Smart Transaction Picking in Tier-to-tier SBS/RS by Deep *Q*-Learning

Bartu Arslan, Banu Y. Ekren
Department of Industrial Engineering
Yasar University
No:37-39, Bornova, Izmir, Turkey
bartuarslan@yahoo.com, banu.ekren@yasar.edu.tr

Abstract
By the rapid growth of e-commerce, the intralogistics sector is facing new challenges. Intralogistics sector requires more flexible, scalable processes with maximum reliability and availability. They are complicated and interconnected systems, whose all components are required to be perfectly coordinated with each other for optimal functionality. In this work, we study an intralogistics technology, shuttle-based storage and retrieval system (SBS/RS), where shuttles are tier-to-tier. In this novel system design, in an effort to increase shuttle utilization as well as decrease initial investment cost, shuttles are designed in a more flexible travel manner so that they can change their tiers within an aisle by using a separate lifting mechanism. Due to the complexity of such system design as well as aiming to obtain fast transaction process time by the decreased number of shuttles in the system, we implement a Deep *Q*-Learning (DQL) approach to let shuttles select the best transaction to process based on its targets. We compare the performance of the DQL by the average cycle time per transaction performance metric with the other well-known selection rules, First-in-First-Out (FIFO) and Shortest Process Time (SPT). Results show that Deep *Q*-Learning approach produces better results than those FIFO and SPT.

Keywords
SBS/RS, Deep Reinforcement Learning, Deep *Q*-Learning, Simulation, Optimization

1. Introduction
By the Industry 4.0 developments established on collaboration of automated systems within facilities, warehouse managers are also interested in investing enterprise-wide automation. For instance, they are increasingly deploying goods with RFID tags helping the transparency and traceability of products as well as implementation of robotic technologies to transform material handling technology.

The Global Automated Material Handling (AMH) Market was valued at USD 50,544.6 million in 2019 and is expected to reach USD 101,487.3 million by 2025 (ReportLinker 2020). For instance, the Census of Fatal Occupational Injuries, conducting statistics under the U.S. Bureau of Labor Statistics, has established that workplace injuries have decreased by 25% in 10 years, due to the utilization of automated technologies in the workplace helping to boost the automated material handling market (ReportLinker 2020).

The automated storage and retrieval system (AS/RS) market size was valued at $7,351 million in 2019, and is projected to reach $12,928 million by 2027 (Allied Market Research 2020). AS/RSs are utilized for inventory management systems in manufacturing centers, distribution facilities, and warehouses. They are developed on computer-controlled systems for autonomous storage and retrieval of loads from one location to another. These systems are used for high-volume loads facilities for quick, accurate, reliable, and low-cost solutions.

The AS/RS technology providers aim to develop the system designs providing maximized storage capacity, increased performance, reduced energy costs, and increased inventory accuracy and customer service. For instance, a mini-load AS/RS technology that is market by Dematic Group, provides ultra-high-speed load handling referred as shuttle-based storage and retrieval system (SBS/RS) mostly utilized in distribution centers and raw material stores that are specially designed for ultra-high-speed load handling (Dematic n.d.). This design is referred as tier-captive SBS/RS throughout this work.

Figure 1 shows the current and the estimated market share of AS/RS technologies based on their types. It is observed that, all AS/RS types are estimated to have increased market by 2027. Note that, mini-load's market size is at the second order among all types.

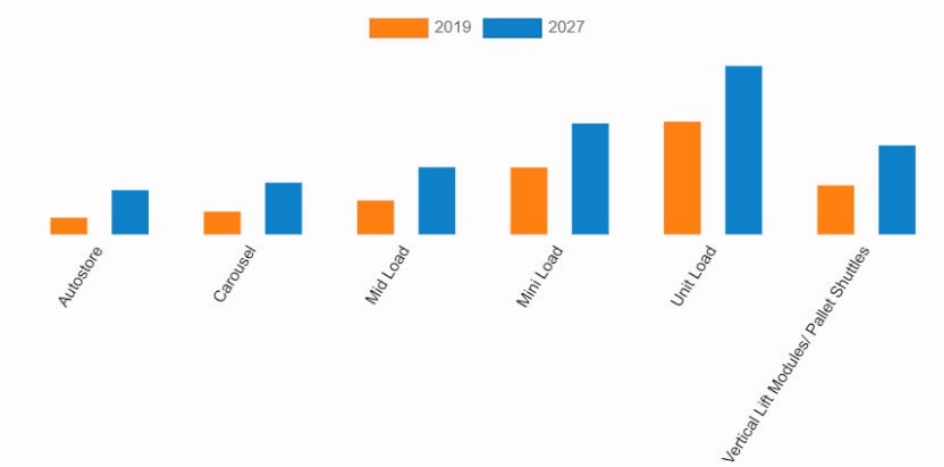

Figure 1. Current and estimated AS/RS market share based on types (source: Allied Market Research, 2020)

