YAŞAR UNIVERSITY

GRADUATE SCHOOL

MASTER THESIS

A MACHINE LEARNING APPLICATION FOR TRANSACTION PICKING IN A TIER-TO-TIER SBS/RS

BARTU ARSLAN

THESIS ADVISOR: ASSOC. PROF. (PHD) BANU YETKİN EKREN

M.SC. IN INDUSTRIAL ENGINEERING PROGRAM

PRESENTATION DATE: 17.06.2021

BORNOVA / İZMİR
JULY 2021

ABSTRACT

A MACHINE LEARNING APPLICATION FOR TRANSACTION PICKING IN A TIER-TO-TIER SBS/RS

Arslan, Bartu
MSc, Industrial Engineering
Advisor: Assoc. Prof. Banu Y. EKREN
July 2021

With the recent growth of e-commerce, the order profiles have shifted towards smaller quantities with faster delivery time requests of customers. This change has led to companies seek for fast transaction processing automation technologies in operations of warehouses. Shuttle-based storage and retrieval system (SBS/RS) is an automated warehousing technology mostly utilized in large distribution centers because of its capability of processing high transaction rate. While the advantage of this system is its capability of processing high transaction rate by the excess numbers of shuttles in the system, a disadvantage is that the average utilization of shuttles is very low, compared to the lifting mechanisms in the system. Since a dedicated shuttle is assigned at each tier of an aisle, this system is also referred as tier-captive SBS/RS in literature. In an effort to balance the utilization levels of shuttles and lifts, a novel design referred as tier-to-tier SBS/RS is introduced. In that design, there is decreased number of shuttles in the system so that they are allowed to travel between tiers by using a separate lifting mechanism specifically dedicated for travel of them. This novel design not only balances the service lifts and shuttles, but also decreases the initial investment cost for the system by the decreased number of shuttles. However, those advantages cause a disadvantage, that is increased average cycle time per transaction performance metric in the system. In this thesis, in an effort to contribute on decreasing average cycle time per transaction performance metric, we apply a machine learning methodology for smart transaction processing in the system. Specifically, we apply Reinforcement Learning and Deep Reinforcement Learning methods for transaction selection of shuttles. The proposed approaches are compared with well-known First-in-First-out

(FIFO) and Shortest Process Time (SPT) selection rules. The results show that the proposed approaches outperform both FIFO and SPT rules, significantly.

Keywords: Tier-to-tier SBS/RS, Reinforcement Learning, Deep Q-learning, Simulation modelling, Automated Warehousing

ÖZ

KATTAN KATA YOLCULUK EDEN SBS/RS’TE İŞLEM SEÇİMİ İÇİN BİR MAKİNE ÖĞRENMESİ UYGULAMASI

Arslan, Bartu
Yüksek Lisans Tezi, Endüstri Mühendisliği
Danışman: Doç. Dr. Banu Y. EKREN
Temmuz 2021

E-ticaretin son zamanlarda büyümesiyle, sipariş profilleri daha küçük miktarlarda ve daha hızlı teslimat süreleri olacak şekilde değişti. Bu değişiklik, şirketlerin depo operasyonlarında hızlı işlem işleme otomasyon teknolojileri aramasına sebep oldu. Mekik tabanlı depolama ve çekme sistemi (SBS/RS), yüksek işlem miktarlarını işleme yeteneği nedeniyle çoğunlukla büyük dağıtım merkezlerinde kullanılan otomatik bir depo teknolojisidir. Bu sistemin avantajı, sistemdeki fazla sayıda mekik ile yüksek işlem miktarlarını işleme kabiliyeti iken, dezavantajı mekiklerin ortalama kullanımının, sistemdeki asansör mekanizmalarına göre çok düşük olmasıdır. Bir koridorun her katına özel bir mekik atandığından, bu sistem aynı zamanda literatürde sabit katlı SBS/RS olarak da anılır. Mekiklerin ve asansörlerin kullanım seviyelerini dengelemek amacıyla, kattan kata yolculuk eden SBS/RS olarak adlandırılan yeni bir tasarım tanıtıldı. Bu tasarımda, sistemde mekiklerin sayısı azalmıştır. Özellikle taşınmaları için ayrılmış ayrı bir asansör mekanizması kullanılarak katlar arasında hareket etmelerine izin verilir. Bu yeni tasarım yalnızca asansörleri ve mekikleri dengelemekle kalmaz, aynı zamanda servis araçlarının sayısının azalmasıyla sistemin ilk yatırım maliyetini de düşürür. Bununla birlikte, bu avantajlar bir dezavantaja dönüşür. Sistemdeki performans ölçütü olan işlem başına ortalama döngü süresinin artmasına neden olur. Bu tezde, işlem başına ortalama döngü süresini azaltmaya katkıda bulunmak amacıyla, sistemde akıllı işlem işleme için bir makine öğrenimi metodolojisi uyguluyoruz. Spesifik olarak, servis araçlarının işlem seçimi için Pekiştirmeli Öğrenme ve Derin Pekiştirmeli Öğrenme yöntemlerini uyguluyoruz. Önerilen yaklaşım, iyi bilinen İlk-Giren-İlk-Çıkar (FIFO) ve En Kısa İşlem Süresi (SPT) seçim kuralları ile karşılaştırılır. Sonuçlar, önerilen metotların her iki kuralı da önemli ölçüde aştığını göstermektedir.

Anahtar Kelimeler: Kattan kata yolculuk eden SBS/RS, Pekiştirmeli Öğrenme, Derin Q-öğrenimi, Benzetim modeli, Depolama sistemi

ACKNOWLEDGEMENTS

I would like to thank my supervisor Banu Y. Ekren for guiding me in this subject and being very helpful along the way.

I would also like to thank my mother, my sisters, and my fiancée for always believing in me and being very supportive.

Bartu ARSLAN

İzmir, 2021

TEXT OF OATH

I declare and honestly confirm that my study, titled “A MACHINE LEARNING APPLICATION FOR TRANSACTION PICKING IN A TIER-TO-TIER SBS/RS” and presented as a Master’s Thesis, has been written without applying to any assistance inconsistent with scientific ethics and traditions. I declare, to the best of my knowledge and belief, that all content and ideas drawn directly or indirectly from external sources are indicated in the text and listed in the list of references.

Bartu ARSLAN

Signature

……………………………………

July 13, 2021

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

SYMBOLS AND ABBREVIATIONS

ABBREVIATIONS:

AS/RS	Automated Storage and Retrieval System
AVS/RS	Autonomous Vehicle Storage and Retrieval System
SBS/RS	Shuttle-based Storage and Retrieval System
AGV	Automated Guided Vehicles
GA	Genetic Algorithm
FIFO	First-in-First-Out
SPT	Shortest Process Time
ML	Machine Learning
RL	Reinforcement Learning
DRL	Deep Reinforcement Learning
DQN	Deep Q-Network
OQN	Open Queueing Network

SYMBOLS:

T	Number of tiers in the system.
B	Number of bays in a tier.
S	Total number of shuttles in the system.
W	Distance between two adjacent bays.
H	Distance between two adjacent tiers.
t	Transaction mean inter-arrival time.
C_{avg}	Average cycle time per transaction.
F_{avg}	Average flow time per transaction.
W_{avg}	Average waiting time per transaction.
WT	Average number of transactions waiting in the queue.

V_s	Maximum velocity of shuttles.
V_l	Maximum velocity of lift 1.
V_{sl}	Maximum velocity of lift 2.
A_s	Acceleration of shuttles.
A_l	Acceleration of lift 1.
A_{sl}	Acceleration of lift 2.
D_s	Deceleration of shuttles.
D_l	Deceleration of lift 1.
D_{sl}	Deceleration of lift 2.
US_{avg}	Average utilization of shuttles.
UL_{avg}	Average utilization of lift 1.
USL_{avg}	Average utilization of lift 2.

CHAPTER 1
INTRODUCTION

Industry 4.0 developments, outbreak of Covid-19 and following e-commerce increase have led to increased investment on automated technologies within the facilities. The advancement on technologies such as RFID tags to deploy the goods, IoT implementations such as sensors have facilitated implementation of robotic technologies in warehouses to increase efficiency, reduce accidents within the facilities and increase the traceability of the products.

A report of Mordor Intelligence (2020) shows that in 2020, Global Automated Material Handling (AMH) Market was valued at $55.907,4 million and expected to reach $112.083 million by 2026. According to another report of Mordor Intelligence (2020b), the Automated Storage and Retrieval (AS/RS) market is valued at $16.576,2 million in 2020 and is projected to reach $29.244,73 million by 2026. These reports show that AS/RS market has a big share in AMH market and expected to grow even more in the near future.

AS/RS is an automated warehouse technology and it is designed to buffer, store and retrieve products (Romaine, 2020). This design is mostly utilized in inventory management systems, warehouses, and distribution centers. One of the biggest benefits of this system is that it does not require a large floor space, which is an important issue in warehouse operations. In Figure 1.1, Allied Market Research (2020) shows the market shares by type for 2019 and 2027. It is observed that all types of AS/RS market tend to increase in the future.

