INTERNATIONAL JOURNAL OF PRODUCTION RESEARCH
https://doi.org/10.1080/00207543.2022.2148767

RESEARCH ARTICLE

OPEN ACCESS

Transaction selection policy in tier-to-tier SBSRS by using Deep Q-Learning

Bartu Arslan[a] and Banu Yetkin Ekren[b,c]

[a]Department of Industrial Engineering & Innovation Sciences, Eindhoven University of Technology, Eindhoven, Netherlands; [b]Department of Industrial Engineering, Yasar University, Izmir, Turkey; [c]School of Management, Cranfield University, Bedfordshire, UK

ABSTRACT
This paper studies a Deep Q-Learning (DQL) method for transaction sequencing problems in an automated warehousing system, Shuttle-based Storage and Retrieval System (SBSRS), in which shuttles can move between tiers flexibly. Here, the system is referred to as tier-to-tier SBSRS (t-SBSRS), developed as an alternative design to tier-captive SBSRS (c-SBSRS). By the flexible travel of shuttles between tiers in t-SBSRS, the number of shuttles in the system may be reduced compared to its simulant c-SBSRS design. The flexible travel of shuttles makes the operation decisions more complex in that system, motivating us to explore whether integration of a machine learning approach would help to improve the system performance. We apply the DQL method for the transaction selection of shuttles in the system to attain process time advantage. The outcomes of the DQN are confronted with the well-applied heuristic approaches: first-come-first-serve (FIFO) and shortest process time (SPT) rules under different racking and numbers of shuttles scenarios. The results show that DQL outperforms the FIFO and SPT rules promising for the future of smart industry applications. Especially, compared to the well-applied SPT rule in industries, DQL improves the average cycle time per transaction by roughly 43% on average.

ARTICLE HISTORY
Received 15 March 2022
Accepted 2 November 2022

KEYWORDS
Logistics; SBSRS; automated warehousing; deep reinforcement learning; DQN; agent-based simulation

1. Introduction

The recent COVID-19 outbreak has led to the growth of e-commerce and the acceleration of digital transformation in industries (Unctad 2021). For instance, since lockdowns have pushed businesses and consumers to go digital, it has paved the way for increased online purchasing. Increase in e-commerce has also led to increased customer expectations towards faster and cheaper delivery requests resulting in increased material handling equipment sales. Warehouse automation statistics indicate that the automation investment in warehouses and distribution centres have been increasing continuously (Statista 2021).

Automated storage and retrieval system (AS/RS) is a robotic technology for automated warehouses designed for buffering, storing, and retrieving products (Romaine 2020). They are commonly utilised in large warehouses and distribution centres. The essential advantage of AS/RS technology is reduced labour in the system. Hence, it provides a reduced number of human-based errors and injuries, increased accuracy, efficiency, productivity, etc. According to Allied Market Research (2020), mini-load AS/RS takes a significant place in the overall AS/RS market. *c*-SBSRS is a widely applied SBSRS technology in warehouses, which is mostly utilised for mini-load transaction processing and in micro-fulfilment centres (Scriven 2021). In that technology, the loads are stored and retrieved by tier-captive shuttles that can move horizontally through a dedicated aisle of a tier providing ultra-high transaction process capacity (Carlo and Vis 2012; Lerher 2015; Lerher et al. 2016; Lerher et al. 2015a, 2015b). The retrieval process in SBSRS starts with a shuttle movement, where it first picks up the load from a storage location and drops it off at the buffer area to be picked up by the lifting mechanism installed at each cross-aisle connected with that buffer area. Then, the lift carries the load at the input/output location. In a storage process, the process usually starts with a lifting operation. The lifting mechanism brings the load at the destination tier to be picked up by the shuttle at that tier. Later, the shuttle stores the load at the regarding storage address. Since there is a dedicated shuttle in each tier of an aisle, typically the average utilisation of those shuttles is very low compared to the average utilisation of lifts.

Recently, an alternative SBSRS design, *t*-SBSRS, was studied to alter the handicap of that low shuttle utilisation case in *c*-SBSRS (Ha and Chae 2018a, 2018b; Jerman

CONTACT Banu Yetkin Ekren banu.ekren@yasar.edu.tr; Banu.YetkinEkren@cranfield.ac.uk School of Management, Cranfield University, Bedfordshire, UK

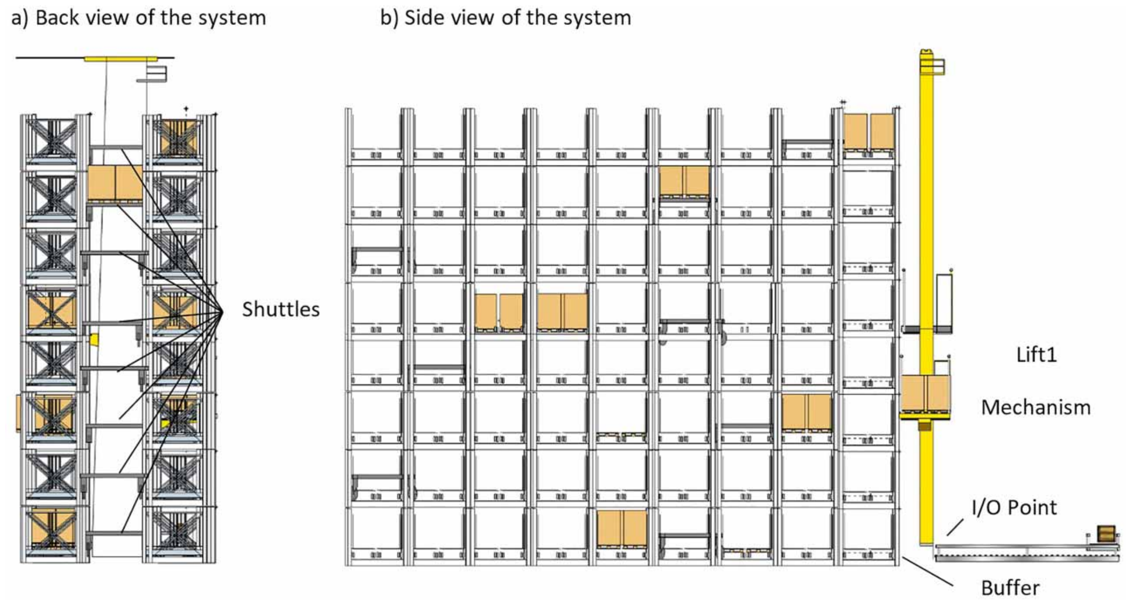

Figure 1. A *c*-SBSRS design. (a) Back view of the system. (b) Side view of the system.

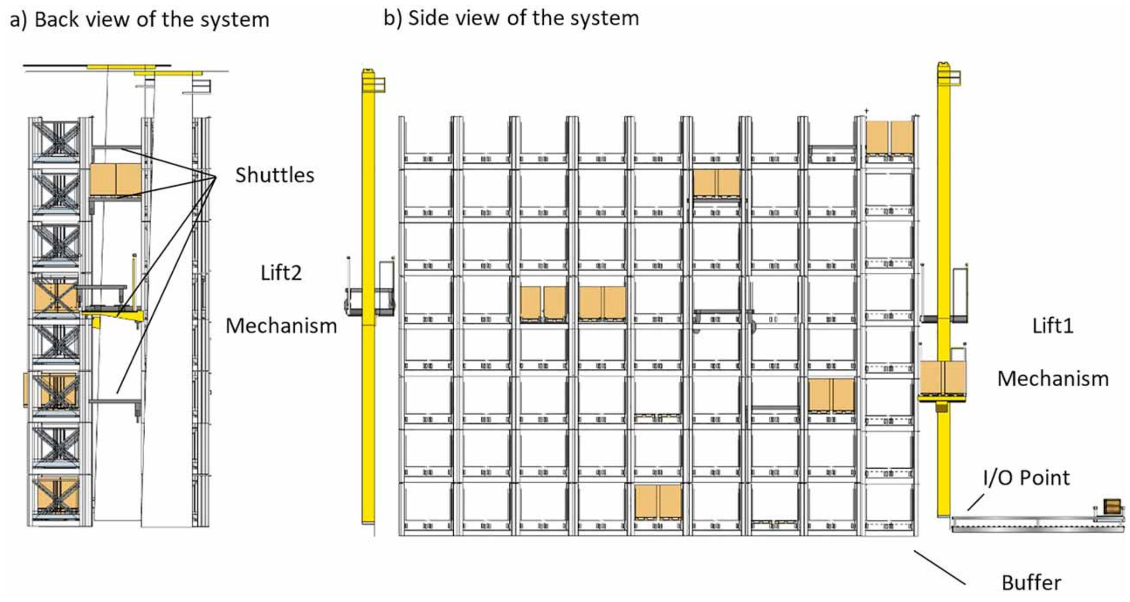

Figure 2. A *t*-SBSRS design.

et al. 2021; Küçükyaşar, Ekren, and Lerher 2021; Lerher, Ficko, and Palčič 2021; Zhao et al. 2019). With the *t*-SBSRS design, one of the aims is to balance the average utilisation of shuttles and lifts. By allowing shuttles to travel between tiers flexibly, the total number of shuttles may be decreased in the system which would also contribute in reduction in the initial investment cost of the system. Figures 1 and 2 show the *c*-SBSRS and *t*-SBSRS designs for a single aisle, respectively. In Figure 1, while there are as many shuttles as tiers in an aisle, in Figure 2 there are fewer shuttles than the ones in Figure 1. Note that, in Figure 2, Lift1 is installed at each cross-aisle to transfer the loads between tiers, and, Lift2 is mounted at the other side of each aisle to transfer the shuttles between tiers within that aisle.

