Journal of Manufacturing Systems 56 (2020) 175–187

Contents lists available at ScienceDirect

Journal of Manufacturing Systems

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

Deep reinforcement learning for a color-batching resequencing problem

Jinling Leng[a], Chun Jin[a,*], Alexander Vogl[b], Huiyu Liu[c]

[a] School of Economics and Management, Dalian University of Technology, Dalian, China
[b] BMW Brilliance Automotive Ltd., Shenyang, China
[c] School of Mechanical Engineering and Automation, Northeastern University, Shenyang, China

ARTICLE INFO

Keywords:
Deep reinforcement learning
Color-Batching problem
Virtual car resequencing
Production control
Automotive industry

ABSTRACT

In automotive paint shops, changes of colors between consecutive production orders cause costs for cleaning the painting robots. It is a significant task to re-sequence orders and group orders with identical color as a color batch to minimize the color changeover costs. In this paper, a Color-batching Resequencing Problem (CRP) with mix bank buffer systems is considered. We propose a Color-Histogram (CH) model to describe the CRP as a Markov decision process and a Deep Q-Network (DQN) algorithm to solve the CRP integrated with the virtual car resequencing technique. The CH model significantly reduces the number of possible actions of the DQN agent, so that the DQN algorithm can be applied to the CRP at a practical scale. A DQN agent is trained in a deep reinforcement learning environment to minimize the costs of color changeovers for the CRP. Two experiments with different assumptions on the order attribute distributions and cost metrics were conducted and evaluated. Experimental results show that the proposed approach outperformed conventional algorithms under both conditions. The proposed agent can run in real time on a regular personal computer with a GPU. Hence, the proposed approach can be readily applied in the production control of automotive paint shops to resolve order-resequencing problems.

1. Introduction

Reinforcement Learning (RL) has made tremendous progress in board games [1], image processing [2], and real-time strategy games [3]. Nevertheless, there are few applications in the production control of the automotive manufacturing industry. Determining how to integrate advanced artificial intelligence algorithms into the conventional manufacturing industry has provoked extensive attention in recent years [4]. In this paper, we apply Deep Reinforcement Learning (DRL) to the production control of the Color-batching Resequencing Problem (CRP) of automotive paint shops.

In automotive manufacturing plants, production control engineers strive for high production sequence adherence of the body shop, paint shop and assembly line to prescheduled assembly production dates and sequences to ease the management and operation of a mixed-model paint shop production line [5,6]. However, in real-world paint shop applications, if consecutive production orders do share an identical painting color, this explicitly causes costs for solvent-cleaning the robots and painting nozzles [7]. Therefore, production engineers of paint shops pursue grouping orders with the same color as an identical color block, named color-batching, to minimize the consumption of paint cleaning and water pollution [7]. The problem of minimizing the total costs of color changeovers by resequencing the production sequence is defined as the CRP in this paper.

The objective of the CRP can be evaluated by the minimal Number of Color changeovers (NC) and the minimal Cost of Color changeovers (CC) criteria. On the one hand, the NC objective assumes that the color changeover costs are the same regardless of colors. In this case, the objective of the CRP is simply to minimize the number of color changeovers [8,9]. On the other hand, the CC objective assumes that the color changeover costs vary from color to color. In this case, the objective of the CRP is to reduce the sequence-dependent costs of paint color changeovers [10,11].

To obtain color-oriented batches of orders, mix bank (MB) buffer systems, also named selectivity banks, are widely used for color-batching resequencing [7]. MBs are installed in production lines in the format of parallel in-line conveyors. MBs have the advantages of low investment [7] and high similarity to the real-world industry applications [8,12]. Each order refers to one car body, as shown in Fig. 1. Car-bodies from the upstream conveyor enter parallel lanes of an MB following a storage strategy at the storage junction. Car-bodies are released from lanes of the MB by a release strategy to the downstream conveyor at the retrieval junction. This complex storage and release control strategy alters the upstream storage-sequence to be a

* Corresponding author.
E-mail address: jinchun@dlut.edu.cn (C. Jin).

https://doi.org/10.1016/j.jmsy.2020.06.001
Received 21 January 2020; Received in revised form 1 June 2020; Accepted 2 June 2020
Available online 20 June 2020

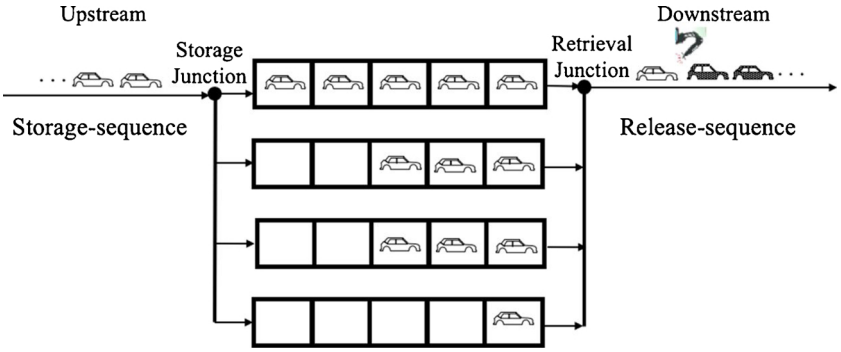

Fig. 1. An MB with four lanes, each having five spaces.

downstream color-batched release-sequence. At a practical scale, the CRP is a non-trivial problem because the number of cars with the same color is usually more than spaces per lane and because the number of colors is more than the number of lanes. Therefore, it is mandatory to develop control strategies of resequencing orders to format an optimal color-batched order sequence.

In an MB, an order cannot pass another order in its lane. If orders with the same color are inevitably split by orders with other colors in the same lane, they would incur color changeovers. A Virtual Car Resequencing (VCR) technique was proposed [13] to improve the flexibility of MBs. VCR kept the physical positions of the car-bodies and switched the color attributes of orders that have the identical car-model attributes to reduce the color changeover costs. The CRP integrated with VCR is known to be an NP-hard problem [7,13–15]. Though many studies have implemented optimization methods [8–11,16,17] for solving the CRP, these methods have obvious limitations in solving large-sized instances in real-world applications.

To address these issues at a practical scale, in this paper, we applied a Deep Q-Network (DQN) algorithm to the CRP integrated with the VCR technique. We proposed a Color-Histogram (CH) model to significantly reduce the number of possible actions from the factorial to the MB capacity to the number of color attributes so that the DQN algorithm can be used to solve the CRP integrated with the VCR technique. A DQN agent, which utilizes deep neural networks to learn a Q-value function, is trained in an RL environment to minimize the costs of color changeovers for the CRP.

Two experiments were conducted and evaluated in terms of minimizing NC and CC. In experiment I, the objective was set to minimize the NC from the perspective of theoretical exploration. The attributes of orders were set to be uniformly distributed. This setup has been widely used in academic studies of the CRP. The experimental results showed that the proposed approach outperforms benchmarks in terms of performance and processing time. In experiment II, the objective was set to minimize the CC objective from the perspective of the practical application of the CRP in paint shops. The CRP parameters, which included the distributions of color and car-model attributes and the sequence-dependent cost matrix, were derived from historical data of a real-world paint shop. The experimental results proved that the agent could learn and adapt to the CRP parameters and achieved significantly better performance than the benchmark methods. The proposed approach fulfilled the gap between theoretical exploration and practical application of the CRP.

There are two contributions in this paper. First, a CH model is proposed to describe the CRP as a Markov Decision Process (MDP). Compared to directly applying a DQN to the problem, our model reduces the number of possible actions so that the DQN algorithm can solve this NP-hard problem. Second, we propose a DQN-based DRL algorithm to solve the CRP integrated with the VCR technique. The DQN agent, based on the convolutional neural network, can learn and adapt to the practical CRP parameters, i.e., the distribution of orders and the metric of color changeover costs, and outperformed the benchmark approaches in a practical-scale simulated environment.

The remainder of the paper is organized as follows. Literature review is given in Section 2. In section 3, we describe the CH model and model the CRP as an MDP, which can be solved by the DQN algorithm. The details of the DQN algorithm are outlined in section 4. In section 5, we describe the experiments and analysis of the results. Section 6 involves our conclusions.

2. Literature review

2.1. Color-batching resequencing problem

Numerous studies have focused on the paint shop CRP in combination with MB buffer systems. Some studies have developed storage and release strategies to alter order sequences [8–11,16,17]. These studies adopted mathematical programming methods such as exact branch-and-bound algorithms [18], mixed-integer linear program [9,10], heuristics [10,16], integrated simulation and mathematical programming approaches [8], integrated simulation and heuristic algorithms [17], and dynamic programming algorithms [10,11]. However, the performance of these algorithms is limited without involving VCR.

The VCR technique was proposed to improve the color-batching performance of the CRP [13–15]. As proven in [14], the CRP in combination with VCR has a complexity of $O(a^n)$ when solved by a dynamic programming approach, where a denotes the number of orders in total and n denotes the product of the number of colors and the number of models. In a typical paint shop application in the real world, the scale of the CRP is considered to be approximately $a = 1000$, $n = 200$. Xu and Zhou provided four heuristic rules and a beam search algorithm to solve VCR of the CRP [13]. However, the algorithm was applied only to limited-scale experiments ($a = 56$, $n = 200$). Sun and Han [15] proposed a heuristic rule for the CRP with VCR and applied the algorithm at a practical scale. Their computational results outperformed those of simple rules and branch-and-bound algorithms proposed in [18]. However, their VCR technique was based on randomly swapping the color attributes of two orders each time; this approach used non-trivial computational time and was highly unstable as the number of orders increased, thus limiting its performance.

