Computers & Industrial Engineering 149 (2020) 106749

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

Deep reinforcement learning based AGVs real-time scheduling with mixed rule for flexible shop floor in industry 4.0

Hao Hu, Xiaoliang Jia *, Qixuan He, Shifeng Fu, Kuo Liu

School of Mechanical Engineering, Northwestern Polytechnical University, 127 West Youyi Road, Beilin District, Xi'an, Shaanxi 710072, PR China

ARTICLE INFO

Keywords:
Automated guided vehicles
Real-time scheduling
Deep reinforcement learning
Industry 4.0

ABSTRACT

Driven by the recent advances in industry 4.0 and industrial artificial intelligence, Automated Guided Vehicles (AGVs) has been widely used in flexible shop floor for material handling. However, great challenges aroused by the high dynamics, complexity, and uncertainty of the shop floor environment still exists on AGVs real-time scheduling. To address these challenges, an adaptive deep reinforcement learning (DRL) based AGVs real-time scheduling approach with mixed rule is proposed to the flexible shop floor to minimize the makespan and delay ratio. Firstly, the problem of AGVs real-time scheduling is formulated as a Markov Decision Process (MDP) in which state representation, action representation, reward function, and optimal mixed rule policy, are described in detail. Then a novel deep q-network (DQN) method is further developed to achieve the optimal mixed rule policy with which the suitable dispatching rules and AGVs can be selected to execute the scheduling towards various states. Finally, the case study based on a real-world flexible shop floor is illustrated and the results validate the feasibility and effectiveness of the proposed approach.

1. Introduction

Nowadays, manufacturing enterprises is facing tremendous pressure aroused by fierce market competition and rising manufacturing costs (Zhang, Ma, Yang, Lv, & Liu, 2018; Ma, Zhang, Lv, Yang, and Wu, 2019). Therefore, continuously improving production efficiency and reducing costs has always been the focus on manufacturing companies. Driven by the recent advances in Internet of things (IoT) and industrial Artificial Intelligence (AI), lots of information technology (RFID, embedded device, augmented reality, etc.) and industrial robot (Robotic arm, mobile robot, etc.) have been widely adopted in production shop floor (Wang, Zhang, Liu, & Wu, 2019; Lin, Yu, Zhang, Yang, Zhang, & Zhao, 2017; 2018;; Mourtzis, Zogopoulos, & Xanthi, 2019). Automated Guided Vehicles (AGVs), a kind of mobile robot for material handling, have been regarded as one of the most promising technology and applied to various shop floors and warehouse logistics for material supply operations owing to high degree of autonomy and flexibility (Vis, 2006; Michalos, Kousi, Makris, & Chryssolouris, 2016; Kousi et al., 2016; Demesure et al., 2018; Wang, Zhang, & Zhong, 2020; Kousi, Koukas, Michalos, & Makris, 2019).

AGVs scheduling is one of the main tasks of AGVs control, and it aims at finding the optimal transport time and equipment for each task, while respecting the production specifications provided by the company (Saidimehrabad, Dehnaviarani, Evazabadian, & Mahmoodian, 2015; Wallace & Andrew, 2001). AGVs scheduling can directly determine the efficiency and overall performance of the AGV system, so that it has been paid excessive attentions by both industry and academia for a long time (Berman, & Edan, 2002; Singh, Sarngadharan, & Pal, 2011; Miyamoto, & Inoue, 2016; Kousi, Michalos, Makris, & Chryssolouris, 2016). Traditional static scheduling approach usually assumes that all the task information is stable and obtained in advance, and then establish an analytical model and solve it with the heuristic algorithm. Mousavi et al. (2017) proposed a multi-objective AGVs scheduling approach in a Flexible Manufacturing System (FMS) using a hybrid of genetic algorithm and particle swarm algorithm. Ulusoy, Sivrikayaşerifoglu, and Bilge (1997) introduced a genetic algorithm approach to solve the problem of simultaneous scheduling of machines and a number of identical AGVs in an FMS so as to minimize the makespan. Nevertheless, it is unrealistic to be informed all the task information beforehand in a real-world shop floor. On the other hand, many uncertainties (such as urgent task, task reworks, etc.) also exist on such a dynamic and complex shop floor environment (Zhang, Zhu, & Lv, 2018). Therefore, the static

* Corresponding author.
E-mail addresses: kimi77@mail.nwpu.edu.cn (H. Hu), jiaxl@nwpu.edu.cn (X. Jia), heqx@mail.nwpu.edu.cn (Q. He), npufushifeng@mail.nwpu.edu.cn (S. Fu), liukuo@mail.nwpu.edu.cn (K. Liu).

https://doi.org/10.1016/j.cie.2020.106749
Received 21 January 2020; Received in revised form 20 May 2020; Accepted 8 August 2020
Available online 14 August 2020

scheduling approach is insufficient for the complicated real-world shop floor.

In recent years, with the help of IoT technology, many scholars focused on the real-time scheduling in AGVs and production system to address the dynamics in shop floor operation environment. Zhang et al. (2015) presented an optimization method for material handling, the tasks are assigned to vehicles using priority policy based on the real-time status of the vehicles. Li et al. (2018) proposed a mechanism for multiple AGVs scheduling in intelligent warehouse system faced with simultaneous multiple customer demands. Mourtzis and Vlachou (2018) introduced a cloud-based cyber-physical system with the help of IoT to enable adaptive shop-floor scheduling and condition-based maintenance. In the literature, various dispatching rules are frequently used to dispatch AGVs in the real-time scheduling problem (Sabuncuoglu, 1998; Klei & Kim, 1996; Singh, Sarngadharan, & Pal, 2011). These rules determine how the transport tasks are assigned to specific AGVs, and some typical rules are first come first served (FCFS), shortest travel distance (STD), earliest due date first (EDD), longest waiting time (LWT), and nearest vehicle first (NVF), etc.

Previous studies of the real-time scheduling problem domain indicate that using a multiple dispatching rules (MDRs) strategy can enhance the production performance to a greater extent than using a single rule over a given scheduling interval in the shop floor control system (Shiue, Lee, & Su, 2018). Meanwhile, the MDRs can also empower the AGVs scheduling with more adaptation because each rule has its advantage and insufficiency respectively. Conventional MDRs mainly selects a proper scheduling rule according to the specific situation based on a knowledge base. Chen et al. (2011) proposed a multiple-criteria real-time scheduling approach for multiple-load carriers, the approach can select an appropriate rule from the knowledge base using machine learning. However, the drawback of the conventional MDRs is that the static knowledge base cannot respond to changes in the shop floor environment. To solve this issue, some scholars adopted reinforcement learning (RL) to MDRs to achieve an adaptive scheduling. RL is a machine learning method that can constantly adjust the agent's behavior through trial and error (Kaelbling, Littman & Moore, 1996). An reinforcement learning based approach for a multiple-load carrier scheduling problem was presented by Chen, Xia, Zhou and Xi (2015), and they developed a Q(λ) RL algorithm based scheduling approach to improve the throughput of the assembly line while reducing the travel cost. Wang and Usher (2005) applied Q-learning algorithm to solve the agent-based production scheduling question, and it enables a single machine agent to learn to select the best dispatching rule. Aydin and Oztemel (2000) also introduced an intelligent agent based dynamic job shop scheduling method using reinforcement learning.