This paper studies SBS/RS technology by implementing a DQL operational approach in transaction processing. The initial design of SBS/RS market by Dematic group has ultra-high speed due to excess amount of shuttles running in the system. Specifically, in each tier of an aisle of the warehouse, there is a dedicated shuttle which can perform two-way horizontal travel through a single axis. There is a lifting mechanism in each aisle providing vertical travel for loads. Because there is a single lifting mechanism in each aisle, while there are multiple shuttles dedicated at each tier, average utilization of shuttles is very low compared to average utilization of lifts. In an effort to balance those utilizations, we propose a novel SBS/RS design where the number of shuttles is decreased within an aisle, so that shuttles can travel between tiers. We refer this novel system design as tier-to-tier SBS/RS. Figure 2 shows a single aisle view from side and top views for the proposed tier-to-tier SBS/RS. The storage transactions arrive at the I/O point by a conveyor system. There is a lifting mechanism referred as Lift 1 in Figure 2, to transfer the loads between tiers and I/O point. If the transaction is addressed at the first tier, Lift 1 is not utilized. There are two types of transaction requests in the system, storage and retrieval. In the case of storage, Lift 1 carries the transaction from the I/O point to destination tier. In the case of retrieval, Lift 1 travels to the load tier address to pick it up from the buffer location to bring it at the I/O point. Meanwhile, Lift 2 may transfer the shuttle to a different tier than its current tier, if the shuttle requires to change its current tier. Note that Lift 1 mechanism has two loads capacity which can work independently.

Compared to the traditional tier-captive SBS/RS, in tier-to-tier SBS/RS, a shuttle tends to complete longer travel time due to the ability of changing tiers. This may also cause an increase in average cycle time per transaction performance metric. Here, our motivation to implement a machine learning algorithm, DQL, is to prevent so much increase in that cycle time per transaction performance metric.

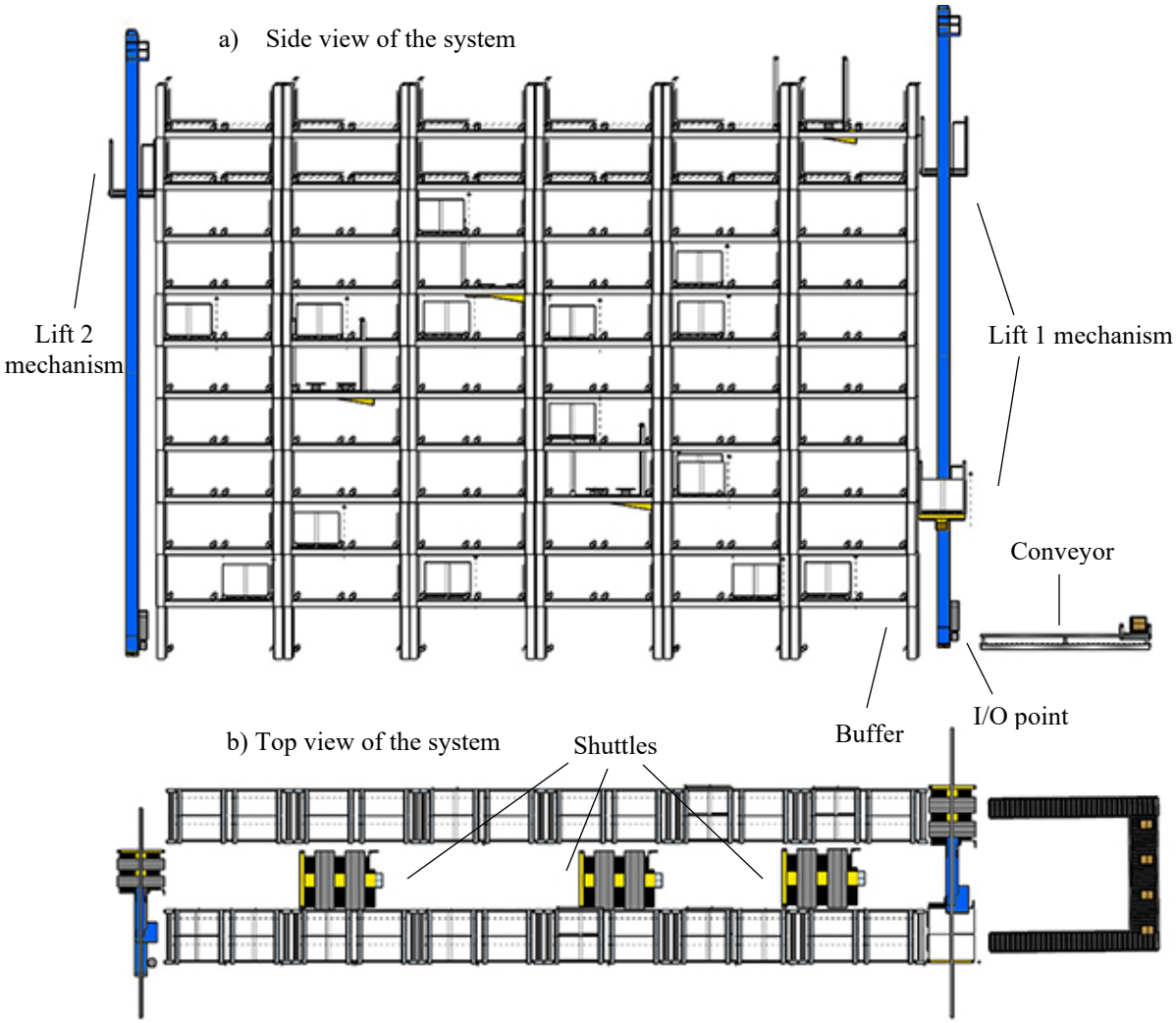

Figure 2. An aisle view of the proposed tier-to-tier SBS/RS design.