To simplify the problem, some studies [8,9,15,16] have considered that the objective of the CRP is to minimize the NC. These studies assumed that the cost of different color changeovers was an identical value and described by a binary variable. However, in the real-world paint shops, the cost of color changeovers varies from color to color. This is because the cost of color-changing depends on the sequence of the two colors. For example, if the color changes from a light color to a dark color, the dark paint can cover the residual light paint, but when a darker color changes to a lighter one, the first few painted bodies after the switch will incur fatal painting defects [17,19]. A high percentage

of defects will cause a reduction in the production volume, increasing the cycle time and waste the energy of the overall plant. Therefore, algorithms in [10,11] have used a sequence-dependent cost matrix to describe the setup costs for different colors and minimize the CC objective. However, these algorithms have not been thoroughly investigated in a practical-scale production environment. In this paper, we evaluate the proposed approach using both the NC and CC objectives at a practical scale.

2.2. Deep reinforcement learning

Since existing algorithms based on dynamic programming or heuristic rules struggle to solve the CRP at a practical scale, we propose using DRL for the CRP. DRL has already been studied in the production scheduling field [20–25]. In particular, the Deep Q-Network (DQN) algorithm [26] proposed by Google DeepMind has been widely used due to its ability to fit various types of problems. For example, Woo et al. proposed a DQN approach for earthwork scheduling [23]. Waschneck [24] proposed multiple DQN agents to solve a global production scheduling problem. Lin et al. proposed a multi-class DQN for a job-shop scheduling problem [25]. Considering the great success that was achieved by applying DQNs to many scheduling problems, it is tempting to apply the DQN algorithm to the CRP integrated with VCR technique. However, directly applying DQN to the CRP integrated with VCR can be challenging, since during VCR process, the action space of swapping the color attributes of orders has tremendously large dimensions. In the worst cases, the number of possible actions is factorial to the MB capacity. If the DQN algorithm is directly implemented, the dimension of the output vector of the Q-network will be unacceptably large.

3. Modeling

3.1. Problem characterization of the CRP

A color changeover is counted if two consecutive orders are painted in different colors. A painting color changeover will incur an additional cost. To minimize the consumption of spray robots cleaning and water pollution, color-oriented batches of bodies are applied. Conventionally, the CRP can be described as follows.

A storage-sequence contains G orders that are to be filled in and then released from an MB. Each order is marked by two attributes: car-model and color. Let M denote the number of different car-models and C denote the number of different colors, index $c = 1,2, \ldots C,\ m = 1,2, \ldots M$.

The storage-sequence is described as two attribute-based sequences:

$$IC = [ic_1, ic_2, \ldots, ic_G],\ \forall\, g \in [1, G],\ 1 \le ic_g \le C \tag{1}$$

$$IM = [im_1, im_2, \ldots, im_G],\ \forall\, g \in [1, G],\ 1 \le im_g \le M \tag{2}$$

Where, IC denotes the color attributes of orders in the storage-sequence, and IM denotes the car-model attributes of orders in the storage-sequence.

The orders in the storage-sequence are popped out and filled into an MB which consists of L lanes, with index $l = 1,2, \ldots L$, each of which is a linear First-In-First-Out (FIFO) conveyor of length N, with index $n = 1,2, \ldots N$. The size of the MB is defined as $MB\_size = L \times N$. The MB is initialized with a storage level, i.e., work-in-process (WIP) inventory, of orders. Each time, one order from the storage-sequence is filled into the MB. After each storage process, one order will be released from the MB. We assume that the total number of orders in the MB is constantly equal to WIP. The storage and release processes continue until the storage-sequence is empty; thus, an episode of the CRP is completed, and a release-sequence of length G is formed.

The color and car-model attributes of a release-sequence are donated as the following two sequences:

$$RC = [rc_1, rc_2, \ldots, rc_G],\ \forall\, g \in [1, G],\ 1 \le rc_g \le C \tag{3}$$

$$RM = [rm_1, rm_2, \ldots, rm_G],\ \forall\, g \in [1, G],\ 1 \le rm_g \le M \tag{4}$$

Note that VCR is conducted before each release and exchanges the color attributes among orders with identical car-model attributes.

Three decisions have to be made in each iteration: the storage procedure, the VCR process and the release procedure. The storage procedure decides to which lane an order should go, and the release procedure decides from which lane an order should be released. In an episode of the CRP, three decision sequences IL, RL and VL are generated.

$$IL = [il_1, il_2, \ldots, il_G],\ \forall\, g \in [1, G],\ 1 \le il_g \le L \tag{5}$$

$$RL = [rl_1, rl_2, \ldots, rl_G],\ \forall\, g \in [1, G],\ 1 \le rl_g \le L \tag{6}$$

Where, IL denotes the storage decision sequence, and RL denotes the release decision sequence. For the VCR process, each decision is described by a matrix $V \in \mathbb{R}^{WIP \times WIP}$, whose element $V_{i,j} \in \{0,1\}$ denotes whether an order i in the MB will transfer its color attribute to an order j. The VCR decision sequence is denoted as

$$VL = [vl_1, vl_2, \ldots, vl_G],\ \forall\, g \in [1, G],\ vl_g \in \mathbb{R}^{WIP \times WIP} \tag{7}$$

A color changeover from color i to color j is denoted by $CCM_{i,j}$, which is an element of the color cost matrix $CCM \in \mathbb{R}^{C \times C}$. If the objective of the CRP is the NC, the color cost matrix can be assumed to be

$$CCM = 1 - E \tag{8}$$

where E is an identical matrix.

If the objective is the CC, the cost matrix is derived from production experience.

The overall objective of the CRP is to determine the three decision sequences (5, 6, 7) to minimize the cost of total color changeovers of the release-sequence.

$$\min_{\mathrm{IL,RL,VL}} \sum_{g=1}^{G-1} CCM_{rc_g, rc_{g+1}} \tag{9}$$

This is subject to the following constraints:

1) Storage distribution:

The elements of IC and IM follow the color and car-model distributions D_c and D_m, respectively. These distributions can be assumed to be uniform or derived from the historical data.

2) The FIFO constraints:

The lanes of the MB are FIFO conveyors; therefore, orders can only be released from the front of a lane or filled into the end of a lane.

3) Storage constraints:

Orders must be filled into a lane that contains less than N orders.

4) Release constraints:

Orders must be released from a non-empty lane.

5) VCR constraints:

Orders can only transfer their color attributes to orders that have identical car-model attributes.

Fig. 2. presents an illustration of the formulation of the CRP. In this example $L = 3,\ N = 5,\ MB_{size} = 15,\ WIP = 3,\ C = M = 3,\ G = 10$.

In practice, the scale of the CRP is usually large, e.g., $L = 10,\ N = 10,\ MB\_size = 100,\ WIP = 70,\ C = 20, M = 10,\ G = 1000$.

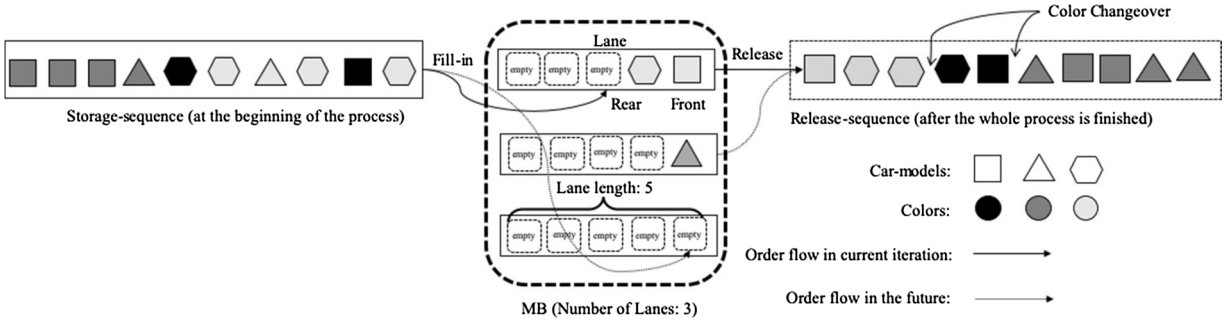

Fig. 2. An illustration of the CRP with an MB buffer system.

Since conventional programming algorithms do not fit large-scale problems, we proposed using a DQN to solve the CRP in a practical scale. In this paper, because the CRP has a discrete action space, the dimension of the output vector of the Q-network of the DQN algorithm [26] is equal to the number of possible actions. Considering a case of $L = 10$, $WIP = 70$, the output of the Q-network will have a length of $L \times L \times WIP! \approx 10^{100}$, which is not implementable. Since the DQN algorithm cannot be directly applied to the CRP with VCR, we proposed a CH model for the CRP with VCR to reduce the possible actions to an implementable size.

3.2. Color-histogram modeling of the problem

We find that it is not necessary to record all colors of orders in the MB independently. In contrast to modeling orders in the MB with both color and car-model attributes as in previous studies [15], we propose a new model that records a histogram of the color attributes of all orders in the MB, named a color-histogram (CH) model. The CH model is compatible with VCR because the color attributes of orders with the same model attributes can be arbitrarily exchanged without explicit color exchange actions. With the proposed model, the number of possible actions of the DQN agent can be significantly reduced. This model consists of a tensor SM and matrix MC that are integrated to describe the state of the MB, as shown in Fig. 3.

