COMPLEX SYSTEM MODELING AND SIMULATION
ISSN 2096-9929 05/06 pp 131–144
Volume 1, Number 2, June 2021
DOI: 10.23919/CSMS.2021.0013

Scheduling Storage Process of Shuttle-Based Storage and Retrieval Systems Based on Reinforcement Learning

Lei Luo, Ning Zhao*, and Gabriel Lodewijks

Abstract: The Shuttle-Based Storage and Retrieval System (SBS/RS) has been widely studied because it is currently the most efficient automated warehousing system. Most of the related existing studies are focused on the prediction and improvement of the efficiency of such a system at the design stage. Hence, the control of existing SBS/RSs has been rarely investigated. In existing SBS/RSs, some empirical rules, such as storing loads column by column, are used to control or schedule the storage process. The question is whether or not the control of the storage process in an existing system can be improved further by using a different approach. The storage process is controlled to minimize the makespan of storing a series of loads into racks. Empirical storage rules are easy to control, but they do not reach the minimum makespan. In this study, the performance of a control system that uses reinforcement learning to schedule the storage process of an SBS/RS with fixed configurations is evaluated. Specifically, a reinforcement learning algorithm called the actor-critic algorithm is used. This algorithm is made up of two neural networks and is effective in making decisions and updating itself. It can also reduce the makespan relative to the existing empirical rules used to improve system performance. Experiment results show that in an SBS/RS comprising six columns and six tiers and featuring a storage capacity of 72 loads, the actor-critic algorithm can reduce the makespan by 6.67% relative to the column-by-column storage rule. The proposed algorithm also reduces the makespan by more than 30% when the number of loads being stored is in the range of 7–45, which is equal to 9.7%–62.5% of the systems' storage capacity.

Key words: Shuttle-Based Storage and Retrieval System (SBS/RS); reinforcement learning; scheduling

1 Introduction

1.1 Background

During the COVID-19 pandemic, online shopping has become the norm. Warehouses play a very important role in facilitating online shopping, and highly efficient and autonomous facilities are urgently needed. An increasing number of distribution centers are utilizing Shuttle-Based Storage and Retrieval Systems (SBS/RSs) in their high-density warehouses to meet the increasing throughput. Figures 1 and 2 show the side and top views of a typical SBS/RS, respectively. The system consists of a lift that raises or lowers loads and a fleet of shuttle carriers in each tier of the rack. The lift does the vertical movement of the loads while the shuttle carriers do the horizontal movement. When properly controlled, an SBS/RS can be very efficient[1].

Different layout configurations and control procedures

• Lei Luo and Ning Zhao are with School of Mechanical Engineering, University of Science and Technology Beijing, Beijing 100083, China. E-mail: 369662347@qq.com; nickzhao@me.ustb.edu.cn.
• Gabriel Lodewijks is with School of Aviation, University of New South Wales, Sydney 2052, Australia. E-mail: lodewijks@unsw.edu.au.
* To whom correspondence should be addressed.
Manuscript received: 2021-04-07; revised: 2021-05-25; accepted: 2021-05-31

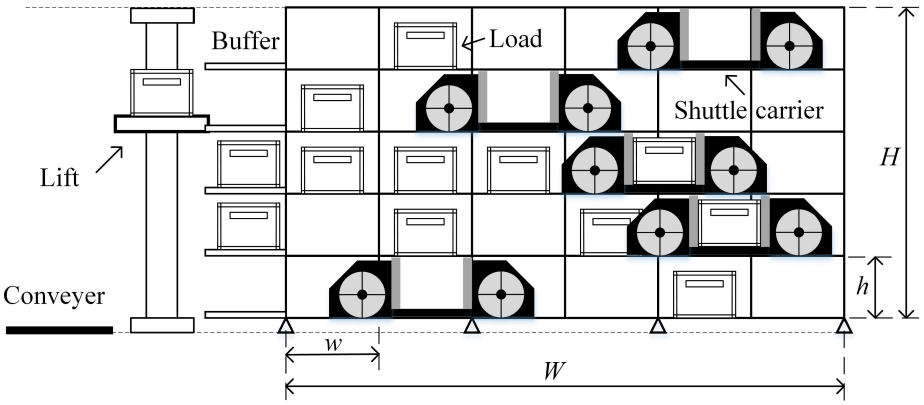

Fig. 1 Side view of an SBS/RS.

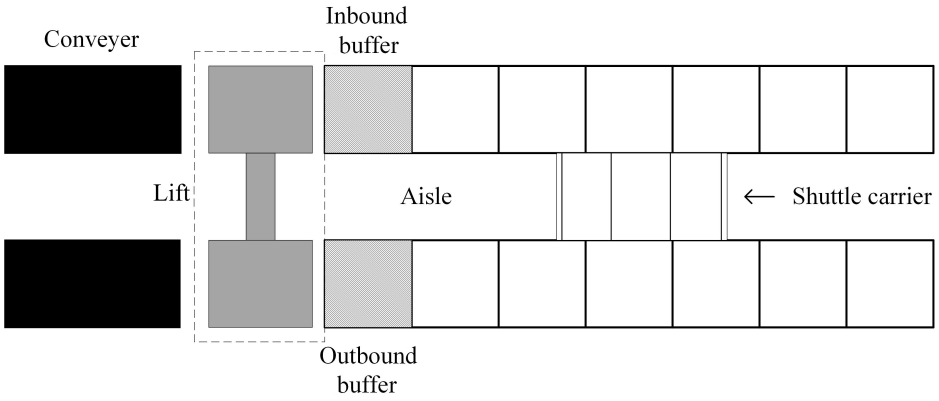

Fig. 2 Top view of an SBS/RS.

in SBS/RSs may lead to different performance outcomes. The length of a rack can be expected to affect the average time needed to store a load in it. The height of the rack also affects the time needed by a lift to vertically move a load. However, controlling these moves simultaneously is a challenging task, and predicting and improving the performance of the systems are difficult because of their complexity. Therefore, most previous studies on SBS/RSs have focused on the prediction and improvement of the throughput of these systems. Malmborg[2] (2002) was the first to investigate Autonomous Vehicle Storage and Retrieval Systems (AVS/RSs), which use the same technology as SBS/RSs. A continuous Markov chain that models horizontal and vertical material flows was used to calculate the expected storage-retrieval cycle time and throughput.

Kuo et al.[3] (2007) proposed M/G/V and M/G/L queuing models to estimate the waiting time of vehicles and lifts. Fukunari and Malmborg[4] (2008) proposed a network queuing approach to predict the cycle time of storage and retrieval transactions of AVS/RSs. Ekren et al.[5] (2010) developed simulation models to identify the factors affecting the performance of AVS/RSs and performed regression analysis to determine the relationship between rack configuration and system performance. Ekren and Heragu[6] (2011) utilized a simulation method to determine the optimal number of vehicles and lifts in a system that results in high performance under various predefined storage rack configuration scenarios. Ekren et al.[7] (2013) proposed an analytical model, called the semi-open queuing network model, to determine system performance. They applied this model to the analysis of a warehouse in France that utilizes an AVS/RS. Marchet et al.[8] (2012) presented an open queuing network approach to estimate the performance (transaction cycle time and waiting time) of an SBS/RS. Lerher[9] (2013) studied energy regeneration models for SBS/RSs. In the study, the energy consumption of the SBS/RS was identified as another performance indicator in the design of a warehouse, and a model considering the reduction of energy consumption was proposed. Lerher[10] (2016) proposed a travel time model for an SBS/RS. On the basis of the proposed travel time model, the expected cycle time for the single- and dual-command cycles of the SBS/RS was estimated. Lerher[11] (2016) combined

previous studies and proposed an estimation model for calculating the throughput and energy consumption of an SBS/RS. Zhao et al.[12] (2016) used a simulation method to identify the best configuration on the basis of numerous experiments. They presented a simulation model of a multielevator tier-captive SBS/RS and analyzed its throughput performance.

