Since January 2020 Elsevier has created a COVID-19 resource centre with free information in English and Mandarin on the novel coronavirus COVID-19. The COVID-19 resource centre is hosted on Elsevier Connect, the company's public news and information website.

Elsevier hereby grants permission to make all its COVID-19-related research that is available on the COVID-19 resource centre - including this research content - immediately available in PubMed Central and other publicly funded repositories, such as the WHO COVID database with rights for unrestricted research re-use and analyses in any form or by any means with acknowledgement of the original source. These permissions are granted for free by Elsevier for as long as the COVID-19 resource centre remains active.
Hybrid combinatorial remanufacturing strategy for medical equipment in the pandemic

You Shang, Sijie Li
School of Economics and Management, Southeast University, Nanjing 211189, China

ARTICLE INFO

Keywords:
Remanufacturing
Recovery option
Combinatorial optimization
Reinforcement learning
Real-time decision making

ABSTRACT

The COVID-19 pandemic hit the medical supply chain, creating a serious shortage of medical equipment. To meet the urgent demand, one realistic way is to collect abandoned medical equipment and then remanufacture, where the disassembled modules are shared with all stock-keeping units (SKUs) to improve utilization. However, in an emergency, the equipment should be processed sequentially and immediately, which means the decision is short-sighted with limited information. We propose a hybrid combinatorial remanufacturing (HCR) strategy and develop two reinforcement learning frameworks based on Q-learning and double deep Q network to find the optimal recovery option. In the frameworks, we transform HCR problem into a maze exploration game and propose a rule of descending epsilon-greedy selection on reweighted valid actions (DeSoRVA) and Expertate knowledge dictionary to combine the cost-minimizing objective with human judgment and the global state of the problem. A real-time environment is further implemented where the quality status of the in-transit equipment is unknown. Numerical studies show that our algorithms can learn to save cost, and the larger scale of the problem is, the more cost-down can be achieved. Moreover, the sophisticated knowledge refined by Expertate is effective and robust, which can handle remanufacturing problems at different scales corresponding to the volatility of the pandemic.

1. Introduction

The COVID-19 pandemic is rampant around the world, putting heavy pressure on the medical supply chain (Mehrotra et al., 2020). Particularly, new dangerous variants (such as Delta and Omicron) of coronavirus spread faster in these two years according to WHO reports. Patients struggle in and out of hospitals where ventilators and other medical equipment face serious shortages which are collected and listed by the US FDA. To some extent, the severe situation is due to limited supply and insufficient inventory, both of which are arranged for daily demand and unable to handle such an unexpected outbreak (Qi et al., 2021). What is worse, since the medical equipment consists of multiple modules (parts or components), its production can be suspended at any time even if only one module is out of stock (Vanhooydonck et al., 2021). Therefore, it is essential to utilize the existing resources as much as possible to improve the supply, especially during the pandemic.

In this regard, the manufacturer can recycle the abandoned medical equipment for remanufacturing to fill the gap between supply and demand (Mutha et al., 2016). In the short term, due to the intensive use, medical equipment will inevitably fail, whose majority are discarded as medical wastes. In the long term, due to the fluctuation of the pandemic, it can be expected a decrease in the demand for medical equipment but an increase in abandoned equipment when the peak of cases has just crossed. Thus, remanufacturing is significant in meeting urgent needs and avoiding overproduction. Even though some of the products are non-repairable, their modules (e.g., durable parts like the data processing chips) are mostly reusable. After repair, replacement, and other procedures, a certain proportion of the products can be restored as the remanufactured version.

Li and Shang (2021) first propose a hybrid remanufacturing pattern for the PCB-based product that its modules are connected to the printed circuit board (PCB) and find it suitable for the product with high material costs and low operating costs. Since the medical equipment, like computed tomography (CT) scanner, Magnetic Resonance Imaging (MRI) system, proton therapy system, etc., features such cost composition, we extend the hybrid remanufacturing to the medical equipment in the pandemic, and both used and new modules are jointly used to remanufacture for the urgent demand. Different from the electronic product (e.g., mobile phones, PCs) discussed in Li and Shang (2021), medical equipment encounters constraints of processing sequence and real-time in remanufacturing. Processing sequence brings...
the principle of “first come first dispose” due to the urgent need and the logistics process, that is to say, the modules in the in-transit equipment cannot be used in the in-processing equipment. Real-time means that remanufacturing decisions should be made in time based on incomplete information, for example, the quality status of the in-transit collected product is unknown before delivery. The existing algorithms do not perform well for the problem with sequential and real-time constraints (Liang et al., 2021).

In the hybrid combinatorial remanufacturing (HCR) problem of medical equipment, the most important is to decide on recovery options sequentially and stepwise to meet the urgent demand in time, which is defined as the decision sequence by considering the production cost in this paper. To visually examine the problem structure, we transform the original problem into a maze exploration game, where we can explore the solution by reinforcement learning (RL), a popular paradigm for sequential decision-making under uncertainty. The maze exploration frameworks are achieved by utilizing Q-learning and Double Deep Q Network (DDQN) to learn better decisions. In the frameworks, we propose a rule named descending epsilon-greedy selection on reweighted valid actions (DeSoRVA), which avoids short-sighted decisions and local-optimal solutions (Dittrich & Fohlmeister, 2020), accelerating the convergence of algorithms. We also propose a knowledge dictionary indexed by Espertate to generalize the experience into sophisticated knowledge. DeSoRVA integrates the exploration experience and human judgment, and the Espertate-indexed knowledge dictionary is a universal state representation for HCR problem.

The rest of the paper is organized as follows. The next section reviews the relevant literature. HCR problem is described in Section 3, and the optimization model is constructed. In Section 4, the equivalent maze problem and two RL frameworks are proposed. Section 5 presents numerical examples. Section 6 concludes the paper.

2. Literature review

Focus on the COVID-19 pandemic, the relevant research pays attention to the distribution of medical supplies and emergency production (Wu et al., 2020), whereas remanufacturing is seldom involved (Li & Shang, 2021). There are mostly two relevant research areas to our work: emergency production and RL-based production decisions. Emergency production is particularly important to fill the demand gap with limited resources. Vanhooydonck et al. (2021) study emergency production for surgical masks, respirators, and other personal protective equipment (PPE) by optimizing the assembly procedure and reducing the dependence on foreign suppliers. Qi et al. (2021) propose an IT-driven rapid production digital twin system for the emergency response to the production resumption, material distribution, and service delay in the COVID-19 pandemic, which provides optimized scheduling, distributed collaboration, and remote services. In addition to the COVID-19 pandemic, various catastrophes including extreme weather, terrorist attacks, and production accidents can also cause a shortage and need an emergency response. For instance, Li and Ou (2020) study the optimal order strategy for the stochastic demand when some parts need to be replenished by emergency channels with higher cost and find that the optimal order quantity of the complementary parts can still be independently determined. Shin and Lee (2020) propose a stochastic dynamic model for patient transport priorities and hospital selection based on status information of patients and hospitals, which is solved by a heuristic algorithm. Wang and Nie (2019) take traffic congestion as one key factor affecting material supply and propose a two-stage emergency supplies distribution model to plan emergency supplies pre-positioning and post-disaster transportation.