We record the car-model attributes of orders rather than the color attributes. Tensor SM describes the car-model distribution of orders in the MB. Element $SM_{l,n,m}$ of tensor SM represents that an order with the car-model attribute m is located in lane l and position n of the MB.

We employ a matrix MC to describe the distribution of colors in each type of car-models. The rows of MC correspond to each color, and the columns of MC correspond to each type of car-models. The matrix can be interpreted as a job list that indicates how many orders of each color and car-model must be produced from the unpainted orders in the MB. By modeling the MB in such a manner, orders with the same car-model attributes can switch their color freely. The matrix is initialized with zeros when the MB is empty. Every time that an order with color c and car-model m is filled into an MB, the element $MC_{c,m}$, which corresponds to the color and model attributes of the order, will increase by 1.

Based on the CH description of the MB, we redefine the corresponding storage and release procedures of the color-batching resequencing with an MB to suit the CRP to the DQN algorithm.

We separate the release procedure into two steps: Step 1. Color selection: selecting a color to be the current color C_c. The DQN agent conducts the color selection step, which is explained in Section 3; Step 2. Release by color: finding an order that can be painted in the color C_c. Considering $C_c = c$, an order with car-model m in an arbitrary lane l that is subject to $MC_{c,m} > 0$, $SM_{l,1,m} > 0$, will be selected. The corresponding element in the MC decreases by 1. If no orders can be selected, one color change will happen. Then step 1 will be called. Fig. 4 shows the pseudo-code of the release-by-color process.

The storage procedure delivers the first order of the storage-sequence and inserts it into a certain lane of the MB. The storage process aims to cluster orders with identical car-model attributes in one lane. By

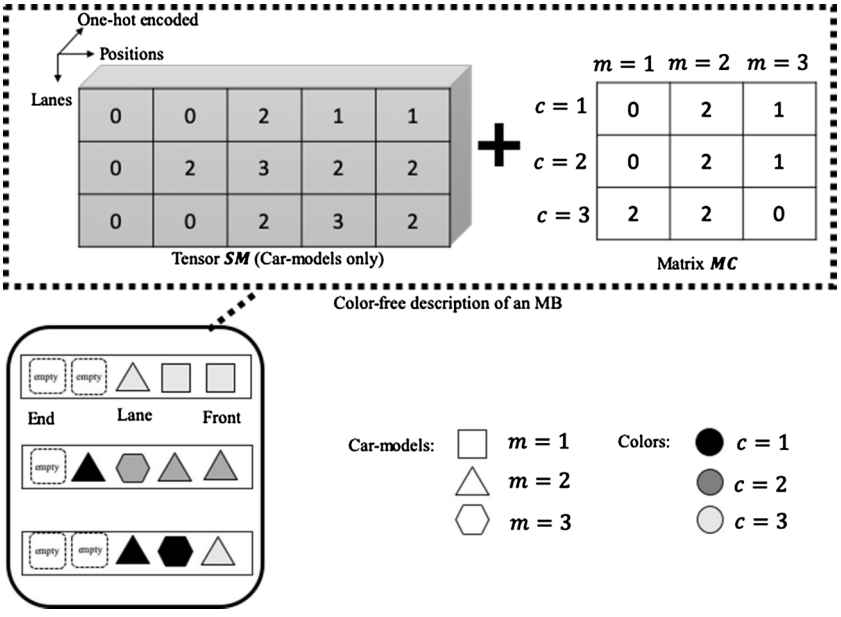

Fig. 3. Illustration of the CH model.

Algorithm 1: Release by color

Input: C_c, *SM, MC*
Output: whether any orders can be released

```
c = C_c
for l= 1,2...L do
    for m= 1,2...M do
        if MC_{c,m} > 0 & SM_{l,1,m} > 0 then
            MC_{c,m} ← MC_{c,m} − 1   # update MC
            release the first order on lane l of SM
            return True   # an order is released in the current color
        endif
    endfor
endfor
Return False   # no orders can be released in the current color
```

Fig. 4. Pseudo-code of the release by color procedure.

Algorithm 2: Storage Rules

Input: *Storage-sequence, SM, MC*
Output: results of fill-in

```
F ← pop the first order from the storage-sequence
m ← the car-model attribute of the F
c ←the color attribute of the F
for l= 1,2...L do
    if lane l is full then
        continue
    endif
    if (car-model attribute of rear order on lane l) = m then
        insert the order to lane l of SM
        MC_{c,m} ← MC_{c,m} + 1  # update MC
        exit
    endif
endfor
l ←select a random lane   # No lanes satisfy the condition
insert the order to lane l of SM
MC_{c,m} ← MC_{c,m} + 1  # update MC
Return True
```

Fig. 5. Pseudo-code of the storage procedure.

creating large order clusters, the car-model distribution of the orders in position $n = 1$ of each line can be diversified. This increases the selection options in step 2 of the release procedure. The pseudo-code of the storage process is shown in Fig. 5.

In this way of modeling, the VCR process needs no swapping decisions anymore. The storage procedure follows an Algorithm 2, which does not require any decisions. The only decision needed in the CRP is selecting the current color. The CRP can now be described as follows.

The definitions of *IC*, *IM*, *RC*, *and RM* are the identical to (1), (2), (3) and (4), respectively. The storage procedure following Algorithm 2 and the release-by-color procedure following Algorithm 1 are iteratively processed. If Algorithm 1 returns False, a color selection decision is made, and then the release-by-color procedure will be processed again. The color selection decision sequence is defined as

$$CS = [cs_1, cs_2, \ldots, cs_G] \tag{10}$$

Each decision cs_g is defined as

$$cs_g = \begin{cases} 0 & Algorithm\ 1\ returns\ True \\ c & Algorithm\ 1\ returns\ False,\ select\ color\ c \end{cases},\ c \in [1, C], g \in [0, G] \tag{11}$$

The color change cost is defined identically to that in the previous model. With the proposed a CH model, the objective of the CRP is to find the optimal decision sequence (10) that minimizes the cost of total color changeovers of the release sequence:

$$\min_{CS} \sum_{g=1}^{G-1} CCM_{rc_g, rc_{g+1}} \tag{12}$$

This is subject to the following constraints:

1) Storage distribution:

The elements of *IC* and *IM* follow the color and car-model distribution D_c and D_m, respectively. These distributions can be assumed to be uniform or derived from the historical data.

2) The FIFO constraints:

The lanes of the MB are FIFO conveyors; therefore, orders can only be released from the front of a lane or filled into the end of a lane.

3) Storage constraints:

Orders must be filled into a lane that contains less than N orders.

4) Release constraints:

Orders must be released from a non-empty lane.

5) Color selection constraints:

For any selected color c, there must be at least one order in the MB subject to $\exists\ l \in [1, L], \exists\ m \in [1, M]\ MC_{c,m} > 0,\ SM_{l,1,m} > 0$.

Under practical conditions, i.e., $L = 10,\ N = 10,\ MB_{size} = 100,\ WIP = 70,\ C = 20, M = 10, and\ G = 1000$, with the proposed a CH model, the number of possible decisions is $C = 20$. Therefore, the DQN algorithm can be implemented for the CRP with a reasonable dimension of the output vector of the Q-network. To solve the CRP with the DQN, the CRP must be modeled as an MDP.

3.3. Markov decision process

The MDP is a subclass of discrete-time systems that requires a sequence of decisions to achieve a certain goal [27,28]. Considering an agent that interacts with an environment, an MDP consists of four elements.

$$<S,\ A,\ T,\ r> \tag{13}$$

Where, S is the state of the system, A is a set of possible choices for the decision, T is the transition probability distribution from one state to another, and r is the immediate reward between two consecutive actions. By definition, an MDP is subject to

$$T(s, a, s') = p_s(s'|s,\ a) \tag{14}$$

Where, s denotes the current state of the system; a denotes the decision made in the current state; s' denotes the next state of the system; and p_s denotes the transition probability distribution. This can be interpreted as that the next state of the system depends only on the state of the system and the decision made for the current moment. The agent will receive an immediate reward r from the environment after every action a is executed under state s, which is subject to

$$p_r = p_r(s, a, r) \tag{15}$$

Where, p_r represents the probability of obtaining reward r at state s given action. In an MDP, the agent observes the state s of the environment and takes action a based on policy π.

$$a = \pi(s) \tag{16}$$

Then, the environment will accept the action and return a new state s' and an immediate reward r. s' is taken as a new state, and the process will iteratively continue until reaching the end condition $t = \tau$. The

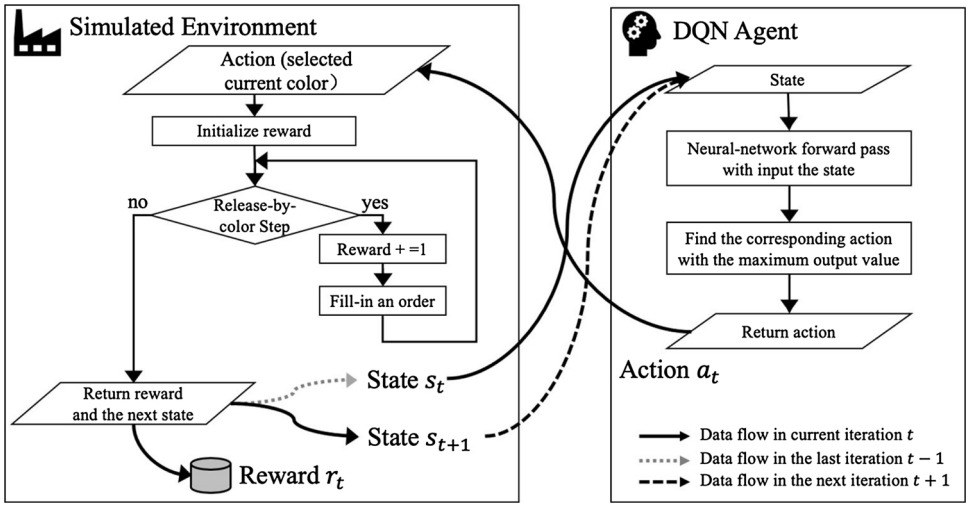

Fig. 6. Dataflow between the DQN agent and the environment.

overall goal of an MDP is to maximize the accumulated reward with an optimal decision policy π^*.