Many researchers also have focused on special configurations or special control policies. Carlo and Vis[13] (2012) were the first to study storage systems with multiple lifts and shuttles. They proposed an integrated look-ahead heuristics strategy to study the scheduling problem of two nonpassing lifts in an SBS/RS. Their strategy considerably improves the system in terms of total handling times and throughput relative to a single lift system. Zhao et al.[14] (2018) extended Carlo and Vis' research[13] by considering the acceleration and deceleration of lifts. They proposed a collision-free lift trajectory prediction method and found it to be more effective than the constant velocity method in decreasing the makespan. They also used the genetic algorithm to find the best order of all tasks. Similarly, Xin et al.[15] adopted the genetic algorithm in their work on a collision-free multirobot station. Lerher[10] (2016) proposed a travel time model of a double-deep SBS/RS. This special model expands the storage capacity of such systems. Lerher et al.[16] (2021) later presented analytical travel time models for the computation of cycle times and throughput performance of AVS/RSs with multitier shuttle vehicles. They considered a multitier shuttle vehicle that travels in a horizontal direction and moves in a vertical direction simultaneously in a storage rack. Tappia et al.[17] (2016) proposed a novel queuing network model to estimate the performance of single-tier and multitier shuttle-based compact systems. D'Antonio et al.[18] (2018) proposed analytical models for evaluating the performances of deep-lane AVS/RSs. Zou et al.[19] (2016) focused on the study of the parallel movement of lifts and shuttles. They proposed a parallel processing policy and reported a significant performance improvement. All the aforementioned studies aimed to improve the performance of SBS/RSs by finding other efficient layout configurations or control policies.

In general, capacity or throughput is the most important key performance indicator, because it represents the efficiency of the system. Another way to analyze throughput is by studying the makespan, which refers to the duration of a sequence of tasks. The shorter the makespan, the more efficient the system. Therefore, most researchers have proposed methods to predict or evaluate the performance of SBS/RSs to guide distribution centers in the design of new versions of such systems. However, some distribution centers may have already been utilizing SBS/RSs. In such distribution centers, the layout and scheduling strategies have been determined. Hence, a key issue is to improve the efficiency of existing systems. This gap may be addressed through further research.

1.2 Reinforcement learning in scheduling

In this work, reinforcement learning is applied to improve an existing system's performance. For an existing system, changing its layout is generally difficult because doing so is a costly and time-consuming endeavor. Therefore, the scheduling control should be modified to improve the performance of the system.

With different columns and different tiers in the rack of an SBS/RS, the makespan of storing a load at a certain storage location in the rack depends specifically on that location. Consider the process of storing a series of loads column by column. When the first load arrives, the control system needs to determine a storage place following the rules used. Thereafter, a series of movement commands are issued to store the load in the selected storage place. When the succeeding loads arrive, the control system performs the same process of finding a storage place and issuing movement commands. This type of control is usually based on some empirical, rigid rules. One empirical rule for the control system, for example, is always storing loads column by column on one side of the shelf. Further details about empirical rules are discussed in Section 2. As empirical rules are robust, their application tends to be relatively simple. However, they may not always be very efficient because the control system does not gain experience over time. Therefore, in this work, the application of reinforcement learning, which allows the control system to learn from previous experiences, will be studied.

Reinforcement learning is an area of machine learning that is concerned about how intelligent agents ought to take actions in an environment so as to maximize the notion of cumulative reward. The cumulative reward in an SBS/RS is to maximize the throughput or minimize the makespan. Reinforcement learning is one of the three basic machine learning paradigms, with the two other ones being supervised learning and unsupervised learning. Reinforcement learning is now widely used in image recognition, data processing, decision making,

and many other fields. Alpha Go[20] (2016) is a globally popular Go player that uses reinforcement learning to learn how to play Go. Following full training, it can beat professional human Go players. To some extent, the game Go is similar to the storage process of an SBS/RS. At the beginning of the game, the player needs to carefully observe the game board to obtain useful information, such as which board positions are occupied by his or the opponent's pieces, and on which position the player can put the next piece. The player formulates many ideas about the possible outcomes of the succeeding moves on the basis of the combination of information available. After considering all options available, the player must then decide on the position on which to place the specific piece. After the opponent's turn, the player goes through the same process, including observing, deciding, and executing. Such a process continues until the end of the competition. Similarly, for the storage process in an SBS/RS, the first step is to observe. When a load arrives at the input point, information about empty storage places and the location of the lift and shuttle carriers should be obtained. The second step is to decide. The reinforcement learning algorithm, which plays the same role as the player in Go, decides and chooses a storage place for this load. The third step for the SBS/RS is to execute and store the load in the selected storage place. This loop repeats until all the loads are stored.

In sum, the scheduling and storage procedure in an SBS/RS involves observing, deciding, and then executing. Empirical rules and other existing methods for controlling the storage process make a series of decisions for all loads. The time needed to execute these decisions varies for each rule because different rules follow different logics in decision making. Contrary to rigid empirical rules, reinforcement learning is a flexible, intelligent controller that learns from the environment through observing and makes decisions to help the system minimize the makespan.

1.3 Objectives

As mentioned in Sections 1.1 and 1.2, changing the layout of an existing SBS/RS is difficult. Therefore, exploring different methods to schedule the storage process is necessary to improve system performance. Empirical rules, such as storing loads column by column, are easy but rigid. Reinforcement learning is a flexible method that can make sound decisions and improve itself. It can also improve the scheduling of the storage process of an SBS/RS.

The objective of this work is to study the effects of applying reinforcement learning to the makespan of the scheduling and storage process of an SBS/RS. The actor-critic (AC) reinforcement learning algorithm is used. A three-dimensional SBS/RS model is developed to provide a learning environment for the AC algorithm to train its neural networks. Experiments are conducted to determine the efficiency of the reinforcement learning algorithm relative to empirical rules.

2 Problem Statement

2.1 Storage scheduling environment

The SBS/RS studied in this work is a common one that comprises a lift at the Input/Output (I/O) side of the rack and several shuttle carriers at each tier of the rack. These shuttle carriers serve their own tiers. A typical storage transaction starts with the arrival of a new load. After the control center of the system decides where the load should go, the lift goes to the I/O layer to obtain the load and then moves to the target tier. When the lift arrives at the determined level, it needs to check whether the inbound buffer at this level is occupied or not. The lift then places the load into the inbound buffer when it is empty. The shuttle carrier at the target level continues the storage transaction by obtaining the load from the inbound buffer and placing it in the correct storage position. The next storage transaction starts before the end of the previous one. Specifically, it starts when a new load arrives, and the lift is released regardless of the state of the shuttle carriers. The lift and shuttle carriers run according to the commands from the controller. That is, their movements depend on the determined storage location for a load. Therefore, storage scheduling entails identifying the best storage place for each load and minimizing the makespan of the total scheduling and storage process.