The applicability of the travel of autonomous vehicles (i.e. shuttles) between tiers has also been studied well by Ekren (2020b), Ekren et al. (2010), Ekren and Heragu (2010), Ekren and Heragu (2011), Ekren et al. (2013), Ekren et al. (2014) where the system is referred to be autonomous vehicle-based storage and retrieval system (AVSRS). Differently, in *t*-SBSRS, the system is mainly designed for lightweight mini-loads processing, where loads are usually carried in totes.

In the *t*-SBSRS design, shuttles may tend to have longer travel than the *c*-SBSRS design, due to their travels between tiers. Once again, while the disadvantage of this novel design would be the increased travel time of shuttles, the advantage would be the possible decreased number of shuttles in the system. In *t*-SBSRS, since there are three different service providers, as two separate lifting mechanisms and shuttles that might be utilised synchronously, this situation may cause increase in the system's operation complexity when deciding which transactions to process first. To alter those handicaps, with the help of recent technological and information technology developments providing fast processing solutions, we study an intelligent modelling approach (i.e. a machine learning (ML) algorithm) that can take into consideration real-time data and information tracking from the current environment as well as future possible states while making a smart decision. Specifically, we apply a dynamic selection rule by DQL, for intelligent transaction selection of shuttles to decrease the average process time of a transaction in the system.

With recent technological developments, the computational power of computers has increased, and ML algorithms gained popularity. Hence, it might be viable to embed those complex control algorithms in smart machines so that practically it would be possible to apply the developed theoretical algorithms. In most fields, the development of mathematical models may take longer times for optimal search of stochastic environments. Hence, we prefer studying a deep reinforcement learning (DRL) approach developed on a model-free *Q*-Learning approach. The algorithm predicts the optimal policy by the environment dynamics (i.e. transition and reward functions). DRL is an appropriate approach where there is no prior or historical data as in our case (Tong et al. 2020; 2019). Besides, for a sequential decision problem (e.g. when a shuttle becomes available it selects a proper transaction to process), DRL is applied well on such problems which can also take into consideration the future possible states (Tong et al. 2020; Takahashi and Tomah 2020). The main research question of this paper is: *How does a Deep Q-Learning method in transaction selection provide a travel time advantage for shuttles compared to the static (i.e. heuristic) selection methods in the proposed novel t-SBSRS?*

Note that the DQL solution procedure applied in this paper can also be utilised for any job selection problem in such queuing systems. It is known that the SPT selection rule is applied well for job selection problems from queues in industry problems. This rule provides good performance metrics, especially from the average cycle time per job result. The contribution of this work is also by the comparison of the performance of the applied DQL method with a well-known SPT selection method under different warehouse designs.

Table 1. Abbreviations used in Table 2.

Abbreviation	Definition	Abbreviation	Definition
DQL	Deep *Q*-Learning	BE	Bellman Equation
SBSRS	Shuttle-based storage and retrieval system	RELU	Rectified Linear Unit
AS/RS	Automated storage and retrieval system	DQN	Deep *Q* Network
c	Tier-captive SBSRS	FQN	Fork-join queueing network
t	Tier-to-tier SBSRS	SIM	Simulation
OQN	Open queueing network	HEU	Heuristic
ML	Machine learning	ANA	Analytical
AVSRS	Autonomous vehicle-based storage and retrieval system	IP	Integer programming
DRL	Deep reinforcement learning	FB	Free-balancing
SPT	Shortest process time	RL	Reinforcement learning
FIFO	First-come-first-out (serve)	PA	Performance analysis
PR	Performance prediction	OP	Optimisation

The proposed operational approaches in this paper are simulated to test the hypothesis of whether they produce better performance results than mainly the SPT rule. The benefit of the implementation of DQL method for operation efficiency in *t*-SBS/RS would also provide significant benefits for warehouse managers from both operational and cost perspectives. Besides, not only warehouse managers, but also technology solution providers marketing those technologies could also benefit from the proposed applications effectively. Table 1 summarises the abbreviations used throughout the paper.

In Section 2, a literature survey summarising *c*-SBSRS, *t*-SBSRS, and ML related works on task scheduling is presented. In Section 3, we explain the ML methodology. In Section 4, we show the application of the method on the studied *t*-SBSRS. Section 5 summarises the results and comments. Section 6 provides a conclusion part.

2. Literature review

The reviewed literature works are summarised in Table 2 by categorising the SBSRS works based on the studied shuttle types, applied methods, and objectives. The sign 'X' shows where the regarding literature paper is placed in terms of system type, method, and objective. Each work is detailed in the following sub-section.

2.1. AVSRS literature

The current literature papers mostly focus on AVSRS and *c*-SBSRS. Marchet et al. (2011) study an AVSRS

Table 2. Literature studies on SBSRS.

Shuttle type			Method								Objective		
Literature	*c*	*t*	OQN	FQN	SIM	HEU	ANA	IP	FB	RL	PA	PR	OP
Marchet et al. (2011)	X		X								X		
Marchet et al. (2013)	X	X			X						X		
Carlo and Vis (2012)	X					X					X		
Lerher (2015)	X						X				X		
Lerher (2018)	X	X					X				X		
Lerher et al. (2015a)	X				X		X				X		
Lerher et al. (2015b)	X						X				X		
Lerher et al. (2016)	X						X				X		
Ekren, Sari, and Lerher (2015)	X				X						X		
Wang, Mou, and Wu (2015)	X						X						X
Tappia et al. (2017)	X		X		X						X		
Zou et al. (2016)	X			X								X	
Ekren (2017)	X				X							X	
Ekren et al. (2018)	X						X					X	
Eder (2019)	X		X									X	
Ekren (2020a)	X				X						X		
Ekren (2020b)		X			X						X		
Ekren and Akpunar (2021)	X		X				X					X	
Ha and Chae (2018a)									X		X		
Ha and Chae (2018b)		X					X				X		
Zhao et al. (2019)		X						X			X		
Küçükyaşar, Ekren, and Lerher (2021)	X	X			X						X		
Lerher, Ficko, and Palčič (2021)		X					X				X		
Jerman et al. (2021)		X			X						X		
Marolt, Kosanić, and Lerher (2022)	X				X						X		
The current Paper		X			X	X				X			X

and present an analytical model to predict some critical performance metrics from the system. They propose an open queueing network model for the solution process. Marchet et al. (2013) compare two configurations' performances: AVSRS and *t*-AVSRS using simulation modelling. Ekren (2020b) studies the design of AVSRS from a multi-objective optimisation procedure. In that study, the conflicting objectives are C_{avg} and energy consumption per transaction outputs. Marolt, Kosanić, and Lerher (2022) study multiple-deep *c*-AVSRS. They propose to increase the depth of each tier in the warehouse simultaneously to increase the average shuttle utilisation and decrease lift utilisation. They perform performance analysis on various pre-defined policies.

2.2. *c-SBSRS literature*

Carlo and Vis (2012) study a new variant of *c*-SBSRS, which includes two non-passing lifting mechanisms as a vertical travel provider. The performance of that system is compared with a traditional single lifting system design. In an effort to decrease space utilisation, Lerher (2015) studies a *c*-SBSRS with double-deep storage compartments. Lerher et al. (2015a) compare the system performance of *c*-SBSRS under alternative warehouse designs and, they show the benefits of utilisation of SBSRS. The results indicate that the proposed system reduces the average cycle time and increases throughput capacity. Lerher et al. (2015b) and (2016) develop analytical travel time models to calculate the performance of tier-captive SBSRS in their later work. They include velocity and acceleration scenarios for shuttles and lifts in the model and validate them by the simulation results. Ekren, Sari, and Lerher (2015) study the advantages of applying a class-based storage policy in *c*-SBSRS. They compare the class-based storage policy results with a random storage one. Wang, Mou, and Wu (2015) study a *c*-SBSRS design by proposing an analytical model for the job selection problem. They propose a sorting genetic algorithm for the solution of the proposed multi-objective optimisation problem. Tappia et al. (2017) study a *c*-SBSRS, and they present queueing network models predicting several performance metrics from the system. Zou et al. (2016) propose a fork-join queueing network model to predict some outputs from a *c*-SBSRS. A study by Ekren (2017) shows warehouse design trade-offs for *c*-SBSRSs. Ekren et al. (2018) develop an SBSBRS performance calculator estimating travel time, variance, and energy consumption outputs from a pre-defined *c*-SBSRS design. That initial work's outputs are utilised in their later works. Eder (2019) proposes an open queueing network (OQN) to solve a limited capacity problem. That paper estimates several performance outputs from the studied *c*-SBSRS. Ekren (2020a) shows a design of experiment work by the simulation to identify statistically significant effective factors on *t*-SBSRS. Later, Ekren and Akpunar (2021) propose another comprehensive tool developed by an

OQN to estimate several performance outputs from a pre-defined *c*-SBSRS warehouse design.

