Proceedings of the 2023 Winter Simulation Conference
C. G. Corlu, S. R. Hunter, H. Lam, B. S. Onggo, J. Shortle, and B. Biller, eds.

SIMULATION ANALYSIS OF A REINFORCEMENT-LEARNING-BASED WAREHOUSE DISPATCHING METHOD CONSIDERING DUE DATE AND TRAVEL DISTANCE

Sriparvathi Shaji Bhattathiri
Michael E. Kuhl

Industrial and Systems Engineering Department
Rochester Institute of Technology
81 Lomb Memorial Drive
Rochester, NY 14623, USA

Ankita Tondwalkar
Andres Kwasinski

Computer Engineering Department
Rochester Institute of Technology
81 Lomb Memorial Drive
Rochester, NY 14623, USA

ABSTRACT

As the adoption of autonomous mobile robots in warehouses and other industrial environments continues to increase, there is a need for methods that can effectively dispatch robots to meet system demand. Real-time dispatching of autonomous mobile robots can be very complex, but simple rule-based methods are typically used for this task. In this paper, a reinforcement-learning-based dispatching method for intralogistics (RLDI) is proposed. RLDI is warehouse layout independent and takes into consideration task due dates and the travel distance. The algorithm is trained and tested in a simulation environment that represents a small warehouse. Monte Carlo simulation analysis is used to explore the capabilities and limitations of the established RLDI. The performance of the method is compared to the shortest distance dispatching rule in single and multi-agent environments under various levels of due date tightness. Experimental results demonstrate the potential for using reinforcement learning methods for warehouse dispatching.

1 INTRODUCTION

The use of automated material handling in warehouses has increased dramatically in recent years. In fact, in 2021 the sales of automated guided vehicles (AGVs) and autonomous mobile robots (AMRs) increased by 36% from 2020 (MHI 2022). In warehouses, AGVs and AMRs are mainly used for material handling (intralogistics). For example, in a warehouse that stores finished and packaged goods this includes storing packages that arrive at loading docks in trucks as well as retrieving packages that need to be sent to customers. The dock area may have limited capacity and it is imperative to put packages away before more trucks arrive and there is no space to unload them. Further, the packages that need to be sent to customers have a delivery date associated with them that can be met only if the packages are retrieved and loaded into trucks at the dock in a timely fashion. This necessitates meeting due dates for pick-up and drop-off tasks performed by AGVs and AMRs.

The assignment of intralogistics tasks can be seen as a dispatching problem due to the dynamic nature of warehouse operations and the lack of advanced notification/pre-arrival information (Le-Anh and De Koster 2006). If pre-arrival information is known scheduling methods could be used to obtain the optimal task assignments for vehicles over a time horizon. However, without this pre-arrival information, traditional scheduling algorithms are not designed to deal with the stochastic nature of task demand. Thus, in intralogistics, dispatching rules are often employed. Some of the common techniques used include first come–first serve, shortest distance, highest priority or first encounter–first serve looping technique (Le-Anh and De Koster 2006).

In the decision-making pertaining dispatching of vehicles for pick-up and drop-off tasks, there are two factors that are of prime importance: travel distance and on-time delivery. Simple dispatching rules such

as shortest distance or early due date, focus on one of these factors at the cost of the other. To address the issue of efficient, on-time delivery, we propose a novel method called reinforcement-learning-based dispatching for intralogistics (RLDI). RLDI is a layout independent reinforcement learning approach for dynamic dispatching in warehouses that takes into consideration both travel distance and due date. A scalable stochastic simulation environment is created to represent a typical warehouse with storage and retrieval functions. The simulation environment is used to generate data to train and test the reinforcement learning algorithm. In addition, the simulation environment is used to conduct an experiment that explores the capabilities and limitations of the RLDI algorithm in comparison to other common dispatching rules.

The remainder of the paper is as follows. In Section 2, we discuss the relevant related work in dispatching and artificial intelligence methods. In Section 3, we provide discuss the background of the reinforcement learning methodology on which we develop the RLDI algorithm. The RLDI methodology is presented in Section 4. An experimental performance evaluation is presented in Section 5. Finally, in Section 6, we draw conclusions and discuss future work.

2 RELATED WORK ON REINFORCEMENT LEARNING FOR DISPATCHING

The literature abounds with the use of reinforcement learning in robotics. Many of the studies are based on simulation environments. Reinforcement learning has been successfully used in path planning (Rasheed et al. 2022; Wesselhöft et al. 2022). It has also been used in other decision-making problems with great success like in the dispatch of manufacturing tasks, assignment of tasks to vehicles in urban environments, and ride-sharing for taxis, to name a few (Stricker et al. 2018; Qin et al. 2021; Castagna and Dusparic 2022). In this section, we discuss the use of reinforcement learning in dispatch, real-time scheduling, and task allocation to AMRs.