1.1 Objectives

The objectives of this study can be summarized to be:

- Searching an alternative SBS/RS design balancing the utilizations of shuttles and lifts compared to the traditional tier-captive SBS/RS.
- In the proposed tier-to-tier SBS/RS, since the number of shuttles in the system is decreased, searching a smart transaction selection algorithm resulting with decreased average cycle time per transaction performance metric in the system.

Once again, we propose a novel tier-to-tier SBS/RS design where there is decreased number of shuttles in the system compared to its tier-captive design. Here, shuttles can travel between tiers. Due to having decreased number of shuttles in the system, there might be higher shuttle utilizations and lower investment costs in this design. In an effort to improve the average cycle time per transaction performance metric, we propose an intelligent transaction selection algorithm developed on Deep *Q*-learning.

2. Literature Review

In this section, the literature review is divided into three parts. First, current studies on tier-captive, and second, tier-to-tier SBS/RS studies are summarized. Last, Deep *Q*-learning based studies are examined.

Most of the current studies in literature focus on tier-captive systems. Marchet et al. (2011) model a tier-captive SBS/RS design and applies an open queueing network modelling approach. In their later work, Marchet et al. (2013) study the design trade-offs in the same design, by using simulation modelling. This study shows that decreasing the number of aisles decreases setup costs in the system. Carlo and Vis (2012) study a novel tier-captive SBS/RS design. In their design, two lifts that cannot pass each other are utilized and the system is compared to a system with single lifting mechanism. Lerher (2015) studies a double-deep tier-captive SBS/RS, resulting in more efficient use of floor space. Ekren et al. (2015) study class-based storage policy for tier-captive SBS/RS. Ekren (2017) studies design trade-offs for a tier-captive SBS/RS for different warehouse designs.

Recent studies are performed by Ekren and Akpunar (2020), Ekren (2021a) and Ekren (2021b). Ekren and Akpunar (2020) study a queuing network-based tool also presented in a web site. Ekren (2021a) presents a simulation based experimental design analysis to identify significant factors affecting performance of a tier-captive SBS/RS. Ekren (2021b) studies a multi-objective optimization procedure for design of a tier-captive SBS/RS.

Ha and Chae (2018a) study tier-to-tier SBS/RS for the first time in the literature. In their design, a single lift carries both loads and shuttles. They compare the traditional system with free balancing. Later on, Ha and Chae (2018b) develop a decision model that determines the number of shuttles in a tier-to-tier SBS/RS. Zhao et al. (2019) develop an integer programming scheduling model for a tier-to-tier SBS/RS. The results suggest that operation costs and capital investments are reduced significantly. Küçükyaşar et al. (2020) compare tier-captive and tier-to-tier SBS/RS in their recent work. Performance metrics such as investment costs, throughput rates and average energy consumptions are compared, and the results suggest that tier-to-tier system could be better with regard to these performance metrics.

Mao et al. (2016) implement deep reinforcement learning (DRL) for resource management problems in computer systems and networks. The results show that DRL algorithms are feasible for application to large-scale systems. Gazori et al. (2020) focus on the task scheduling of fog-based IoT applications and they aim to minimize the computation costs and service delays. They use Double Deep *Q*-learning (DDQL) and the results show that DDQL works better than some of the basic algorithms in terms of performance metrics such as computation cost, energy consumption and service delay. Tong et al. (2020) propose a Deep *Q*-learning task scheduling (DQTS) algorithm for a dynamic task-scheduling problem of cloud computing. The results indicate that compared to standard algorithms, DQTS performs better. Takahashi and Tomah (2020) propose a DRL method that controls multiple AGVs. The results suggest that the proposed algorithm finds optimal or near-optimal solutions and has a very high performance in new environments.

To the best of our knowledge, it is the first time, we implement a reinforcement learning algorithm for smartly operation of an SBS/RS design. By the proposed study, different from the literature works, we implement an adaptive and training-based algorithm for the intelligent shuttle agents to let them select the waiting transactions according to the best reward gain.

3. Methods

3.1 *Q*-Learning

Reinforcement Learning (RL) is a machine learning approach utilizing intelligent agent to take actions in an environment to maximize the cumulative rewards. *Q*-Learning is a model-based reinforcement learning algorithm. The environment provides information as state and reward. State refers to the current situation of the environment. Rewards are earned after an action. These reward values are used to update the *Q*-table using Bellman Equation. The agent learns by taking actions iteratively in the environment.

$$Q(s,a) = (1-\alpha)\, Q(s,a) + \alpha\, [R(s,a) + \gamma\, max\, Q(s',a')] \quad (1)$$

In Eq. (1), $Q(s,a)_t$ is the *Q*-value of taking action *a* in state *s*, α is the learning rate, γ is the discount rate and *R(s, a)* is the reward of taking the action *a*, in state *s*.