2.3. *t*-SBSRS literature

t-SBSRS design is first studied by Ha and Chae (2018a). There is a single lifting device in their system that transfers both shuttles and loads between tiers. They compare this system's performance with a traditional design one. The results indicate that fewer shuttles can also produce the target throughput rate than a tier-captive one. In their later work, Ha and Chae (2018b) propose a decision support system computing the required number of shuttles in a *t*-SBSRS. Zhao et al. (2019) propose an integer mathematical modelling by sequencing transactions in a *t*-SBSRS, minimising the idle times in the system. Lerher (2018) studies aisle changing shuttles rather than a tier changing one in automated warehousing. A recent study by Küçükyaşar, Ekren, and Lerher (2021) compares traditional *c*-SBSRS and *t*-SBSRS designs under several performance metrics, including their initial investment costs. The results indicate that well-designed *t*-SBSRSs could decrease the investment costs and increase the performance of the system compared to a *c*-SBSRS. In a recent study, Lerher, Ficko, and Palčič (2021) propose a novel AVSRS design with multiple-tier shuttles. They present analytical models for computing performance metrics such as cycle time and throughput rate from the system. Jerman et al. (2021) propose a novel storage and retrieval system. They analyse the throughput rate output based on different warehouse designs by using simulation modelling. Liu et al. (2021) develop an energy consumption model that estimates maximum throughput and travel time performance metrics for a *c*-SBSRS that operates on a dual-command cycle. Li et al. (2022) develop a mixed integer programming model to jointly optimise multi-item order batching and retrieving problems.

2.4. Machine learning applications

In this section, we also provide the ML applications for warehouse automation problems. Watanabe et al. (2001) apply a *Q*-Learning method to avoid collision and to track navigation of AGVs. Dou, Chen, and Yang (2015) propose a genetic algorithm approach developed on an RL approach for task scheduling in an automated warehouse system. Xue, Zeng, and Yu (2018) present the flowshop scheduling problem in a multi-AGV system by RL approach. The results indicate that the RL agent can find a near-optimal result from its past experience, and it performs better than a multi-agent model. Malus, Kozjek, and Vrabič (2020) apply the multi-agent RL method for autonomous mobile robots to solve an order dispatching problem.

Differently, Mao et al. (2016) apply a DRL for a resource management problem for computer network systems. The research shows that DRL is applicable for large-scale systems. Tong et al. (2020) implement DQL for dynamic job sorting problems in cloud computing. The paper shows that DQL outperforms standard algorithms. Gazori, Rahbari, and Nickray (2020) studied job scheduling problems for IoT implementations. That study aims to minimise computational costs and service delays in the system by implementing Double Deep Q-Learning. Takahashi and Tomah (2020) propose DRL to develop control algorithms in a multi-AGV system. The results show that the algorithm finds near-optimal solutions, and it works very well in dynamic environments. Hao et al. (2020) study a deep reinforcement learning based real-time scheduling model with a mixed rule for AGVs. Yin, Liu, and Wang (2022) present a decentralised framework of multi-task allocation with attention in DRL combining the task assignment balance and path planning for the distribution process. The results show that the proposed approach has feasibility and effectiveness in multi-AGVs task allocation applications.

To the best of our knowledge, there is no ML-based application for job selection and sequencing problems in automated storage and retrieval systems in the literature. Mainly, the applied approaches are typically static job selection methods such as FIFO or SPT for those systems. The novelty of this study is that we show the applicability of ML algorithms for job selection problems in a complex queuing system (i.e. *t*-SBSRS), including three different service providers (i.e. Lift1-2 and shuttles) which are working in conjunction with each other. To test the performance of the applied method, we compare the results of the dynamic approach with the well-applied static job selection algorithms, FIFO and SPT.

3. Methodology

In this section, we show the theoretical background of the utilised methodology.

3.1. Reinforcement learning

RL is an ML method that is developed to define how intelligent agents should take actions in a system based on interaction with the environment. Here, intelligent agents can be trained by trial and error by utilising the information from the taken actions and experiences. Hence, it is based on neither supervised learning nor unsupervised learning. Here, agents learn to react to an environment based-on their experience.

Any RL problem includes the following contents (Sutton and Barto 2015):

(i) *Agent*: an autonomous object (e.g. a robot) controlling the target of concern.
(ii) *Environment*: it defines everything that agents interact with. It is built for the agent to make it visible like a real-world case. States are representations of that current world or environment.
(iii) *Rewards*: it is a score of how the algorithm performs with respect to the environment. The cumulative reward is the goal of the agent to be maximised in an RL problem. The reward is obtained by mapping each state-action pair of the environment to a single value.
(iv) *Policy*: it is the algorithm used by the agent to decide its actions. This part can be model-based or model-free. A model-based algorithm applies the transition and the reward functions to predict the optimal policy. A model-free algorithm predicts the optimal policy without the environment's dynamics, such as transition probabilities. It predicts the 'value function' from experience without a transition or reward. Here, the value function is a function evaluating action taken in a state for all states. The policy can be obtained from that value function.

In this work, since we do not know the transition probabilities and rewards based on state-action relationships in advance, to estimate those, we follow a model-free approach developed on a Q-Learning method by simulating the system. We consider the update rule by the Bellman Equation (BE) defined by (1):

$$Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha[r_t + \gamma \max_a Q(s_{t+1}, a)] \tag{1}$$

By the BE, we update our Q-table after each action is taken. According to the BE, it is aimed to update the current perceived value with the predicted optimal future reward. By that, it is assumed that the agent takes the best-known action in the next step. While taking an action decision, the agent searches all the Q-values of the possible actions of the current state and selects the action with the largest Q-value one. In Equation (1), α represents the learning rate, r_t is the current reward gained at time t, γ is the discount factor causing rewards to lose their value over time. Hence, $\gamma \max_a Q(s_{t+1}, a)$ calculates the maximum reward for the next state to be obtained weighted by the discount factor γ.

Usually, the Q-table is initialised by a zero matrix, meaning that the agent has no information about its state-action relation. The learning process continues until all the Q-table values become stable. For the learning process, the epsilon-greedy approach is utilised. In this approach, the computer creates a random number (i.e. 0–1). If that number is smaller than the epsilon value, ε, the agent takes a random action. This process is called 'exploration' and is performed during the learning period. Otherwise, it proceeds with the action having the highest Q-value one. To increase the exploration possibility, the epsilon value, ε, is started from 1, and it is decreased over time.

The size of the Q-table can be calculated by the total number of states (N) $\times$ the total number of actions (M). Since the matrix's size would significantly affect the agent's training time, large numbers of states and actions may cause the learning problem to become infeasible. To deal with that handicap, we estimate the Q-value function by Deep Neural Networks (DNN), known by their efficiency in approximating functions. Integration of DNNs into the Q-Learning process is known as Deep Q-Learning or Deep Q-Network (DQN) (Mnih et al. 2015). The details of applied DQN are explained in Section 3.2.

3.2. DQL

DQN is a Deep RL method that utilises neural networks to approximate Q-values. Here, a neural network receives states from the environment as inputs. Then, it approximates the Q-values for each action. The Deep RL process is shown in Figure 3.

DNN considers the states as inputs, and it outputs the Q-values of all possible actions for those inputs. The size of the input layers of the DNN is equal to the size of the states. Besides, the size of the output layer is equal to the number of actions.

During the training process, a loss function, the Temporal Difference error function (TD function), the difference between the Q-value of a state-action pair and

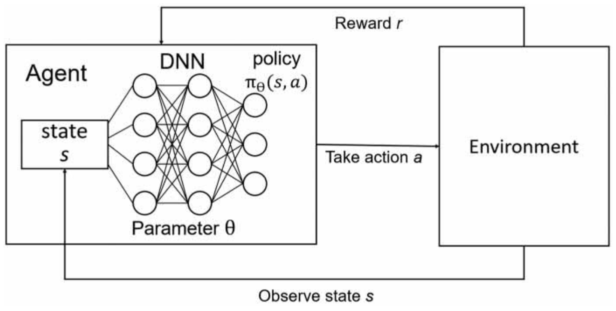

Figure 3. Agent behaviour in DQN (adopted from Mao et al. 2016).