RL is beginning to be applied in production scheduling in recent years. Luo et al. (2021) investigate the real-time scheduling problem with job insertion and propose an online rescheduling framework named two-hierarchy deep Q network (THDQN) according to deep Q network (DQN) and double DQN (DDQN), where the continuous status is transformed into scheduling rules. Dittrich and Fohlmeister (2020) propose a collaborative multi-agent system (MAS) using deep Q learning (DQL) for the scheduling problem and demonstrate advantages with real-time capability and high flexibility. Wang et al. (2021) consider machine failures and rework and propose a dynamic scheduling production method based on deep RL, in which proximal strategy optimization (PPO) is used to find the best scheduling strategy. Zou et al. (2021) propose a multi-carrier energy system (MCES) scheduling algorithm. The results show that the RL framework can effectively and efficiently solve the problem under uncertainties. Kardos et al. (2020) propose a dynamic scheduling algorithm based on Q-learning, which assigns machines to each production step based on real-time information, it is found that RL can improve the average lead time of production orders. Khader and Yoon (2021) construct an RL-based adaptive control system for the stencil printing process, which integrates the clustering algorithm with heterogeneous reward functions to improve Q network performance. Liang et al. (2021) propose a cloud manufacturing service composition model based on the proposed dualing DQN, with priority replay to learn optimal service composition solutions.

Neural networks are suitable to solve emergency production problems with dynamic environments, heterogeneous data, and implicit interactions. Hennig et al. (2021) use neural networks to describe production actions, in which the recurring motifs are clustered first and then used for neural network regression training. Wu et al. (2020) propose a real-time parallel production scheduling algorithm based on the end-to-end neural network for the large-scale demand for masks during the COVID-19 pandemic. Moreover, the reinforcement agent can decide what to do to perform the given task and learn from its experience even without providing prior knowledge in reinforcement learning (Huang et al., 2020). Unlike the existing research, the medical equipment investigated in this paper is orderly delivered rather than parallel production, and then the optimal decision of the in-processing product considerably depends on its sequence number (i.e., its position in the delivery sequence).

3. Problem description and modeling

We consider a medical equipment manufacturer who performs emergency production by remanufacturing. With the two constraints mentioned in Section 1, HCR problem is divided into three categories.

Cat.1: Disordered and static. Neither sequential constraint nor real-time constraint is followed. The manufacturer waits till all collected products are delivered, and all disassembled modules can be used for any compatible product regardless of the sequence that the collected product reaches processing center.

Cat.2: Orderly and static. Only the decision sequence is sequential, where the inventory must not be less than zero. The manufacturer follows the principle of “first come first dispose”, by which the decision is made at $t$, and the production must be completed before or just after the delivery of the collected product. With the monitoring and detecting technologies based on the Internet of Things (IoT), the quality statuses of all collected products can be obtained before delivery.

Cat.3: Orderly and dynamic. Both two constraints are followed by the real-time decision problem with limited information. Different from Cat.2, the quality statuses of in-transit products are unknown in this case. In reality, it is difficult to make decisions with the lack of information.

In this paper, we consider that the manufacturer produces the medical equipment with $J$ SKU (stock-keeping unit) models indexed by $i (1 \leq i \leq J)$, where different models of the same medical equipment are treated as different SKUs, forming a flat product line. Let $S_j$ denote the quantity of $i$-SKU products and assume that the product with sequence number $s_j (1 \leq s_j \leq S_j)$ is currently in processing. The product $s_j$ consists of $J_t (J_t \geq 3)$ generic modules, which are indexed by $j_t$, and distinguished by three kinds of modules (type $A$, type $B$, and PCB).

$\text{Y. Shang and S. Li}$

Computers & Industrial Engineering 174 (2022) 108811
We take Philips Brilliance 16-slice CT scanner as an example to illustrate the three kinds of modules. As shown in Fig. 1, the X-ray tube, inventor, collimator, high voltage generator, and cooling system connect to PCB by cable, socket, and other physical structures. PCB is the foundational building block of medical equipment, the modules with a loose connection to PCB belong to type A, and the modules soldered on PCB belong to type B, which mainly refers to the chips in the data acquisition system. Other ancillary materials or accessories are ignored.

Assume all I SKU models involve $K_A$ models of type A modules, $K_B$ models of type B modules, and I models of PCB (the same as SKUs), and then the total modules are $M_I = I + K_A + K_B$. Let $m_i$ denote the module, where $m_i \in [1, I]$ refers to the PCB model, $m_j \in [I + 1, I + K_A]$ refers to the model of type A, and the rest identifies type B. Some SKUs may use more than one module of the same type, for example, the electrode tube, inventor, collimator, high voltage generator, and cooling system illustrate the three kinds of modules. As shown in Fig. 1, the X-ray tube, inventor, collimator, high voltage generator, and cooling system connect to PCB by cable, socket, and other physical structures.

The quality status of the product depends on its modules. For the collected in-transit products to be reused in remanufacturing, there are two quality statuses, one is known through IoT, that is, the quality of the in-transit product can be perceived by the sensors in IoT. Another is unknown, the quality of the collected product to be delivered is impossible to know. Therefore, the quality statuses of in-transit products are known in Cat.1 and Cat.2, while unknown in Cat.3.

Based on the quality statuses, we define a set $\mathcal{S} = \{u, r, o, p, w\}$ to describe the recovery options, where $u$ means re-manufacturing with a new PCB, $r$ means re-manufacturing with the current PCB, $o$ means complete disassembly, $p$ means partial disassembly, and $w$ means discard as wastes. If the PCB is not repairable, actions $r$ and $o$ are unavailable.

Since there is damage risk when the equipment is disassembled, we assume that the failure rate for disassembly is $\phi \in [0, 1]$, and then the failure probability of disassembly is $1 - (1 - \phi)^s$. The damaged PCB is classified as non-repairable (Li & Shang, 2021).

Let $N_i$ denote the demand of i-SKU product and $n_i$ denote the current yield, thus, the net demand of i-SKU product is $N_i - n_i$. If $N_i - n_i = 0$, the demand is satisfied.

The sequential constraints should be followed in orderly problems (Cat.2 and Cat.3) by two sequences: the delivery sequence and the event sequence. For the delivery sequence, let $S$ denote the total quantity of products, $S = \sum_{i=1}^{I} N_i$, and $s (1 \leq s \leq S)$ denote s-th product in the delivery sequence. Note that $s$ is SKU-independent and refers to a certain $s_r$.