While selecting an action, epsilon-greedy approach is applied. This is a common approach used to manage tradeoff between exploration and exploitation. Each time an action is selected, a random number is created between 0 and 1. If this number is less than an epsilon value ϵ, an action is chosen randomly. Otherwise, the action with maximum *Q*-value is selected. In each step, which means choosing an action, the *Q*-table is updated until it becomes stable. By the

increase of number of states and actions, this approach may become infeasible to implement due to drastic increase of the *Q*-table. This causes an increase for data and hence, time to train agents to learn them. This time, deep *Q*-learning approach might be a good way for dealing with such large state-action relationships, since it uses function approximation. By that function approximation, intelligent agent does not require to experience each state-action case to learn, instead it tends to estimate by a good function fitting. We explain this methodology in Section 3.2.

3.2 Deep *Q*-Learning

In Deep *Q*-Learning (DQL), we consider a deep reinforcement learning approach where neural network method is utilized to approximate the *Q*-value function. State is given to the network as input and the outputs are the *Q*-values of all actions available for that state. In Figure 3, a visual representation for the system is shown. As in *Q*-learning, the agents observe the state *s* from the environment, feeds it to the network as input. As an output, *Q*-values are provided from the network. While selecting an action, the same epsilon-greedy approach is used. After the action *a* is completed, the reward *r* is provided from the environment and it is stored to update the network weights.

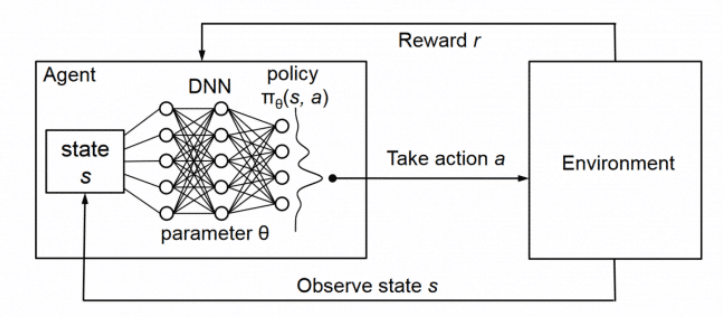

Figure 3. Visual representation of Deep *Q*-Learning (source: Mao et al., 2016)

In order to have more stability, two neural networks are used for the learning process. We create a target and a prediction network. Every *C* iterations, prediction network parameters are copied into the target network. The weights of the networks are updated considering loss function by using mean-squared error (MSE). In Figure 4, the concept is shown visually. Also, the target and prediction functions are shown.

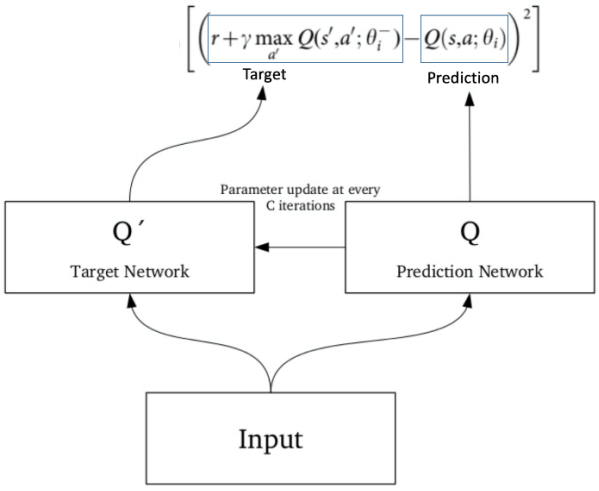

Figure 4. Target Network (source: Choudhary, 2019)

Unlike the *Q*-Learning approach, instead of updating the *Q*-table at each step, {state, action, reward} data is stored in batches. After reaching the predefined batch size, a random {state, action, reward} is selected and used to train the network. This process is called experience replay. This algorithm is proposed in a study of Mnih et al. (2015). The full algorithm is presented below.

Initialize action-value and target action-value functions Q and $\hat{Q}$ with random weights θ
Initialize batch size n, learning rate α, discount rate γ, epsilon ϵ, epsilon decay rate ϵ^{dec}, epsilon minimum value ϵ_{min}
For episode = 1, M **do**
Initialize state s_t
With probability ϵ, select random action a_t, otherwise select a_t = argmax $Q(s, a)$
Execute action a_t, observe reward r_t
Store transition (s_t, a_t, r_t, s_{t+1})
Sample random tuple from transition tuples
$Set\ y_j = \begin{cases} r_j\ , if\ episode\ terminates\ at\ step\ j+1 \\ r_j + \max \hat{Q}(s', a')\ , otherwise \end{cases}$
Perform gradient descent step on $\left(y_j - Q(s, a)\right)^2$
Update ϵ as $\epsilon = \epsilon * \epsilon^{dec}$ if $\epsilon > \epsilon_{min}$, otherwise ϵ_{min}
Every C steps, $\hat{Q} = Q$
End For

This algorithm is applied by simulating the proposed system design. Keras is used to implement the Deep Q-networks to the simulation model. We used Adam as the optimizer, $\alpha = 0.001$, $\gamma = 0.4$, $\epsilon = 1$, $\epsilon^{dec} = 0.999$, $\epsilon_{min} = 0.01$, $n = 64$.