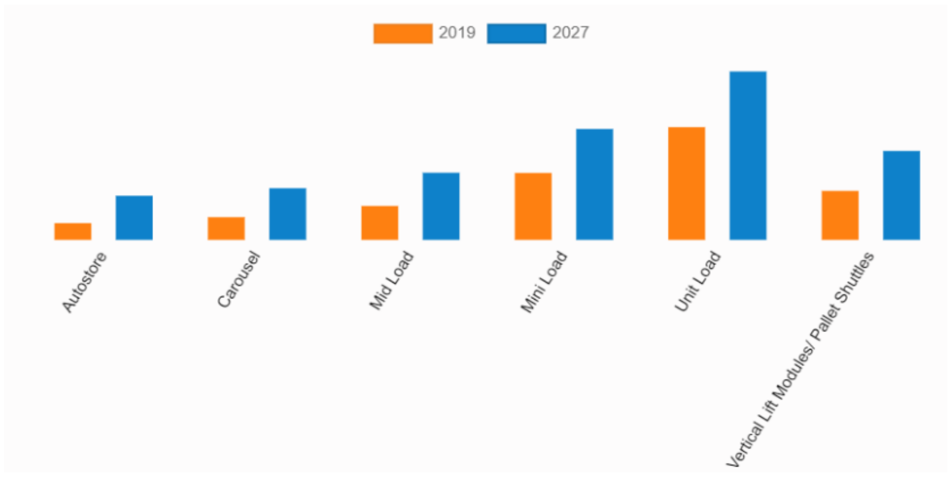


Figure 1.1. AS/RS market share by type

Shuttle-based storage and retrieval system (SBS/RS) is an AS/RS system that is composed of automated vehicles (i.e. shuttles), storage racks and lifts. This system is commonly utilized in mini-load warehouses. A materials-handling systems supplier company named Dematic Group describes this system as "providing fast, increased storage density, increased accuracy and high throughput rates, used in warehouses, factories and distribution centres" (Dematic, n.d.). In this thesis, we refer this system as tier-captive SBS/RS, since the system has a dedicated shuttle in each tier of an aisle. The physical configuration of the tier-captive SBS/RS is shown in Figure 1.2. In a traditional tier-captive SBS/RS, a lifting mechanism is installed at each cross-aisle for travel of loads (i.e. totes) from/to input/output (I/O) points that are located at the ground level of each aisle. We refer this lifting mechanism as Lift 1 as shown in Figure 1.2. A single lifting table is installed at left and right side of this mechanism helping to double the working capacity. Shuttles provide horizontal movement for totes between buffer and destination bays of the totes. One of the handicaps of this design is due to the excess numbers of shuttles in the system, the average utilizations of them are very low compared to the lifting mechanisms which are mostly bottlenecks.

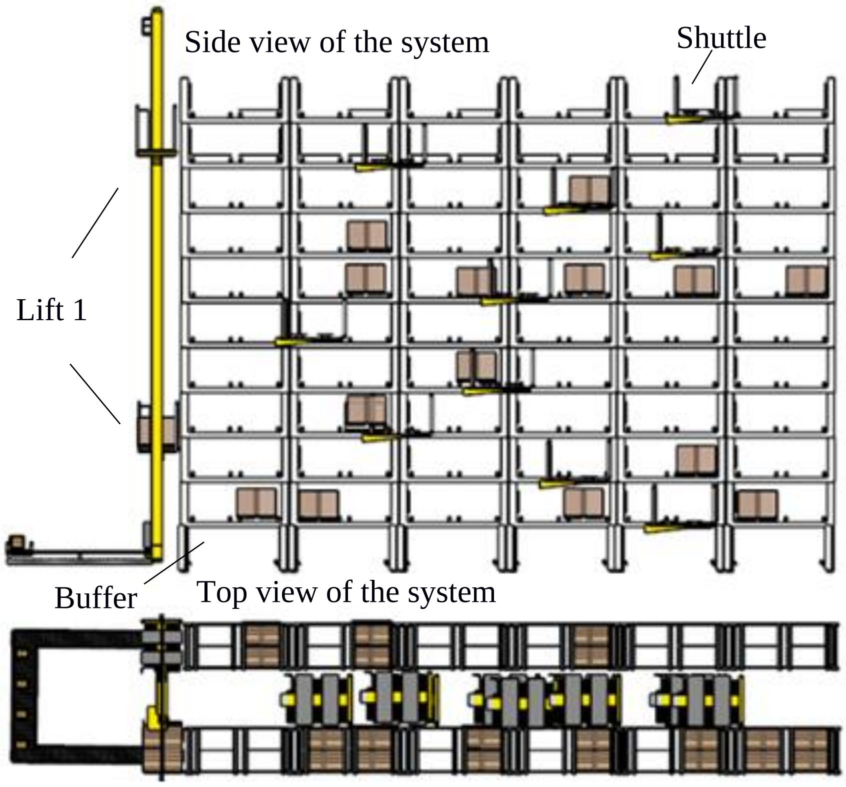


Figure 1.2. Tier-captive SBS/RS design

In this thesis, we study a novel SBS/RS design where the number of shuttles is decreased in the system so that those shuttles are allowed to travel between tiers. We refer this new design as tier-to-tier SBS/RS whose figure is shown in Figure 1.3. Note that instead of having a dedicated shuttle in each tier, the number of shuttles is decreased in the system so that we allow shuttles travel between tiers. This lifting mechanism dedicated for travel of shuttles between tiers is referred as Lift 2 in Figure 1.3. With this design change, our aim is to balance the resource utilizations, lift and shuttles. In addition, since there is less number of shuttles, initial investment cost might be decreased (Küçükyaşar et al., 2020).

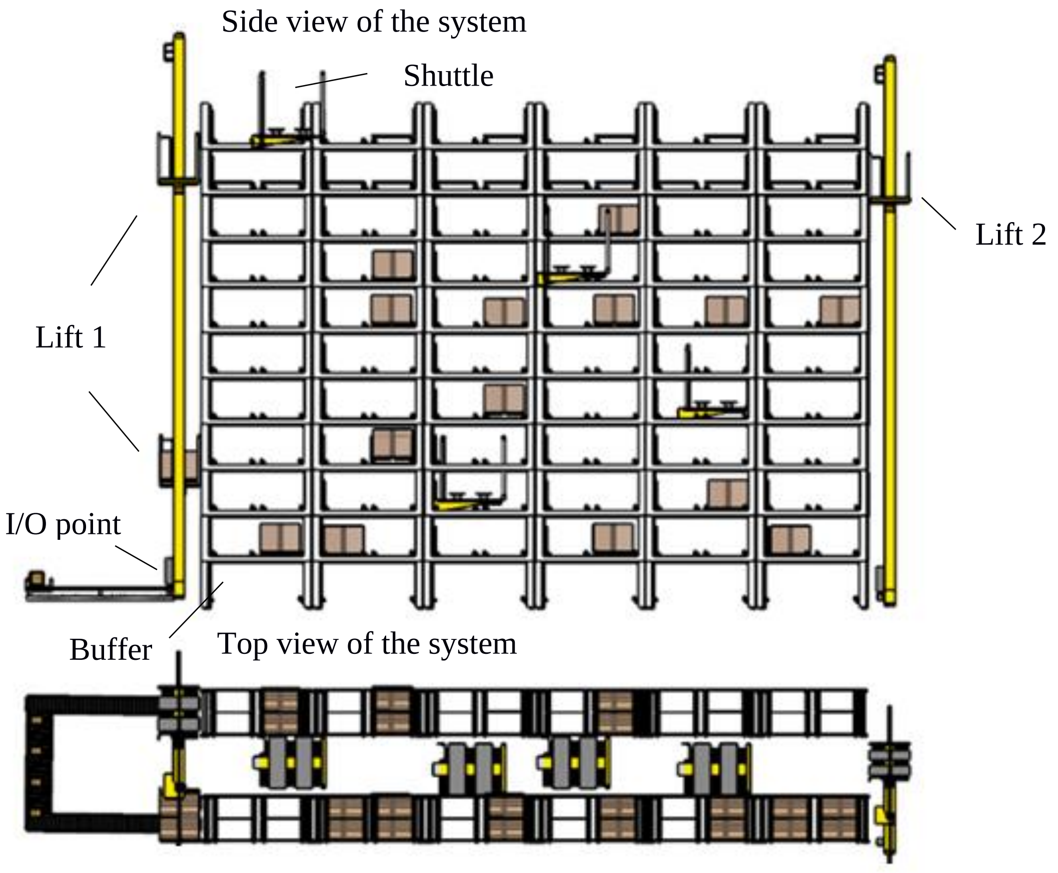


Figure 1.3. Tier-to-tier SBS/RS design

Although the proposed system design balances the utilizations of lifts and shuttles and tends to decrease the initial investment costs, since the shuttles are able to travel between tiers, the average travel time of shuttles may tend to increase. This might cause increase of average cycle time of a transaction in the system. Here, cycle time is the time between when a transaction request is created in the system until it is disposed. Hence, cycle time performance metric includes waiting time of transactions in the system. In this work, we consider not only average cycle time per transaction performance metric but also average flow time per transaction performance metric which considers time between when a transaction is selected by a shuttle until it is disposed. In an effort to reduce average cycle time per transaction performance metric in the system, we apply a smart transaction selection policy, RL, in the system.

The popularity of machine learning algorithms has increased recently due to increase of computing power of processors and graphics cards. Since developing analytical models for most real-world problems is hard and when an assumption changes in the system, that model may become invalid, more adaptive and learning algorithms might be proper for today's dynamic industry environments. In this case, we study a *Q*-

learning algorithm because it is easy to implement, and might provide good results even for complex system designs. Hence, in this thesis, we search two main research questions:

- Q1: Can a machine learning algorithm applicable for a tier-to-tier SBS/RS?
- Q2: If so, does it produce better results compared to the traditional algorithms?

For Q1, we propose a RL solution approach by using *Q*-learning. We compare the results with traditional algorithms such as First-in-First-out (FIFO) and Shortest Process Time (SPT). For Q2, we also develop a Deep Reinforcement Learning (DRL) model using Deep *Q*-learning (DQL) and compare the results with FIFO and SPT.

1.1. AVS/RS Studies

One of the earliest studies is completed by Ekren et al. (2010). They apply design of experiment to identify factors that affects the performance of an autonomous vehicle storage and retrieval (AVS/RS). They consider the average cycle time, average vehicle utilization and average lift utilization. Ekren & Heragu (2011) present a simulation-based performance analysis of an AVS/RS. They aim to find the optimal values for number of autonomous vehicles and lifts in the system. Marchet et al. (2011) study a tier-captive AVS/RS and estimate the performance of the system through analytical model based on open queueing network. The model is validated through simulation. In their later work, Marchet et al. (2013) study trade-offs between tier-captive and tier-to-tier AVS/RS by using simulation and propose a design framework. Ekren et al. (2013) present an analytical model for an AVS/RS by using semi-open queueing network (SOQN) modelling. They use an approximate method to solve the SOQN to obtain the performance measures. Ekren et al. (2014) model the AVS/RS by using an SOQN approach. They solve the network by applying matrix-geometric method. D’Antonio et al. (2019) propose an analytical model for an AVS/RS to evaluate the energy consumption. They validate their models by their simulation results. Ekren (2020b) studies a hierarchical solution approach for an AVS/RS design. The model aims to minimize two performance measures, average cycle time and average energy consumption per transaction performance metrics. Pareto-optimal solutions are provided in that work. A recent work is completed by Lerher et al. (2021). In their study, they propose a novel AVS/RS design with multiple-tier shuttle vehicles. They present an analytical model to estimate the performance of the system. Another recent study was made by Jerman et al. (2021). They propose a novel AVS/RS design including lifts that carry a shuttle that move along the aisle. They utilize simulation modelling approach to analyze the throughput rate performance of the system.