Although good efforts have been made in the above studies, the limitation of performance and learning efficiency still exists on these methods as the complexity of the environment increases (Qiu, Yu, Yao, Jiang, Xu, & Zhao, 2019). Traditional RL algorithm (Q-learning, Sarsa, etc.) commonly used a policy table to store all the possible states and corresponding actions. Their performance could decrease greatly when the states become more complex (the number of states will increase significantly), which can be regard as a high-dimensional data problem (Wan, Li, He, & Prokhorov, 2019). Deep Reinforcement Learning (DRL) has achieved impressive successes and attracted much attention over the last several years, especially Deepmind Company proposed deep q-network (DQN) and then developed AlphaGo which defeated the human champion in the game of Go that shows enormous potential for DRL to solve sequential decision problem (Mnih et al., 2015; Silver et al., 2016). Recently, they have developed a DRL agent which leverages the computational functions of grid cells and can reach the abilities of mammal-like navigational (Banino et al., 2018). DRL utilizes the deep neural network to capture the complex state features instead of policy table, such that the loss of state information is greatly reduced. Some applications of DRL in electric vehicle charging scheduling (Wan, Li, He, & Prokhorov, 2019) and active visual object detection (Han, Liu, Sun, & Zhang, 2019) also have proven the superiority and effectiveness of DRL.

To enable an efficient and adaptive scheduling, a novel real-time scheduling approach to AGVs using DRL is proposed to the flexible shop floor to minimize the makespan and delay ratio in this paper. The dominating real-time information on AGVs system, including the number of tasks, the average remaining time of tasks, the average travel distance of tasks, and the status of AGVs (working or idle), are taken as continuous input states. And then the proposed approach can learn to choose the optimal dispatching rule of the optional rules according to various states through a great number of training. Finally, the proposed approach is analyzed and evaluated by a case study to validate its reliability and effectiveness. The main contributions of this paper can be summarized as follows.

(1) The problem of AGVs real-time scheduling is formulated as a Markov Decision Process (MDP). The key elements including state representation, action representation, reward function, and optimal mixed rule policy, are introduced in detail.
(2) A DQN based method is presented to deal with AGVs real-time scheduling. By using this method, the optimal dispatching rule and AGV can be selected after a great number of training.
(3) A case study based on a practical flexible shop floor is illustrated, and the experimental validations are performed to validate the effectiveness and reliability of the proposed approach.

The remainder of this paper is organized as follows. In Section 2, we formulate the problem of AGVs real-time scheduling as an MDP. The DQN based AGVs real-time scheduling is discussed in Section 3. Case studies are illustrated with Section 4, including the description of training scenes and comparison experiments. Conclusion and future works are presented in Section 5.

2. Problem formulation

In this paper, the problem of AGVs real-time scheduling is formulated as an MDP and can be described by a six-element tuple (S, A, P, γ, R, π). Where S is the state set involves all the possible states s in the scheduling process; A is the action set including all the executable actions a consists of dispatching rules and AGVs; P is the probability of transition from current state s to next state s' when taking action a; γ is the discount factor; $R(s, a, s')$ is the reward function used to evaluate the action after a transition. π is the policy that represents the mapping from the state set S to the action set A, policy π(s, a) denotes the probability of taking a specific action a in a given state s. The definition of state, action, reward function and optimal mixed rule policy are detailed as follows.

2.1. State representation

The state is used to characterize the status of the system so as to guide the decision. A state at time t can be defined as a vector $s_t = (N_t, T_{art}, D_{adt}, A_{st}, A_{vt})$ considering the task status and the AGVs status.

(1) N_t is the number of the current tasks, which represents the amount of the workload of the current transportation tasks.
(2) T_{art} represents the average remaining time of the current tasks.

$$T_{art} = \frac{\sum_{k=1}^{N_t} t_{krt}}{N_t} \tag{1}$$

Where t_{krt} is the remaining time of the k th task of the current tasks. This indicator mainly reflects the average urgency of the current tasks.

(3) D_{adt} represents the average travel distance of the current tasks.

$$D_{adt} = \frac{\sum_{k=1}^{N_t} d_{kdt}}{N_t} \quad (2)$$

Where d_{kdt} is the travel distance of the k th task of the current tasks. This indicator reflects the average workload of each task.

(4) A_{st} denotes the working status of all alternative AGVs, and it can be represented by a binary number.

$$A_{st} = n_1 n_2 \ldots n_i \quad (3)$$

Where n_i represents the working status of the AGV i, and 0 means "Idle" while 1 means "Working". For example, "100" means AGV1 is working, AGV2 and AGV3 are idle.

(5) $A_{vt} = \{v_1, v_2, \cdots v_i\}$ is the normal driving speed of the AGVs and can be represented as a vector, and v_i is the normal driving speed of the AGV i.

2.2. Action representation

Action represents the scheduling behavior of the AGVs system and is defined by a vector $a_t = (Ru_t, AGV_t)$ consists of scheduling rule and AGV type. Dispatching rule determines the sequence in which the tasks are assigned, and AGV type specifies the vehicle to be dispatched by the selected task. In this paper, five typical dispatching rules, namely FCFS, STD, EDD, LWT, and NVF are mainly considered and described in Table 1 (Sabuncuoglu, 1998; Klei & Kim, 1996; Ho, & Liu, 2006). $Ru_t = \{\text{FCFS} = 1, \text{STD} = 2, \text{EDD} = 3, \text{LWT} = 4, \text{NVF} = 5\}$, $AGV_t = \{1 = AGV1, 2 = AGV2, \cdots, i = AGVi\}$. With the combined actions, the task can be selected by specific rules and dispatched to corresponding AGV at time t.

2.3. Reward function

Reward function is designed to estimate the action and optimize the policy. This study aims at reducing the makespan and delay ratio of AGVs real-time scheduling. In theory, lots of indicators can be considered in the reward function such as makespan, energy consumption, delay ratio, maintenance cost, equipment utilization, etc., since the reward is a scalar. Nevertheless, the focus on this study is to validate the effectiveness and efficiency of the DRL based approach. Thus, makespan and delay ratio which reflecting efficiency are selected according to the main requirements of the flexible shop floor (Sabuncuoglu, 1998). To evaluate the two indicators in the same dimension, the concept of the time cost and delay cost are introduced as follows.

$$C_{ikd} = \beta(t_{ik} - t_{kr}) \quad (4)$$

$$C_{id} = \sum_k C_{ikd} \quad (5)$$

$$C_d = \sum_i C_{id} = \beta \sum_i \sum_k (t_{ik} - t_{kr}) \quad (6)$$

Where C_{ikd} denotes the delay cost obtained after AGV i executing the task k and t_{ik} is the actual transport time. t_{kr} is the remaining time of task k. β is the delay cost coefficient. C_{id} denotes the total delay cost of AGV i and C_d represents the total delay cost of the whole scheduling process.

$$C_{ikT} = \lambda t_{ik} \quad (7)$$

$$C_T = \lambda T \quad (8)$$

Where C_{ikT} is the time cost for AGV i executing the task k, and C_T denotes the time cost of the whole scheduling process. λ is the time cost coefficient. T is the makespan of the whole scheduling process. The performance of scheduling will be improved as the above costs decrease. Therefore, the reward function is defined based on the delay cost and time cost as follows.

$$R_t = \mu_1(c_{ad} - C_{ikd}) + \mu_2(c_{aT} - C_{ikT}) \quad (9)$$

$$R_f = \mu_1(C_{ad} - C_d) + \mu_2(C_{aT} - C_T) \quad (10)$$

Where R_t is the current reward used to evaluate an action for each individual task, and R_f is the final reward used to evaluate the overall performance of the scheduling. c_{ad} and c_{aT} are the average delay cost and the average time cost for a task, while C_{ad} and C_{aT} represent the average delay cost and the average time cost for the whole scheduling respectively. μ_1 and μ_2 are the weight factors.