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{p_s,p_r}[\sum_{t=1}^{\tau} r_t \mid p_s = T(s_t, a_t, s_{t+1}),\ p_r = p_r(s_t, a_t, r_t),\ a_t = \pi(s_t)] \tag{17}$$

3.4. Modeling the CRP as an MDP

We consider the release-by-color procedure and the storage procedure as the environment in an MDP. An agent based on a DQN algorithm proposed in [26] will observe the state of the MB and select the current color to be painted as an action. Fig. 6 shows the flow chart of the simulated environment and the agent.

In iteration t, the state S_t is observed by the agent. The agent will output an action a_t, which represents the current color C_c. The action a_t will be passed to the simulated environment. The simulator will simulate the storage and release procedures until the color has to be changed again. Then, a reward r_t will be calculated and recorded in a database. The details of how to initialize and update the reward are introduced in Section 3.4.3. The next state S_{t+1} will be ready to be observed by the agent in the next iteration.

3.4.1. Parameterization of states

We define the state as four variables: $s = \{SM \in \mathbb{R}^{L\times N\times M}, MC \in \mathbb{R}^{C\times M}, LC \in \mathbb{R}^{C\times 1}, SS \in \mathbb{R}^{1}\}$.

Concretely, tensor SM describes the car-model distribution of the orders in the MB. The car-model of each order is one-hot encoded at the corresponding position of the MB.

$$SM_{l,n,m} = \begin{cases} 1 & \text{order in lane } l \text{ on position } n \text{ have car-model attribute m} \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

Matrix MC is precisely the matrix that describes the distribution of colors in each type of car-models. Element $MC_{c,m}$ of matrix MC represents the number of orders in color c and car-model m that are required to be produced from the unpainted orders in the MB.

Vector LC is a one-hot-encoded vector of the color of the last painted order, whose element is LC_c.

$$LC_c = \begin{cases} 1 & c \text{ is the last painted color} \\ 0 & \text{otherwise} \end{cases} \tag{19}$$

Scalar SS is the ratio of the length of the storage-sequence to the number of all planned orders.

$$SS = \frac{L_{in-seq}}{L_{plan-seq}} \tag{20}$$

Where, $L_{plan-seq}$ denotes the number of orders planned to be produced and is equal to the total length of the storage-sequence at the beginning. L_{in-seq} is the length of unpainted orders. The orders are popped from the front of the storage-sequence one by one until the storage-sequence is empty.

3.4.2. Parameterization of actions

Action a is a one-hot- encoded vector of the current selected color and is coded by a binary vector of length C.

$$a = [a^1, a^2, ..., a^C] \tag{21}$$

$$a^c = \begin{cases} 1 & c \text{ is the selected color} \\ 0 & \text{otherwise} \end{cases},\ c = 1. ..C \tag{22}$$

After an action is taken, the orders in the selected color will be painted, and the environment will transit to the next state.

3.4.3. Parameterization of rewards

A sequence-independent cost matrix CCM is introduced to describe the color changeover costs between different colors. $CCM_{i,j}$, which is the element on row i and column j of CM, denotes the cost of color changeovers from the color i to color j.

Based on the flowchart on the left in Fig. 6, when an action changes the current color from $C_c = i$ to $C_c = j$, the reward r is initialized as $-CCM_{i,j}$, and the r increases by 1 every time an order in color j is released. The reward r is only recorded when no orders can be released by the Release-by-color step. The recorded reward r is equal to:

$$r = uc + (-CCM_{i,j}) \tag{23}$$

where uc denotes the number of orders that have been painted between the current state and the next state. r is positively correlated with the number of orders and negatively correlated with the cost.

3.4.4. Parameterization of the goal function

The goal of an MDP is to find an optimal policy π^*, which can be described in (17). In this CRP, τ in (17) is the total number of color changeovers. The summation in (17) covers from the beginning to the end of the CRP. $p_s = T(s_t, a_t, s_{t+1})$ represents the probability distribution of transitions. $p_r = p_r(s_t, a_t, r_t)$ represents the probability of obtaining reward r_t at state s_t given action a_t, *where* $a_t = \pi(s_t)$ is the action taken under state s_t following policy π. Following the optimal policy π^*, the agent can maximize the expected accumulated reward for the whole process.

Considering (23) and (17), since the first term uc in (23) is the number of orders painted in each batch, the summation of uc through the whole process must be $L_{plan-seq}$, which is constant. Therefore, to maximize the accumulated reward is to minimize the color changeovers costs. Finding π^* in (17) can be interpreted as finding a strategy to paint the orders with the minimal total cost of color changeovers.

4. Solving the CRP with deep reinforcement learning

4.1. Deep reinforcement learning

With the rapid development of deep learning techniques, DRL has been widely used to solve MDP problems. We employ a DQN algorithm [26] to solve the CRP. Define Q function as:

$$Q_\pi(s_{now}, a_0) = \mathbb{E}_{p_s,p_r}\left[\sum_{t=now+1}^{\tau} r_t, p_s = T(s_t, a_t, s_{t+1}),\ p_r = p_r(s_t, a_t, r_t),\ a_t = \pi(s_t)\right] + r_{now} \tag{24}$$

Given the state s_{now} and the action a_0, the Q-function expects the accumulated reward from now to the end of the MDP. The first term in (24) represents the expectation of the accumulated reward from $now+1$ to the end of the MDP. The other term $r_{now} = \mathbb{E}_{p_r}[r_t p_r = p_r(s_{now}, a_0, r_t)]$ represents the expectation of the immediate reward by the current action a_0.

In the CRP, the possible actions given by any state are discrete. If the Q -value for all actions given any states can be estimated, the decision can be made by comparing every $Q(s, a)$ value of all possible actions and selecting the action with the maximum Q-value $Q(s, a^*)$. We approximate the Q function with a neural network.

$$Y = N(s; \theta) \tag{25}$$

Where, N denotes the neural network with parameters θ; the output vector $Y \in \mathbb{R}^{C\times 1}$ denotes the Q-value corresponding to each possible action. The network can be optimized by the Bellman Equation,

$$Q_{\pi^*}(s_t, a) = r_t + Q_{\pi^*}(s_{t+1}, a^*) \tag{26}$$

Where, π^* denotes the optimal policy and $a^* = \pi^*(s_{t+1})$ denotes the action that will be taken by the optimal policy. Considering (25), the optimal policy is to apply the action that maximizes the Q-value,

$$a^* = \arg\max_a Y(s_t; \theta)_a \tag{27}$$

where $Y(s_t; \theta)_a$ denotes the element of vector $Y(s_t; \theta)$.

4.2. Architecture of the network

We use a convolutional neural network, namely a Q-network, to approximate the Q function. The architecture of the Q-network is shown in Fig. 7. All layers are attached with a Rectified Linear Unit (ReLU) function as the activation function.

Input tensor SM contains the car-model information of the orders in the MB, and the matrix MC contains the color information of the orders in the MB. The vector LC represents the last painted color. These three inputs must be considered together to decide which color is the best to paint. We proposed a Correlation Module (CM) to correlate this information. The output of the CM is a tensor that contains the correlated information of these inputs.

The network contains two convolutional blocks: Conv0 consists of 4 convolutional layers, whose kernel sizes are 3×3. The first three layers have 256 kernels and the last layer has 32 kernels. Conv1 consists of 4 convolutional layers. The kernel sizes of the first and last layers are 1×1. The kernel sizes of the other two layers are 3×3. The first three layers have 256 kernels, and the last layer has 32 kernels. The output of the two convolutional blocks is flattened and reduced to 256 dimensions by two fully connected layers, i.e., FC0 and FC1 in Fig. 7, and then fused by another fully connected layer, i.e., FC-fuse in Fig. 7.

The rest of the network is based on Dueling DQN [29]. Note that the input scalar SS represents the progress of the total work and is only related to the V-value instead of the advantage value. Therefore, this scalar is only fed to the layer corresponding to the estimation of the V-value. The definitions of the V-value and advantage value can be referred to in [29].

Fig. 8 shows the layout of the CM. The input tensor SM represents the car-model attribute distribution of orders in the MB. Since lanes in the MB are FIFO conveyors, only the order at position $n = 1$ of a lane can be immediately released. It is very important to consider the orders near position $n = 1$ of each lane when selecting the next painting color because these orders are very likely to be released shortly.