1.2. Tier-captive SBS/RS Studies

Tier-captive SBS/RS designs have been studied well in the current literature. A novel SBS/RS study is proposed by Carlo & Vis (2012). Their design includes two non-passing lifts and they compare this design's performance with a single lifting system design. That paper studies a look-ahead strategy heuristic and treats the system as a scheduling problem. Lerher (2015) presents an analytical model for analyzing the performance of a double-deep SBS/RS design. Lerher et al. (2015a) presents a performance evaluation method for an SBS/RS and compare the system performance with a crane-based AS/RS design through simulation. They analyze the mean cycle time and throughput capacity as performance metrics. Lerher et al. (2015b) present an analytical model to evaluate performance metrics of an SBS/RS. Ekren et al. (2015) apply a class-based storage policy methodology where they also optimize the rack design of SBS/RS. They use simulation for the modelling purpose. Wang et al. (2015) propose an analytical model to solve task scheduling problem for an SBS/RS. They also implement sorting genetic algorithm to solve multi-objective optimization function. Lerher et al. (2016) present a method to calculate the throughput performance of an SBS/RS. Tappia et al. (2016) present an analytical model by using a novel queueing model to estimate the performance of an SBS/RS. Zou et al. (2016) present a fork-join queueing network approach for estimating the performance metrics for an SBS/RS. Ekren (2017) provides a graph-based solution for an SBS/RS to evaluate the performance of the system using simulation modelling. She analyzes the average utilization of lifts and average cycle times of transactions. Eder, (2019) presents an analytical model using continuous time open queueing network with limited capacity approach for an SBS/RS. He aims to evaluate the performance of the system. The model is validated through simulation. Ekren (2020a) performs an experimental design for an SBS/RS. She aims to identify the significant factors that affect the performance of the system by using simulation modelling. Ekren & Akpunar (2021) propose a tool that computes performance metrics for SBS/RS. They apply an open queueing network modelling approach.

1.3. Tier-to-Tier SBS/RS Studies

Ha & Chae (2018a) study an SBS/RS design with a single lifting mechanism that can transfer both shuttles and loads. They compare the system performance with a traditional tier-captive design by using simulation modelling. The results show that the targeted throughput rate can be obtained by using less number of shuttles in the system. Later, Ha & Chae (2018b) develop a decision model based on the travel time model. They aim to determine the number of shuttles using this model. Zhao et al. (2019) develop an integer-programming model to decrease the idle time of lifts and shuttles. They aim to minimize the total task time. Küçükyaşar et al. (2020) compares the traditional tier-captive SBS/RS with tier-to-tier. They consider initial investment costs, throughput rates and average energy consumption per transaction performance metrics. They utilize simulation modelling.

1.4. Machine Learning Studies for AGVs

In the current literature, there are no existing studies that implement a machine learning approach on an AVS/RS. Hence, we review machine learning studies that are mostly implemented for autonomous guided vehicles (AGVs). Watanabe et al. (2001) study *Q*-learning approach for collision avoiding and navigation problems for AGVs. Dou et al. (2015) combine genetic algorithm and RL to solve task scheduling problem for mobile robots in a warehouse. Xue et al. (2018) propose RL method for multiple AGVs to solve flow-shop scheduling problem. They aim to minimize the average job delay. Their study shows that the RL method works better than multi-agent scheduling method. Malus et al. (2020) propose a multi-agent RL for order dispatching of autonomous mobile robots in a dynamic production environment.

1.5. Deep *Q*-Learning Studies

Mao et al. (2016) present an initial work of a deep reinforcement learning implementation for a resource management problem. Their results show that the method performs very similar to state-of-the-art heuristics and is adaptable for changing conditions. Gazori et al. (2020) propose a Double Deep *Q*-learning (DDQL) approach to minimize computation costs and long-term service delays. They use target network and experience replay techniques. Tong et al. (2020) implement Deep *Q*-learning method for dynamic task scheduling in a cloud computing environment. They utilize simulation modelling approach. Takahashi & Tomah (2020) implement Deep *Q*-learning for controlling of multiple AGVs. They utilize simulation modelling, and the results show that the proposed method provides near-optimal solutions.

CHAPTER 2
Q-LEARNING APPROACH

In this chapter, we present a *Q*-learning approach for transaction selection in a tier-to-tier SBS/RS. First, we explain the RL and *Q*-learning methods. Then, we give the simulation model details, and implementation of the algorithm. Finally, we discuss the results.

2.1. Reinforcement Learning and *Q*-Learning Methods

RL is a machine learning approach that utilizes agents to choose actions in an environment to maximize the cumulative rewards (Ou et al., 2019). In this method, an agent collects the information from the environment and selects the most appropriate actions, regarding to that information. In Figure 2.1, the interaction between environment and agent is shown.

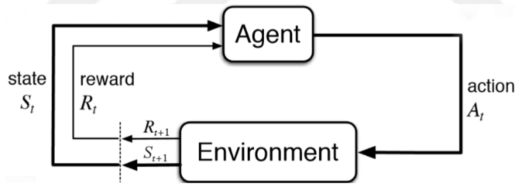


Figure 2.1. Interaction of agent and environment

The agent observes the environment and hence, the state information. By considering the current state of the environment, agent performs an action. Upon completing the action, the action is either rewarded or penalized. Aim of the agent is to maximize the cumulative rewards. Since this interaction continues until the agent is trained well in an environment, this iterative learning method is called reinforcement learning. Sutton & Barto (2015) defines four key elements in a RL problem as policy, reward function, value function and the model of the environment.

A RL model consists of:

- *Agent:* An entity performing actions to gain rewards in an environment.
- *Environment:* The world agent interacts with.

- *State (s):* Current situation of the environment.
- *Action (a):* All possible actions an agent can make.
- *Reward (r):* A feedback given from the environment to evaluate the action.
- *Value function:* Expected reward within a state.

In this study, we implement a model-free RL algorithm, Q-learning. In Q-learning, agent selects the actions based on Q-table. Q-table stores the Q-values of state-action pairs. In order to calculate Q-values of the pairs, we use Bellman's Equation as the value function, shown in Eq. (1).

$$Q^{new}(s_t, a_t) \leftarrow (1 - \alpha)\, Q(s_t, a_t) \; + \; \alpha\, [r_t \; + \; \gamma\, max\, Q(s_{t+1}, a)] \qquad (1)$$

In the equation, s_t represents the state at time t, a_t represents the taken action at time t, $Q(s_t, a_t)$ represents the Q-value of the state-action pair, α is the learning rate, r_t is the reward and γ is the discount rate. $Q^{new}(s_t, a_t)$ is the updated Q-value. The old value is weighted by (1-α). $[r_t \; + \; \gamma\, max\, Q(s_{t+1}, a)]$ is the target value, $max\ Q(s_{t+1}, a)$ being the maximum reward that can be obtained for the next state.

Commonly, Q-table is initialized as a zero matrix, meaning that the agent is not informed about how to behave. Initially, epsilon-greedy approach is used for selecting actions. In this approach, an epsilon value ϵ is initialized as 1 to increase exploration of the system and decreased over time. A random number between 0 and 1 is generated and if the number is smaller than the epsilon value, a random action is taken. Otherwise, the action with highest Q-value is selected (Wei et al., 2017).

This Q-table becomes stable after the learning period and epsilon value is decreased to a very small number, meaning that almost all the time the agent selects the action with the highest Q-value.

The weakness of this method is that since no approximation is made and Q-table requires all state-action pairs to be realized in order to update the values, the learning period takes very long time for large number of states and actions.

2.2. Simulation Model

Due to complexity of tier-to-tier SBS/RS, we utilize simulation modelling approach. The system is simulated by using the Arena Simulation 16.0 commercial software. The notations used for the model are summarized in Table 2.1.

Table 2.2.1. Notations used for the model

Notation	Unit	Description
C_{avg}	*sec.*	Average cycle time per transaction
F_{avg}	*sec.*	Average flow time per transaction
W_{avg}	*sec.*	Average waiting time per transaction
t	*sec.*	Transaction mean inter-arrival time
Vs	*m/s*	Maximum velocity of shuttles
Vl	*m/s*	Maximum velocity of lift 1
Vsl	*m/s*	Maximum velocity of lift 2
As	m/s^2	Acceleration of shuttle
Al	m/s^2	Acceleration of lift 1
Asl	m/s^2	Acceleration of lift 2
Ds	m/s^2	Deceleration of shuttle
Dl	m/s^2	Deceleration of lift 1
Dsl	m/s^2	Deceleration of lift 2
US_{avg}	%	Average utilization of shuttle
UL_{avg}	%	Average utilization of lift 1
USL_{avg}	%	Average utilization of lift 2
W	*m*	Distance between two adjacent bays
H	*m*	Distance between two adjacent tiers
T		Number of tiers in the system
B		Number of bays in a tier
S		Total number of shuttles

For a storage transaction, the process can be summarized as:

1. Shuttle chooses a storage transaction.
2. If the transaction's destination tier is different than the shuttle's tier, shuttle moves to Lift 2, and Lift 2 carries the shuttle to the destination tier. Meanwhile, Lift 1 moves to the I/O point and picks up the shuttle.
3. Lift 1 moves to the destination tier and drops off the tote at the buffer location.

4. Shuttle moves to the buffer location and picks up the tote.

5. Shuttle moves to the storage bay address of the transaction and drops off the tote.

For a retrieval transaction, the process can be summarized as:

1. Shuttle chooses a retrieval transaction.

2. If the transaction's destination tier is different than the shuttle's tier, shuttle moves to Lift 2, and Lift 2 carries the shuttle to the destination tier. Meanwhile, Lift 1 moves to destination tier.