For the event sequence, let $t (1 \leq t \leq T)$ denote the time point, and $T$ denote the deadline for shipment before which the remanufacturing must be completed. We assume that every product $s$ is delivered around $t$, whose recovery action will always be finished before the delivery of $s + 1$. The relationship between the two sequences can be remarked as $T \leftrightarrow S$, treating $S$ as an alternative tag of the deadline.

According to the sequential constraints and the practical significance, we assume that $N_i < S$, which means the urgent demand can be satisfied by combinatorial remanufacturing. If $N_i \geq S$, the disassembly of collected products cannot match the demand, far away from the focus of this paper.

Define $\Omega = r\cup o\cup w$ as the action space, where $y_m = \{y_m[1], y_m[2], \ldots, y_M\}$ and $y_m$ means extra procurement of one piece for module $m$. At each decision, there is only one action is chosen from $\Omega$. If one chosen action belongs to $r$, the current delivered product will be processed (i.e., remanufactured, disassembled, or dropped), leading to the next product, i.e., $s_j \rightarrow s_j + 1$ and $t \rightarrow t + 1$ in the proposed algorithms.

However, if action $y_m$ is chosen, the corresponding module should be procured as a new supply to satisfy remanufacturing, which is represented by neither $s_j$ nor $r$ increasing.

For i-SKU, since $s_j$ is the current product on processing, there remain $S_i - s_j$ products expected to be delivered as planned, while the demand for $N_i - n_i$ products is not met. The inventory of module $m_j$ is denoted as $k_m$. The above information is denoted as the raw state $Z_0$ (i.e., the raw data), as shown in (1).

$$ Z_0 = [S_1 - s_1, \ldots, S_I - s_j, N_1 - n_1, \ldots, N_I - n_I, k_1, \ldots, k_M]. $$

(1)

Let $a_i (a_i \in \Omega)$ denote the action at $t$, and $d_{i,j,a}$ denote her decision. If $a_i$ is taken on $s_j$, $d_{i,j,a} = 1$. The reward of $a_i$ is denoted as $r$, which is represented by $C_{i,j,a}$, $C_{i,j,a}$ is the remanufacturing cost of product $s$, and can be obtained by the costing method in Li and Shang (2021), where the product structure, quality status, operation steps, and potential costs related to disassembly risks are involved.

The manufacturer is to minimize her total remanufacturing cost, as shown in (2).

$$ \min_{a_i} C = \sum_{i=1}^{I} \sum_{s_j=1}^{S_i} \sum_{a_i \in \Omega} d_{i,j,a} C_{i,j,a}, t \leftarrow s_j - 1 \leq t \leq T. $$

(2)

In (2), $t \leftarrow s_j$ means $t$ and $s_j$ are one-to-one mapping, i.e., $t$ and $s_j$ are equivalent. $C_{i,j,a} \in \{c_1, c_2, \ldots, c_M\}$ is the unit procurement cost of $m_j$.

For all collected products, the size of the state space depends on the availability of recovery options. By defining the cardinal number function, card($\Omega$), the state space can be denoted as $\prod_{i=1}^{I} \text{card}(\Omega_i)$, where $\Omega_i$ is a subset of $\Omega$, i.e., the action space for product $s$. In industrial production, both $S$ and $\Omega$ are usually large, HCR problem is NP-hard, and it is difficult to traverse the state space completely.
At \( t \), the state transition matrix of \( Z_t \) by \( a_t \) is denoted as \( X_{a_t} \) and given in (3).

\[
X_{a_t} = \begin{bmatrix}
e_{u,i} & e_{u,n_t} & e_{u,n_j} \\
e_{v,i} & e_{v,n_t} & e_{v,n_j} \\
e_{w,i} & 0 & e_{w,n_t} \\
e_{x,i} & 0 & e_{x,n_j} \\
e_{y,i} & 0 & e_{y,n_j} \\
0 & 0 & 0
\end{bmatrix}
\]

(3)

where \( \mathbf{1}_{M \times M} \) is an identity matrix of order \( M \), \( \mathbf{0}_{M \times J} \) is a zero matrix of order \( M \times J \), \( \mathbf{0}_{J \times I} \) is a zero matrix of order \( I \times J \), and \( e_{u,i} = \begin{bmatrix} e_{u,i,1} & \cdots & e_{u,i,J} \end{bmatrix} \) and \( e_{u,j} = \begin{bmatrix} e_{u,n_t,j} & \cdots & e_{u,n_j,j} \end{bmatrix} \), and others by analogy. \( e_{n,i} \) denotes the offset to the corresponding \( Z_t \) by taking \( a_t \). The first five lines lead to the recovery option in \( t \), and the next \( M \) lines involve \( y_{n,t} \).

According to (3), action \( a_t \) will lead to the following transformations:

\[
S_j - s_j = S_j - s_j + e_{u,i}, \quad N_i - n_i = N_i - n_i + e_{u,j}, \quad 0 \leq e_{u,i}, e_{u,j} \leq 1;
\]

\[
m_j = e_{u,n_t,j}, \quad -(J - 1) \leq e_{u,n_j,j} \leq 0 \quad \text{up to} \quad J - 1 \text{ modules are replaced in addition to the PCB}.
\]

Similarly, we can derive the results of other actions. Then, the state transfer equation can be expressed by (4).

\[
Z_{t+1} = Z_t + X_{a_t} \cdot D_t, \quad \text{if } a_t \in \mathcal{A}, \quad \text{otherwise } Z_{t+1} = Z_t
\]

(4)

where \( D_t \) is the decision set, which is given in (5).

\[
D_t = \begin{bmatrix}
d_{z,i} & d_{z,j} & d_{z,i,o} & d_{z,i,p} & d_{z,i,w} & d_{z,i,j} & \cdots & d_{z,i,y} & d_{z,j} & \cdots & d_{z,j,y}
\end{bmatrix}^T
\]

(5)

where \( d_{z,t} \) is 1 if \( a_t \) is chosen, otherwise 0.

Since only one element is equal to 1 and the remaining elements are equal to 0 in (5), (4) is equivalent to selecting a row in (5) as the state offset.

For three categories of HCR problem, Cat.1 can be constructed by a mixed integer linear programming (MILP) model, the optimal decision can be obtained by branch and bound algorithm. Cat.2 and Cat.3 are the problems that the manufacturer encounters in remanufacturing decisions, while Cat.1 is just used for reference.

4. Equivalence problem and algorithms

4.1. Decision traps and assumption

In remanufacturing decision-making, there are three traps: the short-sightedness of decision-making, the implicitness of decision results, and the specificity of states.

(1) The short-sightedness of decision-making

The manufacturer usually makes emergency production decisions based on her experience and risk attitude. Driven by the pessimistic criterion (or Maximin criterion), she will complete production as early as possible and tend to remanufacture even if the quality of collected products is not good, while the manufacturer with optimistic criterion (or Maxmax criterion) prioritizes the cost-down, she prefers the disassembly in the early stage to provide spare parts for the later production stage.