Three sub-tensors are extracted from tensor SM, representing the car-model distributions of orders at positions $n = 1$, positions $n = 1,\ 2$ and positions $n = 1,\ 2,\ 3$ of the corresponding lanes. These sub-tensors are summed on the first dimension and then merged as a tensor of size $3 \times 1 \times M$. This tensor, which contains only car-model information, is then merged by element-wise minimization with tensor MC, which contains the information of both car-model and color. Note that these tensors are repeated to match the dimension before merging. Finally, the last painted color must be considered due to the different color changeover costs between different colors. Thus, vector LC is repeated and merged by concatenation.

Theoretically, adding more layers to the neural network can achieve a comparable capacity of the network as applying the proposed CM does. However, in practice, we found that compared with adding more layers, applying the proposed CM can stabilize the training and significantly improve the performance.

4.3. Training

The training of the neural network is based on the double DQN method used in [30]. The network was initialized with two copies: $N_q(\theta_q)$ as the target network, $N_p(\theta_p)$ as the policy network. Different from [30], some actions in the CRP may be invalid. When the agent needs to select a color, the environment tries every color with the release-by-color step virtually. If no orders can be released by any color attribute, these colors are marked as invalid. A binary vector $mask_t = [ma_1, ma_2, ..., ma_C]$ is provided to the agent to avoid invalid colors.

$$ma_c = \begin{cases} 1 & c \text{ is the available color} \\ 0 & \text{otherwise} \end{cases} \tag{28}$$

The agent read the state s_t from the environment and take action a_t^*,

$$a_t^* = \underset{a}{\text{argmax}}\, N_q(s_t)_a * mask_t \tag{29}$$

As shown in Fig. 6, the environment paints as many orders with the selected color as possible. Then the environment provides a new state s_{t+1}, a new mask $mask_{t+1}$, a binary number indicates whether it is the end of an episode end_t, and a reward r_t of this action.

$$end_t = \begin{cases} 0 & \text{t is the end of an episode} \\ 1 & \text{otherwise} \end{cases} \tag{30}$$

The environment records a set $EX_t = \{s, \text{mask}_t, s_{t+1}, \text{mask}_{t+1}, a_t, r_t, end_t\}$ every time when a decision is made by the agent. All records are stored in a set named replay buffer $Rep = \{EX_1, EX_2, ..., EX_{now}\}$, where now is the number of decisions made in an episode of the simulation (from the start to the current state).

The parameters of network N_q are updated every U steps. A mini-

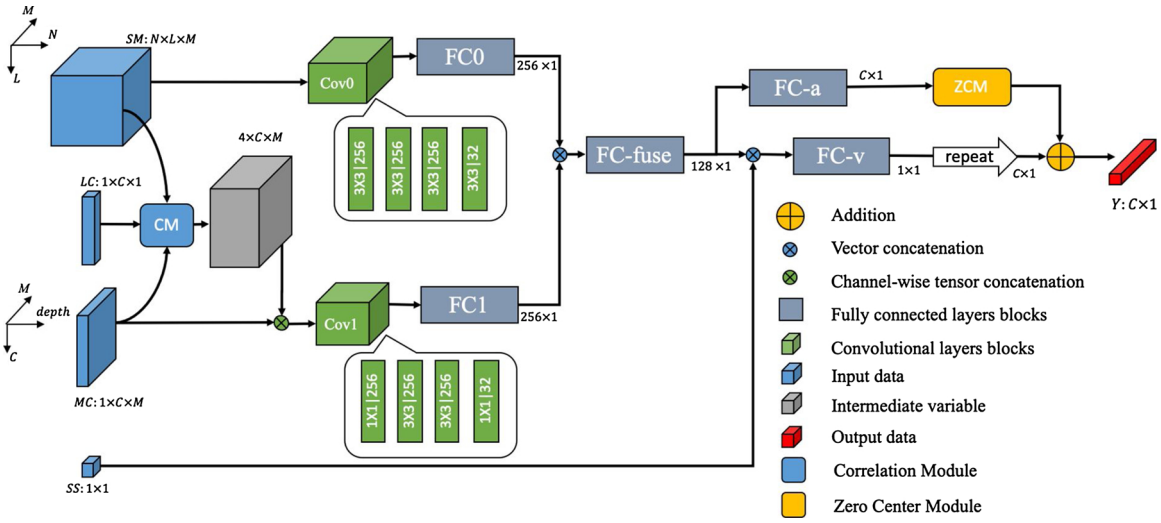

Fig. 7. The architecture of the Q-network.

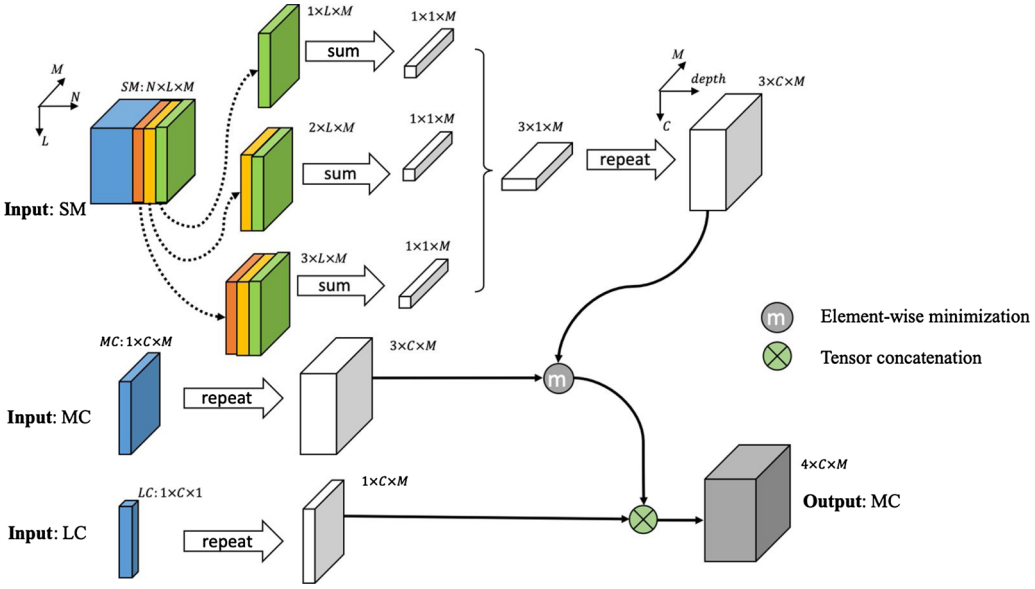

Fig. 8. The layout of the Correlation Module.

batch of experiences are sampled from the replay buffer. The losses are averaged among the mini-batch. The loss is defined as,

$$\text{Loss}= ||N_q(s_t, a_t) - N_p(s_{t+1})_{a^*} \times end_t - r_t,\ a^* = \underset{a}{\text{argmax}}\, N_q(s_{t+1})_a * mask_{t+1}\,||_2 \tag{31}$$

The target network N_q and the policy network N_p synchronizes every Y episodes.

The pseudo-code of the training process is shown in Fig. 9.

5. Numerical study

Two experiments are conducted and evaluated in terms of the NC and CC objectives of the CRP, respectively. In experiment I, the objective is set to minimize the NC from the perspective of theoretical exploration. The attributes of orders are set to be uniformly distributed. This setup is widely used in academic studies of the CRP. In experiment II, the objective is set to minimize the CC from the perspective of the practical application of the CRP. The CRP parameters, which include the distributions of color and car-model attributes and the sequence-dependent cost matrix, are derived from historical data of a real-world paint shop.

5.1. Data generation and setup

To test the effectiveness and performance of the proposed DQN algorithm, we conduct a series of tests based on the parameters that are listed in Table 1. We take [15] as our benchmark. We use the same settings as the benchmark to facilitate an accurate comparison of the performance of the proposed algorithm. The MB in each test consisted of L lanes with length N; C_{pre} presents the storage percentage of the MB, which is $C_{pre} = WIP/(L \times N) \times 100\%$. Reference [15] proposed that the best C_{pre} level for color-batching performance was 60 % (for MB 7×8, 10×10) and 70 % (for MB 5×6). Hence, we compare the performance based on this best C_{pre} level. All experiments were conducted in a computer-simulated environment. The car-model attribute of planned orders obeys the distribution D_m, and the color attribute obeys the distribution D_c; these distributions are generated in different experiments. In each experiment, $L_{plan-seq} = 1000$. Initially, the $WIP = 0$ in the MB. Orders are not released until filled-in orders increase the WIP

Algorithm: Training of the Q network

Input:
$N_q(\theta_q)$ # target network
$N_p(\theta_p)$ # policy network
U # update interval
Y # synchronize interval
Output: $N_p(\theta_p)$

```
initialize θ_q, θ_p, e ← 0
while e < total training episodes do
    e ← e+1
    SS ← 0
    Initialize in-sequence
    L_in−seq ← L_plan−seq
    while L_in−seq > 0 do
        obtain s_t, mask_t from simulated environment
        select color a based on (18)
        initialize reward r ← −CCM_{i,j}
        while True do
            obtain SM, MC, C_c from simulated environment
            result ← call Algorithm 1 (C_c, SM, MC)
            if result do
                break
            endif
            call Algorithm 2 (in-sequence, SM, MC)
            SS ← SS+1
            update reward r ← r+1
        endwhile
        Obtain s_{t+1}, mask_{t+1} from environment
        insert record {s_t, s_{t+1}, mask_t, mask_{t+1}, a, r} to the replay buffer
        s_t ← s_{t+1}
        mask_t ← mask_{t+1}
        if SS % U =0 then
            sample data from the replay buffer
            calculate loss according to (20)
            update N_p(θ_p)
        endif
    endwhile
    if e % Y=0 then
        N_q(θ_q) ← N_p(θ_p)
    endif
endwhile
return N_p(θ_p)
```

Fig. 9. The pseudo-code of the training process.

and the desired C_{pre} is reached. The experiments end when the length of the storage-sequence $L_{in-seq} = 0$.