2.4. Optimal mixed rule policy

Policy is a very important element in MDP, it tells how to choose an action according to different states. Since multiple dispatching rules are used as actions, the mixed rule policy can be estimated by the action-value function $Q^\pi(s,a)$ as follow.

$$\begin{aligned} Q^\pi(s,a) &= E_\pi\{R|s_t = s, a_t = a\} = E_\pi\{R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^k R_{t+k+1}\} \\ &= E_\pi\{\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | s_t = s, a_t = a\} \end{aligned} \quad (11)$$

Where the action-value $Q^\pi(s,a)$ represents the future expected discount rewards when taking action a at state s. γ is the discount coefficient, and R_{t+k+1} is the current or final reward at time t. Thus, the problem of AGVs real-time scheduling is converted to an MDP problem of which the objective is to find an optimal mixed rule policy π^* that can maximize the reward as follow.

$$Q^{\pi^*}(s,a) = max E_\pi\{R|s_t = s, a_t = a\} = \max_\pi Q^\pi(s,a), \forall s \in S, \forall a \in A \quad (12)$$

Table 1
Dispatching rules.

Rule Name	Description	Advantage	Insufficiency
FCFS	The task is selected in the order of arrival	FCFS can guarantee overall efficiency and smoothness of the scheduling	FCFS cannot effectively meet other indicators than efficiency, such as urgent tasks and travel costs
STD	The task with the shortest travel distance will be selected first	STD can improve overall efficiency to a certain extent	STD may cause long waiting time for the tasks with longer travel distance
EDD	The task with the earliest due date will be selected first	EDD can guarantee the completion of the urgent tasks and reduce the delay rate	EDD may increase transportation costs and reduce overall efficiency
LWT	The task with the longest waiting time will be selected first	LWT can effectively reduce the waiting time of tasks to ensure production efficiency	LWT cannot effectively meet other indicators than waiting time, such as efficiency and travel costs
NVF	The vehicle will select the task with the nearest load point	NVF can effectively reduce the travel costs of AGVs	NVF may cause long waiting time for the tasks with farther load point

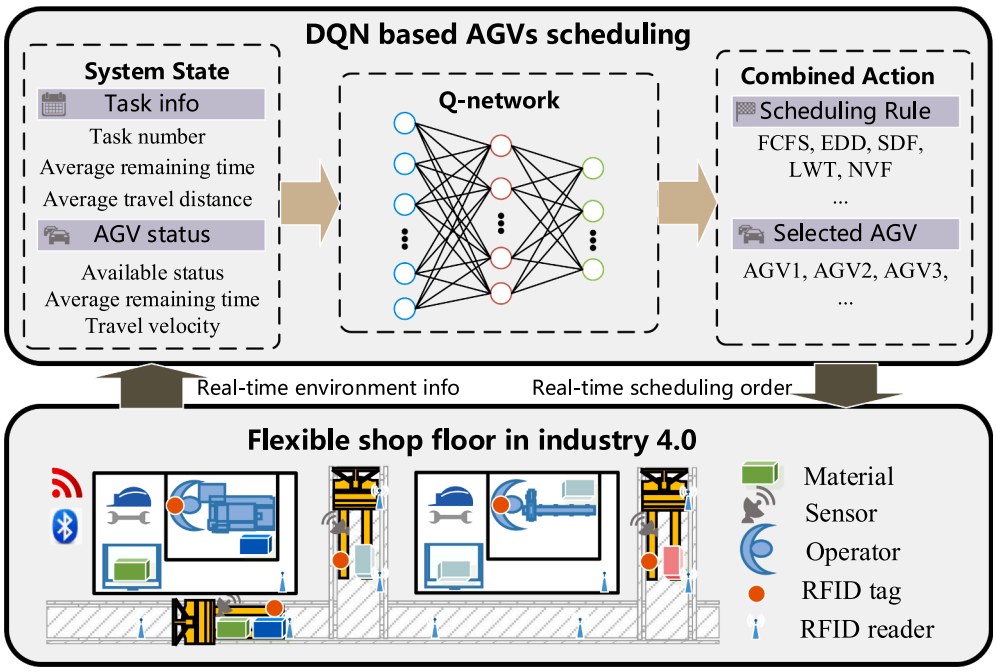


Fig. 1. Architecture of AGVs real-time scheduling approach using DRL.

3. Proposed approach

3.1. Architecture of the approach

As mentioned in the above section, to achieve an optimal mixed rule policy, a DRL based AGVs real-time scheduling approach for the flexible shop floor is proposed, of which the architecture is shown as Fig. 1. This architecture consists of two layers, namely flexible shop floor in industry 4.0, and DQN based AGVs scheduling. Then DQN based AGVs scheduling is further divided into System State module, Q-network module, and Combined Action module. During the production execution stage, when the new tasks arrive or an AGV finishes its work, the scheduling will be triggered. Then the flexible shop floor layer sends the real-time status of production system captured by IoT devices (sensors, RFID, etc.) to System State module. After processing the complex real-time information, System State module extracts key state information, which is mainly divided into task information and AGVs status. This state information is then sent to Q-network module which consists in the calculation core of the architecture. Q-network module continuously performs training and learning by using DQN algorithm, which is detailed in the next part, on the input state and then outputs the results to Combined Action module. Following that, Combined Action module receives the calculation result and interprets it as the selected dispatching rule and AGV. At last, the chosen rule and AGV will be fed back to flexible shop floor layer as the order to guide the AGVs real-time scheduling.

3.2. Training algorithm of DQN

DQN aims at utilizing the neural network as a nonlinear approximation to optimal action-value function $Q(s,a,\theta)\rightarrow Q^{*}(s,a)$, where θ denotes all the parameters of the neural network (Han et al., 2019). Thus, the objective of the training process is to reduce the estimation error between the network and optimal action-value function by updating the parameters of the neural network iteratively. In this study, two neural networks, namely main Q-network and target Q-network, are designed with the same neural network structure. It is worth mentioning that in many DRL based applications, the high-dimensional images data are often used as the input state (camera picture captured by robot, game image, etc.). Such that convolutional layers and pooling layers are usually used in the neural network for feature extraction (Wan, Li, He, & Prokhorov, 2019; Arulkumaran, Deisenroth, Brundage, & Bharath, 2017). However, the features of the state have been clear defined and can be easily obtained in this study. Thus, two Q-networks are designed by using a fully connected neural network consists of one input layer, two hidden layers and one output layer, and the parameters of the neural network refer to the weights and biases of different neuron nodes (Mnih et al., 2015). More detail about the structure of Q-network is shown in Table 2.