The simulation details, model assumptions along with the notations used are summarized in the following section.

3.3 Simulation Model

Simulation modelling is applied due to complexity of the proposed tier-to-tier SBS/RS design. The model is completed in the Python library, SimPy.

Actions are picked whenever a shuttle is available, and the transaction queue is not empty. Those details are given in Section 3.6. Whenever a shuttle picks a waiting transaction in its queue, this information is sent to other resources. Other resources are: Lift 1 and Lift 2 mechanisms. They also start their travels with the shuttle movement simultaneously if they are available. If the shuttle tends to change its current tier, Lift 2 is informed. Remember that Lift 2 mechanism is dedicated for travels of shuttles between tiers. Then, Lift 2 travels to the shuttle's current tier, and picks it up to travel to the destination tier. Meanwhile, unless the transaction tier is the first tier, Lift 1 also starts its travel. If the transaction tier is the first tier, then Lift 1 is not utilized. All the assumptions considered for this simulation model are provided below:

- Mean arrival rates for storage and retrieval transaction follow Poisson distribution and they are equal, $\lambda_S = \lambda_R$.
- Transactions addresses are assigned randomly.
- Transactions arrive at the I/O location and enter a single queue.
- The loading and unloading times are ignored.
- Lifts apply First-in-First-out transaction selection rules.
- Shuttles and lifts wait at their last process points after releasing the entities.
- Maximum velocity of shuttles and lifts are set to 2 *m/s*.
- Acceleration and deceleration values of shuttles and lifts are set to 2 m/s^2.
- The distance between two adjacent tiers is set to 0.35 *m*.
- The distance between two adjacent bays is set to 0.5 *m*.
- Five independent replications are completed for the FIFO and SPT models.

The notations along with their units are summarized in Table 1. We detail the studied DQL in the following sections.

Table 1. The notations used for the model

Notation	Description	Unit
VS_{max}	Maximum velocity of the shuttles	m/s
as_v	Acceleration value of the shuttles	m/s^2
ds_v	Deceleration value of the shuttles	m/s^2
VL_{max}	Maximum velocity of the lifts	m/s
al_v	Acceleration value of the lifts	m/s^2
dl_v	Deceleration value of the lifts	m/s^2
L	Distance between two adjacent bays	m
H	Distance between two adjacent tiers	m
US_{avg}	Average shuttle utilization	%
UL_{avg}	Average utilization of Lift 1	%
ULS_{avg}	Average utilization of Lift 2	%
C_{avg}	Average cycle time of transactions	s
F_{avg}	Average flow time of transactions	s
$1 / (\lambda_S+\lambda_R)$	Mean inter-arrival time for transactions	s
T	Number of tiers in a single aisle	
B	Number of total bays in a single side and tier of an aisle	
S	Number of total shuttles in the system	

3.4 Agents

Here, the shuttles are treated as agents in the system so that they can make smart transaction selection according to estimated reward obtained by the DQL. Whenever a shuttle becomes available, it selects a transaction and a lift to perform the task.

3.5 State Space

In the studied DQL, states are defined to be:
$S(k)$ = (Current tier of available shuttle (k), current bay of available shuttle (k), current tier of Lift 1, availability of Lift 1, current tier of Lift 2, availability of Lift 2)

Note that the tiers can take values from 1 to T, bays are integers from 1 to B, availability is a binary number {0,1} where 0 represents no availability and 1 represents availability.

3.6 Action Space

Actions represent the attributes of transactions waiting in shuttle queue to be processed. The action space is defined to be:

$A(k)$ = (Transaction's tier address, transaction's bay address, transaction's type, selected Lift 1 table). For instance, when selected action is a storage transaction whose address is 3rd tier and 10th bay, with the left-hand side Lift 1 table, then this action is represented by: (3, 10, 0, 1). Transaction type 0 represents storage and 1 represents retrieval.

Since all transactions representing actions are not always available, the shuttle picks from available actions. Also, to avoid collision, a shuttle cannot pick a transaction whose address is at a tier where another shuttle exists there, or another shuttle is heading to that tier. The available actions space is defined by the following:

- There must be at least one transaction waiting in the queue with corresponding tier, bay and type.
- The transaction must not be at another shuttle's tier or its destination tier.