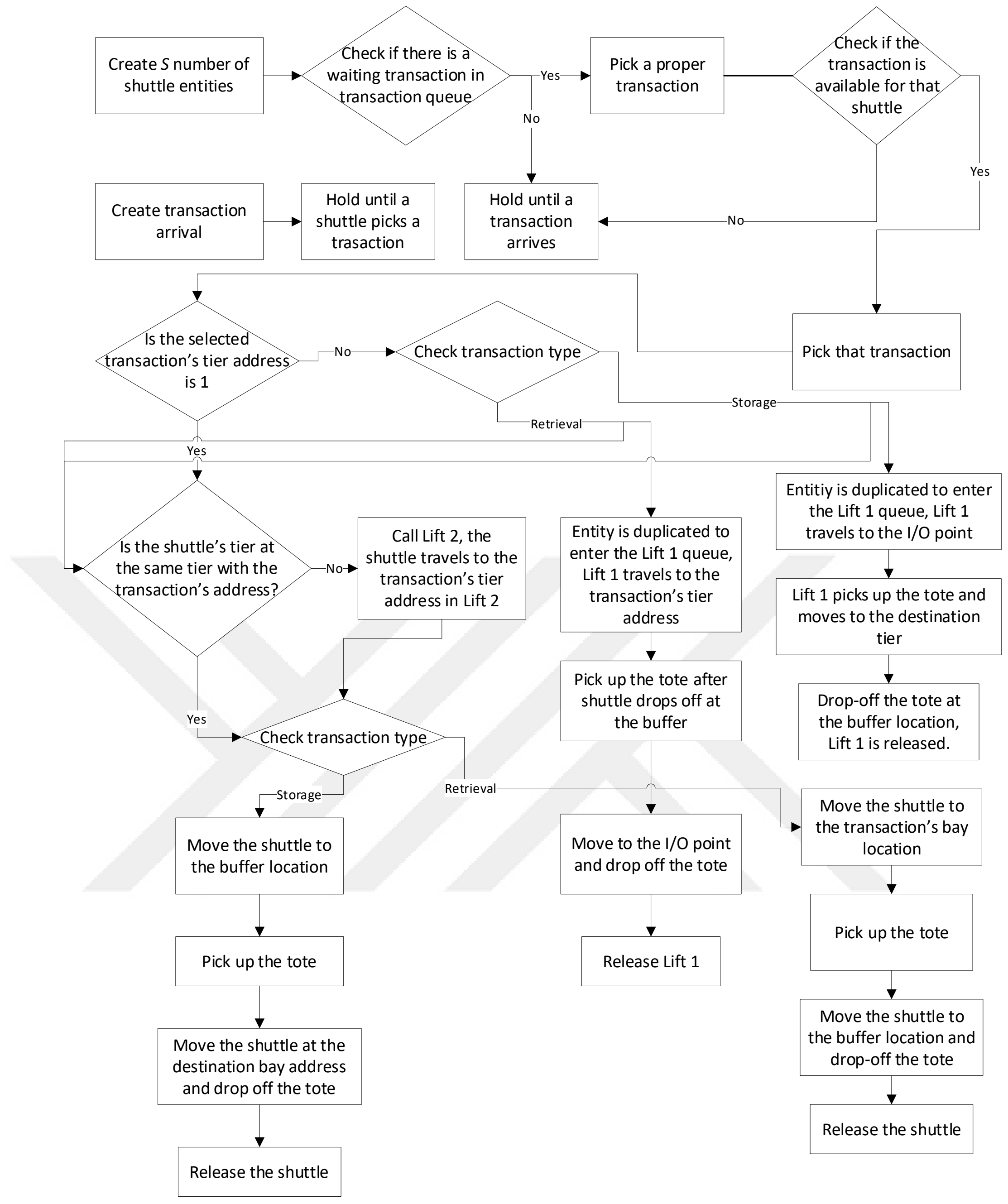
3. Shuttle travels to the retrieval bay address and picks up the tote.

4. Shuttle moves to the buffer location.

5. Lift 1 picks up the tote from the buffer location.

6. Lift 1 moves to the I/O point and drops off the tote.

The flow chart of this simulation model is provided in Figure 2.2.



Create *S* number of shuttle entities
Check if there is a waiting transaction in transaction queue
Yes
No
Pick a proper transaction
Check if the transaction is available for that shuttle
Yes
No
Create transaction arrival
Hold until a shuttle picks a trasaction
Hold until a transaction arrives
Pick that transaction
Is the selected transaction's tier address is 1
No
Check transaction type
Storage
Retrieval
Yes
Is the shuttle's tier at the same tier with the transaction's address?
No
Call Lift 2, the shuttle travels to the transaction's tier address in Lift 2
Entity is duplicated to enter the Lift 1 queue, Lift 1 travels to the transaction's tier address
Entitiy is duplicated to enter the Lift 1 queue, Lift 1 travels to the I/O point
Lift 1 picks up the tote and moves to the destination tier
Pick up the tote after shuttle drops off at the buffer
Drop-off the tote at the buffer location, Lift 1 is released.
Yes
Check transaction type
Storage
Retrieval
Move the shuttle to the buffer location
Move to the I/O point and drop off the tote
Move the shuttle to the transaction's bay location
Pick up the tote
Release Lift 1
Pick up the tote
Move the shuttle at the destination bay address and drop off the tote
Move the shuttle to the buffer location and drop-off the tote
Release the shuttle
Release the shuttle

Figure 2.2. Tier-to-tier SBS/RS simulation flow chart

The verification of the model is done by debugging the codes and animating the system. Also, creating controlled transactions and tracing those transactions helped us verifying the simulation model. The RL algorithm is integrated in the simulation model by using the Visual Basic (VBA) interface in the Arena 16.0 software. The assumptions for the studied simulation model are provided below:

- Transactions arrive at I/O points and enter a single queue.

- Storage and retrieval transaction mean interarrival rates are equal and follow Poisson distribution.
- When a shuttle becomes available, it selects a transaction from that common queue.
- To avoid collisions of shuttles, the available shuttle agent does not select a transaction if another shuttle is located or heading towards that possible transaction's destination tier.
- After selecting a transaction, the entity is duplicated and sent to Lift 1 queue. If necessary, Lift 2 is called as well.
- Lift 1 and Lift 2 operates with First-in-First-out (FIFO) rules.
- If the transaction address is at the first tier, Lift 1 is not utilized.
- Lift 2 is dedicated only for shuttle travels.
- Lift 1 has two lifting tables that can travel two totes independently.
- Lift 2 can only carry one shuttle at a time.
- Shuttles and lifts stay at their last process points as dwell point policy.

For system design inputs such as height of tiers and length of bays, we use parameters from Lerher et al. (2015), Lerher et al. (2015), Ekren et al. (2018), Ekren (2020a). The input parameters for the warehouse are given below.

- $W = 0.5$ *m*., $H = 0.35$ *m*.
- $Vs = Vl = Vsl = 2$ *m/s*.
- $As = Al = Asl = Ds = Dl = Dsl = 2$ m/s^2

2.3. Implementation of *Q*-Learning

For our problem, we define shuttles as agents. Each time a shuttle picks a transaction from a common queue, state information is obtained from the environment. We define the states as $S(k) = (i, j)$:

$S(k)$ = (Current tier of Lift 1, Current tier of the agent shuttle k)

Here, k represents the shuttle that is selecting a transaction from the queue, i represents the current tier of Lift 1 and j represents the current tier of shuttle k. From the Q-table, agent checks the current positions of both Lift 1s and chooses the Lift 1 side with the higher *Q*-value. In this case, the state information takes integer values between 1 and *T*, the number of tiers in the system.

The actions are defined as the attributes of the waiting transactions.

A(k) = (Tier address of the transaction, transaction type)

For agent shuttle k, tier addresses of transactions are integers between 1 and *T*, transaction type is either 0 (for storage) or 1 (for retrieval).

In case two shuttles are available at the same time, the first released shuttle selects an action (i.e., transaction) first. However, since the experiments are designed in a way that average shuttle utilizations are very high, this case rarely occurs. Since the state space consists of tiers of the lift and shuttle, and the action space consists of tier addresses and transaction types, the *Q*-matrix size is equal to multiplication of states and actions, $T^2 \times 2T$.

The immediate reward function is defined by (2):

$$R(s,a) = \frac{1}{flow\ time(a)} \tag{2}$$

The queue information is not included in states, therefore including waiting times in reward function would not correlate with states. Also, to reward the agent for having smaller flow time, inverse of the flow time is utilized as reward function. The details of the *Q*-learning algorithm are explained below.

1. Initialize *Q*-values for all *Q(s, a)* where $s \in S$, $a \in A$
2. Observe the state *S(k) = (i, j)*
3. With probability ϵ, select and execute random action a, otherwise select a = max *Q(s, a)*
4. Observe the reward and update *Q(s, a)* using Eq. (1)
5. If terminal, end the simulation, otherwise update ϵ and go to step 2.

In our problem, we initialized the parameters $\epsilon = 0.8$, $\alpha = 0.1$, $\gamma = 0.2$. Every 10 days, we decreased ϵ by 0.2 and α is decreased by 0.02. *Q*-values are not initialized as 0 to reduce the learning period of the model. The analytical calculations are made by considering the travel time of shuttles and lifts, without considering the waiting times.

Since the system is highly stochastic, we choose small number for learning rate. Also, since the next available actions cannot be known, discount rate is small.

The *Q*-learning algorithm is coded by utilizing VBA interface in the Arena software. We complete some experiments to verify and validate the model. In Figure 2.3, the average cycle time per transaction output is shown during the learning period. It can be observed from the figure that the average cycle time is initially very high and decreases over time. Note that initializing *Q*-values decrease the learning period significantly.