2.2 Three-dimensional model for SBS/RS

A three-dimensional model of an SBS/RS was built with the software Plant Simulation on the basis of a discrete event simulation (Fig. 3). The model is a parameterized model so that the number of tiers and columns can be easily adjusted.

The following assumptions were made when building the SBS/RS model:

(1) The model consists of storage racks on both sides (left and right).

(2) The position of the input and output points is at the second tier of the rack. The input point is set for a

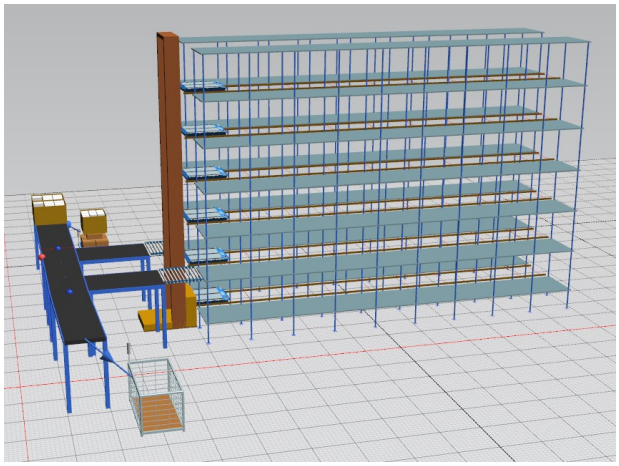

Fig. 3 Model of an SBS/RS.

load that needs to be stored, and the output point is set for load retrieval.

(3) The buffer position is at the beginning of each tier.
(4) The lift and shuttle carriers run at a constant speed.
(5) The shuttle carriers only work on their own tiers.
(6) The SBS/RS has one control system.

2.3 Empirical rules in SBS/RS scheduling

Several empirical storage rules can be used to schedule the storage process. Table 1 shows four typical rules. Under the rule "ColByCol", the load is be stored column by column on one side of the rack until the side is full. Then, the succeeding loads are similarly stored at the other side of the rack. The rule "ColByCol_alternate" is similar to "ColByCol", but it selects storage positions alternately on both sides of the rack. The rule "TierByTier" indicates that the storage positions are selected tier by tier and side by side. The "TierByTier_alternate" rule has the same meaning as "ColByCol_alternate", it stores loads row by row alternately on both sides of the shelf.

The SBS/RS model used in the simulation to study the performance of the four rules consists of six tiers and six columns. The system has a storage capacity of 72 loads (6 tiers ×6 columns ×2 sides). The height and width of each storage unit of the shelf are equal to 1.2 m. The speed of the lift and shuttle carriers is 1 m/s and is assumed to be constant. Two input variables are used in each experiment: the number of loads to be stored and the rule used. The output variable is the makespan, which is the time needed to store a certain number of loads under a certain rule. The outcomes of the experiments are shown in Fig. 4 and Table 2. The x-axis represents the number of loads stored, and the y-axis represents the makespan.

From the experiment results, the first conclusion obtained is that different rules result in a different makespan. From this experience, the rule ColByCol performs better than any other rules. Relative to that

Table 1 Several empirical storage rules.

Number	Rule	Notation
1	Store loads column by column on one side of the shelf	ColByCol
2	Store loads column by column alternatively on both sides of the shelf	ColByCol_alternate
3	Store loads row by row on one side of the shelf	TierByTier
4	Store loads row by row alternatively on both sides of the shelf	TierByTier_alternate

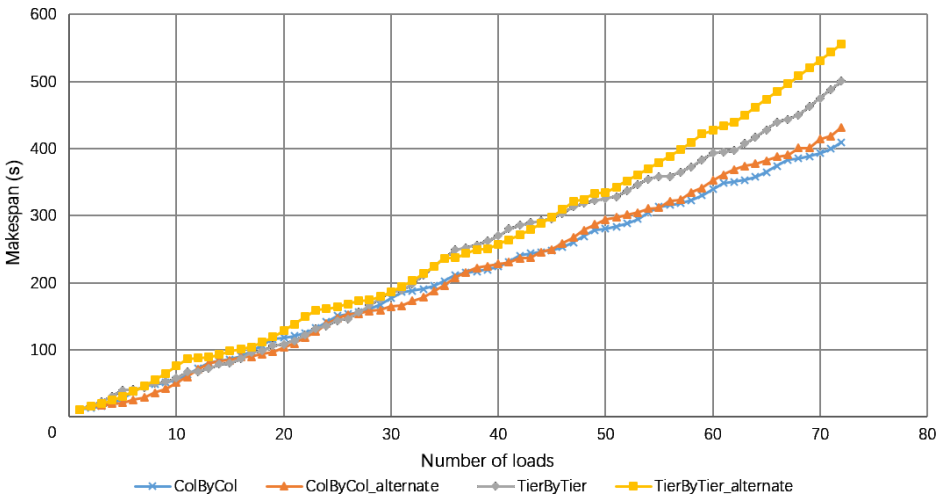

Fig. 4 Makespan as a function of number of loads using the four rules.

Table 2 Makespan for each empirical storage rule.

Number	Rule	Total time (s)
1	ColByCol	409.07
2	ColByCol_alternate	431.27
3	TierByTier	500.27
4	TierByTier_alternate	556.27

of TierByTier_alternate and ColByCol_alternate, the makespan of ColByCol is reduced by 26.5% and 5.14%, respectively.

The second conclusion is that the performance of the scheduling and storage process following a specific rule depends on the number of loads to be stored. Hence, a rule may perform well when the number of loads stored is high, but performs less when the stored number is low. Take for example two polylines of ColByCol and ColByCol_alternate. The orange polyline is lower than the blue one at the beginning. This condition indicates that ColByCol_alternate performs better than ColByCol. However, the result is reversed upon the arrival of the 45th load, with ColByCol showing a better performance. That is, if only 45 or fewer loads need to be stored, then ColByCol_alternate is the best rule. However, if the number of loads to be stored is more than 45, then ColByCol is the better.

Empirical rules can effectively control the scheduling and storing process, but each one cannot outperform another by more than 26.5%. Moreover, no rule performs the best during the whole storage process. Therefore, a highly efficient and smart method is required. In this work, the application of the reinforcement learning algorithm to the storage process is evaluated to gage its performance.

3 Reinforcement Learning in Storage Scheduling

3.1 Actor-critic algorithm

In this work, the performance of the reinforcement learning algorithm in a scheduling and storage process of an SBS/RS is studied. As described previously, reinforcement learning is an area of machine learning that concerns about how intelligent agents take actions in an environment so as to maximize the notion of cumulative reward[21]. Two neural networks, namely, the actor and the critic, are used in the AC reinforcement algorithm. These neural networks enable the algorithm to learn how to maximize the notion of cumulative reward over many steps.

Figure 5 schematically shows the learning process. The process starts with an initiation phase. The agent observes the environment, which is the status of the SBS/RS in this case. On the basis of the observation, the agent picks an action by using the knowledge from previous actions and then proceeds to implement the selected action. Here, the actions are storage decisions. After the implementation of the action, the agent evaluates the new state of the system. If the termination condition, e.g., the makespan of the storage process is no longer decreasing, is met, then the learning process ends.