The numerical examples in Section 5 verify the short-sightedness between short-term costs and long-term cumulative costs, where the lowest cost obtained by the empirical decision is always higher than the optimized result from the global perspective.

(2) The implication of decision results

The decision action affects not only the current and the accumulated production costs, but also the available actions in the later stage. For instance, when the remaining products in i-SKU before deadline \( S \) are equal to the current net demand, i.e., \( S_j - s_j = N_i - n_i \), we call this moment the critical point in time. In reality, the critical point in time may be realized in advance after taking more actions \( o, p, \) and \( w \), and then these three actions become unavailable. Such a relationship between the action and the availability is implicit and cannot be expressed explicitly by mathematical expressions. Therefore, it is difficult to solve such dynamic programming problems, especially in large-scale industrial production.

(3) The specificity of states

The raw state \( Z_t \) provides the availability of actions, however, the information in \( Z_t \) is valuable and limited to the specific problem at \( t \). For any other problem, \( Z_t \) makes no sense. That is to say, due to the specificity of states, if the quality status of in-transit products is unknown, the experience cannot be used, leaving a “black box” to the decision maker. The decision maker only makes the decision based on her preference, which is short-sighted.

4.2. Equivalent problem of maze exploration

To find the optimal remanufacturing decision, we first transform HCR problem into an equivalent maze exploration game, which is illustrated in Fig. 2. The maze explorer located in “start” looks for the exit of the maze, whose walking path is equivalent to the production decision. Since there are \( I \) SKUs, multiple explorers simultaneously explore their mazes. When exploring, the explorer’s state keeps changing, which represents inventories of modules, where higher than or less than 0 represents the inventory or the shortage.

In Fig. 2, the location (0, 1) marked “start” is the entry of the maze. It means that for \( i \)-SKU, the first collected product has just been delivered, and the current yield is 0. The maze explorer can choose to move right, lower right, or stay still. Note that, the explorer is not allowed to move up or left (these directions are nonsense in reality). The pink regions indicate the critical point in time (i.e., the boundary) where moving right is not permitted. This is because the quantity of in-transit products before \( T \) equals the net demand, and actions \( o, p \), and \( w \) will result in the urgent demand never being met.

(1) Move right: disassembly or discard, corresponding to actions \( o, p \), and \( w \). If these actions are taken, the current delivered product is consumed, i.e., \( s_j \leftarrow s_j + 1 \), and the yield of the remanufactured product \( n_i \) remains the same. Thus, the explorer moves from \((s_j, n_i)\) to \((s_j + 1, n_i)\). Specifically, if action \( o \) or \( r \) is selected, the inventory of the module, i.e., the explorer’s state, will increase, and the increment is determined by the quality status of the current collected product, the disassembly risk, and the selected recovery option \( o \) is fully disassembly and \( r \) is partial disassembly.

(2) Move lower right: remanufacturing, corresponding to actions \( u \) and \( r \). If these two actions are taken, not only the current collected product is consumed, the yield also increases, i.e., the explorer moves from \((s_j, n_i)\) to \((s_j, n_i + 1)\). For action \( u \), the yield increases by one, i.e., \( e_{n,i} = 1 \). For action \( r \), \( e_{n,i} = 1 \) if and only if the disassembly risk is 0 or the disassembly does not involve any type \( B \) module, otherwise \( 0 < e_{n,i} < 1 \). Moreover, with the consumption of the module, its inventory decreases, and the decrement depends on the quality status of the current delivered product and the availability of its PCB (the PCB will be replaced in action \( u \) while used in action \( r \).

(3) Stay still: module procurement, corresponding to action \( y_{n,j} \) (Note: This action is taken, the inventory of the module increases, whereas \( s_j \) and \( n_i \) remain the same. When the explorer finds an exit from the maze, it means that he obtains a feasible solution to HCR problem. Considering disassembly risk and expectation, the feasible solutions are continuous regions (green) in Fig. 2.

4.3. Exploration rule and knowledge dictionary

The dotted blue line in Fig. 2 draws the path through the maze, which also indicates a feasible solution for HCR problem. For the exploration, the movement can be decided freely, which depends on one's
subjective thinking pattern and risk preference. Accordingly, we propose four common strategies, each of which represents an exploration pattern and guides the remanufacturing decision.

Strategy I: Fast exploration strategy. This strategy does not allow the explorer to stay still, and the state can only be adjusted in the exit region, which means that action $y_w$ is unavailable until the demand is satisfied. At the same time, actions $u$, $r$, $o$, $p$, and $w$ are available unless the PCB cannot be repaired (actions $r$ and $o$ are disabled at this point) or the remaining in-transit collected products are insufficient (actions $o$, $p$, and $w$ are disabled at this point). In the region of the exit, actions $u$ and $r$ are no longer available, and the module will be replenished from disassembly (i.e., $o$ and $p$) or procurement ($y_w$).

Strategy II: Fast exploration strategy with instant replenishment. The explorer will check and adjust the state just after each move, and out-of-stock modules are immediately replenished according to action $y_w$. Then, the manufacturer proceeds to recovery decisions ($u$, $r$, $o$, $p$, and $w$), which is the same as the fast exploration strategy.

Strategy III: Safe exploration strategy. The explorer first judges the consequences of taking actions before each move and avoids any negative state. For remanufacturing, this means that the manufacturer acquires modules from one of the actions $o$, $p$, and $y_w$ when its inventories are insufficient to remanufacture ($u$ and $r$) with the current collected product.

Strategy IV: Safe exploration strategy with on-demand replenishment. The explorer evaluates the consequences of taking actions, whereas the replenishment is on-demand, i.e., the manufacturer first decides whether to remanufacture. If action $u$ or $r$ is selected, the manufacturer takes action $y_w$, before remanufacturing when the inventory is insufficient. The difference with the safe exploration strategy is the sequential relationship between remanufacturing action and module procurement.

Since the remanufacturing can be executed without checking the sufficiency of on-hand materials by strategy I and strategy II, the two strategies are not applicable for Cat.2 and Cat.3 problems. Besides, Cat.1 problem can be formulated as a MILP model, its optimal solution dominates solutions by these two strategies. Thus, in the numerical experiment, we will only use strategy III and strategy IV to provide sequential-compatible solutions.

Generally, the available paths depend on valid actions at $t$, i.e., $\text{card}(Q_t)$. Although the sequential constraints have reduced $\text{card}(Q_t)$, the exponential explosion of the action space requires performance tuning. In this regard, we propose the DeSoRVA rule to refine the action space, including two parts: reweighting of actions and descending selection.