The primary steps of the training algorithm are presented as shown in Fig. 2. In each episode, a set of complete execution process information of a task captured from the shop floor will be firstly stored in the reply memory library as a record which is denoted as a vector $Re(s_t,a_t,R_t,s_{t+1})$. After that, the stored records are randomly sampled from the replay memory library as to avoid excessive correlation of the networks (Wan, Li, He, & Prokhorov, 2019). The sampled records will be sent to main Q-network and target Q-network, respectively. These two Q-networks have the same neural network structure but different parameters. The parameters of main Q-network is denoted as θ, while θ' denotes the parameters of target Q-network. Following that the value of action-value functions will be predicted by the two networks, and then the loss function can be calculated according to the error of the two networks. It is noticed that the action-value function $Q(s_t,a_t;\theta)$ is directly calculated by main Q-network, while the action-value function $Q(s_t,a_t;\theta')$ is calculated after target Q-network predicts the maximum $Q(s_{t+1},a_{t+1};\theta')$ denoted as $maxQ(s_{t+1};\theta')$:

Table 2
Structure parameters of two Q-networks.

Layer	Node number	Activation function	Description
Input layer	5	None	Complete status information s_t
Hidden layer 1	8	Tanh	None
Hidden layer 2	8	Tanh	None
Output layer	15	Softmax	Dispatching rule and AGV

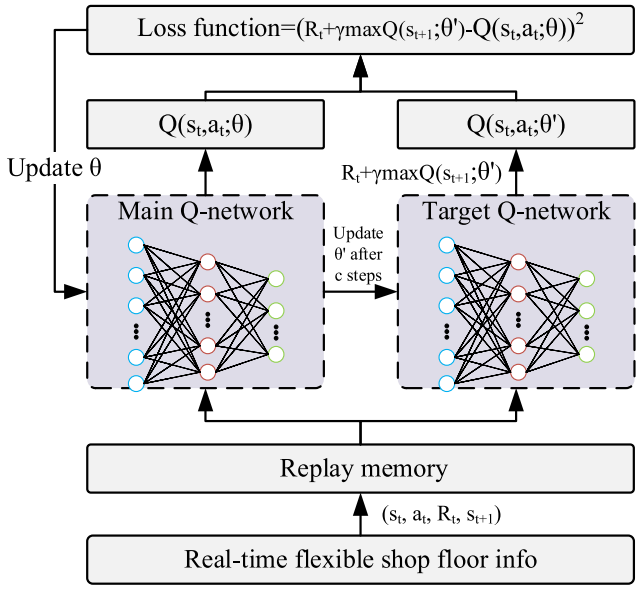


Fig. 2. Primary steps of the training algorithm.

$$Q(s_t, a_t; \theta') = \begin{cases} R_t, s_t = \text{Terminal state} \\ R_t + \gamma max Q(s_{t+1}; \theta'), \text{else} \end{cases} \quad (13)$$

Thus, the loss function $L(\theta)$ is defined as:

$$L(\theta) = (Q(s_t, a_t; \theta') - Q(s_t, a_t; \theta))^2 = (R_t + \gamma max Q(s_{t+1}; \theta') - Q(s_t, a_t; \theta))^2 \quad (14)$$

Then the parameters θ of main Q-network are updated by using the gradient descent algorithm which aims to reduce the loss function iteratively and is widely used in deep learning (Mnih et al., 2015).

$$\theta_{t+1} = \theta_t - \eta \nabla_{\theta_t} L(\theta_t) \quad (15)$$

At last, the parameters θ' of target Q-network will be replaced by the θ of main Q-network after every C step to synchronize between the two networks with a certain delay, and the episode will continue to repeat until the reward function converges or the number of trainings reaches a certain value. The complete algorithm is shown in Algorithm 1.

Algorithm 1: The training algorithm of DQN.

Initialize replay memory D;
Initialize the parameters θ of main Q-network;
Initialize the parameters $\theta' = \theta$ of target Q-network;
For episode = 1,M **do**
  Obtain the current state s_t;
  For time t = 1,T **do**
    With probability ε select a random action a_t;
    Otherwise select action $a_t = argmax Q(s_t, a_t; \theta)$;
    Execute action a_t and then observe the reward R_t and next state s_{t+1};
    Store record $Re(s_t, a_t, R_t, s_{t+1})$ in D;
    Sample random record $Re(s_j, a_j, R_j, s_{j+1})$ from D;
    Calculate $Q(s_j, a_j; \theta')$ of target Q-network in (13);
    Calculate Loss function $L(\theta)$ in (14);
    Update the parameters θ of main Q-network with gradient descent in (15);
    Every C steps set $\theta' = \theta$;
  End For
End For
Output the parameters of network;

3.3. Real-time scheduling with optimal mixed rule policy

The well-trained main Q-network will be used as the optimal mixed rule policy to guide the AGVs real-time scheduling. The detailed process of AGVs real-time scheduling with optimal mixed rule policy can be seen in Fig. 3. At first, a scheduling request is triggered when a new task arrives or an AGV finishes its work and turns to idle status. Then the current system state is calculated and achieved from flexible shop floor and sent to the well-trained main Q-network. Consequently, the optimal action will be calculated by the well-trained main Q-network and the dispatching rule and AGV selected are expressed accordingly. The status of the selected AGV has to be checked before dispatching the task. If the selected AGV is available, then the task chosen by dispatching rule will be assigned to it; otherwise, the optimal action will be recalculated until the task is assigned. At last, the selected AGV will receive the scheduling order and finish the chosen task.

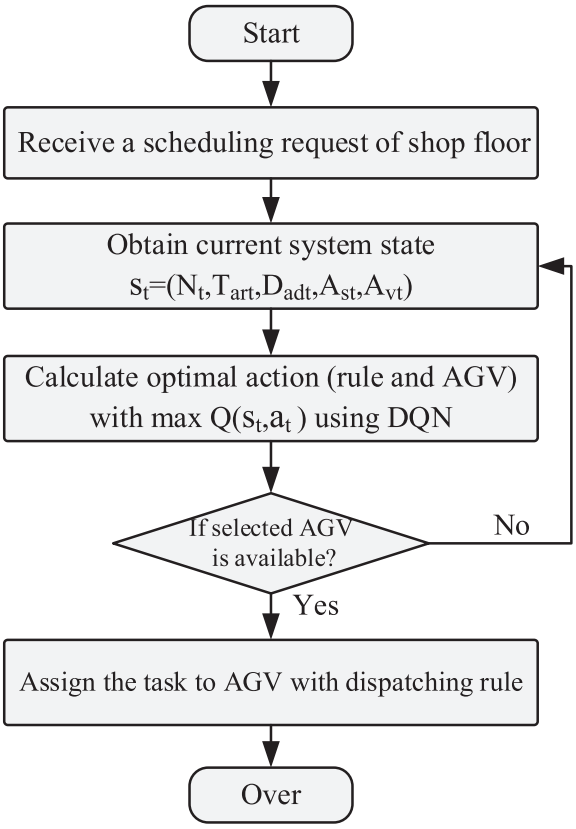


Fig. 3. AGVs real-time scheduling with optimal mixed rule policy.