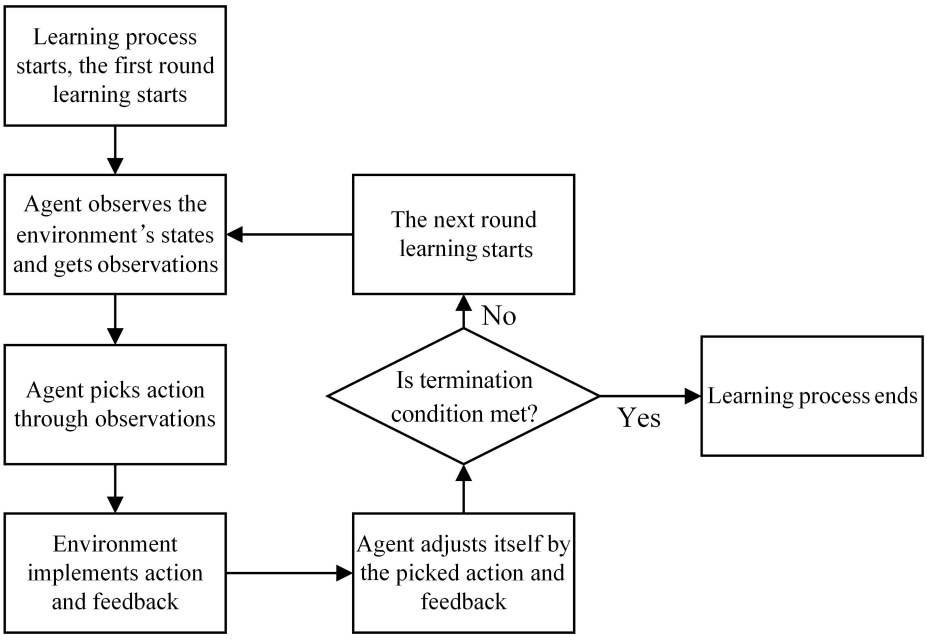

Fig. 5 Learning process.

If the termination condition is not met, then the next round of the learning process starts with the knowledge gained in the previous round.

Figure 6 shows in detail how the AC algorithm works. The AC algorithm consists of two neural networks. The first is called "actor", which picks an action from all available actions on the basis of observations. The second is called "critic", which estimates the score of the determined action selected by the actor[22]. This neural network also scores the action on the basis of observations. The neural network is a data fitting tool. A large number of data are received and fitted through multiple linear or nonlinear layers. The neural network then determines the fitting result of the input data. In the AC algorithm, these two neural networks are updated after each round of simulation to enhance their fit according to the series of actions, scores, and rewards. The rewards come from the effect of the implementation of the actions.

3.2 Observation values in SBS/RS scheduling

As mentioned in the last paragraph of Section 3.1, observations play a vital role. Valid observations can be used by the AC algorithm to make good decisions. By contrast, invalid observations have no use. In the AC algorithm, four types of observations are used to describe the current state of the SBS/RS: (1) current Response Time Matrix (RTM), (2) last few successive RTMs, (3) available storage place expressed in a matrix (permit position matrix), and (4) the last few decisions expressed in a matrix (last selected position matrix). The first observation is the current RTM, which represents the response time for the incoming load. The second observation is the last few RTMs. This observation includes several RTMs of the last few successive loads, so that the last few states of the SBS/RS can be considered. The third observation is the permit position matrix, which describes the available storage place. A value of 0 indicates an available place, while a value of 1 indicates an occupied place in the matrix. The last one is the last selected position matrix, which contains the information about the last few selected storage places. A value of 0 represents the selected position.

The RTM plays a fundamental role in the AC algorithm. When the target storage location of a new load is known, the conditions of the lift and shuttle carriers should be evaluated to predict the time needed to store the new load. However, the conditions of the lift and shuttle carriers of the SBS/RS are difficult to describe in a brief mathematical language because of their dynamic nature and accompanying uncertainties. For example, when the lift is on the third floor of the rack, its status of either waiting for a request or just passing by is difficult to establish. When a shuttle carrier is carrying a load at a certain place, its state of attending to another request in its task queue is not easy to distinguish despite the similarity in the description of the location. If the shuttle carrier has another request in its task queue, then it must finish the request before responding to a new load. If it has no other task, then it can respond to a new load immediately.

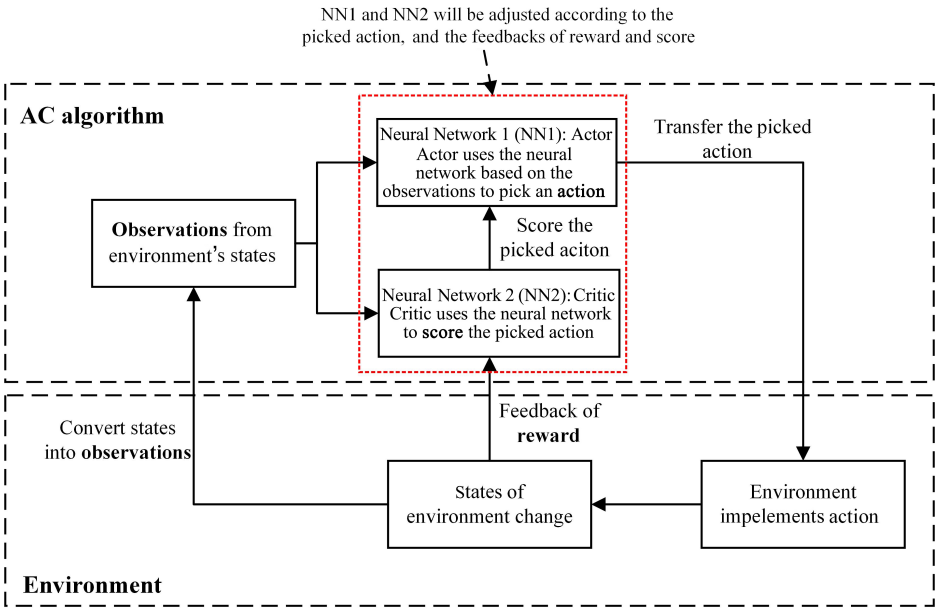

Fig. 6 Function of AC algorithm.

Given the fact that the conditions of the lift and shuttle carriers of the SBS/RS are difficult to describe, an RTM is proposed herein. The proposed RTM can clearly describe the effects of the two facilities by representing the response time of all available storage places to the new load. Before the storage place for a new load is selected, a series of response times are calculated under the assumption that the load is deposited in every available storage place. Figure 7 presents a practical example of the calculation process. Assume the availability of one storage place at $\mathrm{Row}_i\mathrm{Col}_j$. By following the six steps described herein, the response time of the available storage place can be determined.

Step 1: Obtain the current time ($\mathrm{time_{start}}$) and free time ($\mathrm{time_{lift\_free}}$) of the lift.

Step 2: Determine the start time ($\mathrm{time_v}$) of the vertical motion of the lift. If $\mathrm{time_{lift\_free}}$ is greater than $\mathrm{time_{start}}$, then the lift cannot elevate the load at this point. If $\mathrm{time_{lift\_free}}$ is less than $\mathrm{time_{start}}$, then the lift can elevate the load immediately. That is

$$\mathrm{time_v} = \max(\mathrm{time_{start}}, \mathrm{time_{lift\_free}}).$$

Step 3: Calculate the elevating time ($\mathrm{time_{lifting}}$) of the lift. Consider the position of the lift and its request in the task queue in calculating the $\mathrm{distance_{lift}}$ that the lift needs to run. Therefore,

$$\mathrm{time_{lifting}} = \mathrm{distance_{lift}}/\mathrm{speed_{lift}}.$$

Step 4: Similar to Step 2, determine the start time ($\mathrm{time_h}$) of the horizontal motion of the shuttle carrier by using the following equation:

$$\mathrm{time_h} = \max(\mathrm{time_v} + \mathrm{time_{lifting}}, \mathrm{time_{shuttle\_free}}),$$

where $\mathrm{time_{shuttle\_free}}$ represents the free time of the shuttle at tier i.