3.7 Immediate Reward Function

Note that the goal of this study is to minimize average cycle time of a transaction in the system. Cycle time is the time between when a transaction is created until it is disposed from the system. Namely, it also includes waiting times of transactions in queues. We consider the flow time of transactions in the reward function. Here, flow time is the time between when a transaction is selected by a shuttle until it is disposed from the system. In another word, it only includes waiting times of transactions in lift queues. Currently, since we did not include any queue related state

definition in DQL, rewards might not correlate with the states. In order to achieve our goals, immediate rewards are explained as in Eq. (2)-(4):

$$MIFV_t = \frac{1}{\max(flowtime)} \tag{2}$$

$$MAFV_t = \frac{1}{\min(flowtime)} \tag{3}$$

$$R(s, a)_t = \frac{\frac{1}{flowtime(a)} - MIFV_t}{MAFV_t - MIFV_t} * 100 \tag{4}$$

The reward shown in Eq. (4), is normalized by taking multiplicative inverse of flow times. This idea is motivated from a study also aiming to minimize the time where the results show good success (Gazori et al., 2020). Eq. (2) and Eq. (3) calculates the minimum and maximum flow time values until time *t*, respectively. Then, we apply a standard normalization formula to current flow time value, which is *1/flowtime(a)*. Since these values are very small, to track the improvement of the learning algorithm easier, this value is multiplied by 100.

4. Data Collection

For the system inputs, we use the data from Ekren et al. (2018), Ekren (2017), Lerher et al. (2015), Lerher et al. (2015), where parameters are defined to be: Poisson distribution for arrivals of transactions, maximum velocity of shuttles and lifts that can reach and acceleration, deceleration values of velocities are defined to be 2 m/s and 2 m/s^2, respectively. Length of two adjacent bays and height of two adjacent tiers are 0.5 m. and 0.35 m. respectively. We have 5 tiers, 25 bays and 2 shuttles for our experiments. Those parameters are summarized below:

- $VS_{max} = VL_{max} = 2\ m/s$
- $as_v = ds_v = al_v = dl_v = 2\ m/s^2$
- $L = 0.5\ m$
- $H = 0.35\ m$
- $T = 5$
- $B = 25$
- $S = 2$

5. Results and Discussion

5.1 Numerical Results

After simulating the system in Python SimPy with Deep *Q*-learning algorithm, six scenarios are experimented as shown in Table 2. According to that, we compare three transaction selection algorithms (i.e., FIFO, SPT, and DQL) with two different arrival rate scenarios. Here, in FIFO algorithm, the transactions are selected according to their arrival times. Namely, first arriving transaction to the system is given priority in shuttle's selection. In shortest-process-time (SPT) algorithm, the waiting transactions are selected according to their estimated cycle times. Namely, the one having the least estimated cycle time is given priority in processing.

We consider five number of tiers ($T = 5$), twenty-five bays ($B = 25$) and two shuttles for the warehouse design. After, experimenting the six scenarios, it is observed that Deep *Q*-Learning approach outperforms both FIFO and SPT rules in terms of average cycle time per transaction performance metric. The results are summarized in Table 2. The average cycle time and flow time values are given in 95% confidence interval.

Table 2. Conducted experiments and their results

Algorithm	UL_{avg}	ULS_{avg}	US_{avg}	C_{avg}	F_{avg}	$1 / (\lambda_S + \lambda_R)$
FIFO	41%	53%	88%	56.65 ± 1.89	13,83 ± 0.006	7.6
SPT	41%	49%	85%	39.57 ± 0.51	13.42 ± 0.006	7.6
DQL	38%	45%	80%	34.2	12.41	7.6
FIFO	43%	56%	93%	127.62 ± 6.06	13.87 ± 0.013	7.2
SPT	43%	50%	89%	49.19 ± 0.86	13.3 ± 0.013	7.2
DQL	39%	46%	84%	40.75	12.33	7.2

From Table 2, it can be observed that DQL outperforms the FIFO rule in two arrival rate scenarios. The least average cycle time per transaction is always obtained by the DQL approach. In addition, lower utilization rates are obtained, meaning more transactions could be processed in the same period. Although, no queueing information is given about the environment, obtaining better results by DQL is promising for the future works where we could involve more information about the environment.

5.2 Graphical Results

The results are summarized by Figure 5 and Figure 6. Figure 5 compares the three selection approaches results based on the average cycle time per transaction performance metric values. It can be observed that FIFO produces the largest cycle time value. DQL produces the best results under both arrival rate scenarios.

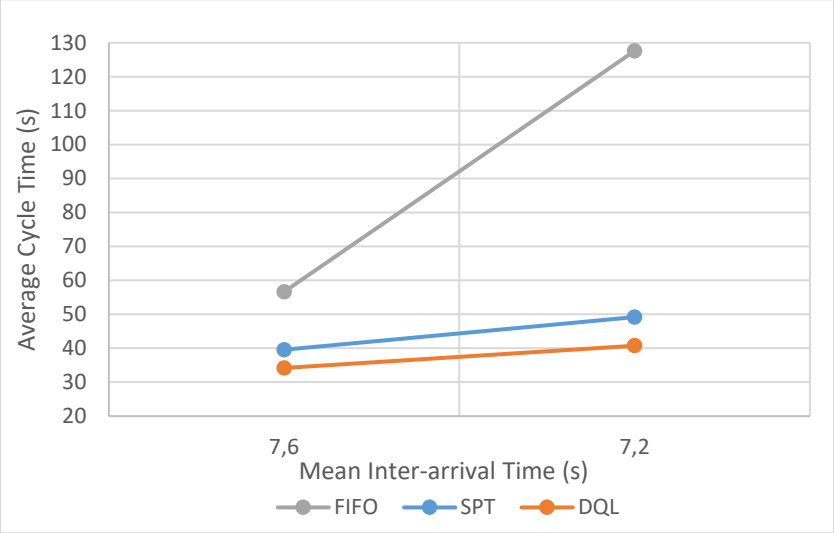

Figure 5. Comparison of three algorithms under average cycle time per transaction