4. Case study

In this section, the experimental analysis is performed by a simulation case based on a real-world flexible shop floor to verify the effectiveness of the proposed approach.

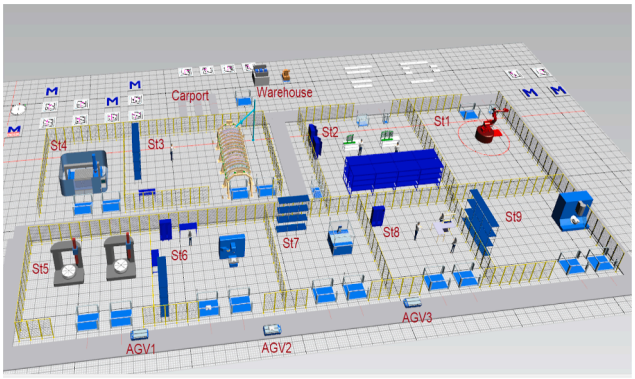


Fig. 4. Case illustration of the flexible shop floor.

4.1. Experimental platform and case description

In this study, a professional discrete event simulation software platform, namely Siemens Tecnomatix, is utilized to establish the simulation scenes of a flexible shop floor environment. As shown in Fig. 4, the flexible shop floor is mainly composed of nine different stations surrounded by a ring-like path, and each station has two buffers for loading and unloading material. There are three AGVs with different normal travel speed working in the shop floor for material handling. A warehouse and carport locate on the up side of the shop floor. The path is designed to allow AGVs run in both side. To simplify the complexity of the problem, the navigation method of all AGVs is the shortest route policy and the possible collisions between AGVs are ignored.

4.2. Implementation of DRL based AGVs real-time scheduling

To enable the proposed DRL based AGVs real-time scheduling, a DQN program is developed using TensorFlow and integrated with the simulation platform as seen in Fig. 5. The implementation consists of two programs, the DQN program developed on TensorFlow encoded with python and the simulation program developed on Tecnomatix encoded with built-in SimTalk programming language. The simulation program includes several subprograms to provide specific functions, and they are shop floor controller, task controller, state controller, communicator, scheduler, respectively. In each episode of learning, during the runtime of the simulation program, the shop floor controller simulates the production process, and the task controller is responsible for generating and managing the current real-time tasks. When a scheduling is triggered, the real-time information on the current tasks and AGVs is sent to the state controller. Formatted state records will be generated by the state controller after data processing and sent to the communicator afterwards. The communicator establishes a network connection with the DQN program through the TCP/IP protocol and sends the state records to the DQN program. The optimal combined action is calculated by the DQN program and sent back to the scheduler to execute the scheduling. After that an optimized task is assigned to a selected AGV based on the optimal dispatching rule. When an AGV finishes its task, the complete execution process information on the task will be sent to DQN program and stored in the reply memory library.

Generally speaking, the parameter updating of neural networks is a time-consuming process that may block prompt scheduling (Pan, Zi, Chen, Zhou, & Wang, 2018). To avoid this situation, we have adopted an asynchronous network update mechanism that allows the parameters update at the end stage of an episode when all tasks are completed. It also means that main Q-network mainly plays the role of a decision maker in the scheduling process.

4.3. Training process

Since the implementation of DQN requires a large amount of training, a set of scenes is designed with simulation platform based on the research of a real-world shop floor. In these training scenes, 800 tasks are randomly generated for each episode. Each task contains several key properties including task ID, from-location, to-location, remaining time, and estimated distance, an example of tasks is shown in Table 3.

To fully reflect the complexity and uncertainties of the dynamic shop floor environment, which widely exist on the real-world production system, as much as possible, some important experimental parameters and constraints are set and described as follows.

Table 3
Task information.

Task ID	From-location	To-location	Remaining time	Estimated Distance
T0001	St3	St19	123 s	96 m
T0002	Warehouse	St4	52 s	54 m
T0003	St6	St4	77 s	80 m
T0004	St7	St5	300 s	113 m

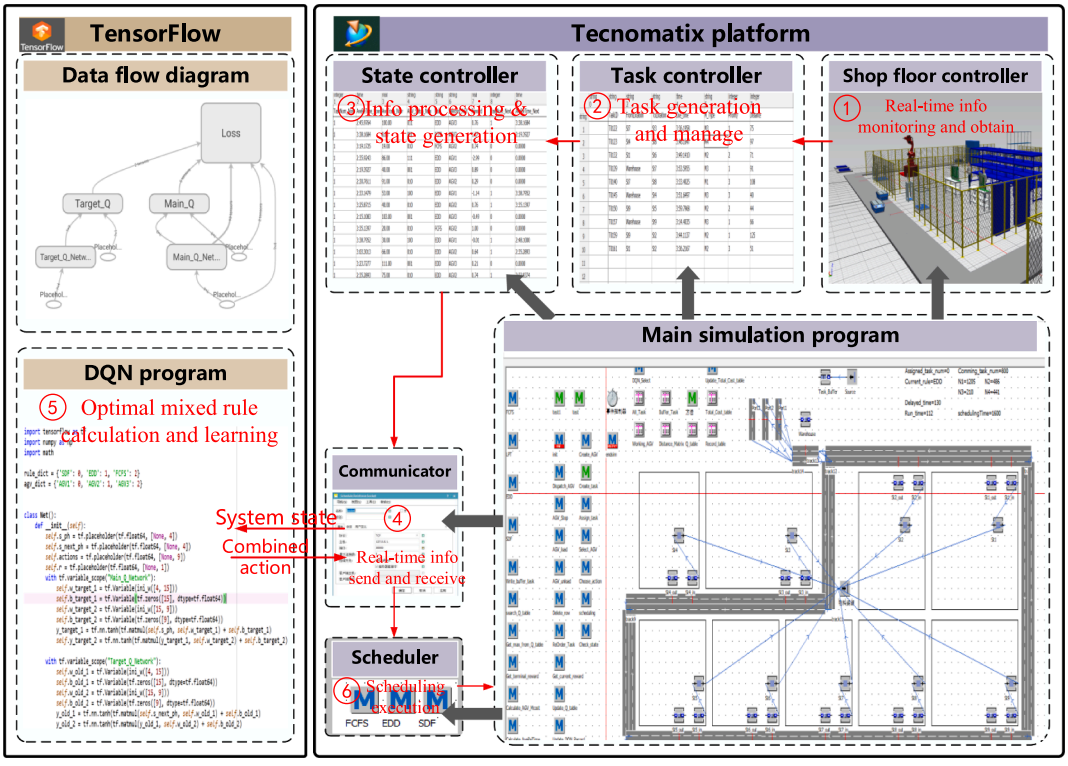


Fig. 5. Implementation of the DQN based AGVs real-time scheduling.