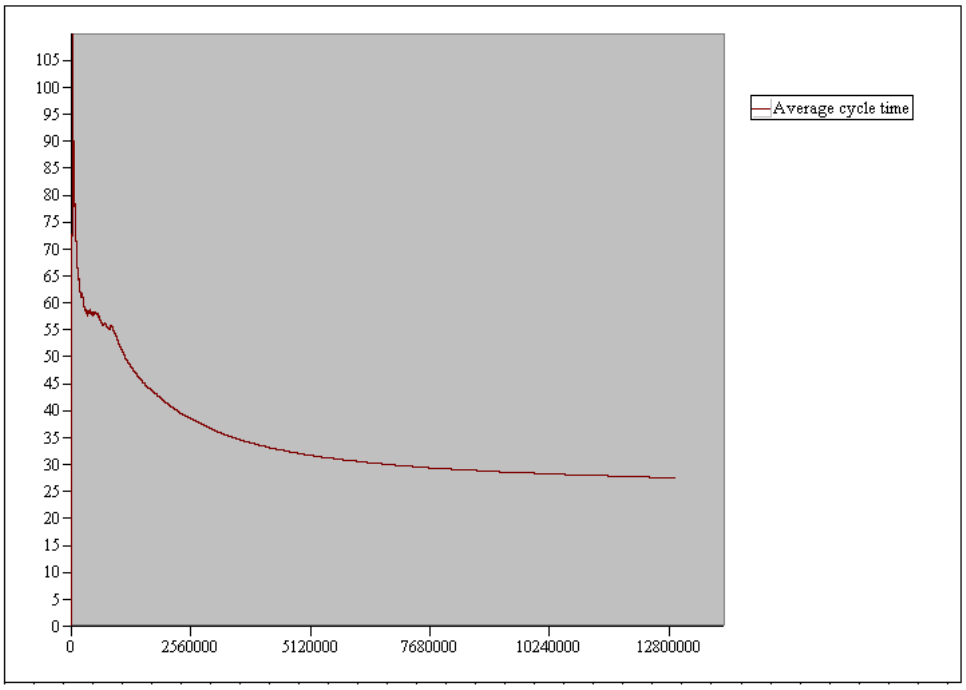


Figure 2.3. Average cycle time per transaction during the training period

The training period for this model is assumed to be around 3,000,000 seconds. In Figure 2.4, it can be observed that after shuttle agents are trained well, average cycle time is stabilized.

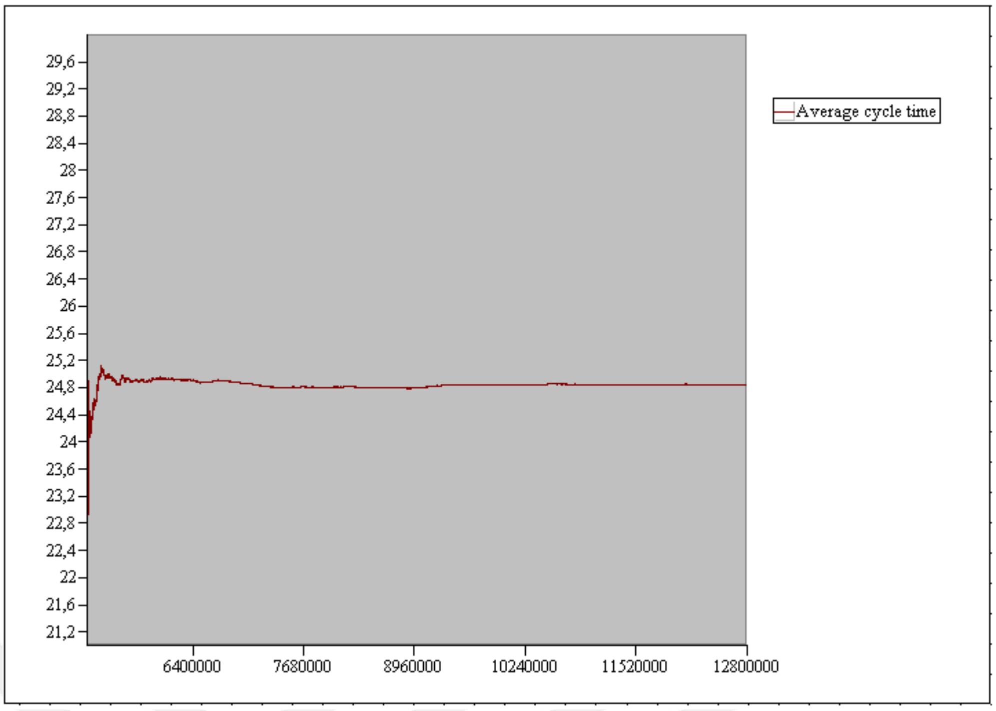


Figure 2.4. Average cycle time per transaction after learning period

2.4. Results

Remember that as the number of tiers increase in the system, the *Q*-matrix size increases exponentially. Due to limitation of *Q*-learning, we conduct experiments having up to 15 number of tiers. The experiments are summarized in Table 2.2. In total, we conduct 4 different warehouse designs for FIFO, SPT and RL rules. In addition, two different arrival scenarios are examined. In single type, the transactions arrive the system one by one. In batch scenario, the transactions arrive in batches.

Table 2.2. Conducted experiments

Warehouse Design #	*T* - *N*s	Scheduling rule	Arrival scenario
1	15 - 5	FIFO	Single type
2	15 - 3	SPT	Batch
3	12 - 4	RL	
4	10 - 3		

In FIFO rule, the shuttles pick the earliest arriving transactions from the queue. In SPT rule, the shuttles pick the transaction whose tier address is the closest one to its current tier. For these rules, 10 independent replications are made. The transaction types and addresses are assigned randomly, assuming equal probability. The results are given considering 95% confidence intervals.

The arrival rates are adjusted so that the resources run at a high utilization, considering the FIFO rule since it produces the highest results for our performance metrics. We first conduct the experiment on the FIFO rule and fix the arrival rate for other rules. In Table 2.3, experiments considering single arrival scenario are summarized.

Table 2.3. Experimental results for single transaction arrivals

Rule	WH Design #	UL_{avg}	USL_{avg}	US_{avg}	C_{avg}	F_{avg}	W_{avg}	t
FIFO	1	87%	93%	90%	55.49 ± 0.46	18.51	36.98 ± 0.45	3.8
RL	1	83%	82%	81%	24.8	16.87	7.93	3.8
SPT	1	83%	81%	81%	24.28 ± 0.21	16.8	7.48 ± 0.09	3.8
FIFO	2	67%	79%	91%	61.59 ± 1.07	17.39	44.2 ± 1.06	6
RL	2	64%	70%	85%	28.21	16.35	11.86	6
SPT	2	64%	69%	85%	27.74 ± 0.06	16.31	11.43 ± 0.18	6
FIFO	3	78%	88%	93%	101.85 ± 3.5	17.05	84.8 ± 3.5	4.3
RL	3	74%	74%	84%	24.85	15.45	9.4	4.3
SPT	3	74%	74%	83%	24.37 ± 0.03	15.38	9.01 ± 0.15	4.3
FIFO	4	76%	87%	92%	102.86 ± 2.94	16.48	86.38 ± 2.93	4.2
RL	4	71%	69%	81%	22.9	14.53	8.37	4.2
SPT	4	71%	69%	81%	22.67 ± 0.02	14.5	8.17 ± 0.09	4.2

Frim Table 2.3, it is observed that RL and SPT produces very close results and they both outperform FIFO rule. The reason of that might be due to the definition of the reward function. Since we consider flow times of transactions as reward function, it resembles to the SPT rule. Note that due to state definition, which only considers information about the tiers status, the agent does not sense enough information to produce better results than SPT.

With the idea of having more transaction options in selecting a transaction from a queue may produce better results, we also conduct experiments by considering batch arrivals. In Table 2.4, the results for batch arrivals are given. In this table, *t* represents the number of transactions arrive at the system every 10 minutes.

Table 2.4. Experimental results for batch transaction arrival

Rule	WH Design #	UL_{avg}	USL_{avg}	US_{avg}	C_{avg}	F_{avg}	W_{avg}	WT	t
FIFO	1	87%	93%	91%	306.16 ± 5.12	18.66	287.5	76	158
RL	1	58%	13%	57%	178.98	12.81	166.17	44	158
SPT	1	58%	13%	58%	188.96 ± 2.56	12.95	176.01	47	158
FIFO	2	67%	79%	91%	299.14 ± 4.78	17.44	281.7	47	100
RL	2	45%	15%	56%	173.05	11.35	161.7	27	100
SPT	2	45%	14%	56%	177.31 ± 2.69	11.41	165.9	28	100

FIFO	3	79%	90%	93%	384.83 ± 4.86	17.08	367.75	86	140
RL	3	52%	10%	57%	179.06	11.02	168.04	39	140
SPT	3	52%	10%	57%	183.93 ± 2.07	11.14	172.79	40	140
FIFO	4	77%	88%	92%	399.74 ± 5.08	16.51	383.23	94	143
RL	4	52%	8%	56%	175.98	10.44	165.54	40	143
SPT	4	51%	8%	56%	180.18 ± 2.47	10.58	169.6	41	143

In this experimental design, it is observed that RL produces better results compared to SPT rule as well. As explained before, this is caused by the fact that agent has more actions available and can select more intelligently.

Although the average utilization for resources is high for FIFO rule, the other rules work with low utilizations. In this case, we ignore the FIFO rule and conduct the experiments for SPT and RL with higher utilizations. In Table 2.5, the results for the experiments are given. In this table, *t* represents the number of transactions arrive at the system each stated minutes in that table.

Table 2.5. Experimental results for batch transaction arrival

Rule	**WH Design #**	UL_{avg}	USL_{avg}	US_{avg}	C_{avg}	F_{avg}	W_{avg}	*WT*	*t*
RL	1	87%	21%	86%	175.75	12.99	162.76	64	150/6.4 min.
SPT	1	86%	20%	87%	181.82 ± 1.56	13.12	168.7	66	150/6.4 min.
RL	2	73%	17%	86%	249.87	11.05	238.82	63	150/9.5 min.
SPT	2	69%	15%	85%	252.56 ± 2.36	11.06	241.5	64	150/9.5 min.
RL	3	81%	19%	86%	160.81	11.23	149.58	52	120/5.8 min.
SPT	3	78%	18%	86%	161.52 ± 0.94	11.22	150.3	52	120/5.8 min.
RL	4	74%	12%	87%	126.62	10.55	116.07	41	100/4.7 min.
SPT	4	77%	17%	85%	132.97 ± 1.12	10.97	122	43	100/4.7 min.

The results show that even in higher utilization for resources, the RL rule produces slightly better results.

CHAPTER 3
DEEP *Q*-LEARNING APPROACH

After noticing the handicaps of simple *Q*-learning application in Chapter 2, ignoring more detailed information from environment, in this chapter, we study a Deep *Q*-learning approach for transaction selection in a tier-to-tier SBS/RS. First, we explain the Deep *Q*-learning method. Then, implementation of the algorithm for the simulation model are provided. Finally, the results are given.

3.1. Deep Q-Learning Method

Deep *Q*-learning (DQL) or Deep *Q*-network (DQN) is a combination of RL and deep neural network methods. In this method, deep neural networks are utilized to approximate the *Q*-values. The agent feeds the state information of environment to the neural network as an input and the network estimates the *Q*-values for each action available for that state. This process is shown in Figure 3.1 below.