The hyperparameters in RL are fixed among all experiments. The learning rate is 0.0001, the target network synchronizes every 10 episodes, the replay buffer size is 10000, and the min-batch size of the training is 128. The Q-network is trained independently in each experiment. In each experiment, the network is trained for 200 episodes, and the mean cost of the 10 latest episodes is taken as the validation criterion. Six tests are conducted in the experiments. Each test is performed for 10 episodes, and the results are averaged.

Experiments were performed on a desktop with an Intel I7 Central Processing Unit (CPU), NVIDIA 1080Ti graphics card and 64 GB of Random Access Memory (RAM). The program was coded in Python 3.6 [31] and the interference of the Q-network was executed on the graphics card. The operating system was Ubuntu 14.04.

5.2. Experiment I: Performance of the algorithm for the NC objective

5.2.1. Benchmarks

We take [15] as a benchmark for comparison with our proposed DQN algorithm to investigate the performance of the color-batching control strategy under uniform distributions of the color and car-model attributes of the orders. The benchmark and our work have similar backgrounds and assumptions but differ in the modeling of the CRP and color-batching strategies. On the one hand, the CRP in the MB in the benchmark is modeled with both car-model and color attributes. Our proposed approach employs a CH model. On the other hand, the color-batching control strategies in the benchmark are based on integrated heuristic rules and the VCR technique. Our proposed control strategies are conducted by a DQN agent that is trained by the DQN algorithm.

Based on the assumption that orders in the MB with the same car-model attribute can swap their color attributes, the benchmark proposed VCR in combination with the MB and randomly swapped the color attributes between orders with the same car-model attribute to improve the performance. The termination condition of the random swapping was originally proposed to be reaching the maximum number of iterations or reaching the lower bound of the color changeover costs. The performance and efficiency of the VCR can be balanced by adjusting the maximum number of iterations.

In experiment I, we implement the benchmark in three ways, which are named bench [15], bench10, and bench0, to explore the different preferences between performance and efficiency. Bench [15] and bench10 conduct integrated heuristic algorithms and the VCR technique to optimize the color-batching problem, and the maximum numbers of iterations in these algorithms are 100 and 10, respectively. Bench0 only applies the heuristic rule-based storage and release approach of [15] and skips the VCR technique.

5.2.2. Results analysis

In this experiment, the performance of the proposed DQN algorithm is evaluated via the NC objective, which is set to minimize the total number of color changeovers. For two color attributes i and j, the number of color changeovers is defined as (8).

The results of the numerical study and a comparison with bench [15], bench10 and bench0 are tubulated in Table 2. In all 12 tests of experiment I, the colors and car-models of the orders follow two uniform integer distributions $D_{c1} = unif(1, C)$ and $D_{m1} = unif(1, M)$, respectively. $\%Adv1$ represents the improved color-batching performance (NC) of the proposed DQN algorithm compared with that of a heuristic algorithm and is formulated as (32). $\%Adv2$ represents the improved color-batching performance (NC) of bench [15] compared with that of bench0. $\%Adv3$ represents the improved color-batching performance (NC) of the proposed DQN algorithm compared with that of the heuristic rules of bench0.

$$\%Adv1 = \frac{(\text{Bench [15] NC} - \text{DQN NC})}{\text{DQN NC}} \times 100\% \tag{32}$$

Table 1
Parameter values for computational experiments.

Parameter	Notations	Benchmark [15]	Values in experiment I	Values in experiment II
MB configuration	$L \times N$	5×6, 7×8, 10×10	5×6, 7×8, 10×10	5×6, 7×8, 10×10
Storage percentage	C_{pre}	60 %, 70 %	60 %, 70 %	60 %, 70 %
Number of colors	C	10, 20	10, 20	14
Color distribution	D_c	D_{c1}, D_{c2}	D_{c1}	D_{c2}
Number of car-models	M	5, 10	5, 10	7, 10
Car-model distribution	D_m	D_{m1}, D_{m2}, D_{m3}	D_{m1}	D_{m2}, D_{m3}
Number of orders in the upstream storage-sequence	$L_{plan-seq}$	1000	1000	1000

Table 2
Comparison of NCs and computational time at different MB configuration.

Test	$L \times N$	C_{pre}	(C, D_c, M, D_m)	Bench [15]		Bench10		Bench0		DQN		$\%Adv$		
				NC	Time (s)	NC	Time (s)	NC	Time (s)	NC	Time (s)	1	2	3
1	5 × 6	70 %	(10, D_{c1}, 5, D_{m1})	304.6	56.05	315.1	27.16	336.8	0.328	249.2	0.776	22%	11 %	35 %
2			(20, D_{c1}, 10, D_{m1})	462.4	311.6	470.1	52.36	524.9	0.385	514.8	1.380	−10%	14 %	2%
3	7 × 8	60 %	(10, D_{c1}, 5, D_{m1})	223.6	103.7	235.1	58.61	206	0.488	129.9	0.520	72 %	−8%	59 %
4			(20, D_{c1}, 10, D_{m1})	365.5	398.0	383.4	102.0	416.9	0.581	338.5	1.066	8%	14 %	23 %
5	10 × 10	60 %	(10, D_{c1}, 5, D_{m1})	145.9	1292	179.4	308.2	118	0.843	69.4	0.413	110 %	−19%	70 %
6			(20, D_{c1}, 10, D_{m1})	274.5	959.6	304.7	349.9	283.6	0.888	175.9	0.788	56 %	3%	61 %

In all test results, bench [15] consistently performs better than bench10, and these results prove that the number of iterations can affect the color-batching performance when conducting the benchmark algorithm.

In most tests, the proposed DQN algorithm achieves the smallest NC, except in test 2. Especially in tests 3, 5, and 6, $\%Adv1$ shows that the proposed algorithm performs 72 %,110 %, 56 % better than the best benchmarks [15]. Note that the ratios $L \times N/M$ in these tests are larger than those in other tests; thus, the orders with the desired car-model attributes are less likely to be blocked by other orders. In test 2, the NC of the proposed algorithm is 11 % worse than that of the best benchmarks since the second test has the smallest ratio $\times N/M$. The effectiveness of the proposed algorithm is limited because the orders with the desired car-model attributes could very likely be blocked by other orders.

$\%Adv2$ shows the advantage in the NC of bench [15] compared with that of bench0 in tests 1, 2, 4, and 6. However, in tests 3 and 5, the results are the opposite. Although the VCR technique is effective in reducing the number of color changeovers, the optimization performance is unstable. The virtual resequencing of bench [15] in tests 3 and 5 hurts the color-batching performance since randomly swapping the color attributes of the orders in bench [15] is only used for pursuing a short-term reduction in the NC (the orders inside the bank). Minimizing the short-term NC can lead to adverse effects on the long-term color-batching performance. In contrast, $\%Adv3$ shows that the proposed DQN algorithm performs better than bench0 for all tests because the DQN agent was trained to maximize the Q-value, which is the long-term target. By the proposed approach, we solve the CRP in a long-term-oriented and stable way.

The proposed algorithm takes less computational time than bench [15] and bench10, and this result proves the effectiveness of the proposed DQN algorithm in all 6 tests. The bench [15] takes more time when the MB size is larger because the number of VCR iterative operations increases with the size of the MB. In contrast, based on the CH model, the proposed DQN algorithm does not need iterative operations. Therefore, the proposed algorithm is not affected in terms of computational time when the MB size increases. In tests 5 and 6, the computational time of the proposed algorithm is even shorter than of bench0. The proposed algorithm takes less time when the total NC is smaller because the agent only reacts with the environment if the current color has to be changed. Therefore, the proposed algorithm achieves excellent performance with trivial costs in computational time.

Furthermore, Fig. 10 displays comparisons of the training curves of the DQN algorithm and the heuristic algorithm of the benchmark [15]. The dotted lines show the color-batching performance of the benchmark algorithm.

The training curves of comparative tests 1, 3, 5 vs. 2, 4, 6 show that when the numbers of car-model and color attributes of orders are the same, the larger the size of the MB is, the lower the NC and the better the corresponding color-batching performance. Although the convergence speed slightly slows down with an increase in the size of the MB, the proposed algorithm can still converge stably within 100 episodes and 30 min., even in the case of an MB of size 10 × 10 of the MB.

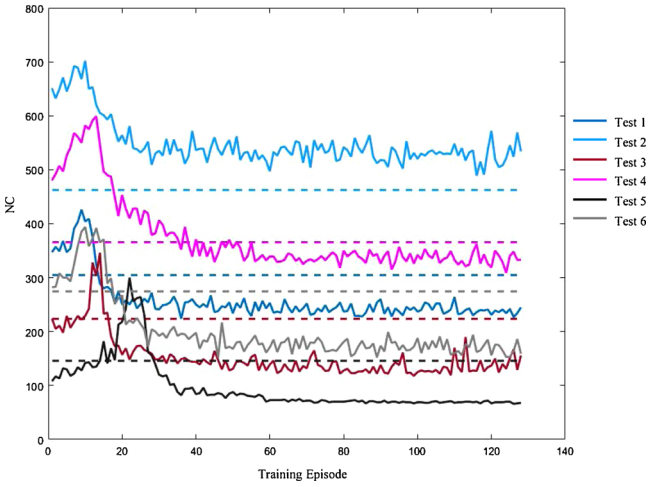

Fig. 10. The training curves of each test in experiment I.