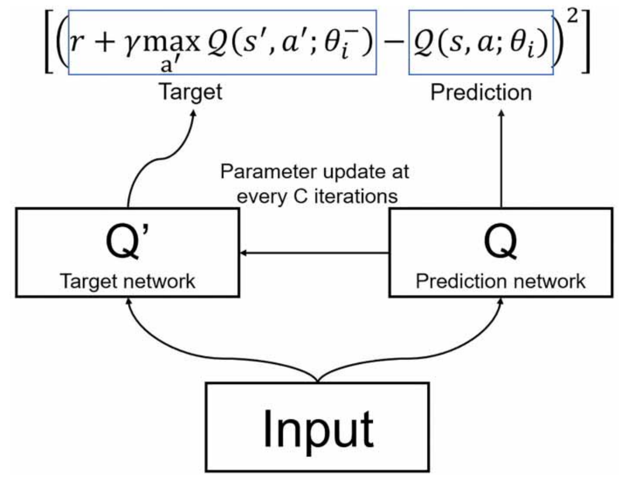
$\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta_i^-)\right) - Q(s, a; \theta_i)\right)^2\right]$
Target
Prediction
Parameter update at every C iterations
Q'
Target network
Q
Prediction network
Input

Figure 4. Updating target network by using prediction network (adopted from Choudhary 2019).

```
Algorithm 1 Deep Q-Learning Algorithm
Initialize: Action-value and target action-value functions $Q$ and $\hat{Q}$ with random weights $\theta$
Initialize: Batch size $n$, learning rate $\alpha$, discount rate $\gamma$, epsilon $\epsilon$, epsilon decay rate $\epsilon^{dec}$, epsilon minimum value $\epsilon_{min}$
 1: for episode = 1, $M$ do
 2:     Observe state $s_t$
 3:     With probability $\epsilon$, select random action $a_t$, otherwise select $a_t$ = argmax $Q(s, a)$
 4:     Execute action $a_t$, observe reward $r_t$
 5:     Store transition $(s_t, a_t, r_t, s_{t+1})$
 6:     Sample random tuple from transition tuples
 7:
          $Set y_j = \begin{cases} r_j, & \text{if episode terminates at step j+1} \\ r_j + max\hat{Q}(s', a'), & \text{otherwise} \end{cases}$
 8:     Perform gradient descent step on $(y_j - Q(s, a))^2$
 9:     Update $\epsilon$ as $\epsilon = \epsilon \times \epsilon^{dec}$ if $\epsilon > \epsilon_{min}$, otherwise $\epsilon_{min}$
10:     Every C steps, $\hat{Q} = Q$
```

Figure 5. Deep *Q*-Learning with experience replay algorithm.

its *Q*-Target value is utilised. Compared to the other DL methods, *Q*-Target is not stable, which makes the training challenging. To obtain more stable training, we utilise two neural networks. One network represents the *Q*-Target, and the other represents the prediction. The two networks have the same architecture. For every *C* iteration, the estimated network parameters are copied at the target network. This method helps the training to be more stable (Choudhary 2019). A visual explanation of the procedure is shown in Figure 4.

After an action is completed, state, action, reward, and next state information are kept in memory. In DQN, as opposed to *Q*-Learning, the *Q*-table is not updated at each step. Instead, state, action, reward, and next state information are kept and fed into the network after reaching a pre-defined size. This method, named as experience replay, is utilised in our study (Mnih et al. 2015). The pseudocode of this algorithm is given in Figure 5. First, the agent observes the state. Then, it selects and executes action according to an epsilon-greedy approach. State, action, reward, next state tuple are kept in memory in each step. After reaching a certain point, we sample from the transition tuples. These tuples are fed into the network and perform gradient descent on the target network. The epsilon is decreased at each iteration and in every *C* step, the prediction network is cloned to the target network. We show the applied procedure for the algorithm in Section 4.

4. Deep *Q*-Learning implementation for SBSRS

We simulate the system to observe the performance of the DQL. The simulation model details are given in Section 4.1.

Table 3. Notations utilised in the model

Notation	Unit	Definition
C_{avg}	s	Average cycle time of a transaction
F_{avg}	s	The average flow time of a transaction
W_{avg}	s	Average waiting time of a transaction
T	s	The mean inter-arrival time of transactions
Vs	m/s	The maximum speed that a shuttle can reach
Vl	m/s	The maximum speed that Lift1 can reach
Vsl	m/s	The maximum speed that Lift2 can reach
As	m/s^2	Acceleration for shuttle velocity
Al	m/s^2	Acceleration for Lift1 velocity
Asl	m/s^2	Acceleration for Lift2 velocity
Ds	m/s^2	Deceleration for shuttle velocity
Dl	m/s^2	Deceleration for Lift1 velocity
Dsl	m/s^2	Deceleration for Lift2 velocity
US_{avg}	%	Average utilisation of a shuttle
UL_{avg}	%	Average utilisation of Lift1
USL_{avg}	%	Average utilisation of Lift2
W	m	Two bays' distance
H	m	Two tiers' distance
T		Number of tiers in an aisle
B		Number of bays on either side of a tier
S		Total number of shuttles

4.1. *Simulation model assumptions*

The simulation model is completed in Python programming, by using the SimPy library. In Table 3, the utilised notations and their units are provided.

In the simulation model, whenever a shuttle becomes available, it checks the status of the waiting transactions in its queue. If there is more than one transaction in its queue, it picks the transaction based on the pre-defined selection approach (SPT, FIFO, or DQN). After the shuttle picks a transaction, the transaction request entity is cloned, and it immediately enters the queue of Lift1. If the process is storage, then, Lift1 moves to the I/O point to receive the tote. Later, it moves to the storage tier with the tote. Meanwhile, if the seized shuttle is located at a tier different from the storage address, it requests Lift2

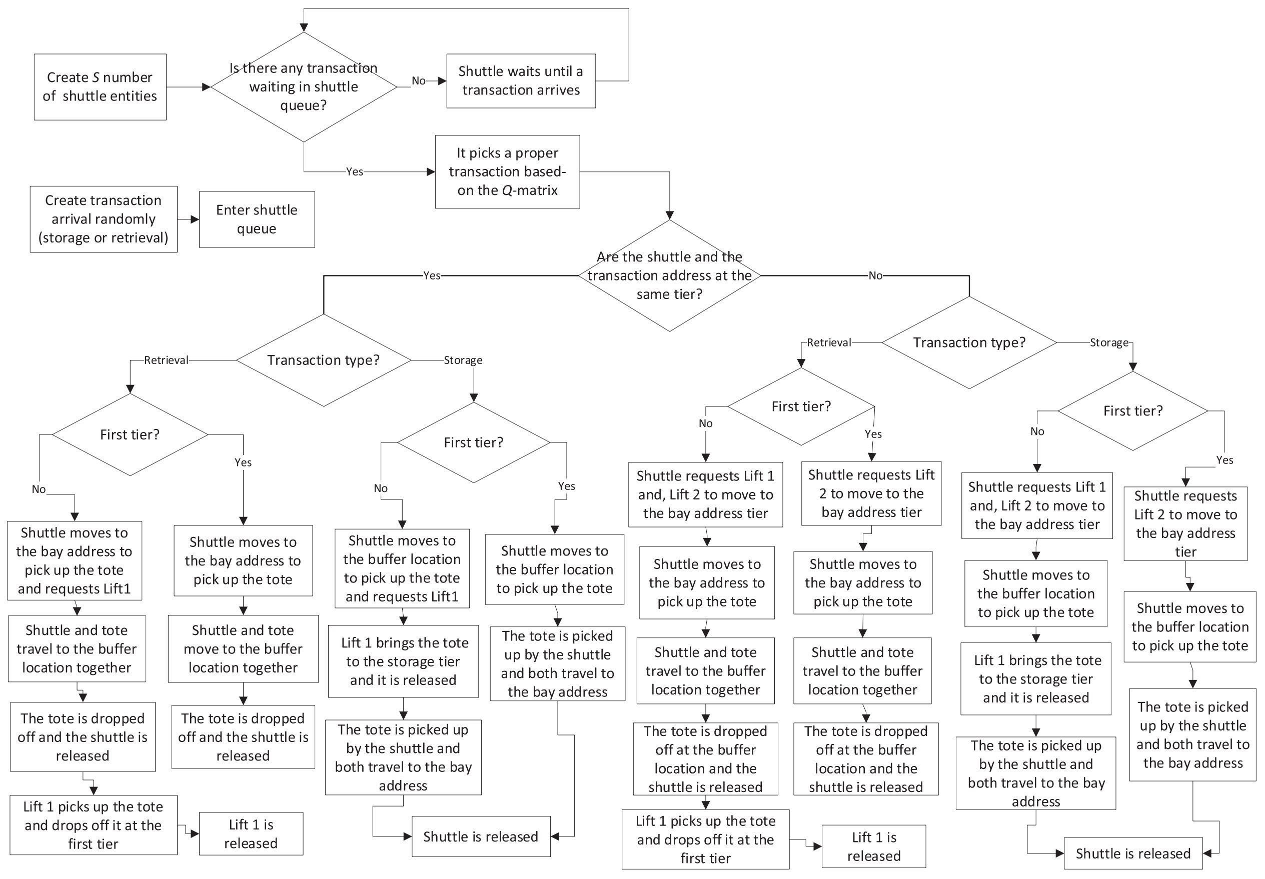
Create S number of shuttle entities
Is there any transaction waiting in shuttle queue?
No
Shuttle waits until a transaction arrives
Yes
It picks a proper transaction based-on the Q-matrix
Create transaction arrival randomly (storage or retrieval)
Enter shuttle queue
Are the shuttle and the transaction address at the same tier?
Yes
No
Transaction type?
Retrieval
Storage
First tier?
No
Yes
Shuttle moves to the bay address to pick up the tote and requests Lift1
Shuttle and tote travel to the buffer location together
The tote is dropped off and the shuttle is released
Lift 1 picks up the tote and drops off it at the first tier
Lift 1 is released
Shuttle moves to the bay address to pick up the tote
Shuttle and tote move to the buffer location together
The tote is dropped off and the shuttle is released
Shuttle moves to the buffer location to pick up the tote and requests Lift1
Lift 1 brings the tote to the storage tier and it is released
The tote is picked up by the shuttle and both travel to the bay address
Shuttle moves to the buffer location to pick up the tote
The tote is picked up by the shuttle and both travel to the bay address
Shuttle is released
Shuttle requests Lift 1 and, Lift 2 to move to the bay address tier
Shuttle moves to the bay address to pick up the tote
Shuttle and tote travel to the buffer location together
The tote is dropped off at the buffer location and the shuttle is released
Lift 1 picks up the tote and drops off it at the first tier
Lift 1 is released
Shuttle requests Lift 2 to move to the bay address tier
Shuttle moves to the bay address to pick up the tote
Shuttle and tote travel to the buffer location together
The tote is dropped off at the buffer location and the shuttle is released
Shuttle requests Lift 1 and, Lift 2 to move to the bay address tier
Shuttle moves to the buffer location to pick up the tote
Lift 1 brings the tote to the storage tier and it is released
The tote is picked up by the shuttle and both travel to the bay address
Shuttle requests Lift 2 to move to the bay address tier
Shuttle moves to the buffer location to pick up the tote
The tote is picked up by the shuttle and both travel to the bay address
Shuttle is released