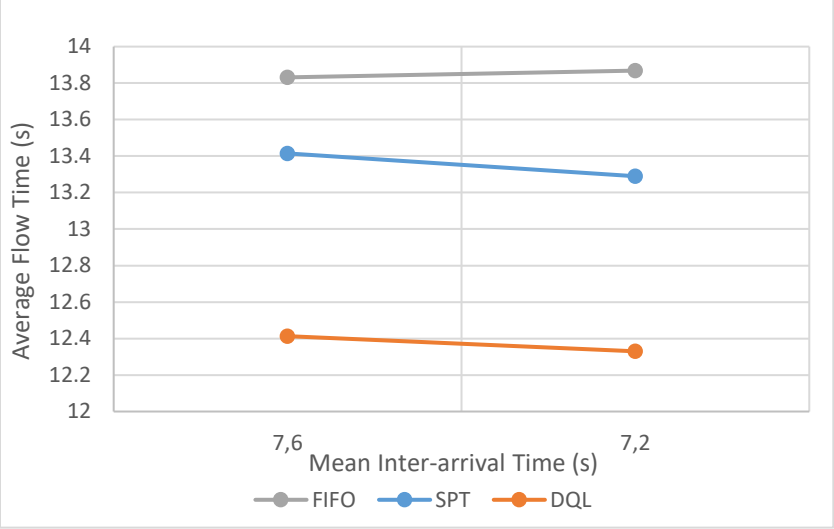

Figure 6. Comparison of three algorithms under mean inter-arrival times

Figure 6 compares the three selection approaches results based on the average flow time per transaction performance metric values. When we compare the average cycle time per transaction performance metrics based on the selection rules, DQL makes an improvement of 40% and 68% compared to FIFO rule, under inter-arrivals of 7.6 and 7.2

seconds, respectively. When we compare the DQL with SPT, DQL makes an improvement of 14% and 17%, under inter-arrivals of 7.6 and 7.2 seconds, respectively.

The results indicate that the DQL algorithm works better even under high utilization environment. This may lead even better results under an intense system condition, for example when the number of tiers and shuttles gets larger.

5.3 Validation

The proposed tier-to-tier SBS/RS is a novel system design. Hence, we do not have a real system to validate the model by comparison. Thus, the validation and verification of the model are completed by debugging the codes and making logical changes and tracking the results accordingly. For instance, we increase arrivals, velocities, etc. or change number of shuttles in the system to test whether or not the results change as expected.

6. Conclusion

This paper studies a Deep *Q*-Learning approach for transaction selection in a tier-to-tier shuttle-based storage and retrieval system. Our aim is to implement a smart selection algorithm for shuttles so that average cycle time of a transaction is decreased. We treat the shuttles as intelligent agents so that they can be trained from their past experiences and make smart decisions in transaction selection. The state information is received from the environment in real time and estimated flow time is calculated by using the DQL algorithm. Based on the results, the best alternative is selected to be processed.

The main motivation of this work is to implement an algorithm that is adaptive to changing conditions and able to learn from its experience. The algorithm iterates over time to learn the best possible assignment of transaction to shuttle. The system is simulated in a Python library, SimPy and Deep *Q*-Network is utilized by using the Keras library.

We compare the DQL algorithm's results with two traditional selection rules, FIFO and SPT. The results show that, for the average cycle time performance metric, 40% improvement is obtained by DQL, compared to FIFO rule, under inter-arrival time of 7.6 sec. Under inter-arrival time of 7.2 sec., this improvement is 68%. When the results are compared with the SPT rule, it is 14% and 17% improvement under inter-arrival time of 7.6 sec. and 7.2 sec., respectively. The results indicate that as the arrival rates increases, namely under higher utilizations of shuttles, the DQL algorithm works better.

As a future work, queue related information of transactions could be included in state definition. Hence, reward function may become more precise contributing better selection policy. Those algorithms can be experimented under different warehousing designs such as number of tiers, racks, shuttles, etc.

References

Allied Market Research., Automated Storage and Retrieval System Market, Available: https://www.alliedmarketresearch.com/automated-storage-and-retrieval-system-market-A06282, January 25, 2021.

Carlo, H., and Iris, V., "Sequencing Dynamic Storage Systems with Multiple Lifts and Shuttles." *International Journal of Production Economics* 140: 844–53, 2012.