2.1 Reinforcement Learning Approaches for Dispatching and Task Allocation

Real-time scheduling is often done based on a rule like shortest distance, first come first serve, highest priority etc. which takes into consideration a single attribute. These methods are heuristic in nature and have it's advantages and disadvantages. For example, the shortest-travel-distance-based systems will lead to shorter empty travel but may lead to long wait times for pick-ups from some areas. In extreme cases, packages at the aisle furthest away from the starting point/docks may never get picked. Thus, overall efficiency of travel will be high but some high-priority packages may not be delivered on time (Klei and Kim 1996; Hu et al. 2020). On the other hand, early due-date-based delivery systems will help in meeting more due-dates but the overall efficiency of travel is low. The cost associated with travel goes up (Hu et al. 2020). This has lead to an interest in multi-criteria heuristic methods that can perform better than single attribute rules on average (Klei and Kim 1996). With the advent of machine learning, traditional artificial neural networks have been employed to solve the problem of dispatch by using it to select the appropriate rule based on training data that encompasses previous experiences using supervised learning (Chen et al. 2011). Traditional artificial neural networks have also been used to assign priority to tasks and then dispatch is done based on the priority and availability of space in the drop-off location(Jeong and Randhawa 2001) .

The main drawback of the use of traditional neural networks is that, the model has a static knowledge base and often does not capture the dependence of the overall performance of the system on a series of decisions (Kulkarni 2012). To overcome this problem reinforcement learning (RL) can be used which utilizes a trail and error method to learn the effect of a decision on the overall performance of an agent in a given environment(Kulkarni 2012).

RL algorithms like Q-table-based RL stores all the states and actions as well as Q values associated with them to make decisions in the future. Such RL algorithms have been used to solve the dynamic scheduling problem (Chen et al. 2015). The performance of such methods reduces with increase in complexity of the environment while DQNs perform very well in such situations (Watkins and Dayan 1992).

DQN reinforcement learning has hence been used to solve real-time dispatch where a central task allocation system is trained to choose one of the dispatch rules, either First Come First Serve (FCFS), shortest travel distance, earliest due date, longest waiting time, or nearest load point task and an AGV. This approach has been shown to be better than using any of the rules by themselves (Hu et al. 2020). The observations used to make the decision are to and from locations, time remaining to complete the current task, and estimated distance for the task being allotted. While there is a negative reward associated with lateness, the due date itself is not part of the observation space (Hu et al. 2020).

Another approach has been to train a reinforcement learning algorithm to help the AMR to choose which package to pick up or drop off based on the current location of all the vehicles. Such an algorithm aims to reduce the empty travel distance and wait time. This does not take into consideration the due date but still outperforms shortest-distance-based dispatch and longest-queue-based dispatch. When there are multiple packages to be picked up and dropped off, some packages would be assigned to other vehicles or maybe already taken care of. Trying to deliver these packages again would be impossible. To avoid taking illegal actions masks can be used (Wei et al. 2022). This enabled the reinforcement learning algorithm to converge faster. This is a common practice when there are multiple actions and there are some actions that cannot be taken (Williams et al. 2017; Ye et al. 2020). While DQN-based reinforcement learning can be used for solving the problem, replacing the traditional neural network with an attention neural network can give better results. (Wei et al. 2022). Similarly, the task allocation problem based on pick-up and drop-off location as well as distance has been solved using other policy-based algorithms like PPO (Agrawal et al. 2023), the due date is however not a part of the observation space.

Yet another approach used for the dispatch of autonomous mobile robots in simulated warehouses is with the help of a system where each AMR bids for a task based on the current observation and the AMR with the highest bid is awarded the job. Once the AMR performs a task, the task is removed from its list of tasks and the learning environment evaluates the actions, where a policy-based algorithm, the actor-critic model, is used for learning. Despite of taking the due date into consideration, this model is layout dependent(Malus et al. 2020).

Further reinforcement learning has proven to be useful in dispatch of vehicles for delivery of packages to customers using drones (Chen et al. 2022), vehicle dispatch for ride-sharing (Guo and Xu 2020; Al-Abbasi et al. 2019; Singh et al. 2021) and even for order dispatch and driver repositioning based on demand (Holler et al. 2019; Lin et al. 2018; Zheng et al. 2022).

Hence, it can be seen from the literature survey that a reinforcement learning algorithm is a promising tool to solve the problem of real-time dispatch and there have been efforts to reduce travel distance and take into consideration the current position of the AMR for the same. In the next section, we present the methodological background for our reinforcement-learning-based algorithm including the accepted definition and terminologies and a summary deep reinforcement learning methods.

3 METHODOLOGICAL BACKGROUND ON REINFORCEMENT LEARNING

Reinforcement learning is a type of machine learning in which the agent autonomously interacts and learns from its surrounding environment. The main elements of an RL system are (Szepesvari 2010):

1. State space $\mathscr{S} = \{S_0, S_1, \ldots, S_i\}$: The different conditions that the environment could be in can be be represented as states. The state space is the set of all possible states $s_t \in \mathscr{S}$ that the environment can be in. The transition from one state to another is influenced by the action an agent takes in the current state.
2. Action space $\mathscr{A} = \{a_0, a_1, \ldots, a_j\}$: The RL is a trial and error of taking actions that steadily facilitate the learning of what actions yield the best results (rewards) corresponding to each state.
3. Reward r_t: A reward defines the goal in the RL system. At time step t, an agent learning with RL executes an action $a_t \in \mathscr{A}$ and, either gets rewarded or penalized depending on the outcome of the action thereby receiving a reward r_{t+1} associated with the result of its action on the environment

and observing the next state s_{t+1} to which the environment has transitioned to. The goal of an agent is to maximize the total reward over the long run such that,

$$r_t = \mathscr{R}(s_t, a_t, s_{t+1}). \tag{1}$$

4. Policy π: Policy is mapping of states of the environment to the actions to be taken in the current state. In other words, a policy represents the agent's way of behaving at a given time.The policy can be either deterministic or stochastic.The deterministic and stochastic policy is denoted by μ and π respectively as;