Figure 6. Flow chart of the agent-based simulation model.

and travels to the storage tier with Lift2. After the shuttle arrives at the storage tier, it moves to the buffer area to pick up the storage tote. Shuttle and tote travel to the bay address together and, the shuttle discharges the load at the bay address.

In the case that the selected transaction is a retrieval transaction, Lift1 is requested immediately. Then, it moves to the destination retrieval tier. After the shuttle drops the load at the buffer area, Lift1 receives the load and carries it to the I/O point. Note that Lift1 is never utilised for both transaction types if the tier of the transaction is first tier. Again, Lift2 is never utilised if the tier of the address and the seized shuttle's tier are same. After the shuttle arrives at the retrieval tier, it brings the load at the buffer area for Lift1's pick up. The scheduling rules of Lift1 and Lift2 are always FIFO not to ruin the transaction processing order of shuttles (note that first, a shuttle is seized then, lifts). The SPT and DQN policies are applied for the transaction selection of shuttles. The simulation model's detailed flow is given in Figure 6.

The considered assumptions in the simulation model are summarised below (Lerher 2015; Lerher et al. 2015a, 2015b; Kucukyasar, Ekren, and Lerher 2021):

- The mean inter-arrival rates follow Poisson distribution and, they are equal for storage and retrieval transactions ($\lambda_S = \lambda_R$).
- We consider a random storage policy.
- Storage transactions arrive at, and retrieval transactions leave from, the I/O point in the warehouse.
- There is a single shuttle queue, where storage and retrieval transaction requests arrive.
- We ignore the loading/unloading times of transactions.
- Lift1 and Lift2 process the waiting transactions in the FIFO order.
- There are two lifting tables in Lift1, and they work independently. Namely, the capacity of that lifting mechanism is two totes.
- The last completion location point is the dwell point of lifts/shuttles.
- $Vs = Vl = Vsl = 2\,\text{m/s}$.
- $As = Al = Asl = Ds = Dl = Dsl = 2\,\text{m/s}^2$.
- $W = 0.5\,\text{m}, H = 0.35\,\text{m}$.
- The number of replications is calculated by considering a desired half-width value and, it is set to minimum of three independent replications.

We debug the codes and apply degenerative changes on the input variables to observe whether we obtain the expected outputs to verify the simulation model. Since there is no real t-SBSRS, we validate the simulation model by involving experts in the modelling as well as by comparing the estimated F_{avg} from the current literature works. In the following section, we explain the applied DQL approach.

4.2. Deep Q-Learning implementation

In this approach, we perform a non-episodic task where the task has no clear ending point, and the shuttle agent is kept trained until the system stops. How we define the main DQN elements is summarised in below subsections.

4.2.1. Agents

The agents are the shuttles in the system. The primary role of shuttle agents is to apply an intelligent transaction selection policy considering possible future states to result in total maximum reward in the long run. Remember that each shuttle agent picks an action (i.e. a transaction) based on the estimated Q-value and keeps that information to train itself according to the gained reward.

4.2.2. State space

The state space is what the agent feeds into the network from environment information. In the studied problem, the state space is defined as:

$S(k)$ = (Current tier of the shuttle k, current bay of the shuttle k, current tier of the first lifting table of Lift1, availability of the first lifting table of Lift1, current tier of the second lifting table of Lift1, availability of the second lifting table of Lift1, current tier of Lift2).

Here, k represents the shuttle that is available and will pick a transaction from its queue. The tier values are integers between 1 and T, and the bay values are integers between 1 and B, availability is either '0' or '1' where '0' shows that the lift is currently not available and '1' represents that the lift is currently available.

4.2.3. Action space

The transactions waiting in the queue of shuttles have several attributes which are considered as actions in the model. Specifically, the action space is defined as:

$A(k)$ = (the tier address of the transaction, the bay location of the transaction, transaction type, and table side of Lift1).

Note that from the action definition, while a shuttle selects a transaction, at the same time, a proper Lift1 table is also selected. To explain how states are defined: we assume that there is a storage transaction in the shuttle queue, whose storage address is at the 5th tier and the 20th bay, then the actions related to this transaction would be (5, 20, 0, 1) and (5, 20, 0, 2) according to the action definition above. 0 represents storage transaction type (i.e. it is 1 for retrieval transactions), and 1 and 2 represent the lifting table side of the Lift1 mechanism. Note that actions become active according to the attributes of the waiting transaction. In addition, a waiting transaction might be ignored from being an active action for the shuttle when there is already a busy shuttle processing at that transaction's tier. This would also help for the prevention of shuttle collisions that could happen at the same tier.

4.2.4. Reward function definition

As mentioned before, the reward is a performance metric from the system. It is the most crucial issue to track for an agent to make an optimal decision for increasing cumulative reward in the long run. In the DQN approach, we aim to minimise the C_{avg} performance metric. Here, we define cycle time as the total time spent in the system. For that, we subtract the time transaction created from the time it is disposed. In other words, it includes all the waiting times of transactions in the system. F_{avg} is the average time a transaction travels in the system, it ignores the waiting time in the shuttle queue. In the defined reward function, to reduce the state space, we do not include the queue related information such as so far waiting time of transactions, arrival time and queue order of transactions. Hence, the cycle time reward values do not correlate with the states. To overcome that issue, the reward function is normalised by using Equations (2)–(4) in order:

$$\mathrm{MINF}_t = \frac{1}{\max(\mathrm{flowtime})} \tag{2}$$

$$\mathrm{MAXF}_t = \frac{1}{\min(\mathrm{flowtime})} \tag{3}$$

$$r_t = \frac{\frac{1}{\mathrm{flowtime}(a)} - \mathrm{MINF}_t}{\mathrm{MAXF}_t - \mathrm{MINF}_t} \times 100 \tag{4}$$

Note that the realised flow time values are stored in the set of *flowtime*. In Equation (2), the inverse of the maximum *flowtime* returns the minimum value of 1/*flow times*. In Equation (3), the inverse of the minimum *flowtime* returns the maximum value of 1/*flow times*. Equations (2) and (3) are utilised for normalising the reward function. Normalisation is generally formalised by $X_{\text{norm}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}}$ (Gazori, Rahbari, and Nickray 2020). Based on this formula, we treat our normalisation function by Equation (4). In Equation (4), *flowtime* (a) represents the current flow time of action a. Since the

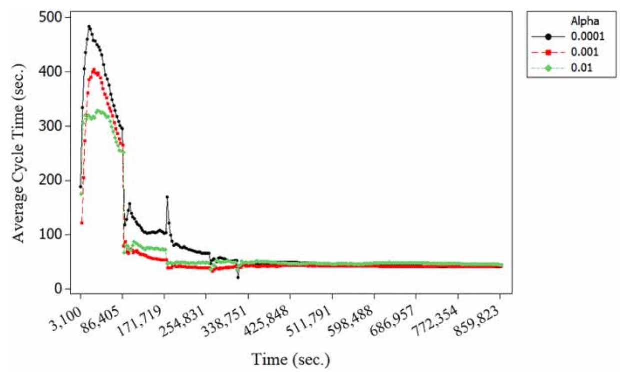

Figure 7. DQN results under three different learning rate scenarios.