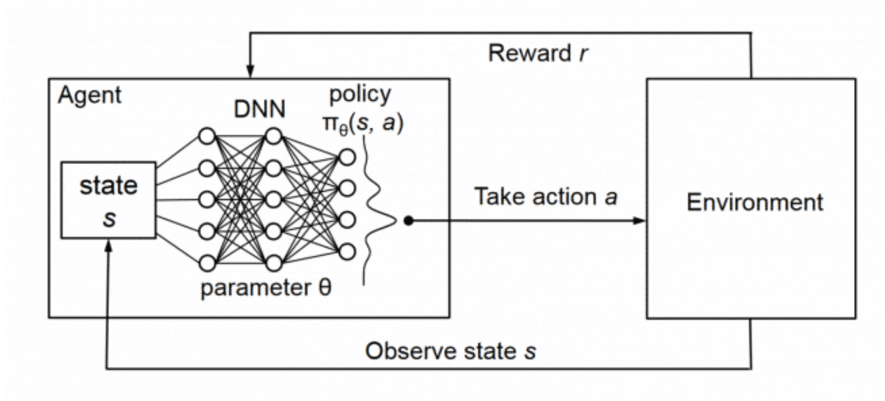
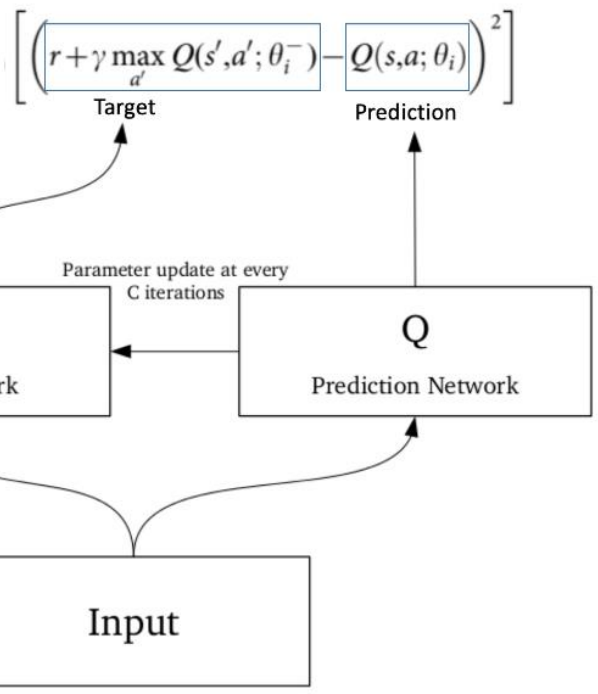


Figure 3.1. Deep *Q*-network process (source: Mao et al., 2016)

The Deep Neural Network (DNN) takes the state information as input and the information is fed through the hidden layer, to the output layer. Each state information is a node of the input layer, meaning that the state size is equal to the input size.

Compared to the other deep learning methods, the target value always changes. This causes the network to be unstable. In order to have more stable training, two neural networks that have the same architecture are utilized. One network represents the target, and one network represents the prediction. After every *C* iteration, which is a hyper-parameter, the prediction network parameters are cloned to the target network (Choudhary, 2019). This method is shown in Figure 3.2.



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

Figure 3.2. Utilizing two networks method (source: Choudhary, 2019)

We also use a method called experience replay for our study. Upon completion of an action, state, action, reward, next state information is stored in memory as tuples. As opposed to *Q*-learning, the network is not updated after each iteration. Instead, a pre-defined number of tuples are sampled from the memory and fed into the network when the number of tuples reach a certain point. This increases the normality of the information, meaning that a better learning result is obtained. This method is studied by Mnih et al. (2015). The algorithm is explained in detail in Figure 3.3. The agent observes the state and executes an action according to epsilon-greedy method. Then, state, action, reward, next state tuple is stored. The stored tuples are sampled after reaching a certain point and fed into the network. Gradient descent on target network is applied and epsilon is decreased. Every *C* step, prediction network is copied into the target network.

```
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 $\left(y_j - Q(s, a)\right)^2$
9:      Update $\epsilon$ as $\epsilon = \epsilon \times \epsilon^{dec}$ if $\epsilon > \epsilon_{min}$, otherwise $\epsilon_{min}$
10:     Every C steps, $\hat{Q} = Q$
```

Figure 3.3. Deep *Q*-Learning with experience replay algorithm

3.2. Implementation of the Method

For the DQN application, the simulation model is coded in Python programming, by using SimPy library. For the implementation of DQN, we utilize TensorFlow and Keras libraries. The flow of the simulation model is explained in Section 2.2. In this approach, we perform a non-episodic task. The model runs until the agent is trained.

As in the *Q*-learning, we treat shuttles as agents. The main goal of these agents is to select transactions in such a way that they maximize the cumulative rewards in long run. The state space is the input that is fed into the network to obtain the approximate *Q*-values for the actions. In this part, we define the state space as:

S(*k*) = (Current tier of shuttle *k*, current bay of shuttle *k*, current tier of first lifting table of Lift 1, availability of the first lifting table of Lift 1, current tier of the second lifting table of Lift 1, availability of the second lifting table of Lift 1, current tier of Lift 2)

Here, *k* represents the shuttle that is ready to pick a transaction from the queue. The tier values are integers between 1 and number of tiers in the system *T*, the bay values are integers between 1 and number of bays in a tier *B*, the availability is either 0 or 1 where 0 represents no availability at that moment and 1 represents that the resource is available. The actions defined in the neural network is the transaction attributes and the selected Lift 1 table. The action space can be summarized as:

$A(k)$ = (the tier address of the transaction, the bay address of the transaction, transaction type, selected Lift 1 table)

Here, the actions only represent available actions for shuttle k, excluding the transactions that can cause a collision, as explained in Section 2.2. Unlike Q-learning, here we implement the lifting table into our actions, to utilize Lift 1 more efficiently. As an example, if there is a retrieval transaction waiting in the queue assigned to the 10th tier, 8th bay, then the action index for this transaction would be (10, 8, 1, 1) and (10, 8, 1, 2) where 1 is assigned for transaction and 0 is assigned for storage items as the third index, 1 represents the left side of the Lift 1 and 2 represents the right side of the Lift 1 as the last index.

As the reward function, similar to the RL approach, we utilize flow time of the transaction. Using cycle time as our reward parameter may result in lower cycle times, however, it may require additional information to the state space to correlate with the reward. Otherwise, waiting times would change the reward and the waiting time information is not provided as an input. Nevertheless, minimizing flow time would result in decreased cycle times. Unlike RL, the reward function is normalized as explained in Eq. (2)-(4).

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

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

$$r_t = \frac{\frac{1}{flowtime(a)} - MINF_t}{MAXF_t - MINF_t} * 100 \tag{4}$$

Here, *flowtime* stores all the flow time values of all actions. In Eq. (2), the inverse of the maximum of these stored values is taken. In Eq. (3), the inverse of the minimum of these values is taken. We use generalized formulation of normalization, $X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$ for our case and it is shown in Eq. (4). *flowtime(a)* represents the flow time of action a. This method is inspired by study of Gazori et al. (2020) and shows good results. The normalized value is very small, hence the value is multiplied by 100 to track the improvement of the algorithm.

As mentioned earlier, TensorFlow and Keras libraries are utilized to apply the DQN to the simulation model. The DQN consists of three dense layers; an input layer, a hidden layer and an output layer. For input and hidden layers, Rectified Linear Unit (ReLU) activation function is selected. The activation function is utilized to transfer the weighted sum of input values to the output node. ReLU is a proper activation function since we deal with positive input values. As optimizer, we utilize “Adam” optimizer with parameters $\alpha = 0.001$, $\gamma = 0.2$, $\epsilon = 1$, $\epsilon^{\text{dec}} = 0.9999$, $\epsilon_{\min} = 0.01$, $n = 64$. Mean Squared Error (MSE) is used as loss function.

We conduct an experiment to decide the learning rate for our problem with parameters $T = 5$, $B = 25$, $S = 2$. In Figure 3.4, it can be seen that $\alpha = 0.001$ is the most suitable for our problem since the parameter learns quicker than $\alpha = 0.0001$ and provides a better result compared to $\alpha = 0.01$.

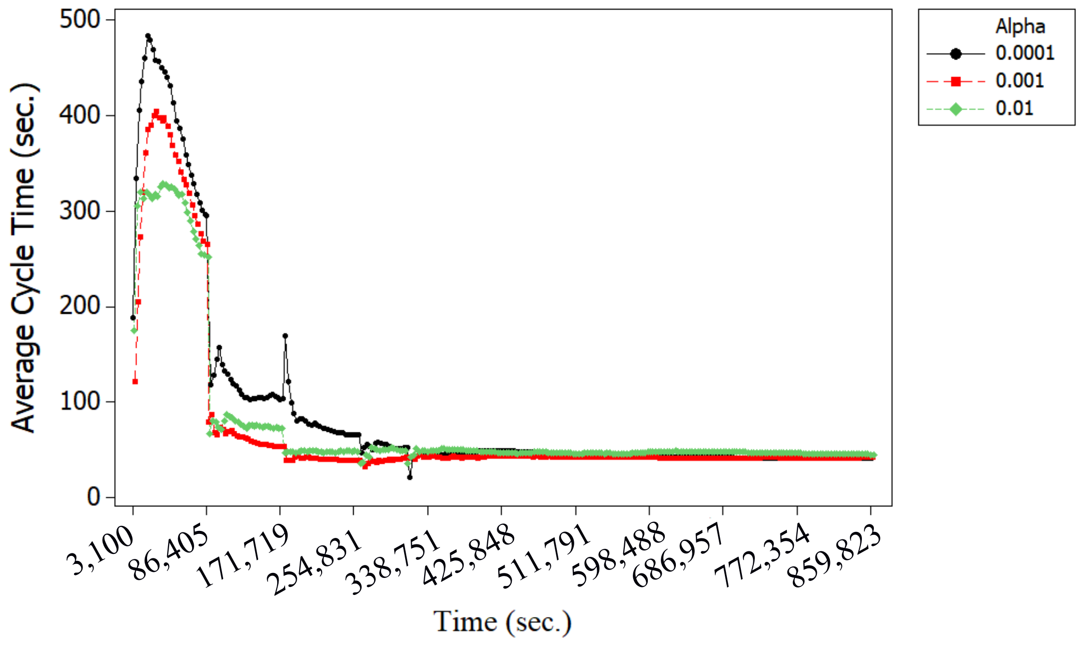


Figure 3.4. DQN results under three different learning rate scenarios

Initially, due to epsilon-greedy approach, the system chooses random actions. This increases exploration to be made and prevents convergence to a local optimum. For this reason, the DQN causes a cost. The static FIFO and SPT rules are assumed to have no computational costs. The cost occurs during learning and may produce worse results compared to static rules. Figure 3.5. shows the computational cost of DQN with the parameters for the warehouse design $T = 5$, $B = 25$, $S = 2$ and DQN parameters of $\alpha = 0.001$, $\gamma = 0.2$, $\epsilon = 1$, $\epsilon_{\text{dec}} = 0.9999$, $\epsilon_{\min} = 0.01$. DQN is compared to SPT in this experiment.