$$a_t = \mu(s_t). \tag{2}$$

$$a_t \sim \pi(.|s_t). \tag{3}$$

The RL algorithms are designed to find the best policy π^* (in this case, a stochastic one) that maximizes the cumulative expectation of the discounted rewards over time denoted by the state function,

$$V_\pi(s) = \sum_{t=1}^{\infty} \gamma^t E[R_t|\pi, (s_0 = s)]. \tag{4}$$

The RL system is a model of the environment, wherein the agent eventually learns to predict the response of the environment to the actions it takes. Given a state and action, the RL model estimates the next state and the reward. There are two approaches to modeling the environment; model-based and model-free. The model-based methods primarily depend on planning whereas model-free methods depend on learning. The agent builds the model while autonomously learning from its interactions with the environment. In model-based learning, the agent will always choose an action that will yield the maximum reward regardless of what the action may cause. In case of the model-free approach, the agent will take an action multiple times for an optimal reward based on the outcome.The need for autonomous learning from the environment calls for the need of a model-free approach.

In particular, our approach will build on the RL method call a Deep Q-Network (DQN) which uses a learning mechanism known as Q-learning. Table-based Q learning (Watkins and Dayan 1992) estimates a Q values through the time difference equation

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[R_{t+1} + \gamma \max_{a'} Q(s'_{t+1}, a') - Q(s_t, a_t)] \tag{5}$$

where s'_{t+1} is the next state, and a' is the action to be taken in the next state, $\alpha \in [0,1]$ is the learning rate, and $\gamma \in [0,1]$ is the discount factor. The table-based Q-learning is not suitable for large action and state spaces since it would take indefinite learning time to explore each state to create the required Q-table. An alternative to the table-based approach is to estimate the $Q(s_t, a_t)$ through artificial neural network (ANN) called deep-Q network (DQN) with parameter vector θ such that $Q(s,a;\theta) \approx Q(s,a)$. Given our set up we use the model free DQN learning (Mnih et al. 2015).

The DQN approximates a state-value function such that $Q(s,a;\theta) \approx Q(s,a)$. In the DQN shown in Figure 1 each agent utilizes one neural network as action-value function approximator $Q_i(x,a;\theta_i)$ and another as target action-value function approximator $\hat{Q}_i(x,a;\theta_i^-)$, where θ_i and θ_i^- denote the parameters for each of the neural networks. At each learning step, the parameters θ_i of each agent's action-value function are updated from a mini-batch of training samples each consisting of tuples <*current state, next state, action taken, target Q-value*> using a gradient descent back propagation algorithm with the error function,

$$L(\theta_i) = \frac{1}{2}[r_i(x,a) + \gamma \max_{\hat{a} \in A} \hat{Q}_i(\hat{x}, \hat{a}; \theta_i^-) - Q_i(x,a;\theta_i)]^2 \tag{6}$$

where $r_i(x,a)$ is the observed reward after taking action a while in state x. Only every c learning steps, the parameters θ_i^- of the target action-value function are updated by copying the updated parameters θ_i of the action-value function. In RL, there is always a need to balance between the exploration and exploitation.The Epsilon greedy policy chooses action with the highest estimated value with probability $1-\varepsilon$ and selects a random action with a probability ε where ε is hyper parameter between 0 and 1 such that

$$a_t = \begin{cases} maxQ_t(a), & \text{if } 1-\varepsilon, \\ \text{any action (a) ,} & \varepsilon. \end{cases} \tag{7}$$

The Boltzmann (Softmax) policy on the other hand chooses actions depending on their estimated values with a temperature constant τ that controls the randomness of the policy. A lower value of τ indicates the policy wanting to exploit more, whereas a higher τ results in a more exploratory policy so that

$$P_t(a) = \frac{exp(\frac{Q_t(a)}{\tau})}{\sum_{i=1}^{n} exp(\frac{Q_t(i)}{\tau})}, \tag{8}$$

where $Q_t(a)$ is the estimated value of action. In Equation (8), the denominator of the exponential function normalizes the probabilities such that the sum of all is 1, which is representative of the fact that the probability distribution over all actions is a valid distribution. The numerator determines the relative probability of choosing an action a compared to other actions, with a higher $Q_t(a)$ resulting in higher probabilities.

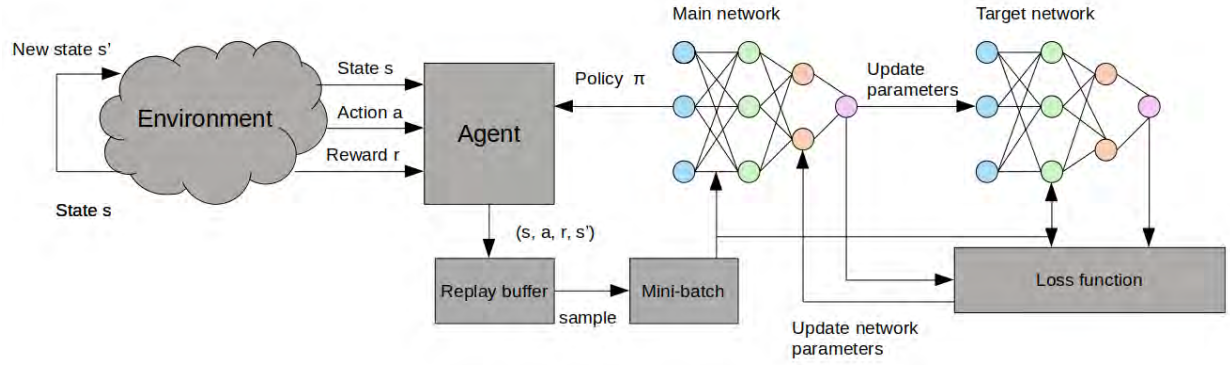


Figure 1: DQN framework.