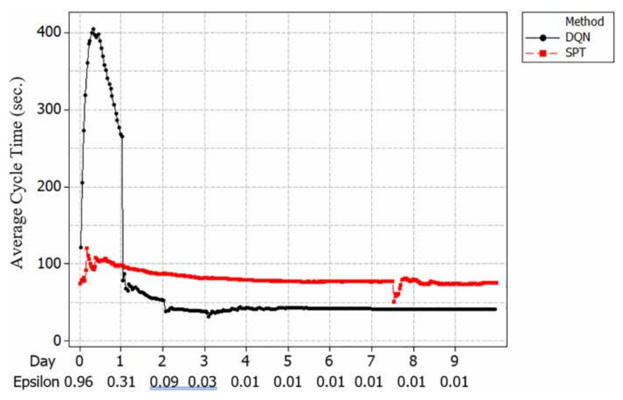

Figure 8. Average cycle time of each day.

normalised values are relatively small, we multiply the values by 100 to track the improvement conveniently.

4.2.5. *DQN results and utilised parameters*

The DQN methodology is applied by using the TensorFlow and Keras frameworks in Python Programming. Here, the DQN model is composed of three dense layers: input, hidden, and output. Rectified Linear Unit (ReLU) is utilised for input layer and hidden layer definition as an activation function. Here, the activation function defines, how the weighted sum of inputs would be transferred into the output node. While positive input values exist, ReLU might be a proper approach to apply. For the optimiser of the network, 'Adam' optimiser is chosen with parameters $\alpha = 0.001$, $\gamma = 0.2$, $\varepsilon = 1$, $\varepsilon_{\text{dec}} = 0.9999$, $\varepsilon_{\text{min}} = 0.01$, $n = 64$. Besides, the Mean Squared Error (MSE) loss function is considered.

The average cycle time performance metric results from the DQN application for the warehouse design with $T = 5$, $B = 25$, $S = 2$ are given in Figure 7 for three different learning rates, α values: 0.01, 0.001 and 0.0001 separately. To observe how long it takes to train an agent, we provide average cycle time versus simulation runtime graph. It is observed that the fast-training time is obtained when $\alpha = 0.001$. This is mainly because the system is highly stochastic, and a small learning rate might work better. In that figure, it can be assumed that after 86,405 s (e.g. roughly after one day), the agents are trained well due to the drastic decrease in the performance output.

4.2.6. *DQL results comparison with SPT*

Since SPT is a well applied selection rule in practice, here first, we compare the DQN results with static the SPT selection rules of shuttles. Note that DQN applies random transaction selections during its exploration period to train agents. During the random selection in the training process, the cycle times of transactions might be high coming with a cost to the company. Namely, some performance metric results may be worse during the training period than the other static algorithms. Figure 8 shows the average cycle time per transaction versus simulation time under the best α value, 0.001, observed from Figure 7. In that experiment, the other DQN parameter values are: $\gamma = 0.2$, $\varepsilon = 1$, $\varepsilon_{\text{dec}} = 0.9999$, $\varepsilon_{\text{min}} = 0.01$. Since the applied method is a non-episodic task, we treat each day as an episode. After each day, the C_{avg} output is reset to zero to trace each episode's average cycle time output. As observed in Figure 8, the DQL algorithm outperforms the SPT algorithm after a single day (on average, 13,091 transactions later). Hence, the training cost would incur in terms of relatively higher average process time of a transaction just in the first day.

Note that the Figure 7 results are for the warehouse design with $T = 5$, $B = 25$, $S = 2$. Since the shuttles are defined as agents, an increased number of shuttles might result in decreased training time due to the increased possibility of facing more state-action cases. Hence, Section 5 provides several warehouse design experiments results by considering different T, B, and S values.

5. Experimental results

Here, as mentioned previously, the DQL method is compared with both SPT and FIFO rules. Here, in the FIFO algorithm, shuttles constantly pick the first arriving transaction in the system. In the SPT rule, the transactions are selected based on the shortest travel distances through their addresses.

We consider eight warehouse designs under different scenarios of the number of tiers, T, the number of bays, B, and the number of shuttles, S, in the system. The experiments are selected by focusing on different number of shuttles, bays and tiers scenarios. Since the simulation

Table 4. Experimental design and their results.

Exp	T	B	S	Algorithm	T	UL_{avg}	USL_{avg}	US_{avg}	C_{avg}	F_{avg}	W_{avg}
1	5	25	2	FIFO	6.6	Burst	Burst	Burst	Burst	Burst	Burst
	5	25	2	SPT	6.6	47%	51%	94%	98.5 ± 6.59	12.94 ± 0.01	85.56
	5	25	2	DQN	6.6	37%	44%	85%	41.67 ± 2.05	11.53 ± 0.13	30.14
2	8	25	3	FIFO	5.2	Burst	Burst	Burst	Burst	Burst	Burst
	8	25	3	SPT	5.2	69%	74%	90%	61.54 ± 4.2	14.69 ± 0.02	46.86
	8	25	3	DQN	5.2	55%	66%	81%	36.52 ± 1.56	12.96 ± 0.17	23.56
3	10	25	4	FIFO	4.2	Burst	Burst	Burst	Burst	Burst	Burst
	10	25	4	SPT	4.2	86%	86%	90%	89.34 ± 7.32	15.51 ± 0.06	73.83
	10	25	4	DQN	4.2	67%	80%	80%	46.06 ± 2.44	13.47 ± 0.25	32.59
4	13	25	5	FIFO	4.4	Burst	Burst	Burst	Burst	Burst	Burst
	13	25	5	SPT	4.4	90%	92%	83%	57.93 ± 2.97	18.64 ± 0.05	39.29
	13	25	5	DQN	4.4	72%	90%	74%	44.13 ± 0.98	16.07 ± 0.22	28.07
5	5	50	2	FIFO	12	42%	57%	97%	567.39 ± 187.65	23.72 ± 2.24	543.66
	5	50	2	SPT	12	41%	52%	94%	138.18 ± 11.73	22.97 ± 0.08	115.21
	5	50	2	DQN	12	34%	46%	86%	69.81 ± 4.46	20.83 ± 0.16	48.98
6	8	50	3	FIFO	9.2	64%	79%	94%	257.14 ± 38.13	26.64 ± 0.05	230.5
	8	50	3	SPT	9.2	62%	74%	92%	114.32 ± 8.87	25.93 ± 0.06	88.39
	8	50	3	DQN	9.2	51%	69%	85%	66.12 ± 3.12	23.7 ± 0.2	42.42
7	10	50	4	FIFO	7.6	Burst	Burst	Burst	Burst	Burst	Burst
	10	50	4	SPT	7.6	77%	87%	91%	147.44 ± 11.51	28.11 ± 0.08	119.34
	10	50	4	DQN	7.6	63%	83%	84%	79.44 ± 3.98	25.57 ± 0.22	53.88
8	13	50	5	FIFO	7.4	Burst	Burst	Burst	Burst	Burst	Burst
	13	50	5	SPT	7.4	86%	93%	88%	155 ± 8.96	32.73 ± 0.15	122.27
	13	50	5	DQN	7.4	72%	93%	83%	104.35 ± 5.21	30.33 ± 0.35	74.02

run of DQN model is time consuming due to the training period, we complete a single long-run simulation by applying batching (Law 2015). Table 4 presents the conducted experiments. Those designs are also studied well in literature (Küçükyaşar, Ekren, and Lerher 2021; Ha and Chae 2018b). The conducted experiments are: $T = 5, 8, 10, 13$; $B = 25, 50$; $S = 2, 3, 4$. The simulation results are also given in Table 4. The mean inter-arrival times are adjusted so that the average utilisation of the bottleneck service provider (i.e. shuttles) is large enough, specifically larger than 90%. To make a fair comparison independent from an effect of a warehouse design, note that we fix the inter-arrival time within a specific warehouse design and compare the performance outputs under that arrival rate. Namely, the arrival rates for the first four experiments are same. However, the following four experiments are different from those initial four experiments. The results are given at 95% confidence intervals. In Table 4, 'Burst' represents that the design model cannot produce a feasible result due to not having a steady-state condition. In other words, the system explodes due to large number of transactions in the system. Since it is time consuming to run the DQN models due to its training periods, and we are interested in the performance results after the learning periods.