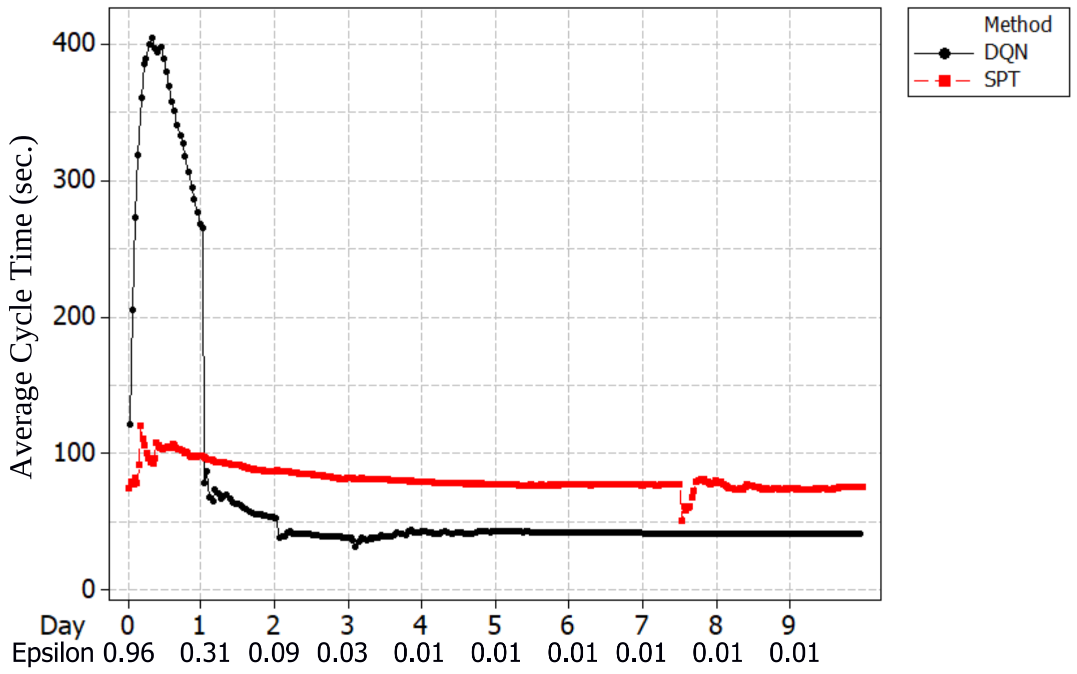


Figure 3.5. Average cycle time comparison for DQN and SPT

As observed in the figure, the DQN starts to outperform SPT in a single day, which is equal to 13,091 transactions on average. Since the applied method is non-episodic, we treat each day as an episode and reset the average cycle times every day, until the epsilon reaches the minimum value. Even though this experiment was made with a relatively low warehouse capacity, the increased number of shuttles also increase the number of agents, meaning that experiencing more states and actions may result in decreased training times.

3.3. Results

As we mentioned before, DQN algorithm is compared with FIFO and SPT rules. In FIFO rule, the shuttles pick the first transaction that entered the system. In SPT rule, the shuttles pick the transaction regarding to shortest travel time considering the address information of the transaction.

We consider eight different warehouse designs, shown in Table 3.1. The tiers, bays and number of shuttles are changed for different experiments. The average cycle times, average flow times, average waiting times, average lift and shuttle utilizations are provided in Table 3.2. The inter-arrival times are adjusted so that one of the resource utilizations is higher than 90%. Within the same experiment, the inter-arrivals are fixed so that we can compare the performance of the rules. The results for FIFO and SPT rules are given at 95% confidence intervals. The number of replications is calculated considering the half-width values. Minimum of three independent replications are made. “N/A” means that the algorithm cannot produce a result. This is because the transaction arrival rates are high so that the system cannot process the transactions.

Table 3.1. Conducted experiments

Warehouse Design #	T	B	S
1	5	25	2
2	8	25	3
3	10	25	4
4	13	25	5
5	5	50	2
6	8	50	3
7	10	50	4
8	13	0	5

Table 3.2. Results of the experiments

WH Design #	**Rule**	***t***	UL_{avg}	USL_{avg}	US_{avg}	C_{avg}	F_{avg}	W_{avg}
1	FIFO	6.6	N/A	N/A	N/A	N/A	N/A	N/A
1	SPT	6.6	47%	51%	94%	98.5 ± 6.59	12.94 ± 0.01	85.56
1	DQN	6.6	37%	44%	85%	41.67	11.53	30.14
2	FIFO	5.2	N/A	N/A	N/A	N/A	N/A	N/A
2	SPT	5.2	69%	74%	90%	61.54 ± 4.2	14.69 ± 0.02	46.86
2	DQN	5.2	55%	66%	81%	36.52	12.96	23.56
3	FIFO	4.2	N/A	N/A	N/A	N/A	N/A	N/A
3	SPT	4.2	86%	86%	90%	89.34 ± 7.32	15.51 ± 0.06	73.83
3	DQN	4.2	67%	80%	80%	46.06	13.47	32.59
4	FIFO	4.4	N/A	N/A	N/A	N/A	N/A	N/A
4	SPT	4.4	90%	92%	83%	57.93 ± 2.97	18.64 ± 0.05	39.29
4	DQN	4.4	72%	90%	74%	44.13	16.07	28.07
5	FIFO	12	42%	57%	97%	567.39 ± 187.65	23.72 ± 2.24	543.66
5	SPT	12	41%	52%	94%	138.18 ± 11.73	22.97 ± 0.08	115.21
5	DQN	12	34%	46%	86%	69.81	20.83	48.98
6	FIFO	9.2	64%	79%	94%	257.14 ± 38.13	26.64 ± 0.05	230.5
6	SPT	9.2	62%	74%	92%	114.32 ± 8.87	25.93 ± 0.06	88.39
6	DQN	9.2	51%	69%	85%	66.12	23.7	42.42
7	FIFO	7.6	N/A	N/A	N/A	N/A	N/A	N/A
7	SPT	7.6	77%	87%	91%	147.44 ± 11.51	28.11 ± 0.08	119.34
7	DQN	7.6	63%	83%	84%	79.44	25.57	53.88
8	FIFO	7.4	N/A	N/A	N/A	N/A	N/A	N/A

8	SPT	7.4	86%	93%	88%	155 ± 8.96	32.73 ± 0.15	122.27
8	DQN	7.4	72%	93%	83%	104.35	30.33	74.02

As seen from the table, FIFO rule generally cannot produce a feasible result because that there is no steady-state condition. For experiment with 5 tiers and 50 bays, DQN outperforms FIFO rule by 88% and for experiment with 8 tiers and 50 bays, DQN produces 74% better results.

Remember that in RL, the algorithm produced very similar results to SPT. Here, DQN outperforms SPT significantly. These two algorithms are compared in Figure 3.6.

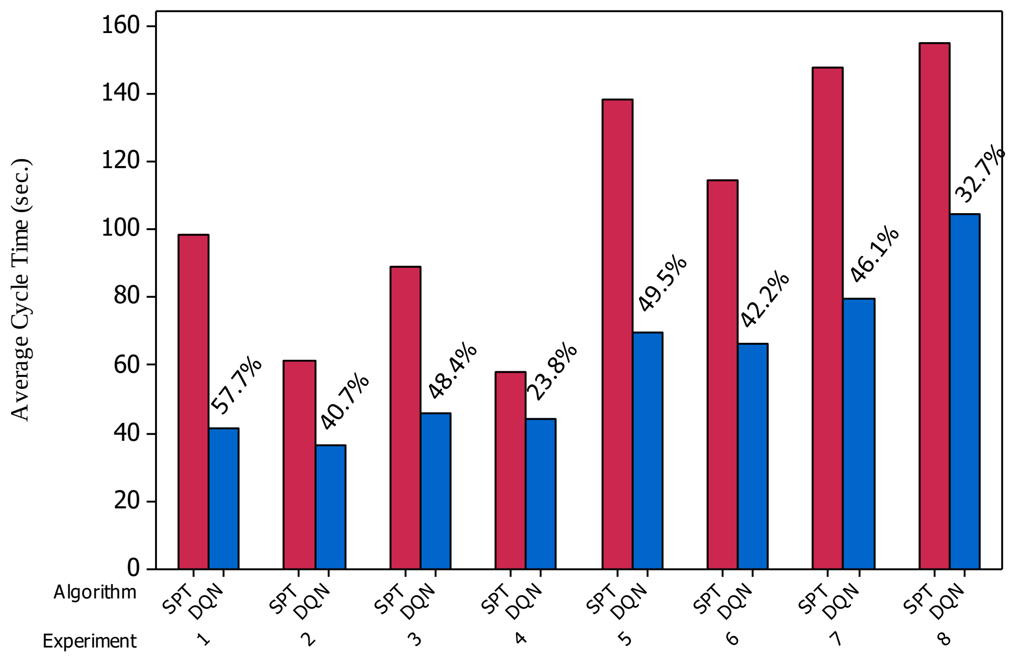


Figure 3.6. Comparison of SPT and DQN rules

On average, DQN algorithm decreases average cycle time by 42.6% and decreases flow time by 10.5%.

CHAPTER 4
CONCLUSIONS AND FUTURE RESEARCH

This thesis studies simulation-based machine learning algorithms (i.e., *Q*-learning and Deep *Q*-learning) for intelligently scheduling of transactions in a tier-to-tier shuttle-based storage and retrieval system. The aim of this study is to implement a machine-learning algorithm for a complex SBS/RS design to decrease average cycle times, flow times and utilizations.