4 RLDI METHODOLOGY

In this section, we describe the methodology for the RLDI algorithm for vehicle dispatching in a warehouse environment. The RLDI approach adapts the general reinforcement learning methodology described in Section 3 to task assignment and path planning in the warehouse environment.

4.1 Warehouse System Description and Logical Representation

The RLDI method requires a representation of a warehouse layout environment as an input. The warehouse layout consists of the various warehouse components including racks, walls, dock, staging areas, aisles, etc. The simulated warehouse environment is used to train and test the dispatching policy for vehicles to perform intralogistics tasks such as storage and retrieval. All of the static components in the warehouse are taken into consideration and logically represented using an occupancy grid. Racks and other forbidden

areas where the AMR cannot travel to in the warehouse are represented as 1 in the occupancy grid. The free path is represented by 0. Figure 2 exhibits an example of the graphical representation of the warehouse and the translation to the logical occupancy grid.

4.2 RLDI Reinforcement Learning Protocol

Given the logical occupancy grid representation of the warehouse environment, the RLDI protocol specifies the various aspects of the reinforcement learning methodology. In addition to the warehouse layout, the set of warehouse vehicles and their current location are given along with a set of pick-up or drop-off tasks that need to be completed. As a vehicle in the warehouse becomes available, the RLDI algorithm is used to assign the vehicle a task from the set of tasks and then dispatch the vehicle. This process is repeated until all of the tasks have been completed. For RLDI the observation space, action space and reward function of the reinforcement learning algorithm are defined as follows:

1. The observation/state space $\mathscr{S}$is defined as a dictionary with three keys:

$$s_t = Dict\{Duedate, Distance, Package\}, \tag{9}$$

   Each of the key represents a specific aspect of the environment state that the agent (warehouse vehicle) observes. The observation space allows the agent to observe the due date, distance to be traveled to complete the pick-up/drop-off task for the package as well as whether the package is yet to be delivered or already delivered. The agent uses this information at hand to learn about the environment and make decisions to select the appropriate action. In particular, the keys are defined as
   (a) *Due date*: Represents the number of time units (from the current time) until the package is due. This key is modeled as a multi-discrete object with an array dimension equal to maximum number of tasks. The due date
   (b) *Distance*: Represents the distance of the location of a task from the agent's current location. This key is modeled as a multi-discrete object with an array size equal to the maximum number of tasks. The distance is determined using the A^* (A-star) algorithm (Hart et al. 1968) which determines the minimum cost path between two points.
   (c) *Package*: Represents whether a task has is available or whether is has already been assigned to another agent. This key is modeled as a multi-binary object with an array size of maximum number of tasks, where 1 indicates that the task is available and 0 indicates that the task has already been assigned.

   The distances and the due dates are normalized between 0 and 100. So, the learned policy is based on the observation of normalized due date and distance, hence making it independent of the layout of the warehouse. This is advantageous in that it enables the model to be extended to use with other warehouse layouts without having to re-train the model.
2. Action space $\mathscr{A} = \{a_0, a_1, \ldots, a_j\}$: The RL is a trial and error of taking actions that steadily facilitate the learning of what actions yield the best results (rewards) corresponding to each state.The action space is defined as the set of $\mathscr{A} = \{a_1, a_2, a_3\}$ of possible actions where a_1, a_2, a_3 represent respectively. The actions refer to delivering one of the active packages.
   (a) *Action Mask:* Action masks are extra information available to the agent during action selection to ensure that only possible actions are chosen. The Q values obtained from the DQN network are multiplied by the action mask and the masked Q values are used for action selection.

3. Reward r_t: The reward function represents the goal of the agent to minimize the lateness in delivery of the package and the travel time and is as follows:

$$\begin{aligned} \mathscr{R}_{active} &= (state[package][action] - 1000) * 1000000, \\ \mathscr{R}_{distance} &= (state[distance][action] * max_{dist}/1000), \\ \mathscr{R}_{duedate} &= 1000 * (deadline - abs(deadline)), \\ \mathscr{R}_{packagenumber} &= ((package_{max} - trips) - abs(package_{max} - trips)) * 1000, \end{aligned} \tag{10}$$

and

$$\mathscr{R} = r_{active} + r_{distance} + r_{duedate} + r_{packagenumber}. \tag{11}$$

The functions from Equations (9-11), are substituted in the corresponding reinforcement learning algorithm functions in Equations (1-8). The RLDI method is trained and tested using the process described in the next section.

4.3 Generation of Data and RLDI Training

The decision on which package to deliver next depends on the distance at which each package is stored as well as the deadline to be met. The deadline takes into account the number of vehicles, the travel time (based on travel distance), the time taken to service the package at the dock, and a tightness factor theta (Ullrich 2013; Schubert et al. 2019). The pick-up location and drop-off locations are randomly chosen using Algorithm 1.