The training curves of comparative tests 1 vs. 2, 3 vs. 4, and 5 vs. 6 point out that when the size and configuration of the MB are the same, increasing the number of car-model and color attributes of orders will increase the NC, but the convergence speed is unaffected. Therefore, the proposed DQN algorithm outperforms in terms of color-batching performance and computational time.

5.3. Experiment II: Performance of the algorithm for the CC objective

As shown above, experiment I illustrates the performance of the proposed approach with uniformly distributed attributes under the NC objective, which is commonly assumed in academic research [8,9]. In practice, the cost of color changeovers is asymmetrical because the difference between two colors can influence the difficulty of cleaning and the risk of reworking. The benchmarks were not proposed to deal with asymmetrical color changeover costs. We change the criterion of VCR from minimizing the NC to minimizing the CC so that the benchmarks can adapt to the asymmetrical *CCM* in this experiment.

In experiment II, we push the experimental environment closer to reality. The discrete distributions of the colors and car-models were based on real-world paint shop production data, as shown in Fig. 11. The cost matrix *CCM* used in this experiment is shown in Table 3. The benchmark was originally proposed to pursue the NC; thus, we implemented the benchmark algorithms with the CC target for comparison. In this implementation, the criterion of VCR is changed from the NC to the CC. The other parameters and the experimental platform are the same as those in experiment I.

Table 4 lists the costs of color changeovers and computational time of the proposed DQN algorithm and the heuristic algorithm of [15] in terms of the CC objective.

As shown in Table 4, the proposed DQN algorithm outperforms the

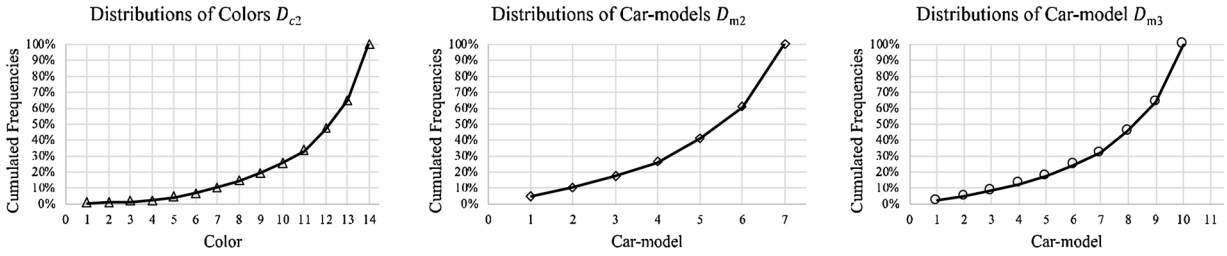

Fig. 11. Distribution of colors and car-models in experiment II.

Table 3
The cost matrix of color changeovers (*CCM*) in experiment II.

j	*i*													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	0	2.1	2.1	0.8	0.1	0.1	0.8	0.1	1.5	1.5	0.1	2.1	0.1	0.1
2	3.4	0	3.4	0.8	1.5	3.4	0.8	0.1	0.1	0.1	0.1	3.4	0.1	0.1
3	6.1	6.1	0	0.1	0.8	0.1	0.1	0.1	1.5	0.1	0.1	0.8	0.1	0.1
4	2.8	2.8	0.8	0	0.1	2.8	0.1	0.8	0.1	2.8	2.8	0.1	0.1	0.8
5	2.1	2.1	0.8	0.1	0	0.1	0.1	0.1	0.1	0.1	0.8	0.1	0.1	0.1
6	2.1	2.1	0.1	0.1	0.1	0	0.1	0.1	0.8	0.1	0.8	0.8	0.1	0.1
7	4.8	0.8	0.8	0.8	0.1	2.8	0	4.8	0.1	2.8	2.8	0.1	0.1	0.1
8	2.8	2.8	2.8	2.8	0.1	0.1	2.8	0	0.1	0.1	0.1	0.1	0.1	0.1
9	2.1	2.1	2.1	0.8	0.8	0.8	0.8	0.8	0	0.1	0.1	0.1	0.1	0.1
10	2.8	2.8	2.8	6.1	0.1	2.8	6.1	0.1	0.1	0	0.1	0.1	0.1	0.1
11	2.8	0.1	0.1	2.8	0.8	0.1	2.8	2.8	0.8	0.1	0	0.1	0.1	0.1
12	2.1	0.1	0.1	2.1	0.1	0.8	2.1	0.1	0.8	0.8	0.8	0	0.1	0.1
13	1.5	1.5	0.1	0.1	0.1	1.5	0.1	0.1	0.1	1.5	0.1	0.1	0	0.1
14	3.4	3.4	0.8	2.8	0.1	2.8	2.8	0.1	0.1	0.1	0.1	0.1	0.1	0

benchmarks in all tests in terms of color-batching performance and computational time. The advantages of the proposed approach are much more significant. As the car-models, colors and cost matrix of color changeovers are not uniformly distributed, the importance of every color attribute and car-model attribute is different. The benchmarks treat every color and car-model evenly. However, the proposed approach can learn the distribution and make decisions by considering the differing importance of the color and car-model attributes.

As shown in Table 4, the conclusions in terms of computational time are similar to those in experiment I. In all 6 tests, the proposed algorithm takes less computational time than bench [15] and bench10. In tests 5 and 6, the computational time of the proposed algorithm is even shorter than that of bench0. In addition, for each test setup, the proposed algorithm takes similar computational times in experiment I and experiment II, while the bench [15] takes much more time in experiment II than in experiment I. In summary, the proposed algorithm can consistently achieve excellent performance with short computational time in all the setups.

The set of plots in Fig. 12. show a comparison of the color-batching performance distributions of 10 episodes in each test. The Y-axis is the total number of color changeovers, and the X-axis is the bench [15], bench10, bench0 and the proposed approach. The smaller the NC of an approach is, the better the performance of that approach.

The proposed approach outperformed all benchmarks and is remarkably faster than bench [15] and bench10. Note that the proposed approach had a smaller variance in performance among the 10 repetitive runs in performance than the benchmarks in all tests. Because the parameters, e.g., color changeover costs and the distributions of the color and car-model attributes, are not uniformly distributed, these unevenly distributed parameters could potentially add variance to the performance. The benchmarks relied on randomly swapping color attributes and suffered from the variance of the parameters; in contrast, the proposed DQN algorithm could learn and adapt to the uneven distributions and achieved consistently high performance.

Fig. 13 shows the training curves of the DQN in each test. Although the convergence speed is different in each test, the DQN can consistently converge within 100 episodes. The performance at the beginning represents the inherent difficulty of the problem because the agent makes nearly arbitrary decisions to explore the environment. The more improvement there is from the beginning to convergence, the more effective the algorithm is. The performance improvements from the beginnings (episodes 0–5) to the convergences in this experiment are far greater than those in the experiment I. This result indicates that the proposed algorithm is more effective under practical experimental parameters. In addition, although tests 1,2,3 and 4 exhibit different performance at the beginning, the performance converges at a similar level in these tests. This results indicates that increasing the bank size from 5×6 to 7×8 is not as effective as an increase from 7×8 to 10×10, which is possibly a valuable indicator for the planning of a paint shop layout.

6. Conclusion

In this paper, we proposed a RL-based algorithm for the CRP. We used a CH model to describe the CRP to reduce the dimension of the action space so that the DQN algorithm can be applied to the CRP. The DQN agent is trained and tested in a simulated environment at a practical scale. Two experiments were conducted to evaluate the proposed approach. In experiment I, the color and car-model attributes of the orders follow uniform distributions, and the target is set as minimizing the NC. Compared with benchmarks, the proposed policy achieved better performance in 5 out of 6 tests in experiment I. In experiment II, the distribution of colors and car-models and the cost of color changeovers were set according to the real production data. The objective of the CRP was to minimize the CC. The proposed algorithm

Table 4
Comparison of CCs and computational time at different MB configuration.