Step 5: Calculate the moving time ($\mathrm{time_{moving}}$)of the shuttle carrier,

$$\mathrm{time_{moving}} = \mathrm{distance_{shuttle}}/\mathrm{speed_{shuttle}},$$

where $\mathrm{distance_{shuttle}}$ represents the total distance that the shuttle needs to run on the basis of the shuttle's current position and the target position.

Step 6: Denote the response time of the storage place as $\mathrm{Row_iCol_j}$. Then $\mathrm{R_{time}} = \mathrm{time_h} + \mathrm{time_{moving}}$.

After these six steps, another storage place is selected, and Step 1 is rerun until the response time of each available storage place is obtained. All the series of response times make up the RTM. If the storage place is unavailable, then the response time is infinite.

3.3 Application of AC algorithm to storage scheduling

The three-dimensional SBS/RS model shown in Fig. 3 plays the role of the environment in the experiment introduced in Section 4. The AC algorithm plays the role of an agent that makes decisions about where each load should be stored. The learning process is shown in Fig. 8. A learning process consists of many rounds of simulations. In one round of the simulation,

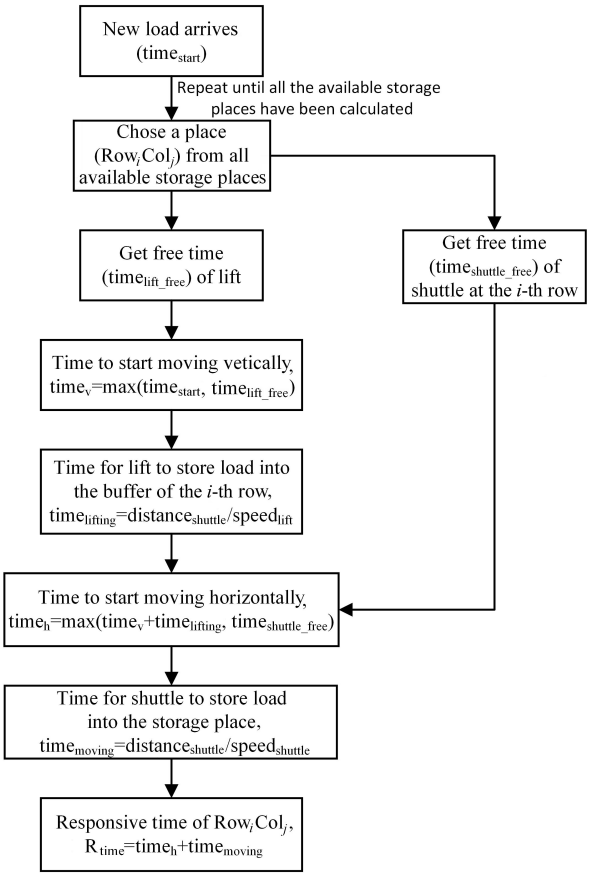

Fig. 7 Calculation of RTM.

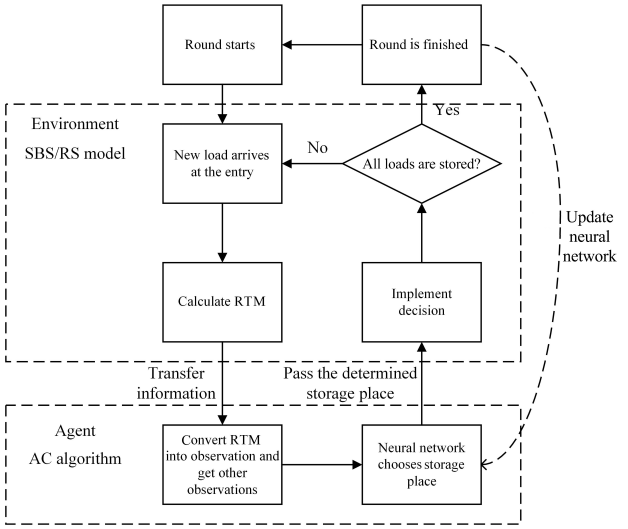

Fig. 8 Reinforcement learning in action on SBS/RS.

the learning process starts when a load arrives at the entrance of the rack. The model collects the required information, the most important is the RTM. Then, all information is sent to the agent, an AC algorithm coded in Python. The AC algorithm then converts the received information into a standard format. The neural network serving as an actor picks an action (in the SBS/RS, an action is a precise storage place), which is then scored by the neural network serving as a critic. The determined action is forwarded to the model so that the load knows its assigned storage place. After the load leaves the entrance, the next load follows, and the process is repeated. When all the loads are stored, this round of simulation is deemed complete, and the next round starts after the neural networks in the AC algorithm are updated.

The two neural networks and the proposed algorithm are built as follows:

(1) The actor neural network, which is usually called the policy network, consists of three convolution layers and a linear layer.

(2) The critic neural network, which is called the value network, consists of four convolution layers and a linear layer. The first three convolution layers are shared with the policy network.

(3) Four types of observations are used. The current RTM and permit position matrix only contain one matrix, and they are relatively fixed. The last few RTMs contain several matrices depending on the number of former states of the system intended to be used in the neural network. A total of 35 matrices are used in the system comprising six columns and six tiers. Hence, the resulting decision is based on the situations of the last 35 loads.

(4) The learning rate is related to the changing amplitude of the parameters of the neural networks and thus affects the networks' fitting capacity. Its initial value is 0.004, and it decreases by 30% every 500 rounds in the learning process.

4 Experiment and Discussion

4.1 Learning process of AC algorithm

A training experiment consisting of 2000 rounds was conducted to train the AC algorithm for the SBS/RS with six columns and six tiers (Fig. 3). Figure 9 shows the results of the training process. The x-axis represents the number of training rounds, and the y-axis represents the time which is spent on storing all 72 loads in each round. At the beginning of the training, all parameters in the neural networks were initialized by the "kaiming_normal_" method, which can make these parameters increasingly uniform. When the training process reached about 700 rounds, the time spent on the scheduling and storage process decreased greatly. After 1000 rounds, the results began to converge. This convergence indicated that the AC algorithm controlled the scheduling and storage process effectively.

4.2 Experiment and analysis

As discussed previously, the four empirical storage rules are ColByCol, ColByCol_alternate, TierByTier, and TierByTier_alternate. The last two rules did not perform particularly well and are thus not discussed further here. A new empirical storage rule called the greed rule was introduced following the proposal of the RTM. For each load under the control of the greed rule, the shortest time spent on reaching the storage place according to the RTM was chosen.