(1) The total number of tasks in an episode is 800, and the maximum number of current tasks N_t is 20.
(2) Task arrival interval obeys a normal distribution $N(45s, 15s)$.
(3) From-location and to-location are randomly generated in St1–St9 and Warehouse, and each of them cannot be both from-location and to-location. In addition, warehouse cannot be to-location. For instance, "from St3 to St3" or "from St3 to Warehouse" are both forbidden.
(4) The normal travel speed of AGV1, AGV2, AGV3, are 1 m/s, 1.2 m/s, and 1.5 m/s, respectively.
(5) When an AGV completes a task and there are no tasks at current time, it will be asked to travel back to the carport.
(6) Since all the tasks are generated randomly, the task of which the remaining time less than 60 s can be regarded as an urgent task, such as T0002 in Table 3.
(7) To simulate the task reworks situation, 3% of all 800 tasks will repeat randomly.
(8) All the materials contained in a task are considered to be packaged beforehand and are within the rated load capacity of the AGVs, and all AGVs can only perform one task at a time.

An episode is used as an example to illustrate the training process intuitively as shown in Fig. 6. Assume that the current tasks are shown in Table 3 and a schedule is triggered at the moment. AGV1 is in working status, AGV2 and AGV3 are in idle status at the same time. According to the definition of state representation (Part 1 in Section 2), the current state s_t can be calculated as $s_t = (4,138,85.8,100,\{1,1.2,1.5\})$. Then the action a_t is predicted as $a_t = (2,3)$ according to the definition of action representation (Part 2 in Section 2) through main Q-network, such that STD rule and AGV3 are chosen to execute the scheduling and task T0002 is assigned to AGV3. After a while, AGV3 completes the task and the current reward R_t is calculated as $R_t = 1.39$ based on actual completion result according to the definition of reward function (Part 3 in Section 2), then the next state $s_{t+1} = (8,178.9,104.6,010,\{1,1.2,1.5\})$ is also observed in the same way as s_t simultaneously. At last, a complete record of the scheduling process Re can be obtained as $Re = ((4,138,85.8,100,\{1,1.2,1.5\}),(2,3),1.39,(8,178.9,104.6,010,\{1,1.2,1.5\}))$ and then stored in the reply memory library for the training afterwards.

The proposed DQN is trained for 3000 episodes to learn the optimal mixed rule policy for AGVs real-time scheduling. The training process spends nearly 11 hours on the Workstation server (i7-6700 CPU@3.40 GHz, 16G RAM). The evolution of the final rewards of DQN is illustrated in Fig. 7 (a). It can be seen that, in the first 1000 episodes, the final rewards are constantly fluctuating and the action policy is relatively random. Then the final rewards grow rapidly during 1000 episodes to 1500 episodes and gradually reaches steady afterwards. It implies an optimal mixed rule policy is achieved. The evolution of the final rewards of EDD and AHP-based method (which is detailed in the next part) is also illustrated in Fig. 7 (b). It can be seen that, since the EDD and AHP-based method belong to single or multiple attribute dispatching rule, the performance of final rewards is relatively stable around specific value. In addition, the performance of final rewards of EDD is more stable than that of AHP-based method. The results also reveal the difference between machine learning-based method and dispatching rule-based method, machine learning-based method requires a number of training to obtain better performance, while dispatching rule-based method can achieve stable performance without any training.

4.4. Analysis and discussion

In this part, the proposed approach is evaluated and compared with a traditional AHP-based dynamic scheduling method (Zhang et al., 2015). This method can be regarded as a multiple attribute dispatching rule (Klei & Kim, 1996), and the tasks are assigned to vehicle according to a comprehensive priority P_c as:

$$P_c = \omega_1(t_{kr}/t_{ar}) + \omega_2(d_{kt}/d_{at}) \tag{16}$$

Where t_{kr} and d_{kt} are the remaining time and estimated distance of task k, respectively. And t_{ar} and d_{at} are the average remaining time and average estimated distance of the current tasks. ω_1 and ω_2 are the weight coefficient which are calculated based on analytic hierarchy process (AHP) from the perspective of efficiency, and timeliness. In this case, the weight coefficient are calculated as $\omega_1 = 0.37, \omega_2 = 0.63$.

Comparisons are also conducted with two classic RL method, namely Q-learning and Sarsa (Kaelbling, Littman & Moore, 1996), and the five dispatching rules (FCFS, STD, EDD, LWT, NVF) separately in the different four scenes with 800 tasks randomly generated. To evaluate these methods comprehensively in terms of efficiency and timeliness, three indicators, namely final rewards R_f, makespan, and delay ratio D_r, are used in the comparison, and delay ratio is calculated as:

$$D_r = N_d/N_T \tag{17}$$

Where N_d and N_T are the number of delayed tasks and the number of total tasks in an episode, respectively. 20 episodes for each scene are carried out, and the results of average performance of all methods are illustrated in Table 4. It is obviously from the results that the proposed approach outperforms others in all three indicators, and the maximum improvements are 9.8% in terms of makespan and 10.9% in terms of delay ratio as seen in Fig. 8. In addition, the AHP-based method can obtain better makespan than Q-learning and Sarsa, but the delay ratio is relatively higher. On the other hand, the results also show that nearly none of the dispatching rules can achieve better performance in both makespan and delay ratio. For example, although EDD can obtain better

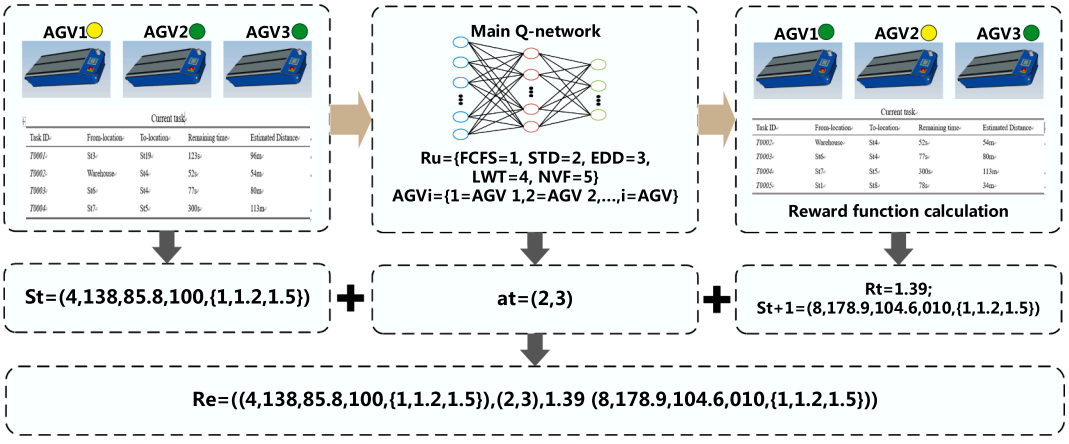
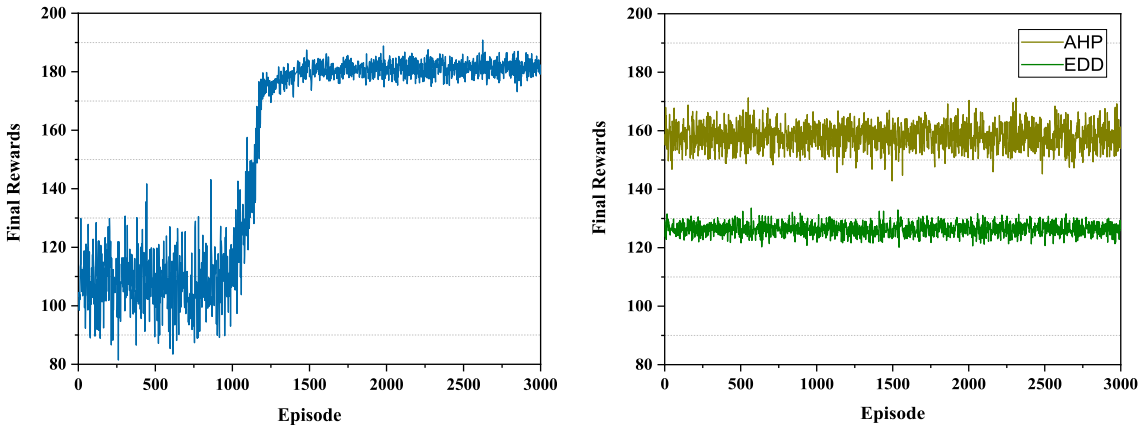


Fig. 6. The training process demonstration for an episode.