Algorithm 1 Data Generation

1: r =$randint(0,1)$
2: if r ==0 do
Pick up task is generated
Pickup location = $randint(1,49)$
Drop off location is the dock (0)
end if
3: else do
Pick up location is the dock (0)
Drop off location =$randint(1,49)$
end else
4: A^* to estimate travel distances and time, where $v = 1ft/sec$
5: Due dates are generated for each pair of pick up and drop off locations as per :
$D_n = p_n + \tau + \Delta$
D_n: due date of the nth package generated
p_n : estimated processing time to complete the package delivery where
$p_n = \sum_{n=1}^{N} \frac{t_{0,n}}{N} + t_{0,n} + s_0 + s_n$
$t_{0,n}$: travel time between the dock and the pick-up/drop-off location of the n^{th} package
s_0 : service time at the dock i.e. the time to load/unload the package (20s)
s_n : service time at the pickup/ drop-off location to load/unload the package (20s)
τ : $\{0,\ldots,\lfloor\theta(\max_{n=1,\ldots,n}\ p_n)(N/K)\rfloor\}$ where N is number of packages, K is number of AMRs
Δ : $\{0,\ldots,\lfloor\theta(\max_{n=1,\ldots,n}\ p_n)\rfloor\}$

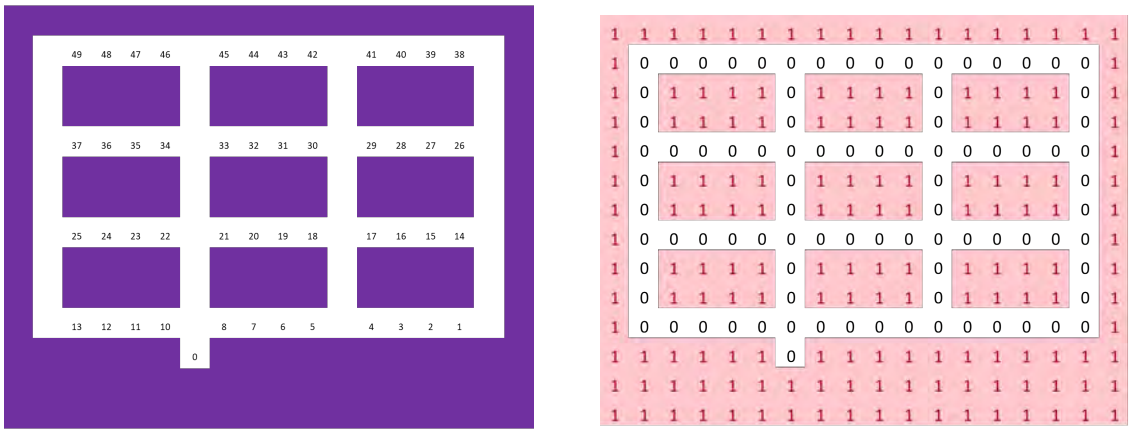
While training the model, the data is generated when the environment is reset for each episode. The rewards are then given to the agent based on the action. The relationship between the state and the action is learned. Different deep neural network architectures are explored for DQN. Parameter and hyper-parameter

tuning is done. Further, the different parameters of reinforcement learning like the exploration rate, discount factor, etc. are explored for faster learning convergence.

5 EXPERIMENTATION AND RESULTS

The experiments are conducted to analyze the potential to use reinforcement learning for the real-time dispatch of packages by taking into consideration the travel distance as well as the due date.

For this study, a small simulated warehouse that is 90 X 70 ft. (6300 sq. ft.) is taken into consideration. The warehouse consists of 49 pick-up and drop-off locations from which the shelves on the two sides can be accessed. Figure 2 is a graphical representation of the small warehouse with numbers 1 to 49 representing the locations to which the AMRs can drive, to pick up items from the shelves and drop off items to be put away on the shelves. 0 represents the dock. The dock has packages arriving in trucks that must be put away on the shelves for storage until it is retrieved to be delivered to customers. The packages that have been retrieved from shelves based on orders are brought out to the dock to be loaded into trucks. In the simulation, it is assumed that the AMR travels at a speed of 5 ft/s.



(a) Graphical warehouse representation.

(b) Logical warehouse representation.

Figure 2: Graphical and logical representation of the simulated warehouse.

The dispatch decision for allocating pick-up and drop-off tasks to AMR is learned RLDI method. For a given instance the maximum number of packages to be delivered is 6. RLDI learns the decision of where to dispatch the AMR by taking into consideration the due date and the travel distance. The training of the model is time-consuming but the decisions made are instantaneous. The training of a single agent for 10,000 epochs takes about 3.5 hours. To train the multi-agent case, it takes 10 hours for 10,000 epochs. For obtaining the preliminary results, various hyper-parameters of the deep neural network used were tuned, the architecture of the model was tuned and different policies were used. The best parameters are reported. Due to the time taken for training, the single agent is trained only for 1,000,000 epochs and the multi-agent for 20,000 epochs. This is due to a computational constraint. The best parameters and hyper-parameters for faster convergence are as seen in Table 1.

The policy learned by RLDI is tested on $1,000$ randomly generated instances where there are 6 packages to be delivered. The due-date tightness parameter theta is adjusted between 0.5 to 1.25 to go from very tight to less tight. First, the policy learned for a single agent is tested. The same combination of packages is delivered based on the shortest distance rule that decides to go to the closest pick-up location next and the shortest overall trip that decides to complete the task with the overall shortest distance. The results obtained for the testing of a single agent in comparison to the rule-based system are shown in Table 2. From the table, it can be seen that the single agent RLDI algorithm performs better than the rule-based systems until the due-dates are very tight. Then, the performances of the algorithms are similar.