Test	$L \times N$	C_{pre}	(C, D_c, M, D_m)	Bench [15]		Bench10		Bench0		DQN		*%Adv*		
				CC	time	CC	time	CC	time	CC	time	1	2	3
1	5×6	70 %	(14, D_{c2}, 7, D_{m2})	168.4	533.9	270.4	60.69	517.2	0.374	101.5	1.223	66%	207 %	410 %
2			(14, D_{c2}, 10, D_{m3})	192.3	540.5	289.2	62.07	525.7	0.373	116.6	1.367	65 %	173 %	351 %
3	7×8	60 %	(14, D_{c2}, 7, D_{m2})	135.1	1189	238.9	138.2	356.5	0.551	69.4	0.859	95%	164 %	414 %
4			(14, D_{c2}, 10, D_{m3})	139.4	1187	230.4	138.1	378.5	0.549	76.6	0.962	82%	172 %	394 %
5	10×10	60 %	(14, D_{c2}, 7, D_{m2})	131.3	3406	215.6	389.2	219.5	0.861	52.7	0.599	149 %	67 %	317 %
6			(14, D_{c2}, 10, D_{m3})	129.3	3476	214.9	374.5	221.2	0.838	66.8	0.716	94%	71 %	231 %

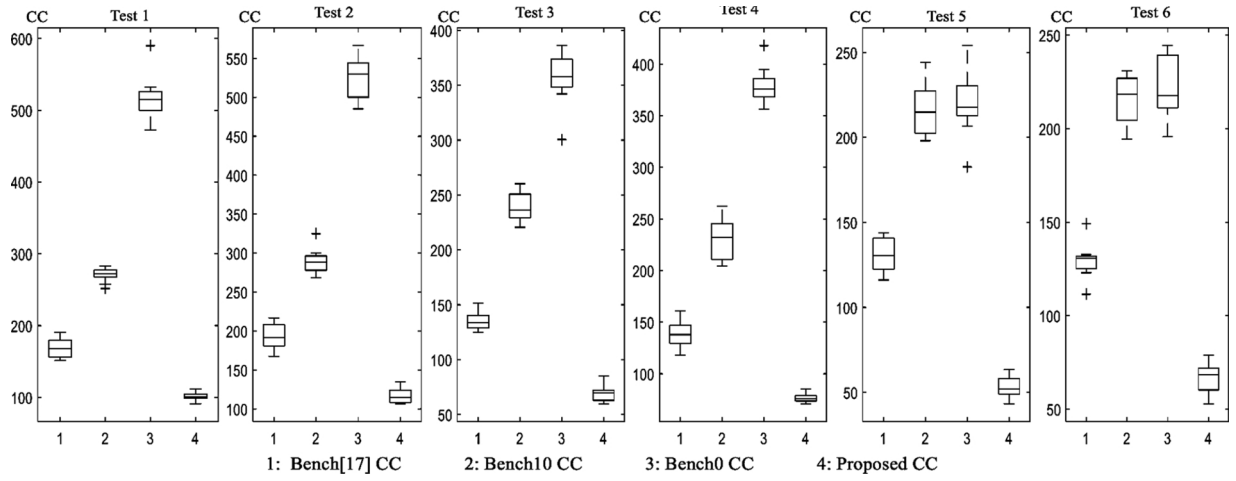

Fig. 12. Performance distributions of CCs in experiment II.

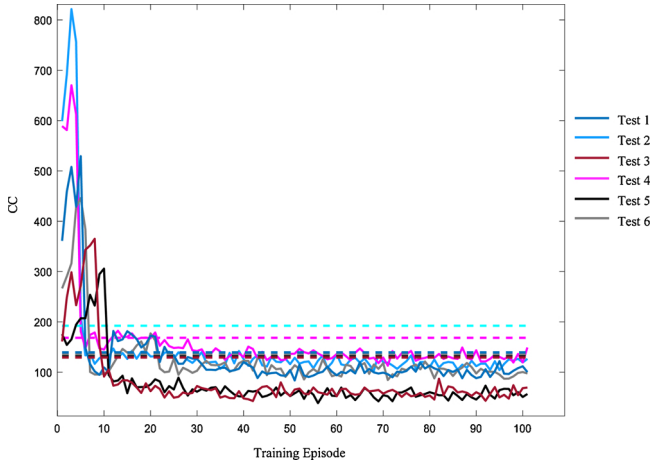

Fig. 13. The training curves of each test in experiment II.

had a 65 %–149 % advantage in performance compared with the benchmarks in all the tests. In both experiments, our policy not only has performed better than most of the benchmarks but also used less computational time. We observed that the proposed RL-based color-batching policy provides production and operation management with stronger sequence flexibility, cost reduction, and environmental friendliness in daily operation.

Declaration of Competing Interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would gratefully acknowledge the support by the National Natural Science Foundation of China (NSFC) as the research program under granted No.71671025.

References

[1] Silver D, Schrittwieser J, Simonyan K, Antonoglou I, Huang A, Guez A, et al. Mastering the game of go without human knowledge. nature 2017;550:354–9.
[2] Ghesu FC, Georgescu B, Zheng Y, Grbic S, Maier A, Hornegger J, et al. Multi-scale deep reinforcement learning for real-time 3D-landmark detection in CT scans. IEEE Trans Pattern Anal Mach Intell 2017;41:176–89.
[3] Vinyals O, Babuschkin I, Czarnecki WM, Mathieu M, Dudzik A, Chung J, et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. Nature 2019;575:350–4.
[4] Sharp M, Ak R, Hedberg Jr. T. A survey of the advancing use and development of machine learning in smart manufacturing. J Manuf Syst 2018;48:170–9.
[5] Moradi H, Zandieh M. An imperialist competitive algorithm for a mixed-model assembly line sequencing problem. J Manuf Syst 2013;32:46.
[6] Asadi N, Jackson M, Fundin A. Implications of realizing mix flexibility in assembly systems for product modularity—a case study. J Manuf Syst 2019;52:13–22.
[7] Boysen N, Scholl A, Wopperer N. Resequencing of mixed model assembly lines: survey and research agenda. Eur J Oper Res 2012;216:594–604.
[8] Elahi MML, Rajpurohit K, Rosenberger JM, Zaruba G, Priest J. Optimizing real-time vehicle sequencing of a paint shop conveyor system. Omega (United Kingdom) 2015;55:61–72.
[9] Taube F, Minner S. Resequencing mixed-model assembly lines with restoration to customer orders. Omega (United Kingdom) 2018;78:99–111.
[10] Ko SS, Han YH, Choi JY. Paint batching problem on M-to-1 conveyor systems. Comput Oper Res 2016;74:118–26.
[11] Hong S, Han J, Choi JY, Lee K. Accelerated dynamic programming algorithms for a car resequencing problem in automotive paint shops. Appl Math Model 2018;64:285–97.
[12] Meissner S. Controlling just-in-sequence flow-production. Logist Res 2010;2:45–53.
[13] Xu Y, Zhou JG. A virtual resequencing problem in automobile paint shops. Proceedings of the 22nd International Conference on Industrial Engineering and Engineering Management 2015;2016:71–80.
[14] Epping T, Hochstättler W, Oertel P. Complexity results on a paint shop problem. Discret Appl Math 2004;136:217–26.
[15] Sun H, Han J. A study on implementing color-batching with selectivity banks in automotive paint shops. J Manuf Syst 2017;44:42–52.
[16] Sun H, Fan S, Shao X, Zhou J. A colour-batching problem using selectivity banks in automobile paint shops. Int J Prod Res 2015;4(53):1124–42.
[17] Leng J, Jin C, Vogl A, Wortmann D. A hybrid simulation model for optimal color-batching resequencing in paint shop. In SummerSim' 19: Proceedings of the 2019 Summer Simulation Conference. 2019. p. 1–12.
[18] Spieckermann S, Gutenschwager K, Voß S. A sequential ordering problem in automotive paint shops. Int J Prod Res 2004;42:1865–78.
[19] Jaehn F, Kovalev S, Kovalyov MY, Pesch E. Multiproduct Batching and scheduling with buffered rework: the case of a car paint shop. Nav Res Logist 2014;61:458–71.
[20] Ou X, Chang Q, Chakraborty N. Simulation study on reward function of re-inforcement learning in gantry work cell scheduling. J Manuf Syst 2019;50:1–8.
[21] Cao Z, Lin C, Zhou M, Huang R. Scheduling semiconductor testing facility by using cuckoo search algorithm with reinforcement learning and surrogate modeling. Ieee Trans Autom Sci Eng 2019;16:825–37.
[22] Wei Y, Pan L, Liu S, Wu L, Meng X. DRL-Scheduling: An Intelligent QoS-Aware Job Scheduling Framework for Applications in Clouds. IEEE Access 2018;6:55112–25.
[23] Woo S, Yeon J, Ji M, Moon IC, Park J. Deep RL with fully convolutional NN to solve an earthwork scheduling problem. 2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC) 2019:4236–42.
[24] Waschneck B, Reichstaller A, Belzner L, Altenmüller T, Bauernhansl T, Knapp A, et al. Optimization of global production scheduling with deep reinforcement learning. Procedia Cirp 2018;72:1264–9.
[25] Lin CC, Deng DJ, Chih YL, Chiu HT. Smart manufacturing scheduling with edge computing using multiclass deep q network. IEEE Trans Industr Inform

2019;15:4276–84.
[26] Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, et al. Human-level control through deep reinforcement learning. Nature 2015;518:529–33.
[27] Hu L, Liu Z, Hu W, Wang Y, Tan J, Wu F. Petri-net-based dynamic scheduling of flexible manufacturing system via deep reinforcement learning with graph convolutional network. J Manuf Syst 2020;55:1–14.
[28] Wang KJ, Nguyen PH, Wu SY, Xue J. Technology portfolio adoption considering capacity planning under demand and technology uncertainty. J Manuf Syst 2018;47:1–11.
[29] Wang Z, Schaul T, Hessel M, Van Hasselt H, Lanctot M, De Freitas N. Dueling network architectures for deep reinforcement learning. 2015. arXiv preprint arXiv:1511.06581.
[30] Van Hasselt H, Guez A, Silver D. Deep reinforcement learning with double Q-learning. in: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence 2016:2094–100.
[31] Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, et al. PyTorch: an imperative style, high-performance deep learning library. Advances in Neural Information Processing Systems, 32. 2019. p. 8024–35.