(1) Reweighting of available actions

According to the epsilon-greedy rule, the explorer has two common policies to choose the action from $Q_t$: randomly or with the highest Q value. Since Q values of available actions are indistinguishable at the beginning, the explorer may waste much time in the case that there is no optimal solution, i.e., action $u$ is repeatedly taken. Thus, the alternatives should be reweighted to avoid the meaningless area.

We design a rational behavior pattern to reweight actions, where $o$ is preferred over $p$ for more disassembled modules, and $r$ is preferred over $u$ for a lower cost. Here "a preferred action" denotes a 10x possibility to be chosen in pairwise comparison, whereas minor alternatives are kept to balancing either breath or depth of searching.

(2) Descending selection

We take the rational behavior pattern to reweight actions during random selection. However, the selected action might still be invalid in some cases. For instance, action $u$ or $r$ is selected when the demand has already been satisfied; action $r$ or $o$ is selected when the PCB is non-repairable. Due to the discretization of continuous states to apply RL framework (Huang et al., 2020; Liu et al., 2020), similar states are merged, where an invalid action may be assigned the highest Q value.

Therefore, the validity of the action should be verified first. If the action is invalid, the alternative with the next highest Q value should be selected, which means turning to sub-optimal action when the first choice is illegal. Such a procedure is repeated until the action is valid.

The pseudocode of DeSoRVA rule is shown in Algorithm 1, where $\epsilon$ is the threshold for applying random selection, and $Q_t$ denotes the action space $Q$ at $t$, which is obtained by Algorithm 1.

Algorithm 1. The pseudocode of DeSoRVA rule is shown in Algorithm 2, where $\epsilon$ is the threshold for applying random selection, and $Q_t$ denotes the action space $Q$ at $t$, which is obtained by Algorithm 2.

Algorithm 2. The pseudocode of DeSoRVA rule is shown in Algorithm 3, where $\epsilon$ is the threshold for applying random selection, and $Q_t$ denotes the action space $Q$ at $t$, which is obtained by Algorithm 3.

Following the proposed DeSoRVA rule, we have Theorem 1.

**Theorem 1.** Following DeSoRVA rule, if there is no valid action, demand must have been met.

**Proof.** Assume that there is no action available for the first time, but the demand has not been met for product $s_j$, i.e., $N_i - n_i > 0$. Let the last and the current collected products be $s_{j'}$ and $s_{j''}$ ($1 \leq j < j''$), respectively. For HCR problem, action $u$ is unavailable if and only if the collected product is delivered after the deadline $T$, then we have $S_j - s_{j''} < 0$ and $S_j - s_{j'} = 0$. Since $N_i - n_i > 0$, there must be another product $s_j$ ($s_j < s_{j''}$) that satisfies $S_j - s_j = N_i - n_i > 0$.

Following DeSoRVA rule, when $S_j - s_j = N_i - n_i > 0$, either action $u$ or $r$ must be selected to reduce $S_j - s_j$ and $N_i - n_i$ to 0. Thus, $N_i - n_i > 0$ is false when $S_j - s_{j''} < 0$, which contradicts the assumption. \(\square\)
To generalize the specific state (i.e., Z_i) of HCR problem, we propose a knowledge dictionary indexed by Esperate. Esperate is a compound word “Espertate” of “Esperanto” and “State” defined in this paper, which represents the universal state of HCR problem and is independent of the raw state. By transforming a specific state into an Esperate state, the manufacturer’s experience and knowledge can be accumulated and refined to sophisticated knowledge. Esperate at t is denoted as E_t, which is shown in (7).

\[ E_t = [E_{i1}, E_{i2}, E_{i3}, E_{i4}, E_{i5}] \]

where \( E_{i1} \) is the quality status of the current product, which is measured by (8); \( E_{i2} \) is the remaining time and reflects the urgency degree; \( E_{i3} = 1 - s_i/S_i \); \( E_{i4} \) is the production progress of the overall production target; \( E_{i5} = \frac{1}{T} \sum_{n_i} (n_i/N_i) \); \( E_{i6} \) is the production progress of the current SKU; \( E_{i7} = n_i/N_i \); \( E_{i5} \) is the relative quantity of in-transit collected products, \( E_{i5} = (s_i - n_i)/(S_i - N_i) \).

\[ E_{i1}(s_i) = \begin{cases} \min(C_{i,a_i}C_{i,a_i}) - \min C_{i,a_i} \\ \max C_{i,a_i} - \min C_{i,a_i} \end{cases} \]

where \( a_i \in \{u, r, \} \); \( s_i \in \{1, \ldots, S_i\} \); and \( C_{i,a_i} \) is set to infinity for an unrepairable PCB.

4.4. Reinforcement learning frameworks

To keep away from the three traps in the remanufacturing decision, we utilize Q-learning and DDQN to develop RL frameworks for HCR problem. Since the objective of HCR problem is to minimize the total cost, we take the cost incurred after taking each recovery decision as

To conclude, the proposed HCR framework with Esperate knowledge dictionary and combined RL algorithms is effective in remanufacturing with a high rate of recovery, reasonable decision-making, and high recovery cost efficiency in the real-world remanufacturing systems.
the reward to stepwisely update Q values. If an action is invalid, its Q value is assigned as negative infinity to prevent the action from being selected.

(1) Q-learning

Based on an initial Q table, the explorer moves according to the DeSoRVA rule and then updates Q value according to the reward of the action derived by the Bellman Equation (see also (9)). The Q-learning framework is shown in Fig. 3, where the states (\(Z_i\) and \(E_i\)) and corresponding Q values of all actions are stored in the Q table.

By searching the current state in the Q table, the explorer finds his action with the highest Q value and then selects his best action by the DeSoRVA rule if it exists; otherwise, the state will be added to the Q table with all-zero Q values and the random selection is performed.

After the action is selected, if \(a_i \in r\), the reward \(r_t\) will be used to update Q value by (9), which is a Bellman Equation.

\[
Q(Z_t, a_t) \leftarrow Q(Z_t, a_t) + \alpha [r_t + \gamma \max\{Q_{t+1}(Z_t, a)\} - Q(Z_t, a_t)].
\]

In (9), \(\alpha \in (0, 1)\) and \(\gamma \in (0, 1)\) denote the learning rate and the discount factor respectively, \(Q(Z_t, a_t)\) is Q value of \(a_t\) for \(Z_t\), and \(Q_{t+1}\) is given in (6). In addition, \(Z_t\) in (6) and (9) can be replaced by \(E_t\) when Esperate is enabled, and (9) will be skipped if taking action \(y_{mj}\) since \(y_{mj}\) is independent on \(t\).

The algorithm for the Q-learning framework is described in Algorithm 3. The initialized state \(Z_0\) consists of three parts, the first \(\{S_0, ..., S_T\}\) is for \(S_t - s_i\) of \(I\) items, the middle \(\{0, ..., 0\}\) is for \(N_i - n_i\) of \(I\) items, and the last \(\{0, ..., 0\}\) is for \(k_{mj}\) of \(M_i\) modules.