Table 1: DQN Parameters and hyper-parameters.

Parameter/Hyper-parameter	Value
Epsilon ε	$\in [0.1, 1]$
Discount factor γ	0.90
Learning rate α	3×10^{-7}
Optimizer	Adam
Number of layers	4

Table 2: Results obtained for testing of the single agent RLDI algorithm.

Decision Making Algorithm	Metric	Due Date Tightness			
		1.50	1.00	0.75	0.50
Shortest Distance	Mean Cumulative Time	322.92	322.92	322.92	322.87
	(Std. Dev.)	(13.54)	(13.54)	(13.54)	(13.82)
	Mean Empty Travel Time	24.82	24.82	24.82	24.78
	(Std. Dev.)	(8.78)	(8.78)	(8.78)	(8.73)
	Mean Percent On-Time	79.78	68.86	60.58	47.10
	(Std. Dev.)	(12.93)	(14.30)	(14.08)	(13.66)
	Mean Lateness	105.54	169.98	227.63	308.39
	(Std. Dev.)	(82.41)	(91.94)	(92.69)	(93.29)
Shortest Complete Trip	Mean Cumulative Time	322.44	322.44	322.44	322.93
	(Std. Dev.)	(14.45)	(14.45)	(14.45)	(15.18)
	Mean Empty Travel Time	24.35	24.35	24.35	24.84
	(Std. Dev.)	(9.76)	(9.76)	(9.76)	(10.14)
	Mean Percent On-Time	75.98	63.65	54.70	44.28
	(Std. Dev.)	(13.48)	(14.40)	(13.38)	(12.26)
	Mean Lateness	124.27	200.90	266.99	350.82
	(Std. Dev.)	(88.51)	(97.23)	(95.90)	(91.6)
RLDI Single Agent	Mean Cumulative Time	332.72	331.21	330.81	329.72
	(Std. Dev.)	(16.28)	(16.21)	(16.44)	(15.99)
	Mean Empty Travel Time	92.72	33.11	32.71	31.63
	(Std. Dev.)	(16.28)	(10.57)	(10.97)	(10.21)
	Mean Percent On-Time	86.15	75.53	64.13	45.90
	(Std. Dev.)	(12.30)	(14.93)	(16.41)	(13.86)
	Mean Lateness	47.25	102.89	176.62	304.06
	(Std. Dev.)	(58.22)	(84.21)	(97.59)	(92.80)

After training the RLDI model for multiple agents, the performance of the multi-agent RLDI model is tested in the warehouse simulation and compared with the multi-agent dispatch based on the shortest distance rule for the same set of packages. The results obtained for the testing of the multi-agent case in comparison to rule-based system are shown in Table 3. With the limited number of training epochs, the trained multi-agent has a performance similar to that of the shortest distance.

When the same trained single agent policy is tested on a bigger map with a different layout as seen in Figure 3, similar performance in terms of on-time trips is observed (Table 4). This shows the layout independent nature of the learned policy.

Table 3: Results obtained for testing of the multi-agent RLDI algorithm.

Decision Making Algorithm	Cumulative Time Mean Seconds (Std. Dev.)	Empty Travel Time Mean Seconds (Std. Dev.)	On-Time Completion Mean Percentage (Std. Dev.)
Shortest Distance	170.00 (10.04)	28.25 (12.16)	68% (0.14)
Shortest Complete Trip	175.95 (12.26)	31.37 (8.51)	65% (0.16)
RLDI Multi-agent	178.26 (13.34)	40.72 (11.88)	68% (0.13)

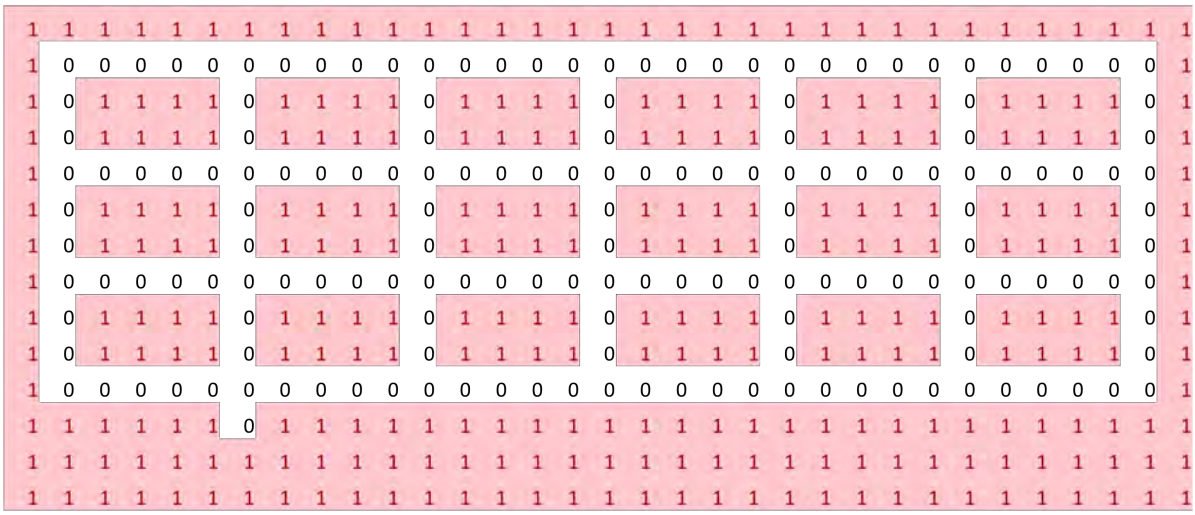


Figure 3: Alternate warehouse layout.