From Table 4, it is observed that in most cases, the FIFO schedule rule cannot produce a feasible solution (e.g. due to a fixed arrival rate). DQN results consistently outperform both FIFO and SPT rules considering the C_{avg} performance metric. When $T = 5$, $B = 50$, $S = 2$, the DQN method decreases the C_{avg} by 57.7%, and when

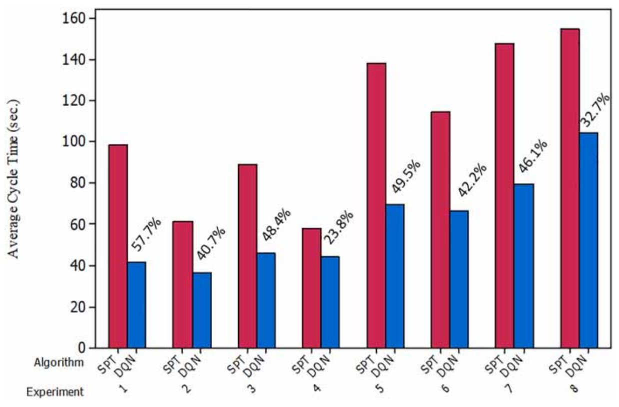

Figure 9. Comparison of SPT and DQN selection policies for C_{avg}.

$T = 8$, $B = 50$, $S = 2$, the DQN method decreases the average cycle time by 49.5% when it is compared to the SPT rule. These results are also summarised in Figure 9. From that figure, it is observed that DQN mainly produces highly better results than the SPT results. This is possibly caused by the effect of the future rewards. SPT can be considered as if a myopic policy, due to focusing on solely the current minimum flow times. However, the DQN algorithm also considers the possible future rewards, and this leads shuttles to travel to tiers/bays so that that the average flow time of transactions is minimised in long term. Therefore, DQN can be considered as a promising job selection approach for the future of smart industries.

6. Conclusion

This paper studies a DQN application for an AS/RS warehouse for the system's transaction selection policy of autonomous vehicles (i.e. shuttles). The studied warehouse, *t*-SBSRS, is developed to alter the handicaps of *c*-SBSRS which are the unbalanced utilisation of shuttles and lifts. Hence, there is more than the required number of shuttles in the system. In the proposed novel *t*-SBSRS, the number of shuttles is reduced by enabling them to move between tiers. That would benefit in facilitating the investment decisions of those automated technologies. To the best of our knowledge there is no limitation to implement such a new design in SBSRS. However, from the ML-based perspective, a good connected IoT environment would be required. There are three separate service providers in the new system. These are: the two lifts for movement of loads and shuttles between tiers separately, and the shuttles for horizontal travel of loads. Due to the decreased number of shuttles in the system, and hence possible increased travel time expectation in the system, we apply an intelligent modelling approach (i.e. DQL) that can take into consideration real-time data and information tracking from current environment as well as future possible states while making smart decisions. Due to the complexity of the system in which those service providers work interactively, we simulate the system to observe how DQL improves average cycle time of a transaction time in the system. We model the systems by using the Python, SimPy library, and the DQN is applied using the TensorFlow and Keras libraries. We compare the proposed system's performance with SPT and FIFO selection rules under different warehouse designs. The results show that DQN outperforms FIFO and SPT rules which is promising for the future of smart industry applications. Especially, when it is compared with a well-applied SPT, on average, DQL decreases the average cycle time output by roughly 43%. Besides, this paper is a promising work on showing how recent ML-based algorithms can be applied on automated warehousing systems promising for the future of smart industry applications.

As the future works, more environmental information, including attributes of waiting transactions in the queue could be included in state information by also exploring fast training learning algorithms. Besides, more experiments can be concluded by also including different velocity profiles of servers and number of tiers in the system. Additionally, different reward functions taking into multiple objectives in the system can also be considered. Last, a performance comparison analysis with a *c*-SBSRS would also be worth studying as a future work.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This work was supported by The Scientific and Technological Research Council of Turkey and Slovenian Research Agency: ARRS [grant number 118M180].

Notes on contributors

Banu Yetkin Ekren is a Senior Lecturer of Logistics and Supply Chain Management at Cranfield School at Management. Previously, Dr Ekren worked as Associate Professor at Yasar University, Izmir, Turkey. She completed her PhD at University of Louisville, KY, USA, in the Department of Industrial Engineering. She was the recipient of the best PhD dissertation award when she graduated in 2009. Her dissertation focused on design and analysis of automated warehouses by using analytical and simulation modelling. Dr Ekren also completed a one-year postdoctoral research in the period of 2011–2012 for a funded contract project with the Defense Logistics Agency (DLA), PA, USA. She developed simulation models for the DLA's largest warehouse to improve its operating performance.

Bartu Arslan is a PhD Candidate at Eindhoven University of Technology (TU/e) in the Industrial Engineering and Innovation Sciences (IE&IS) department working on the project titled AI-Based Replenishment and Order Fulfillment Strategies for Omnichannel Supply Chains. This project is a part of the AI Planner of the Future research programme, supported by the European Supply Chain Forum (ESCF). His current research focuses on applying Deep Reinforcement Learning method for supply chain problems. Bartu Arslan obtained his master's degree in Industrial Engineering at Yasar University (Turkey) in 2021 and his bachelor's degree in Industrial Engineering at Yasar University in 2019. During his master's and last year of his bachelor's, he was a research fellow for a project supported by the Turkish Scientific Council (TUBITAK).

Data availability statement

The authors confirm that the data supporting the findings of this study are available within the article.

References

Allied Market Research. 2020. *Automated Storage and Retrieval System Market*. https://www.alliedmarketresearch.com/automated-storage-and-retrieval-system-market-A06282.

Carlo, H., and I. Vis. 2012. "Sequencing Dynamic Storage Systems with Multiple Lifts and Shuttles." *International Journal of Production Economics* 140: 844–853. doi:10.1016/j.ijpe.2012.06.035.

Choudhary, A. 2019. *A Hands-On Introduction to Deep Q-Learning using OpenAI Gym in Python*. https://www.

analyticsvidhya.com/blog/2019/04/introduction-deep-q-learning-python/.

Dou, J., C. Chen, and P. Yang. 2015. "Genetic Scheduling and Reinforcement Learning in Multirobot Systems for Intelligent Warehouses." *Mathematical Problems in Engineering* 2015: 597956. doi:10.1155/2015/597956.

Eder, M. 2019. "An Analytical Approach for a Performance Calculation of Shuttle-Based Storage and Retrieval Systems." *Production and Manufacturing Research* 7 (1): 255–270. doi:10.1080/21693277.2019.1619102.

Ekren, B. Y. 2017. "Graph-based Solution for Performance Evaluation of Shuttle-Based Storage and Retrieval System." *International Journal of Production Research* 55 (21): 6516–6526. doi:10.1080/00207543.2016.1203076.

Ekren, B. Y. 2020a. "A Simulation-Based Experimental Design for SBS/RS Warehouse Design by Considering Energy Related Performance Metrics." *Simulation Modelling Practice and Theory* 98: 101991. doi:10.1016/j.simpat.2019.101991.

Ekren, B. Y. 2020b. "A Multi-Objective Optimisation Study for the Design of an AVS/RS Warehouse." *International Journal of Production Research*, 1–20. doi:10.1080/00207543.2020.1720927

Ekren, B. Y., and A. Akpunar. 2021. "An Open Queueing Network-Based Tool for Performance Estimations in a Shuttle-Based Storage and Retrieval System." *Applied Mathematical Modelling* 89: 1678–1695. doi:10.1016/j.apm.2020.07.055.

Ekren, B. Y., A. Akpunar, Z. Sari, and T. Lerher. 2018. "A Tool for Time, Variance and Energy Related Performance Estimations in a Shuttle-Based Storage and Retrieval System." *Applied Mathematical Modelling* 63: 109–127. doi:10.1016/j.apm.2018.06.037.

Ekren, B. Y., and S. S. Heragu. 2010. "Approximate Analysis of Load-Dependent Generally Distributed Queuing Networks with low Service Time Variability." *European Journal of Operational Research* 205 (2): 381–389. doi:10.1016/j.ejor.2010.01.022.

Ekren, B. Y., and S. S. Heragu. 2011. "Simulation Based Performance Analysis of an Autonomous Vehicle Storage and Retrieval System." *Simulation Modelling Practice and Theory* 19 (7): 1640–1650. doi:10.1016/j.simpat.2011.02.008.

Ekren, B. Y., S. S. Heragu, A. Krishnamurthy, and C. J. Malmborg. 2010. "Simulation Based Experimental Design to Identify Factors Affecting Performance of AVS/RS." *Computers and Industrial Engineering* 58 (1): 175–185. doi:10.1016/j.cie.2009.10.004.

Ekren, B. Y., S. Heragu, A. Krishnamurthy, and C. Malmborg. 2013. "An Approximate Solution for Semi-Open Queuing Network Model of an Autonomous Vehicle Storage and Retrieval System." *IEEE Transactions On Automation Science and Engineering* 10: 205–215. doi:10.1109/TASE.2012.2200676.

Ekren, B. Y., S. S. Heragu, A. Krishnamurthy, and C. J. Malmborg. 2014. "Matrix-Geometric Solution for Semi-Open Queuing Network Model of Autonomous Vehicle Storage and Retrieval System." *Computers and Industrial Engineering* 68 (1): 78–86. doi:10.1016/j.cie.2013.12.002.