(2) DDQN framework

When the scale of HCR problem is large, it is hard to maintain the Q table due to dimensional explosion in the Q-learning framework. Then, we propose a DDQN framework with two same-structure deep neural networks (DNNs), i.e., the current Q Network and target Q Network, to implement the transformations among states, actions, and Q values. With the DDQN framework, the explorer also follows DeSoRVA rule. Unlike the Q-learning framework, Q values in the DDQN framework are obtained through the neural network, and the two DNNs are updated asynchronously.

The DDQN framework is shown in Fig. 4.

In Fig. 4, \(u^R\) and \(u^D\) denote weights of the two networks, respectively. \(Q^R(E_t, a_t, u^R)\) is the current Q value of \(a_t\) from the output of current Q network given input \(E_t\). If \(a_t \in r\), \([E_t, a_t, E_{t+1}, r_t, \text{fin}]\) will be stored in an experience pool named replay bank, and then the target Q value is calculated by picking random \(M\) samples. The target Q values are denoted as \(Y_m\) (\(m = 1, ..., M\)) in (10).

\[
Y_m = r_t + \gamma Q^D(E_{t+1}, \arg\max_{a_{t+1}} Q_{t+1}(a_{t+1}, u^D)), \quad (10)
\]

Algorithm 3 Q-learning framework for HCR problem.

1: initialize Q table
2: for episode in range (1, EPISODE) do
3: initialize \(t = 1\), \(Z_1 = \{S_0, ..., S_T, 0, ..., 0, 0, ..., 0\}\), \(E_1 = \{0, 0, 0, 0, 0\}\), \(Q_1 = \{0, 0, 0, 0, 0\}\), fin = False  //fin denotes ending mark
4: repeat
5: calculate \(E_t\) by (7)
6: if \(Q(Z_t, a_t)\) or \(Q(E_t, a_t)\) does not exist then
7: append Q table with \(Q_t\)
8: \(\text{epsilon} \leftarrow 1.1\)  //randomly take an action for a new state
9: else
10: generate a random number \(\text{epsilon} \in (0, 1)\)
11: end if
12: \(a_t \leftarrow \text{Get\_Current\_Action}(Z_t, Q_t, \text{epsilon}, \text{strategy})\)
13: if \(Q_t = \emptyset\) then
14: \(\text{fin} \leftarrow \text{True}\)
15: else if \(a_t = y_{mj}\) then  //module procurement does not change \(t\)
16: obtain \(Z_t\) by (4) and record corresponding cost \(C_{i, a_t}\)
17: \(Z_t \leftarrow Z_t\)
18: else
19: obtain \(Z_{t+1}\) by (4) and record corresponding cost \(C_{i, a_t}\)
20: calculate \(r_t\) and \(E_{t+1}\) from \(C_{i, a_t}\) and \(Z_{t+1}\), respectively
21: update Q value by (9)
22: \(Z_t \leftarrow Z_{t+1}\) and \(E_t \leftarrow E_{t+1}\)
23: \(t \leftarrow t + 1\)
24: end if
25: until fin is True
26: for each \(i\) and \(j\) do
27: proceed remaining products and jobs
28: calculate total cost \(C_T\) at \(T\)
29: end for
30: end for

where \(Q_{t+1} = \{Q^R(E_{t+1}, a_{t+1}, u^R) | a_{t+1} \in r\} \), \(a'_{t+1}\) is the action with the highest Q value in \(Q_{t+1}\).
Following Fig. 4, \( Y_n \) is used to update the current Q network by adjusting \( u^R \) to minimize the mean squared error loss (mse_loss) in (11) (Kingma & Ba, 2015).

\[
\min_{u^R} \text{mse loss} = \frac{1}{M} \sum_{m=1}^{M} (Y_m - Q^R(E_m, a_m, u^R))^2.
\] (11)

The objective in (11) can be achieved by Adam (adaptive moment estimation) optimizer, and the corresponding \( u^R \) is used to periodically replace \( u^G \). The asynchronous update can enhance the stability of neural networks by alleviating the divergence of the target Q value when the current Q value rapidly increases (Mnih et al., 2015).

The proposed DDQN framework is realized by the following Algorithm 4, where the initialized state \( Z_t \) is the same as that in Algorithm 3.

5. Numerical studies

5.1. Environment and scheme configuration

In the numerical studies, we continue to take CT scanners as an example to perform remanufacturing in the pandemic. From the investigation of several local suppliers and online used medical equipment stores (e.g., PrizMed Imaging, LBN Medical, Radiology Oncology Systems), the used medical equipment is individually processed (see Fig. 5(a)) and showcased (see Fig. 5(b)), which verifies our assumption. The price considerably depends on technical specifications and individual conditions, while the brand and model do not matter much. For example, a 16-slice CT scanner is around $82,000 to $86,000, and a 64-slice version will cost $125,000 to $140,000. Therefore, we can use the sum of discrete numbers to represent the random but grouped material costs of the medical equipment.

According to the above demonstration and combined with the illustration in Fig. 1, let \( I = 3 \), \( K_A = K_B = 5 \), \( \varphi = 0.1 \), \( \theta = 0.6 \), and \( S = 30, 100, 300, 1000 \), respectively. The costs of actions are set as follows: \( C_{i,a} \in (4.0,6.0) \), \( C_{i,s} \in (0.5,6.5) \), \( C_{i,o} \in (0.25,5.0) \), \( C_{i,p} \in (0.15,2.0) \), \( C_{s,o} \in (0.15,6.6) \), and \( C_{s,a} \in (1.3,5) \) for type \( A \), type \( B \) and PCB, respectively. The operating costs are configured lower than material costs, which is consistent with the cost composition of medical equipment. The quality statuses are randomly generated based on a given quality level, and the data is divided into the training set and testing set. The numeric format of \( \text{Espertate} \) is set to 1–2 by default, which means the decimal place of \( E_{ij} \) is set to 1 and others are 2.

Following Hasselt et al. (2016), the hyperparameters are set as \( a = 0.001 \), \( \gamma = 0.9 \), \( M = 32 \) (batch size), \( P = 100 \) (update period), and both DNNs with 10 layers. \( \epsilon \) is uniformly decreased as shown in (12), which indicates that the exploration begins with a random selection to use knowledge.

\[
\epsilon_{t+1} = \epsilon_t - \frac{\epsilon_{\text{init}} - \epsilon_{\text{fin}}}{\text{EPISODE}}.
\] (12)

where \( \epsilon_{\text{init}} = 1.0 \), \( \epsilon_{\text{fin}} = 0.1 \), \( \text{EPISODE} = 10000 \).

For the four remanufacturing strategies, considering the proposed algorithms and knowledge dictionaries, we have the following seven solution schemes to HCR problem.

PS (Precise Solution). The optimal solution is obtained by branch and bound algorithm, which is applicable to Cat.1 problem since it is treated as a MILP problem.