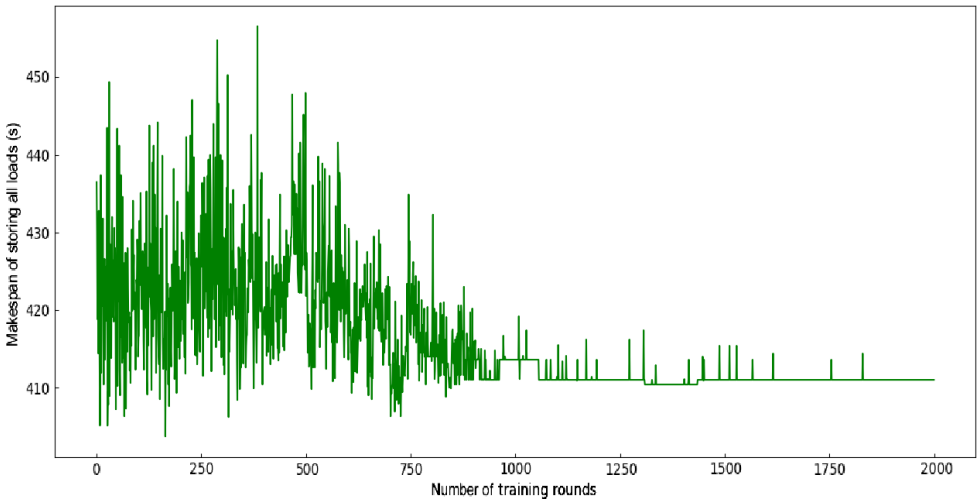

Fig. 9 Training process of AC algorithm on SBS/RS.

The previous experiment described in Section 2.3 was repeated but modified with new experiments following the greed rule and using the AC algorithm. The results are shown in Table 3 and Fig. 10. Figure 10 clearly shows that the greed rule performed better than the ColByCol and ColByCol_alternate rules for all numbers of loads. However, it was outperformed by the AC algorithm. The makespan of the AC algorithm decreased by 6.67%, 1.71%, and 2.55% relative to the makespan of the ColByCol_alternate, ColByCol, and greed rules, respectively.

Table 3 Total time spent by empirical storage rules, the greed rule, and AC algorithm.

No.	Rule	Total time (s)
1	ColByCol_alternative	431.27
2	ColByCol	409.07
3	Greed	412.61
4	AC algorithm	402.07

The reduction of the makespan of the AC algorithm was calculated and compared with that of the other rules to determine the degree of superiority of the AC algorithm in the scheduling and storage process. The results are shown in Fig. 11. The reduction was calculated with the following equation:

$$\text{reduction} = (\text{time}_{\text{other\_rule}} - \text{time}_{\text{AC}})/\text{time}_{\text{other\_rule}},$$

where $\text{time}_{\text{other\_rule}}$ means the makespan of empirical rules and greed rule, and time_{AC} means the makespan of AC algorithm.

When the value of the reduction was greater than 0, the makespan of the AC algorithm was less than that of the other rules. Figure 11 shows that the AC algorithm generally performs well, especially when

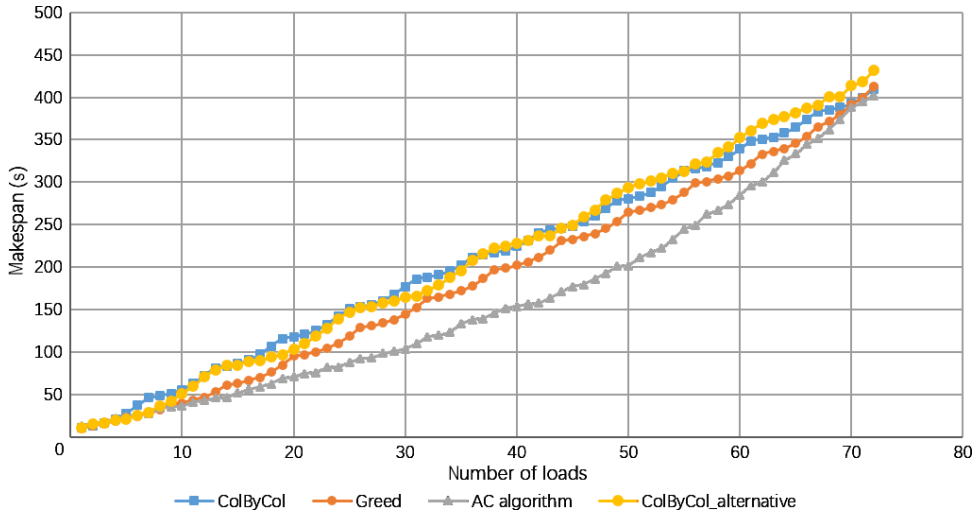

Fig. 10 Makespan of empirical rules, greed rule, and AC algorithm under six tiers and six columns model.

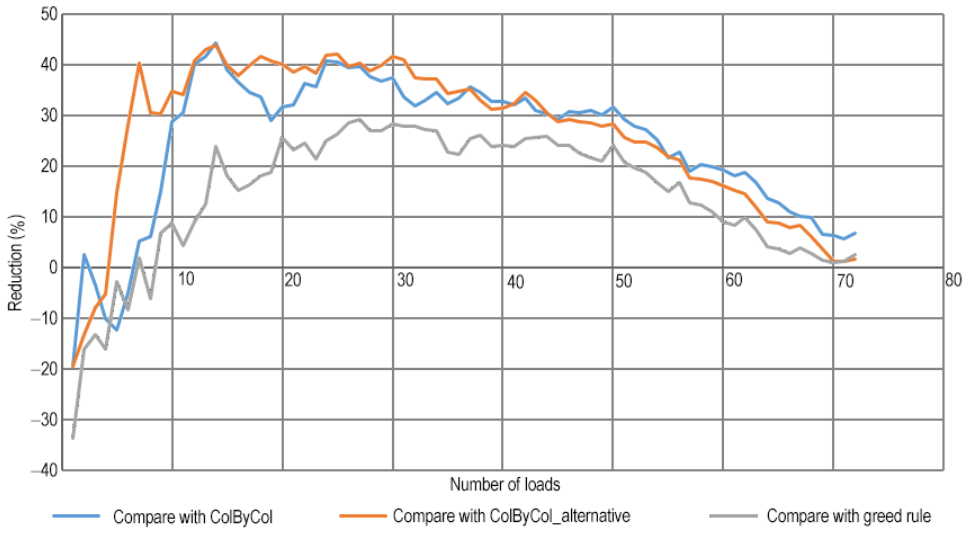

Fig. 11 Makespan of AC algorithm compared with other rules under six tiers and six columns model.

the number of stored loads exceeds 7. At loads greater than 7, the AC algorithm shows a 30% reduction in the makespan relative to ColByCol. When the number of stored loads ranges from 7 to 45, which equates to 9.7% to 62.5% of the system's storage capacity, the AC algorithm still presents a 30% reduction in the makespan. When the number of loads is between 45 and 64, the reduction in makespan still exceeds 10%. As for the greed rule, it is better than ColByCol and ColByCol_alternate rules. However, the AC algorithm shows a 20% reduction in the makespan relative to the greed rule when the number of loads is in the range of 19–52. This result indicates that the AC algorithm is superior to the empirical storage rules for the full range of loads to be stored.