Ekren, B. Y., Z. Sari, and T. Lerher. 2015. "Warehouse Design Under Class-Based Storage Policy of Shuttle-Based Storage and Retrieval System." *IFAC-PapersOnLine* 48 (3): 1152–1154. doi:10.1016/j.ifacol.2015.06.239.

Gazori, P., D. Rahbari, and M. Nickray. 2020. "Saving Time and Cost on the Scheduling of Fog-Based IoT Applications Using Deep Reinforcement Learning Approach." *Future Generation Computer Systems* 110: 1098–1115. doi:10.1016/j.future.2019.09.060.

Ha, Y., and J. Chae. 2018a. "Free Balancing for a Shuttle-Based Storage and Retrieval System." *Simulation Modelling Practice and Theory* 82: 12–31. doi:10.1016/j.simpat.2017.12.006.

Ha, Y., and J. Chae. 2018b. "A Decision Model to Determine the Number of Shuttles in a Tier-to-Tier SBS/RS." *International Journal of Production Research* 57: 1–22. doi:10.1080/00207543.2018.1476787.

Hao, H., J. Xiaoliang, H. Qixuan, F. Shifeng, and L. Kuo. 2020. "Deep Reinforcement Learning Based AGVs Real-Time Scheduling with Mixed Rule for Flexible Shop Floor in Industry 4.0." *Computers & Industrial Engineering* 149 (2020): 106749. doi:10.1016/j.cie.2020.106749.

Jerman, B., B. Y. Ekren, M. Küçükyaşar, and T. Lerher. 2021. "Simulation-Based Performance Analysis for a Novel AVS/RS Technology with Movable Lifts." *Applied Sciences (Switzerland)* 11 (5): 1–16. doi:10.3390/app11052283.

Küçükyaşar, M., B. Ekren, and T. Lerher. 2021. "Cost and Performance Comparison for Tier-Captive and Tier-to-Tier SBS/RS Warehouse Configurations." *International Transactions in Operational Research* 28 (4): 1847–1863. doi:10.1111/itor.12864.

Law, A. M. 2015. *Simulation Modeling & Analysis*. New York: McGraw-Hill.

Lerher, T. 2015. "Travel Time Model for Double-Deep Shuttle-Based Storage and Retrieval Systems." *International Journal of Production Research* 54: 1–22. doi:10.1080/00207543.2015.1061717.

Lerher, T. 2018. "Aisle Changing Shuttle Carriers in Autonomous Vehicle Storage and Retrieval Systems." *International Journal of Production Research* 56 (11): 3859–3879.

Lerher, T., B. Y. Ekren, G. Dukic, and B. Rosi. 2015b. "Travel Time Model for Shuttle-Based Storage and Retrieval Systems." *The International Journal of Advanced Manufacturing Technology* 78: 1705–1725. doi:10.1007/s00170-014-6726-2.

Lerher, T., B. Y. Ekren, Z. Sari, and B. Rosi. 2015a. "Simulation Analysis of Shuttle Based Storage and Retrieval Systems." *International Journal of Simulation Modelling* 14 (1): 48–59. doi:10.2507/IJSIMM14(1)5.281.

Lerher, T., B. Y. Ekren, Z. Sari, and B. Rosi. 2016. "Method for Evaluating the Throughput Performance of Shuttle Based Storage and Retrieval Systems." *Tehnicki Vjesnik - Technical Gazette* 23: 715–723. doi:10.17559/TV-20141022121007.

Lerher, T., M. Ficko, and I. Palčič. 2021. "Throughput Performance Analysis of Automated Vehicle Storage and Retrieval Systems with Multiple-Tier Shuttle Vehicles." *Applied Mathematical Modelling* 91: 1004–1022. doi:10.1016/j.apm.2020.10.032.

Li, H., J. Lyu, L. Zhen, and D. Zhuge. 2022. "A Joint Optimisation of Multi-Item Order Batching and Retrieving Problem for low-Carbon Shuttle-Based Storage and Retrieval System." *Cleaner Logistics and Supply Chain* 4, doi:10.1016/j.clscn.2022.100042.

Liu, Z., Y. Wang, M. Jin, H. Wu, and W. Dong. 2021. "Energy Consumption Model for Shuttle-Based Storage and Retrieval Systems." *Journal of Cleaner Production* 282. doi:10.1016/j.jclepro.2020.124480.

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 Analysis* 69 (1): 397–400. doi:10.1016/j.cirp.2020.04.001.

Mao, H., M. Alizadeh, I. Menache, and S. Kandula. 2016. "Resource Management with Deep Reinforcement Learning." In *Hotnets 2016 – Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, 50–56. doi:10.1145/3005745.3005750.

Marchet, G., M. Melacini, S. Perotti, and E. Tappia. 2011. "Analytical Model to Estimate Performances of Autonomous Vehicle Storage and Retrieval Systems for Product Totes." *International Journal of Production Research* 2011. doi:10.1080/00207543.2011.639815.

Marchet, G., M. Melacini, S. Perotti, and E. Tappia. 2013. "Development of a Framework for the Design of Autonomous Vehicle Storage and Retrieval Systems." *International Journal of Production Research* 51. doi:10.1080/00207543.2013.778430.

Marolt, J., N. Kosanić, and T. Lerher. 2022. "Relocation and Storage Assignment Strategy Evaluation in a Multiple-Deep Tier Captive Automated Vehicle Storage and Retrieval System with Undetermined Retrieval Sequence." *International Journal of Advanced Manufacturing Technology* 118: 3403–3420. doi:10.1007/s00170-021-08169-x.

Mnih, V., K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, et al. 2015. "Human-Level Control Through Deep Reinforcement Learning." *Nature* 518 (7540): 529–533. doi:10.1038/nature14236.

Romaine, E. 2020. *Automated Storage & Retrieval System (AS/RS) Types & Uses*. https://www.conveyco.com/automated-storage-and-retrieval-types/.

Scriven, R. 2021. *Robotic Micro-fulfillment Centres (MFC) – Infographic*. Accessed March 14, 2022. https://www.interactanalysis.com/robotic-micro-fulfillment-centres-mfc-infographic/.

Statista. 2021. *Size of the Warehouse Automation Market Worldwide from 2012 to 2026*. https://www.statista.com/statistics/1094202/global-warehouse-automation-market-size/.

Sutton, R. S., and A. G. Barto. 2015. "An Introduction to Reinforcement Learning." In *A Bradford Book The*. doi:10.4018/978-1-60960-165-2.ch004.

Takahashi, K., and S. Tomah. 2020. "Online Optimization of AGV Transport Systems Using Deep Reinforcement Learning." *Bulletin of Networking, Computing, Systems, and Software* 9 (1): 53–57.

Tappia, E., D. Roy, R. de Koster, and M. Melacini. 2017. "Modeling, Analysis, and Design Insights for Shuttle-Based Compact Storage Systems." *Transportation Science* 51 (1): 269–295. doi:10.1287/trsc.2016.0699.

Tong, Z., H. Chen, X. Deng, K. Li, and K. Li. 2020. "A Scheduling Scheme in the Cloud Computing Environment Using Deep Q-Learning." *Information Sciences* 512: 1170–1191. doi:10.1016/j.ins.2019.10.035.

Unctad. 2021. *How COVID-19 Triggered the Digital and e-Commerce Turning Point*. https://unctad.org/news/how-covid-19-triggered-digital-and-e-commerce-turning-point.

Wang, Y., S. Mou, and Y. Wu. 2015. "Task Scheduling for Multi-Tier Shuttle Warehousing Systems." *International Journal of Production Research* 53 (19): 5884–5895. doi:10.1080/00207543.2015.1012604.

Watanabe, M., M. Furukawa, and Y. Kakazu. 2001. "Intelligent AGV Driving Toward an Autonomous Decentralized Manufacturing System." *Robotics and Computer-Integrated Manufacturing* 17 (1–2): 57–64. doi:10.1016/S0736-5845(00)00037-5.

Xue, T., P. Zeng, and H. Yu. 2018. "A Reinforcement Learning Method for Multi-AGV Scheduling in Manufacturing." In *2018 IEEE International Conference on Industrial Technology (ICIT)*, 1557–1561. doi:10.1109/ICIT.2018.8352413.

Yin, Z., J. Liu, and D. Wang. 2022. "Multi-AGV Task Allocation with Attention Based on Deep Reinforcement Learning." *International Journal of Pattern Recognition and Artificial Intelligence* 36 (9): 2252015. doi:10.1142/S0218001422520152.

Zhao, X., Y. Wang, Y. Wang, and K. Huang. 2019. "Integer Programming Scheduling Model for Tier-to-Tier Shuttle-Based Storage and Retrieval Systems." *Processes* 7 (4). doi:10.3390/pr7040223.

Zou, B., X. Xu, Y. Gong, and R. De Koster. 2016. "Modeling Parallel Movement of Lifts and Vehicles in Tier-Captive Vehicle-Based Warehousing Systems." *European Journal of Operational Research* 254 (1): 51–67. doi:10.1016/j.ejor.2016.03.039.