First, we simulate the system using Arena 16 simulation software and apply *Q*-learning method by using the VBA interface. After, we compare the results of the algorithm with FIFO and SPT rule under different warehouse designs. The results show that RL outperforms FIFO rule, but generates very similar results to SPT rule. The study shows a promising result for the further study of implementing Deep *Q*-learning algorithm.

In the Chapter 3, we implement a Deep *Q*-learning method for the same problem. We utilize Python SimPy for simulation modelling. TensorFlow and Keras libraries are used for implementation of the algorithm. The results are compared with well-known FIFO and SPT algorithms. The results show that Deep *Q*-learning outperforms the other algorithm significantly, which is promising for the future of smart industry applications.

In the future, this thesis can be extended by considering state space expand by including attributes of waiting transactions in queue as well as reward function of cycle time. In addition, more experiments can be conducted by including different velocity profiles of the shuttles and lifts in the system. Finally, multiple objectives can be integrated to the reward function such as energy consumptions of transactions as well.

REFERENCES

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

Carlo, H., & Vis, I. (2012). Sequencing dynamic storage systems with multiple lifts and shuttles. *International Journal of Production Economics*, *140*, 844–853. https://doi.org/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/

D’Antonio, G., Bruno, G., Traini, E., & Lombardi, F. (2019). An analytical model to estimate AVS/RS energy consumption. *IFAC-PapersOnLine*, *52*(13), 24–29. https://doi.org/10.1016/j.ifacol.2019.11.086

Dematic. (n.d.). *Dematic Multishuttle*. Retrieved April 20, 2021, from https://www.dematic.com/en/products/products-overview/storage-systems/dematic-multishuttle/

Dou, J., Chen, C., & Yang, P. (2015). Genetic Scheduling and Reinforcement Learning in Multirobot Systems for Intelligent Warehouses. *Mathematical Problems in Engineering*, *2015*, 597956. https://doi.org/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. https://doi.org/10.1080/21693277.2019.1619102

Ekren, B., & Heragu, S. (2011). Simulation based performance analysis of an autonomous vehicle storage and retrieval system. *Simulation Modelling Practice and Theory*, *19*, 1640–1650. https://doi.org/10.1016/j.simpat.2011.02.008

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. https://doi.org/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*(August 2019), 101991. https://doi.org/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. https://doi.org/10.1080/00207543.2020.1720927

Ekren, B. Y., & Akpunar, A. (2021). An open queuing network-based tool for performance estimations in a shuttle-based storage and retrieval system. *Applied Mathematical Modelling*, *89*, 1678–1695. https://doi.org/10.1016/j.apm.2020.07.055

Ekren, B. Y., Akpunar, A., Sari, Z., & Lerher, T. (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. https://doi.org/10.1016/j.apm.2018.06.037

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

Ekren, B. Y., Heragu, S. S., Krishnamurthy, A., & Malmborg, C. J. (2010). Simulation based experimental design to identify factors affecting performance of AVS/RS. *Computers and Industrial Engineering*, *58*(1), 175–185. https://doi.org/10.1016/j.cie.2009.10.004

Ekren, B. Y., Heragu, S. S., Krishnamurthy, A., & Malmborg, C. J. (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. https://doi.org/10.1016/j.cie.2013.12.002

Ekren, B. Y., Sari, Z., & Lerher, T. (2015). Warehouse design under class-based storage policy of shuttle-based storage and retrieval system. *IFAC-PapersOnLine*, *28*(3), 1152–1154. https://doi.org/10.1016/j.ifacol.2015.06.239

Gazori, P., Rahbari, D., & Nickray, M. (2020). 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. https://doi.org/10.1016/j.future.2019.09.060

Ha, Y., & Chae, J. (2018a). Free balancing for a shuttle-based storage and retrieval system. *Simulation Modelling Practice and Theory, 82*, 12–31. https://doi.org/10.1016/j.simpat.2017.12.006

Ha, Y., & Chae, J. (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. https://doi.org/10.1080/00207543.2018.1476787

Jerman, B., Ekren, B. Y., Küçükyaşar, M., & Lerher, T. (2021). Simulation-based performance analysis for a novel avs/rs technology with movable lifts. *Applied Sciences (Switzerland)*, *11*(5), 1–16. https://doi.org/10.3390/app11052283

Küçükyaşar, M., Ekren, B., & Lerher, T. (2020). Cost and performance comparison for tier-captive and tier-to-tier SBS/RS warehouse configurations. *International Transactions in Operational Research*. https://doi.org/10.1111/itor.12864

Lerher, T. (2015). Travel time model for double-deep shuttle-based storage and retrieval systems. *International Journal of Production Research, 54*, 1–22. https://doi.org/10.1080/00207543.2015.1061717

Lerher, T., Ekren, B. Y., Sari, Z., & Rosi, B. (2015a). Simulation analysis of shuttle based storage and retrieval systems. *International Journal of Simulation Modelling*, *14*(1), 48–59. https://doi.org/10.2507/IJSIMM14(1)5.281

Lerher, T., Ekren, B., Dukic, G., & Rosi, B. (2015b). Travel time model for shuttle-based storage and retrieval systems. *The International Journal of Advanced Manufacturing Technology*, *78*. https://doi.org/10.1007/s00170-014-6726-2

Lerher, T., Ekren, B. Y., Sari, Z., & Rosi, B. (2016). Method for evaluating the throughput performance of shuttle based storage and retrieval systems. *Tehnicki Vjesnik - Technical Gazette*, *23*, 715–723. https://doi.org/10.17559/TV-20141022121007

Lerher, T., Ficko, M., & Palčič, I. (2021). Throughput performance analysis of Automated Vehicle Storage and Retrieval Systems with multiple-tier shuttle vehicles. *Applied Mathematical Modelling*, *91*, 1004–1022.

https://doi.org/10.1016/j.apm.2020.10.032

Malus, A., Kozjek, D., & Vrabič, R. (2020). Real-time order dispatching for a fleet of autonomous mobile robots using multi-agent reinforcement learning. *CIRP Annals*, *69*(1), 397–400. https://doi.org/10.1016/j.cirp.2020.04.001

Mao, H., Alizadeh, M., Menache, I., & Kandula, S. (2016). Resource management with deep reinforcement learning. *HotNets 2016 - Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, 50–56. https://doi.org/10.1145/3005745.3005750

Marchet, G., Melacini, M., Perotti, S., & Tappia, E. (2011). Analytical model to estimate performances of autonomous vehicle storage and retrieval systems for product totes. *International Journal of Production Research*, *2011*. https://doi.org/10.1080/00207543.2011.639815

Marchet, G., Melacini, M., Perotti, S., & Tappia, E. (2013). Development of a framework for the design of autonomous vehicle storage and retrieval systems. *International Journal of Production Research*, *51*. https://doi.org/10.1080/00207543.2013.778430

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533. https://doi.org/10.1038/nature14236

Mordor Intelligence. (2020a). *Automated Material Handling (AMH) Market - Growth, Trends, COVID-19 Impact, and Forecasts (2021-2026)*. https://www.mordorintelligence.com/industry-reports/global-automated-material-handling-market-industry

Mordor Intelligence. (2020b). *Automated Storage and Retrieval System (ASRS) Market - Growth, Trends, COVID-19 Impact, and Forecasts (2021-2026)*. https://www.mordorintelligence.com/industry-reports/automated-storage-and-retrieval-systems-market-industry

Ou, X., Chang, Q., & Chakraborty, N. (2019). Simulation study on reward function of reinforcement learning in gantry work cell scheduling. *Journal of Manufacturing Systems*, *50*(October 2018), 1–8. https://doi.org/10.1016/j.jmsy.2018.11.005

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

Sutton, R. S., & Barto, A. G. (2015). An introduction to reinforcement learning. In *A Bradford Book The*. https://doi.org/10.4018/978-1-60960-165-2.ch004

Takahashi, K., & Tomah, S. (2020). Online optimization of AGV transport systems using deep reinforcement learning. *Bulletin of Networking, Computing, Systems, and Software*, *9*(1), 53–57.

Tappia, E., Roy, D., de Koster, R., & Melacini, M. (2016). Modeling, Analysis, and Design Insights for Shuttle-Based Compact Storage Systems. *Transportation Science, 51*(1), 269–295. https://doi.org/10.1287/trsc.2016.0699

Tong, Z., Chen, H., Deng, X., Li, K., & Li, K. (2020). A scheduling scheme in the cloud computing environment using deep Q-learning. *Information Sciences, 512*, 1170–1191. https://doi.org/10.1016/j.ins.2019.10.035

Wang, Y., Mou, S., & Wu, Y. (2015). Task scheduling for multi-tier shuttle warehousing systems. *International Journal of Production Research, 53*(19), 5884–5895. https://doi.org/10.1080/00207543.2015.1012604

Watanabe, M., Furukawa, M., & Kakazu, Y. (2001). Intelligent AGV driving toward an autonomous decentralized manufacturing system. *Robotics and Computer-Integrated Manufacturing, 17*(1–2), 57–64. https://doi.org/10.1016/S0736-5845(00)00037-5

Wei, Z., Zhang, Y., Xu, X., Shi, L., & Feng, L. (2017). A task scheduling algorithm based on Q-learning and shared value function for WSNs. *Computer Networks, 126*, 141–149. https://doi.org/10.1016/j.comnet.2017.06.005

Xue, T., Zeng, P., & Yu, H. (2018). A reinforcement learning method for multi-AGV scheduling in manufacturing. *2018 IEEE International Conference on Industrial Technology (ICIT)*, 1557–1561. https://doi.org/10.1109/ICIT.2018.8352413

Zhao, X., Wang, Y., Wang, Y., & Huang, K. (2019). Integer programming scheduling model for tier-to-tier shuttle-based storage and retrieval systems.

Processes, *7*(4). https://doi.org/10.3390/pr7040223

Zou, B., Xu, X., Gong, Y., & De Koster, R. (2016). Modeling parallel movement of lifts and vehicles in tier-captive vehicle-based warehousing systems. *European Journal of Operational Research*, *254*(1), 51–67. https://doi.org/10.1016/j.ejor.2016.03.039

APPENDIX 1 – Link to the models and application

The models and applications made on Arena simulation software and Python for this work is submitted on the link below.

https://github.com/bartuarslan/thesiswork