Choudhary, A., A Hands-On Introduction to Deep Q-Learning Using OpenAI Gym in Python, Available: https://www.analyticsvidhya.com/blog/2019/04/introduction-deep-q-learning-python/, January 25, 2021.

Dematic. Dematic Multishuttle, Available: https://www.dematic.com/en/products/products-overview/storage-systems/dematic-multishuttle/, January 25, 2021.

Ekren, B. Y., Graph-based solution for performance evaluation of shuttle-based storage and retrieval system, *International Journal of Production Research*, vol. 55, no. 21, pp. 6516-6526, 2017.

Ekren, B. Y., Akpunar, A., Sari, Z., and Lerher, T., A tool for time, variance and energy related performance estimations in a shuttle-based storage and retrieval system, *Applied Mathematical Modeling*, vol. 63, pp. 109-127, 2018.

Ekren, B.Y., A Simulation-based experimental design for SBS/RS warehouse design by considering energy related performance metrics, *Simulation Modelling Practice and Theory*, in press, https://doi.org/10.1016/j.simpat.2019.101991, 2021a.

Ekren, B.Y., A multi-objective optimisation study for the design of an AVS/RS warehouse, International Journal of Production Research, in press, https://doi.org/10.1080/00207543.2020.1720927, 2021b.

Ekren, B. Y., Akpunar, An open queuing network-based tool for performance estimations in a shuttle-based storage and retrieval system, *Applied Mathematical Modelling* 89, 1678-1695, 2020.

Ekren, B. Y., Akpunar, A., Sari, Z., and Lerher, T., Warehouse design under class-based storage policy of shuttle-based storage and retrieval system, *IFAC-PapersOnLine*, 48 (3), 1152-1154, 2015.

Gazori, P., Dadmehr R., and Mohsen N., Saving Time and Cost on the Scheduling of Fog-Based IoT Applications Using Deep Reinforcement Learning Approach., *Future Generation Computer Systems* 110(xxxx): 1098–1115, 2020.

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

Ha, Y., and Chae J., Free Balancing for a Shuttle-Based Storage and Retrieval System., *Simulation Modelling Practice and Theory* 82: 12–31, 2018b.

Küçükyaşar, M., Ekren, B. Y., and Lerher, T., Cost and Performance Comparison for Tier-captive and Tier-to-tier SBS/RS Warehouse Configurations., *International Transactions in Operational Research*, 2020.

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

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

Lerher, T., Ekren, B. Y., Goran D., and Bojan R, Travel Time Model for Shuttle-Based Storage and Retrieval Systems., *The International Journal of Advanced Manufacturing Technology* 78, 2015.

Mao, H., Mohammad A., Ishai M., and Srikanth K., Resource Management with Deep Reinforcement Learning. *HotNets 2016 - Proceedings of the 15th ACM Workshop on Hot Topics in Networks*: 50–56, 2016.

Marchet, G., Marco M., Sara P., and Elena T., Analytical Model to Estimate Performances of Autonomous Vehicle Storage and Retrieval Systems for Product Totes., *International Journal of Production Research*, 2011.

Marchet, G., Marco M., Sara P., and Elena T., Development of a Framework for the Design of Autonomous Vehicle Storage and Retrieval Systems., *International Journal of Production Research* 51, 2013.

Mnih, V. et al., Human-Level Control through Deep Reinforcement Learning., *Nature* 518(7540): 529–33, 2015.

ReportLinker, Automated Material Handling (AMH) Market - Growth, Trends, and Forecast (2020 - 2025), Available: https://www.reportlinker.com/p05815026/Automated-Material-Handling-AMH-Market-Growth-Trends-and-Forecast.html?utm_source=GNW, January 25, 2021.

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

Tong, Z. et al., A Scheduling Scheme in the Cloud Computing Environment Using Deep Q-Learning., *Information Sciences* 512: 1170–91, 2020.

Zhao, X., Yanyan W., Yunge W., and Ke H., Integer Programming Scheduling Model for Tier-to-Tier Shuttle-Based Storage and Retrieval Systems., *Processes* 7(4), 2019.

Acknowledgements

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

Biographies

Bartu Arslan is an M.Sc. student in the Department of Industrial Engineering. He graduated from the Department of Industrial Engineering, at Yasar University in Izmir, Turkey. He worked as a fellow under Dr. Banu Y. Ekren's supervision for a research project funded by TUBITAK focusing on agent-based simulation modelling of an automated warehousing system. His research interests include machine learning, applications of simulation-optimization in supply chain, logistics, production, and warehousing.

Banu Y. Ekren is a full time academic in the department of Industrial Engineering, at Yasar University in Izmir, Turkey. She got her Ph.D., from University of Louisville in the Dept. of Industrial Engineering, in Kentucky, USA. Her research focuses on future of education, warehousing, stochastic and simulation modelling, modelling supply chains, simulation-based optimization, and design and analysis of automated warehousing. Banu Y. Ekren holds associate professor position at her university and teaches simulation, stochastic modelling and facility planning and logistics courses at undergraduate level. She has several journal and book chapter publications as well as PI of international research projects.