Table 4: Mean (std. dev.) percentage of on-time deliveries when learned policy is tested on a new layout for various levels of due date tightness.

Due Date Tightness	1.25	1.00	0.75	0.50
Shortest Distance	79.2% (0.14)	67.8% (0.15)	59.0% (0.12)	45.2% (0.14)
Shortest Complete Trip	74.8% (0.15)	62.8% (0.15)	54.5% (0.14)	42.5% (0.13)
RLDI Single Agent	86.1% (0.13)	73.3% (0.17)	60.6% (0.17)	44.3% (0.15)

6 CONCLUSIONS AND FUTURE WORK

The experimental results show that the RLDI method is an effective approach for dispatching of multiple AMRs in the intralogistics of warehouses. The observations that influence the decision-making ability of each agent are the warehouse layout, the distance to be traveled to complete the active tasks, and the due dates associated with each task. In our work, we normalize these observations with respect to the maximum distance and maximum due date respectively, hence once trained, the same model could be used to make decisions in a warehouse with a different configuration and size than that used in the simulation. This generalization capability of the learned policy makes it very appealing. The training and testing of the models are in their preliminary stages. There is a need to train the multi-agent algorithm for longer to enable it to gain enough experience to perform better.Additionally, policy-based learning can be further improved further by tuning the hyper-parameters like learning rate and discount factor. Further, the number of packages that are in the system at a time is rather small, and increasing this number would provide better insight into the performance of the model. Policy-based learning can also be improved further by using more sophisticated models like PPO and SAC models. The results obtained from preliminary training and testing show that reinforcement learning can perform better or almost as good as the shortest distance for a finite learning time making it a promising candidate for real-time dispatch in warehouses.

REFERENCES

Agrawal, A., A. S. Bedi, and D. Manocha. 2023. "RTAW: An Attention Inspired Reinforcement Learning Method for Multi-Robot Task Allocation in Warehouse Environments". *arXiv preprint arXiv:2209.05738*.

Al-Abbasi, A. O., A. Ghosh, and V. Aggarwal. 2019. "Deeppool: Distributed Model-free Algorithm for Ride-sharing using Deep Reinforcement Learning". *IEEE Transactions on Intelligent Transportation Systems* 20(12):4714–4727.

Castagna, A., and I. Dusparic. 2022. "Multi-Agent Transfer Learning in Reinforcement Learning-based Ride-Sharing Systems". In *International Conference on Agents and Artificial Intelligence*, 120–130. Setúbal, Portugal: Science and Technology Publications, Lda.

Chen, C., L.-f. Xi, B.-h. Zhou, and S.-s. Zhou. 2011. "A Multiple-criteria Real-time Scheduling Approach for Multiple-load Carriers Subject to LIFO Loading Constraints". *International Journal of Production Research* 49(16):4787–4806.

Chen, C., B. Xia, B.-h. Zhou, and L. Xi. 2015. "A Reinforcement Learning Based Approach for a Multiple-load Carrier Scheduling Problem". *Journal of Intelligent Manufacturing* 26:1233–1245.

Chen, X., M. W. Ulmer, and B. W. Thomas. 2022. "Deep Q-learning for Same-day Delivery with Vehicles and Drones". *European Journal of Operational Research* 298(3):939–952.

Guo, G., and Y. Xu. 2020. "A Deep Reinforcement Learning Approach to Ride-sharing Vehicle Dispatching in Autonomous Mobility-on-demand Systems". *IEEE Intelligent Transportation Systems Magazine* 14(1):128–140.

Hart, P. E., N. J. Nilsson, and B. Raphael. 1968. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". *IEEE Transactions on Systems Science and Cybernetic* 4(2):100–107.

Holler, J., R. Vuorio, Z. Qin, X. Tang, Y. Jiao, T. Jin, S. Singh, C. Wang, and J. Ye. 2019. "Deep Reinforcement Learning for Multi-driver Vehicle Dispatching and Repositioning Problem". In *2019 IEEE International Conference on Data Mining (ICDM)*, 1090–1095. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Hu, H., X. Jia, Q. He, S. Fu, and K. Liu. 2020. "Deep Reinforcement Learning Based AGVs Real-time Scheduling with Mixed Rule for Flexible Shop Floor in Industry 4.0". *Computers & Industrial Engineering* 149:106749.

Jeong, B. H., and S. U. Randhawa. 2001. "A Multi-Attribute Dispatching Rule for Automated Guided Vehicle Systems". *International Journal of Production Research* 39(13):2817–2832.

Klei, C., and J. Kim. 1996. "AGV Dispatching". *International Journal of Production Research* 34(1):95–110.

Kulkarni, P. 2012. *Reinforcement and Systemic Machine Learning for Decision Making*, Volume 1. Hoboken, NJ: John Wiley & Sons.

Le-Anh, T., and M. De Koster. 2006. "A Review of Design and Control of Automated Guided Vehicle Systems". *European Journal of Operational Research* 171(1):1–23.

Lin, K., R. Zhao, Z. Xu, and J. Zhou. 2018. "Efficient Large-scale Fleet Management via Multi-agent Deep Reinforcement Learning". In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1774–1783. New York, New York: Association for Computing Machinery.