RS (Random Selection). One of the available actions is randomly selected at each decision, which is capable of the Cat.2 problem. From Fig. 6(a), it is found that the remanufacturing cost of scheme RS is within a certain range and satisfies the normal distribution. In numerical studies, this scheme is used to be a benchmark.

RQ (Q-learning with Raw state). The Q-learning framework is implemented with the raw state to index the knowledge dictionary.

EQ (Q-learning with \( \text{Espertate} \)) and ED (DDQN with \( \text{Espertate} \)). The Q-learning and DDQN frameworks are implemented with \( \text{Espertate} \) to index the knowledge dictionary.

OQ (One-time decision by Q-learning). Q value is updated to the \( \text{Espertate}-\)indexed Q with \( \text{Espertate} \) (action, score), the corresponding action is taken according to the Q table by matching \( \text{Espertate} \) and DeSoRVA rule.

OD (One-time decision by DDQN). The Q network is updated with the input of \( \text{Espertate} \), and the action is taken according to the output of the current Q network (i.e., the updated Q network) and DeSoRVA rule.

The following Table 1 describes the mapping of HCR problems, remanufacturing strategies, and solution schemes. Note that Schemes OQ and OD get knowledge iteratively trained by schemes EQ and ED, respectively. The training process is exhibited in Fig. 15.

The experiments are conducted in the computing environment with Core i7 2.6 GHz CPU, 32 GB RAM, GTX 1060 6 GB GPU, Windows 10 x64, Python 3.9, Gurobi 9.1.1 x64, Tensorflow 2.5.1, and cuDNN 8.1.1.

From extensive numerical experiments, it is found that remanufacturing costs of schemes RQ, EQ, and ED are close to scheme RS in the first 1000 episodes since \( \epsilon \) is close to 1.0, and an experimental example in Fig. 6(a) shows this finding. However, they are gradually
Fig. 5. The individualized treatment (acquired from lbnmedical.com and oncologysystems.com).

Fig. 6. The histogram of remanufacturing costs.

Table 1
Mappings of HCR problems, remanufacturing strategies, and solution schemes.

| HCR problems | Algorithm or framework                      | Solution schemes |
|--------------|---------------------------------------------|------------------|
| Cat.1        | Branch and Cut                              | PS               |
| Cat.2        | Random selection, Q-learning and DDQN      | RS, RQ, EQ, and ED |
| Cat.3        | Random selection, Q-learning and DDQN      | RS, OQ and OD    |

reduced with the increase of episode and the decrease of epsilon during the next episodes, Fig. 6(b) shows an example. Obviously, remanufacturing costs of schemes EQ and ED are lower than those of scheme RS, which shows that our proposed algorithms are feasible to update the decision-making knowledge and find a better solution.

We then take the expected remanufacturing cost obtained by the last 1000 episodes as the benchmark.

5.2. Solving Cat.1 and Cat.2 problems

Since Cat.1 problem is a MILP problem, it can be solved by the optimization solver with Scheme PS. We use Gurobi Optimizer in numerical studies, which is one of the fastest mathematical programming solvers for MILP problems.

For Cat.2 problem, a glimpse of the numerical experiments of $S = 300$ is shown in Fig. 7. The results indicate that, schemes EQ and ED can learn to save remanufacturing costs, and strategy III dominates strategy IV with lower cost and less fluctuation for both schemes RS and EQ.

From Fig. 7, it is found that the upper bound of remanufacturing costs by strategy III is lower than that by strategy IV, while the lower bound is at the same level. In other words, strategy III generates a better decision for the manufacturer, which is counter-intuitive because strategy IV is more rigorous and effective from the definitions of four strategies. The reason is that strategy III provides a better start to avoid low-value paths in maze exploration and finally contributes to the lower cost. Fig. 8 shows distributions of remanufacturing costs with 10000 episodes for schemes EQ and ED, respectively.

Consequently, only strategy III is involved in next experiments. To examine the impact of the problem scale on the remanufacturing cost, we set $S = 30, 100, 300, 1000$, respectively. The results of schemes RS, RQ, EQ, and ED are shown in Fig. 9. Note that, in Fig. 9(d), there is no result for scheme RQ because the memory usage exceeds the hardware limitation when $S \geq 1000$.

From Fig. 9, there is no significant discrepancy among schemes RQ, EQ, ED, and RS when $S = 30$; when $S = 100$, the cost-down (the saving of remanufacturing cost brought by schemes RQ, EQ, or ED relative to scheme RS) can be observed in the next episodes of both schemes EQ and ED, which indicates meaningful knowledge guidance; when $S \geq 300$, both schemes EQ and ED have more significant cost-down, and the latter performs better.

However, as shown in Fig. 9(d), the remanufacturing cost of either scheme EQ or ED may increase again with the increase of episodes when $S \geq 1000$. The reason is that the problem scale is too big when $S \geq 1000$, and the size of the decision-making space far exceeds the capacity of the knowledge dictionary. In this case, by the merging of raw states, the one-to-one relationship between state and decision becomes many-to-one, which causes the obtained action to no longer be optimal and the remanufacturing cost increases further. The trouble is
to be settled by designing digits of the element in \textit{Espertate}, the detailed discussion can be found in Observation 2.

According to extensive experiments, we have the following Observation 1.

**Observation 1.** \textit{Espertate} contributes to the cost-down, and DDQN shows more advantages in handling larger scale problems.

Regardless of the problem scale, the remanufacturing cost of scheme RQ has never broken the lower bound of scheme RS, which indicates that the knowledge dictionary indexed by raw state cannot provide guidance. In contrast, by \textit{Espertate}, the cost-down is more significant, especially when the problem scale is large. We use a box plot to depict the distribution of remanufacturing costs, as shown in Fig. 10. To evaluate their performance, we draw scheme PS in Fig. 10, but scheme PS is only applicable for Cat.1 problem, the precise solution cannot be obtained by it for Cat.2 problem on large scale. In Fig. 10, the center of each horizontal line in the box plot denotes the expected remanufacturing cost, and the upper and lower denote 1st and 99th percentile remanufacturing cost, respectively.

By comparing schemes EQ and ED, when the problem scale is large and the discretization configuration is appropriate, the cost-down by DDQN is significantly greater than that by Q-learning. When $S \geq 100$, the remanufacturing cost of scheme ED has $130\% \sim 135\%$ to scheme PS, achieving $10\% \sim 15\%$ cost-down than scheme EQ and more than $20\%$ cost-down than scheme RS (about $135\%$ of scheme ED vs. about $155\%$ of scheme RS). This means that, when the scale of the decision problem is large, scheme ED is the most suitable for Cat.2 problem, which provides significant cost-down.

We use knowledge usage to further examine the impact of \textit{Espertate} on remanufacturing, which is defined by the proportion of successful inquiries between the state and the knowledge dictionary. Fig. 11 shows an experimental example, in which scheme ED is not involved since DDQN does not have Q table indexes.