(a) The evolution of DQN (b) The evolution of AHP and EDD

Fig. 7. The evolution of the final rewards in the training.

Table 4
Results of comparison for different AGVs real-time scheduling method.

Scheduling method	Makespan(min)	Delay ratio (%)	Reward
DQN	544.6	12.76	182.33
AHP-based	557.3	18.43	158.12
Q-Learning	576.8	16.22	152.38
Sarsa	561.5	15.37	161.26
FCFS	587.7	20.78	114.86
STD	582.1	22.87	123.97
EDD	596.2	15.21	126.42
LWT	603.9	18.92	118.36
NVF	578.3	23.69	129.82

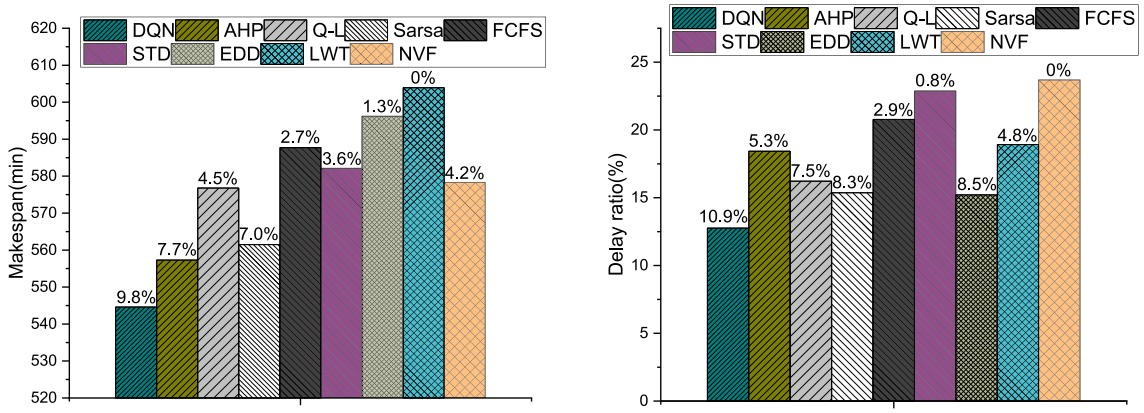
delay ratio (just slightly higher than DQN), it could lead to longer makespan because EDD always select the urgent tasks first. It reveals that the better performance can only be achieved by choosing the most appropriate dispatching rule according to different situations. Therefore, the experimental results can prove the effectiveness and reliability of the proposed approach for AGVs real-time scheduling in the flexible shop floor.

The response time is extremely critical for real-time scheduling methods to fulfill the dynamic shop floor environment. Therefore, the CPU processing time of each AGVs real-time scheduling method is also illustrated with table 5. It can be seen that the CPU processing time of the proposed method is slightly higher than other methods but is still acceptable to real-time scheduling (about 1.2 s for each schedule). The results also show that the time of dispatching rules is obviously shorter than all the three RL methods and AHP-based method because of lower computational complexity.

Based on the above analysis, compared with the traditional AHP-based method, the proposed method can select appropriate dispatching rules according to different states. Such that it can achieve better adaptability, but the weak in adaptability is precisely the drawback of the AHP-based method in which the weight coefficient of different attributes is constant. Compared with the classic RL methods (Q-learning and Sarsa), the proposed method can obtain better overall performance, and the reason is that the Q-network is used to deal with complex high-dimensional state information effectively. Meanwhile, the overall

Table 5
CPU processing time of different AGVs real-time scheduling methods.

Scheduling method	Mean CPU time (s)
DQN	1.22
AHP-based	0.87
Q-Learning	1.06
Sarsa	1.14
FCFS	0.33
STD	0.57
EDD	0.71
LWT	0.65
NVF	0.89



(a) Improvements of makespan (b) Improvements of delay ratio

Fig. 8. Improvements of different AGVs real-time scheduling method.

performance of Sarsa is slightly better than Q-learning, and they both significantly outperform than most dispatching rules. It also reveals that classic RL method can achieve relatively optimized policy but still have limitations in performance improvement.

5. Conclusion and future work

In this paper, to reduce the makespan and delay ratio of AGVs real-time scheduling for the flexible shop floor in industry 4.0, a novel real-time scheduling approach using DRL is proposed. By formulating the problem as an MDP, state representation, action representation, and reward function is introduced to simplify the complex material handling process. With plenty of training in a flexible shop floor based scene, an optimal mixed rule policy can be achieved to select the appropriate dispatching rules and AGVs according to different situations. The results of the case study, including training process and comparison experiments, has validated the effectiveness and adaptation of the proposed approach. The future work will focus on three aspects. The first is the extension of the DRL based approach to real-world AGVs scheduling application and the navigation of AGVs control. The second is the improvement on DRL based approach considering more complex operation situations and uncertainties. The third is the multi-objective optimization of AGVs real-time scheduling considering more objectives such as energy consumption, equipment utilization, maintenance cost, etc.

CRediT authorship contribution statement

Hao Hu: Conceptualization, Methodology, Software, Investigation, Writing - original draft. **Xiaoliang Jia:** Supervision, Project administration. **Qixuan He:** Software, Data curation, Visualization. **Shifeng Fu:** Data curation, Writing - original draft. **Kuo Liu:** Software, Validation.

Acknowledgement

This work was supported in part by the Shaanxi Collaborative Innovation Project under Grant 2016XT-19, and in part by the National Science Foundation of China under Grant 52075452.