Malus, A., D. Kozjek, and R. Vrabič. 2020. "Real-time Order Dispatching for a Fleet of Autonomous Mobile Robots Using Multi-agent Reinforcement Learning". *CIRP annals* 69(1):397–400.

MHI 2022. "Industrial Robots–Trends by Industry". *MHI Solutions* 10(3):28–29.

Mnih, V., K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. 2015. "Human-level Control through Deep Reinforcement Learning". *Nature* 518:529–533.

Qin, W., Y.-N. Sun, Z.-L. Zhuang, Z.-Y. Lu, and Y.-M. Zhou. 2021. "Multi-agent Reinforcement Learning-based Dynamic Task Assignment for Vehicles in Urban Transportation System". *International Journal of Production Economics* 240:108251.

Rasheed, A. A. A., M. N. Abdullah, and A. S. Al-Araji. 2022. "A Review of Multi-agent Mobile Robot Systems Applications.". *International Journal of Electrical and Computer Engineering* 12(4):3517–3529.

Schubert, D., A. Scholz, and G. Wäscher. 2019. "Integrated Order Picking and Vehicle Routing with Due Dates". *OR Spectrum* 40:1109–1139.

Singh, A., A. O. Al-Abbasi, and V. Aggarwal. 2021. "A Distributed Model-free Algorithm for Multi-hop Ride-sharing using Deep Reinforcement Learning". *IEEE Transactions on Intelligent Transportation Systems* 23(7):8595–8605.

Stricker, N., A. Kuhnle, R. Sturm, and S. Friess. 2018. "Reinforcement Learning for Adaptive Order Dispatching in the Semiconductor Industry". *CIRP Annals* 67(1):511–514.

Szepesvari, C. 2010. *Reinforcement Learning: An Introduction*. Morgan & Claypool Publishers.

Ullrich, C. A. 2013. "Integrated Machine Scheduling and Vehicle Routing with Time Windows". *European Journal of Operational Research* 227(1):152–165.

Watkins, C., and P. Dayan. 1992. "Q-Learning". *Machine Learning* 8:279–292.

Wei, Q., Y. Yan, J. Zhang, J. Xiao, and C. Wang. 2022. "A Self-Attention-Based Deep Reinforcement Learning Approach for AGV Dispatching Systems". *IEEE Transactions on Neural Networks and Learning Systems (Early Access)*:1–12.

Wesselhöft, M., J. Hinckeldeyn, and J. Kreutzfeldt. 2022. "Controlling Fleets of Autonomous Mobile Robots with Reinforcement Learning: a Brief Survey". *Robotics* 11(5):85.

Williams, J. D., K. Asadi, and G. Zweig. 2017. "Hybrid Code Networks: Practical and Efficient End-to-end Dialog Control with Supervised and Reinforcement Learning". In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, 665–677. Vancouver, Canada: Association for Computational Linguistics.

Ye, D., Z. Liu, M. Sun, B. Shi, P. Zhao, H. Wu, H. Yu, S. Yang, X. Wu, Q. Guo et al. 2020. "Mastering Complex Control in MOBA Games with Deep Reinforcement learning". In *Proceedings of the AAAI Conference on Artificial Intelligence*, Volume 34, 6672–6679.

Zheng, B., L. Ming, Q. Hu, Z. Lü, G. Liu, and X. Zhou. 2022. "Supply-demand-aware Deep Reinforcement Learning for Dynamic Fleet Management". *ACM Transactions on Intelligent Systems and Technology (TIST)* 13(3):1–19.

AUTHOR BIOGRAPHIES

SRIPARVATHI SHAJI BHATTATHIRI is a student in the Engineering PhD program at Rochester Institute of Technology. Her area of interest is the use of machine learning and AI applied to material handling and human robot interaction. Her email address is ssb6096@rit.edu.

ANKITA TONDWALKAR is a PhD candidate in the Electrical and Computer Engineering PhD program at Rochester Institute of Technology. Her research interest includes cognitive radios, applications of machine learning and reinforcement learning to wireless communications and related domains. Her email address is at3235@.rit.edu.

MICHAEL E. KUHL is a Professor in the Industrial and Systems Engineering Department at Rochester Institute of Technology. His research interests include modeling and simulation of stochastic arrival processes, and the application of simulation to autonomous material handling, healthcare, and manufacturing systems. He is a member of the WSC Board of Directors representing the INFORMS Simulation Society. He has also served WSC as Proceedings Editor (2005), Program Chair (2013), and Mobile App Chair (2014-2019, 2022-2023). His email address is Michael.Kuhl@rit.edu.

ANDRES KWASINSKI is a Professor in the Department of Computer Engineering and Electrical and Computer Engineering PhD Director at the Rochester Institute of Technology. He has co-authored more than one hundred publications in peer-reviewed journals and international conferences. He has also co-authored the books "Cooperative Communications and Networking" (Cambridge University Press, 2009), "3D Visual Communications" (Wiley, 2013), and "Joint Source-Channel Coding" (Wiley-IEEE Press, 2022). He is currently Chief Editor of the IEEE Signal Processing Society Education Center. He has been an Associate Editor for the IEEE Signal Processing Magazine, and Editor for the IEEE Transactions on Wireless Communications and the IEEE Wireless Communications Letters. His current areas of research include cognitive radios and wireless networks, cross-layer techniques in wireless communications and smart infrastructures and networking. His email address is Andres.Kwasinski@rit.edu.