4.3 Experiments with a different SBS/RS configuration

As mentioned previously, the AC algorithm outperforms the empirical storage rules. To investigate whether or not this superiority depends on the configuration of the SBS/RS chosen, this study conducts another experiment involving a model of the SBS/RS containing six tiers and eight columns. The total number of storage places therefore is increased from 72 to 96. Figures 12 and 13 show the results, which indicates that the AC algorithm again outperforms the other rules over the full range of loads to be stored. The makespan of the AC algorithm decreases by 10.97%, 1.65%, and 3% relative to the makespan of ColByCol_alternate, ColByCol, and greed

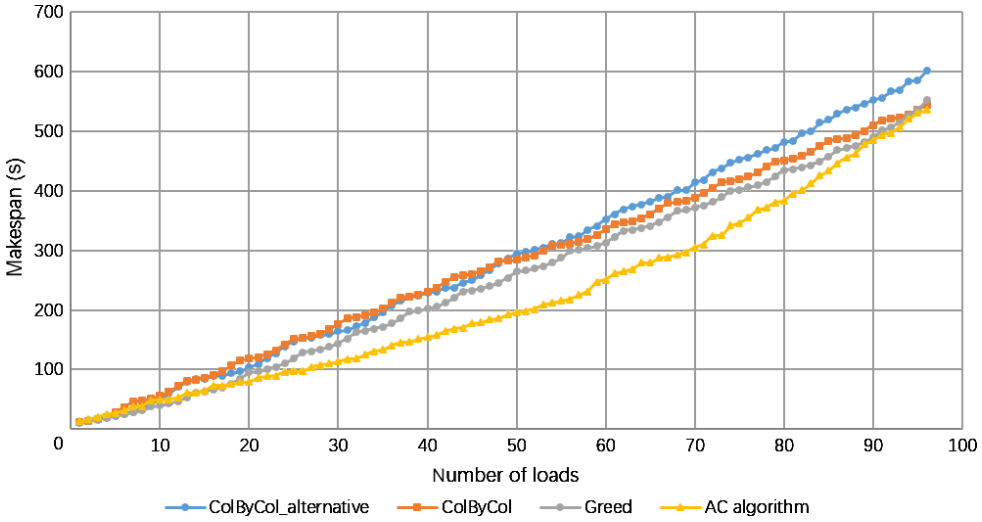

Fig. 12 Makespan of empirical rules, greed rule, and AC algorithm under six tiers and eight columns model.

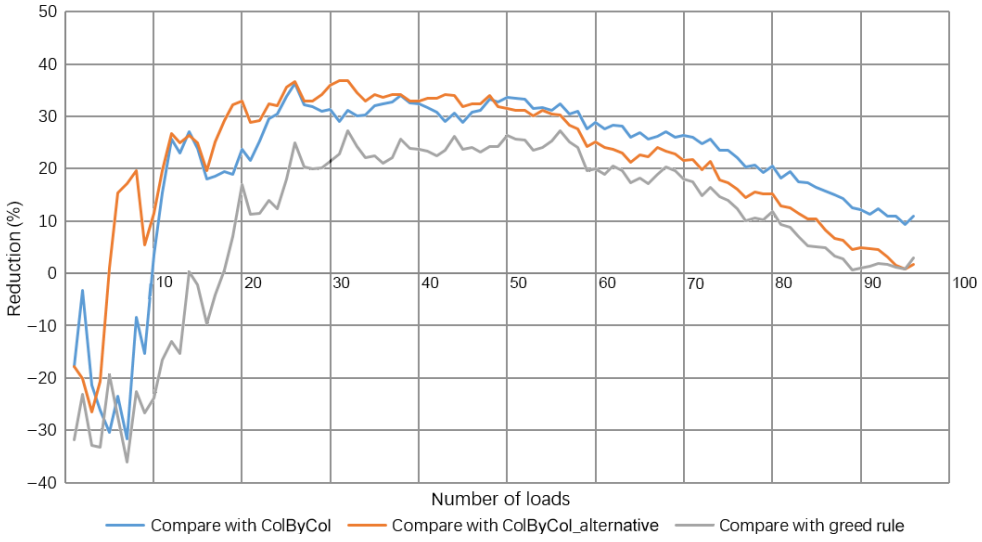

Fig. 13 Makespan of AC algorithm compared with other rules under the six tiers and eight columns model.

rules, respectively. When the number of stored loads is in the range of 11–74, which equates to 11.5%–77.1% of the systems' storage capacity, the AC algorithm achieves a 20% reduction in the makespan relative to the ColByCol rule. When the number of loads is between 74 and 86, the reduction in the makespan still exceeds 10%. The AC algorithm also reduces the makespan by 20% relative to the greed rule when the number of loads is between 26 and 60. The experiment results show that the configuration of the SBS/RS do not affect the superiority of the AC algorithm over the empirical storage rules, but the degree of superiority is different.

According to the results of the two experiments, the effect of the AC algorithm on each model varied. The result of the first experiment (SBS/RS model with six tiers and six columns) shows that when the number of stored loads is in the range of 6 – 56, which equates to 8.33%–77.78% of the systems' storage capacity, the AC algorithm reduces the makespan by more than 20% relative to the ColByCol rule (Table 4). As presented in Table 5, the second experiment (SBS/RS model with six tiers and eight columns) shows that the AC algorithm reduces the makespan by more than 20% when the number of stored loads is in the range of 11–73, which equated to 11.46%–76.04% of the systems' storage capacity. In sum, the AC algorithm performs better in the first experiment than in the second experiment. The difference is attributed to the complicated structure of neural networks. The actor neural network has three convolution layers and one linear layer, while the critic neural network has four convolution layers and one linear layer. Each layer comprises numerous nodes, and their exact number depends on the scale of problem solving. The first experiment involves 72 storage positions, and the second one involves 96 storage positions. These differences lead to the different scales of the two neural networks. The greater the number of storage positions, the more complicated the neural networks, and the greater the number of coefficients. Thus, the AC algorithm performs better in the first experiment than in the second one.

5 Conclusion

This work focuses on improving the efficiency of an existing SBS/RS by scheduling the storage process. The study is a novel one because most existing research has mainly focused on finding the best configurations of SBS/RSs at the design stage. For an existing SBS/RS device, the configuration is fixed, and changing it to improve the performance of the device is costly. Therefore, changing the schedule and storage process is the best option. Empirical storage rules are easy and robust, but they cannot effectively reduce the makespan. Empirical storage rules may be efficient when the number of loads to be stored reaches the rack capacity, but the makespan is relatively long in the overall process. When the AC algorithm is used, along with the two neural networks and the proposed observation values, the resulting scheduling and storage process outperform that based on empirical rules. The AC algorithm outperforms the empirical rules over the full range of rack occupancy. The results of the experiments show that in an SBS/RS with six columns and six tiers, the AC algorithm can reduce the makespan by 6.67% relative to the column-by-column rule. In the storage process, the AC algorithm can reduce the makespan by more than 30% when the number of loads to be stored is between 7 and 45, which is equal to 9.7%–62.5% of the systems' storage capacity. Hence, the AC algorithm can outperform the other empirical rules regardless of the number of loads.

In the future, the performance of the AC algorithm and other reinforcement learning algorithms will be further studied to find other efficient rules for scheduling the storage process of large SBS/RSs (over 120 storage

Table 4 Distribution of makespan reduction in the first experiment.

Number	Lower limit number of the stored loads	Upper limit number of the stored loads	Lower limit of the systems' storage capacity (%)	Upper limit of the systems' storage capacity (%)	Reduction of makespan (%)
1	6	56	8.33	77.78	20–30
2	7	45	9.72	62.50	30–40
3	12	31	16.67	43.06	More than 40

Table 5 Distribution of makespan reduction in the second experiment.

Number	Lower limit number of the stored loads	Upper limit number of the stored loads	Lower limit of the systems' storage capacity (%)	Upper limit of the systems' storage capacity (%)	Reduction of makespan (%)
1	11	73	11.46	76.04	20–30
2	18	56	18.75	58.33	30–40

positions). The utilization of reinforcement learning in retrieval transactions is also an interesting topic for future research. In addition, reinforcement learning can be used in scheduling other complex logistics systems, such as a Robotic Mobile Fulfillment System (RMFS).