References

Aydin, M. E., & Oztemel, E. (2000). Dynamic job-shop scheduling using reinforcement learning agents. *Robotics and Autonomous Systems, 33*(2), 169–178.
Arulkumaran, K., Deisenroth, M. P., Brundage, M., & Bharath, A. A. (2017). Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine, 34*(6), 26–38.
Berman, S., & Edan, Y. (2002). Decentralized autonomous AGV system for material handling. *International Journal of Production Research, 40*(15), 3995–4006.
Banino, A., Barry, C., Uria, B., Blundell, C., Lillicrap, T., Mirowski, P., … Kumaran, D. (2018). Vector-based navigation using grid-like representations in artificial agents. *Nature, 557*(7705), 429–433.
Chen, C., Xi, L. F., Zhou, B. H., & Zhou, S. S. (2011). A multiple-criteria real-time scheduling approach for multiple-load carriers subject to lifo loading constraints. *International Journal of Production Research, 49*(16), 4787–4806.
Chen, C., Xia, B., Zhou, B., & Xi, L. (2015). A reinforcement learning based approach for a multiple-load carrier scheduling problem. *Journal of Intelligent Manufacturing, 26*(6), 1233–1245.
Demesure, G., Defoort, M., Bekrar, A., Trentesaux, D., & Djemai, M. (2018). Decentralized motion planning and scheduling of AGVs in an FMS. *IEEE Transactions on Industrial Informatics, 14*(4), 1744–1752.
Han, X., Liu, H., Sun, F., & Zhang, X. (2019). Active object detection with multistep action prediction using deep Q-network. *IEEE Transactions on Industrial Informatics, 15*(6), 3723–3731.
Ho, Y., & Liu, H. (2006). A simulation study on the performance of pickup-dispatching rules for multiple-load AGVs. *Computers & Industrial Engineering, 51*(3), 445–463.
Klei, C. M., & Kim, J. (1996). Agv dispatching. *International Journal of Production Research, 34*(1), 95–110.
Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research, 4*(1), 237–285.
Kousi, N., Koukas, S., Michalos, G., & Makris, S. (2019). Scheduling of smart intra – factory material supply operations using mobile robots. *International Journal of Production Research, 57*, 801–814.
Kousi, N., Koukas, S., Michalos, G., Makris, S., & Chryssolouris, G. (2016). Service oriented architecture for dynamic scheduling of mobile robots for material supply. *Procedia CIRP, 55*, 18–22.
Kousi, N., Michalos, G., Makris, S., & Chryssolouris, G. (2016). Short-term planning for part supply in assembly line using mobile robots. *Procedia CIRP, 44*, 371–376.
Li, Z., Barenji, A. V., Jiang, J., Zhong, R. Y., & Xu, G. (2018). A mechanism for scheduling multi robot intelligent warehouse system face with dynamic demand. *Journal of Intelligent Manufacturing*, 1–12.
Lin, J., Yu, W., Zhang, N., Yang, X., Zhang, H., & Zhao, W. (2017). A survey on internet of things: architecture, enabling technologies, security and privacy, and applications. *IEEE Internet of Things Journal, 4*, 1125–1142.
Lee, J., Davari, H., Singh, J., & Pandhare, V. (2018). Industrial Artificial Intelligence for industry 4.0-based manufacturing systems. *Manufacturing Letters, 18*, 20–23.
Ma, S., Zhang, Y., Lv, J., Yang, H., & Wu, J. (2019). Energy-cyber-physical system enabled management for energy-intensive manufacturing industries. *Journal of Cleaner Production*, 892–903.
Mousavi, M., Yap, H. J., Musa, S. N., Tahriri, F., & Dawal, S. Z. (2017). Multi-objective AGV scheduling in an FMS using a hybrid of genetic algorithm and particle swarm optimization. *PLOS ONE, 12*(3).
Mourtzis, D., Zogopoulos, V., & Xanthi, F. (2019). Augmented reality application to support the assembly of highly customized products and to adapt to production re-scheduling. *The International Journal of Advanced Manufacturing Technology, 105*, 3899–3910.
Mourtzis, D., & Vlachou, E. (2018). A cloud-based cyber-physical system for adaptive shop-floor scheduling and condition-based maintenance. *Journal of Manufacturing Systems, 47*, 179–198.
Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., … Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature, 518*(7540), 529–533.
Miyamoto, T., & Inoue, K. (2016). Local and random searches for dispatch and conflict-free routing problem of capacitated AGV systems. *Computers & Industrial Engineering, 91*, 1–9.
Michalos, G., Kousi, N., Makris, S., & Chryssolouris, G. (2016). Performance assessment of production systems with mobile robots. *Procedia CIRP, 41*, 195–200.
Qiu, C., Yu, F. R., Yao, H., Jiang, C., Xu, F., & Zhao, C. (2019). Blockchain-based software-defined industrial internet of things: A dueling deep Q-learning approach. *IEEE Internet of Things Journal, 6*(3), 4627–4639.
Pan, J., Zi, Y., Chen, J., Zhou, Z., & Wang, B. (2018). LiftingNet: A novel deep learning network with layerwise feature learning from noisy mechanical data for fault classification. *IEEE Transactions on Industrial Electronics, 65*(6), 4973–4982.
Saidimehrabad, M., Dehnaviarani, S., Evazabadian, F., & Mahmoodian, V. (2015). An Ant Colony Algorithm (ACA) for solving the new integrated model of job shop scheduling and conflict-free routing of AGVs. *Computers & Industrial Engineering*, 2–13.
Singh, N., Sarngadharan, P. V., & Pal, P. K. (2011). Agv scheduling for automated material distribution: A case study. *Journal of Intelligent Manufacturing, 22*(2), 219–228.
Sabuncuoglu, I. (1998). A study of scheduling rules of flexible manufacturing systems: A simulation approach. *International Journal of Production Research, 36*(2), 527–546.
Shiue, Y., Lee, K., & Su, C. (2018). Real-time scheduling for a smart factory using a reinforcement learning approach. *Computers & Industrial Engineering*, 604–614.
Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Den Driessche, G. V., … Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature, 529*(7587), 484–489.
Ulusoy, G., Sivrikayaşerifoglu, F., & Bilge, U. (1997). A genetic algorithm approach to the simultaneous scheduling of machines and automated guided vehicles. *Computers & Operations Research, 24*(4), 335–351.
Vis, I. F. A. (2006). Survey of research in the design and control of automated guided vehicle systems. *European Journal of Operational Research, 170*(3), 677–709.
Wang, J., Zhang, Y., Liu, Y., & Wu, N. (2019). Multiagent and bargaining-game-based real-time scheduling for internet of things-enabled flexible job shop. *IEEE Internet of Things Journal, 6*(2), 2518–2531.
Wang, W., Zhang, Y., & Zhong, R. Y. (2020). A proactive material handling method for CPS enabled shop-floor. *Robotics and Computer-integrated Manufacturing*.
Wallace, & Andrew. (2001). Application of AI to AGV control?agent control of AGVs. *International Journal of Production Research, 39*(4), 709–726.
Wang, Y., & Usher, J. M. (2005). Application of reinforcement learning for agent-based production scheduling. *Engineering Applications of Artificial Intelligence, 18*(1), 73–82.
Wan, Z., Li, H., He, H., & Prokhorov, D. V. (2019). Model-free real-time EV charging scheduling based on deep reinforcement learning. *IEEE Transactions on Smart Grid, 10*(5), 5246–5257.
Zhang, Y., Ma, S., Yang, H., Lv, J., & Liu, Y. (2018). A big data driven analytical framework for energy-intensive manufacturing industries. *Journal of Cleaner Production*, 57–72.
Zhang, Y., Zhu, Z., & Lv, J. (2018). CPS-based smart control model for shopfloor material handling. *IEEE Transactions on Industrial Informatics, 14*(4), 1764–1775.
Zhang, Y., Zhang, G., Du, W., Wang, J., Ali, E., & Sun, S. (2015). An optimization method for shopfloor material handling based on real-time and multi-source manufacturing data. *International Journal of Production Economics*, 282–292.