/10.1007/978-3-030-50371-0_6
OpenAI Gym. https://doi.org/10.48550/arXiv.1606.01540

[21] Kechinov, M. (2020). eCommerce purchase history from electronics store [Data file]. Retrieved from https://www.kaggle.com/datasets/mkechinov/ecommerce-purchase-history-from-electronics-store

[22] Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., & Dormann, N. (2021). Stable-Baselines3: Reliable Reinforcement Learning Implementations. Journal of Machine Learning Research 22 (2021) 1-8

[23] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... & Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems, 32.

[24] Wang, Y., He, H. & Tan, X. (2020). Truly Proximal Policy Optimization. Proceedings of the 35th Uncertainty in Artificial Intelligence Conference, in Proceedings of Machine Learning Research 115:113-122 Available from https://proceedings.mlr.press/v115/wang20b.html.

[25] Cobbe, K. W., Hilton, J., Klimov, O., & Schulman, J. (2021). Phasic policy gradient. In International Conference on Machine Learning (pp. 2020-2027). PMLR.