Acknowledgment

This work was supported by the National Natural Science Foundation of China (No. 52075036) and the Natural Science Foundation of Beijing Municipality (No. L191011).

References

[1] M. Fukunari and C. J. Malmborg, A network queuing approach for evaluation of performance measures in autonomous vehicle storage and retrieval systems, *Eur. J. Oper. Res.*, vol. 193, no. 1, pp. 152–167, 2009.

[2] C. J. Malmborg, Conceptualizing tools for autonomous vehicle storage and retrieval systems, *Int. J. Prod. Res.*, vol. 40, no. 8, pp. 1807–1822, 2002.

[3] P. H. Kuo, A. Krishnamurthy, and C. J. Malmborg, Design models for unit load storage and retrieval systems using autonomous vehicle technology and resource conserving storage and dwell point policies, *Appl. Math. Model.*, vol. 31, no. 10, pp. 2332–2346, 2007.

[4] M. Fukunari and C. J. Malmborg, An efficient cycle time model for autonomous vehicle storage and retrieval systems, *Int. J. Prod. Res.*, vol. 46, no. 12, pp. 3167–3184, 2008.

[5] B. Y. Ekren, S. S. Heragu, A. Krishnamurthy, and C. J. Malmborg, Simulation based experimental design to identify factors affecting performance of AVS/RS, *Comput. Ind. Eng.*, vol. 58, no. 1, pp. 175–185, 2010.

[6] B. Y. Ekren and S. S. Heragu, Simulation based performance analysis of an autonomous vehicle storage and retrieval system, *Simul. Model. Pract. Theory*, vol. 19, no. 7, pp. 1640–1650, 2011.

[7] B. Y. Ekren, S. S. Heragu, A. Krishnamurthy, and C. J. Malmborg, An approximate solution for semi-open queueing network model of an autonomous vehicle storage and retrieval system, *IEEE Trans. Autom. Sci. Eng.*, vol. 10, no. 1, pp. 205–215, 2013.

[8] G. Marchet, M. Melacini, S. Perotti, and E. Tappia, Analytical model to estimate performances of autonomous vehicle storage and retrieval systems for product totes, *Int. J. Prod. Res.*, vol. 50, no. 24, pp. 7134–7148, 2012.

[9] T. Lerher, Modern automation in warehousing by using the shuttle based technology, in *Automation Systems of the 21st Century*: *New Technologies*, *Applications and Impacts on the Environment & Industrial Processes*, D. Arent and M. Freebush, eds. New York, NY, USA: Nova Publishers, 2013, p. 5186.

[10] T. Lerher, Travel time model for double-deep shuttle-based storage and retrieval systems, *Int. J. Prod. Res.*, vol. 54, no. 9, pp. 2519–2540, 2016.

[11] T. Lerher, *Throughput and Energy Related Performance Calculations for Shuttle Based Storage and Retrieval Systems, Energy Science, Engineering and Technology*. New York, NY, USA: Nova Science Publishers, 2016.

[12] N. Zhao, L. Luo, S. P. Zhang, and G. Lodewijks, An efficient simulation model for rack design in multi-elevator shuttle-based storage and retrieval system, *Simul. Model. Pract. Theory*, vol. 67, pp. 100–116, 2016.

[13] H. J. Carlo and I. F. A. Vis, Sequencing dynamic storage systems with multiple lifts and shuttles, *Int. J. Prod. Econ.*, vol. 140, no. 2, pp. 844–853, 2012.

[14] N. Zhao, L. Luo, and G. Lodewijks, Scheduling two lifts on a common rail considering acceleration and deceleration in a shuttle based storage and retrieval system, *Comput. Ind. Eng.*, vol. 124, pp. 48–57, 2018.

[15] J. B. Xin, C. Meng, F. Schulte, J. Z. Peng, Y. H. Liu, and R. R. Negenborn, A time-space network model for collision-free routing of planar motions in a multirobot station, *IEEE Trans. Ind. Inform.*, vol. 16, no. 10, pp. 6413–6422, 2020.

[16] T. Lerher, M. Ficko, and I. Palcic, Throughput performance analysis of Automated Vehicle Storage and Retrieval Systems with multiple-tier shuttle vehicles, *Appl. Math. Model.*, vol. 91, pp. 1004–1022, 2021.

[17] E. Tappia, D. Roy, R. de Koster, and M. Melacini, Modeling, analysis, and design insights for shuttle-based compact storage systems, *Transp. Sci.*, vol. 51, no. 1, pp. 269–295, 2016.

[18] G. D'Antonio, M. De Maddis, J. S. Bedolla, P. Chiabert, and F. Lombardi, Analytical models for the evaluation of deep-lane autonomous vehicle storage and retrieval system performance, *Int. J. Adv. Manuf. Technol.*, vol. 94, no. 5, pp. 1811–1824, 2018.

[19] B. P. Zou, X. H. Xu, Y. M. Gong, and R. De Koster, Modeling parallel movement of lifts and vehicles in tier-captive vehicle-based warehousing systems, *Eur. J. Oper. Res.*, vol. 254, no. 1, pp. 51–67, 2016.

[20] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al., Mastering the game of Go with deep neural networks and tree search, *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[21] J. Y. Hu, H. L. Niu, J. Carrasco, B. Lennox, and F. Arvin, Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning, *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 14413–14423, 2020.

[22] A. G. Barto, R. S. Sutton, and C. W. Anderson, Neuronlike adaptive elements that can solve difficult learning control problems, *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-13, no. 5, pp. 834–846, 1983.

Gabriel Lodewijks received the BEng degree in transport engineering and logistics from University of Twente, the Netherlands in 1990, the MS degree in transport engineering and logistics from Delft University of Technology, the Netherlands in 1992, and the PhD degree in dynamics of transportation systems from Delft University of Technology, the Netherlands in 1996. He is now a full professor and has been the head of the School of Aviation, University of New South Wales, Sydney, Australia, since 2017. He also works as a visiting/guest/chair professor at the University of Witwatersrand, South Africa; Wuhan University of Technology, Beijing University of Science and Technology, and China University of Mining and Technology, all in China; and Newcastle University, Australia. He has over 300 publications including 2 books and 140+ journal papers. His research interests are optimization of maintenance, repair and overhaul processes, automation of air cargo handling systems, tracking and tracing of equipment, components, and people at airports and in aviation related companies, optimization of gate processes and baggage handling procedures to reduce the turnaround time of aircraft, maintaining safety and security in airport logistic processes, and the improvement of passenger experience by streamlining airport logistics.

Lei Luo received the BS and MS degrees from University of Science and Technology Beijing, China in 2015 and 2018, respectively. He is currently a PhD candidate at the School of Mechanical Engineering, University of Science and Technology Beijing. His main research interests include system simulation and optimization, intelligent manufacturing, and digital twin.

Ning Zhao received the BEng and MEng degrees in mechanical engineering from Shandong University, China in 1999 and 2002, respectively, and the PhD degree in mechanical engineering from Beijing Institute of Technology, China in 2005. He is now a professor at the University of Science and Technology Beijing, China. His research interests include simulation modelling and scheduling of logistics system, factory planning and optimization, and digital twin application in manufacturing and logistics system.