From Fig. 11, we have the following findings. If the knowledge dictionary is indexed by raw state, when $S = 100$, the knowledge usage approaches $50\%$ only in the late episodes; when $S = 300$, the knowledge usage becomes even worse. The low usage of scheme RQ (see also Fig. 11(a)) indicates insufficient learning and incomplete guidance, which also explains why the cost range of scheme RQ almost coincides with scheme RS in Fig. 9.

In contrast, if \textit{Espertate} is applied, the knowledge usage is always at a high level, thereby leading to cost reduction. However, when $S = 30,$
both two knowledge dictionaries show a high usage level but no notable cost-down (see also Fig. 9(a)). This means that the positive correlation between the knowledge usage and the cost-down is not applicable when the scale of the problem is small, which is because the optimization space is limited.

To examine the impact of the discretization on remanufacturing, we let $S = 1000$ in experiments. The results are shown in Fig. 12, and we have the following observation.

**Observation 2.** Discretization configuration affects cost-down and stability.

Discretization configuration is defined by “Scheme-X-Y”, which reflects the discretization level of the continuous state space. “Scheme-X-Y” denotes digit(s) of the element in Espertate (i.e., $E_{t1}, \ldots, E_{t5}$) for the chosen scheme, in which “X” is for $E_{t4}$ and “Y” is for other elements. For example, “EQ-1-2” means that scheme EQ is chosen, one digit is used for $E_{t4}$ and others use two digits; “EQ-1-1” means all elements in Espertate use one digit.

From Fig. 12(a), both EQ-1-1 and EQ-2-2 are worse than EQ-1-2 and EQ-2-3; ED-2-2 is also worse than other configurations from Fig. 12(b). Such phenomena indicate that, keeping a one-digit gap between $E_{t4}$ and other elements contributes to a better decision on whether Q-learning
or DDQN is used. In practice, the one-digit gap can distinguish similar quality statuses by production progress.

EQ-1-2 & EQ-2-3 in Fig. 12(a) and ED-1-2 & ED-2-3 in Fig. 12(b) indicate that the remanufacturing cost increases in the latest episodes, which is due to similar states are merged when the discretization level is too low, and optimal actions are unexpectedly substituted with other non-optimal actions, as discussed in Fig. 9(d).

However, a higher discretization level does not necessarily lead to better performance. Comparing ED-2-3 and ED-3-4 in Fig. 12(b), the latter is slightly worse than the former, because the value of the knowledge is limited when the state is too scattered. To further explain the phenomenon, we depict the effect of discretization level on the size of the knowledge dictionary (i.e., the number of entries in the dictionary) in Fig. 13.

By Espertate, the size of the knowledge dictionary is significantly reduced than that of the raw state, especially for the large-scale problem. This means that the knowledge utilization is improved by Espertate. Moreover, the discretization configuration for Espertate also affects the size of the knowledge dictionary. The size of EQ-2-3 is about 10 times larger than EQ-1-2, but the cost-down is not that much, which indicates the redundancy of indexes.

Moreover, it can be concluded that an oversized Q table occupies more time and space, and the operating efficiency is reduced in Q-learning. Although DDQN does not involve the Q table, according to the curve of the remanufacturing cost, the discrimination performance of the neural network is also limited when the input is more complicated, which is caused by the excessively high discretization level.

5.3. Solving Cat.3 problem

For Cat.3 problem, the training set is used to generate sophisticated knowledge, and the performance is observed by the experimental set. We compare the results among schemes RS, ED, OQ, and OD, where scheme ED is used to be a benchmark for the performance of schemes OQ and OD. Similar to Section 5.2, one scheme is meaningful if its remanufacturing cost breaks the lower bound of the remanufacturing cost achieved by scheme RS.

We randomly select four datasets from experimental sets for Cat.3 problem and draw results in Fig. 14.

In Fig. 14, “e”. refers to the expectations to the remanufacturing cost. Note that the remanufacturing costs of schemes OQ and OD remain constant, because the input (i.e., Espertate transformed from the raw state) is fixed, thus the output as the decision sequence for Cat.3 problem is fixed. The remanufacturing cost of scheme OD-3-4 ranges between schemes RS and ED regardless of the dataset. Specifically, it provides 6% ~ 8% cost-down than scheme RS, 4% ~ 7% higher than scheme ED, and is close to scheme ED in general.

In contrast, the cost-down by scheme OD-2-3 than scheme RS is narrowed to 2% ~ 5%, which is the opposite of the Cat.2 problem. Fig. 15 shows the training processes of schemes RS, EQ, and ED to iterate the knowledge, where training datasets are sequentially executed.

In Fig. 15, “c”. denotes continuous iterations on the knowledge. Note that scheme EQ-3-4 again exceeds the hardware limitation in sample 7. It is found that schemes ED-2-3 and ED-3-4 show similar cost-down for the first four datasets, and both dominate scheme EQ-3-4. However, with the iterations on the knowledge, the performance of scheme ED-2-3 gradually degrades during samples 5 ~ 10, and eventually becomes even worse than scheme RS in sample 11. Meanwhile, scheme ED-3-4 performs well from the beginning to the end, and the trained knowledge can handle Cat.3 problem (see also Fig. 14).

Concerning the volatility of the pandemic, the scale of confirmed cases can be either small or big among cyclic waves, where the manufacturer may collect variable numbers of abandoned medical equipment, bringing new Cat.3 problems. To reflect the phenomenon, the quantity of delivered products in the next case is adjusted to be different from the training stage. If the knowledge really involves the essence of HCR, the proposed algorithm should be applicable to the new problem for a changed scale.
The remanufacturing cost of scheme OD-3-4 keeps lower than scheme RS, but the gap is smaller with decreasing in the problem scale. The reason is that the remanufacturing cost decreases as the problem scale become smaller, relatively enhancing the impact of each decision. Any non-optimal decision can enlarge the total remanufacturing cost.

Sample 2 ($S = 100$) in Fig. 16 shows that scheme OD-3-4 (Cat.3 problem) is even superior to scheme ED-2-3 (Cat.2 problem). This counter-intuitive phenomenon indicates the adaptability of our algorithm. The knowledge trained from a larger scale (e.g., $S = 300$) can better distinguish the state for a smaller problem (e.g., $S = 100$), and the shortcomings of the DDQN framework in solving the small-scale problem can be effectively overcome.

6. Conclusion

In considering the sequential delivery and individual treatment, we propose an HCR strategy for remanufacturing the medical equipment to satisfy urgent demand in the COVID-19 pandemic.

Due to the limited and uncertain information, there are decision traps including short-sightedness of the decision, the implicitness of the result, and specificity of the state, hiding optimal decisions in remanufacturing. To visualize the problem structure and recovery options, we first transform the original problem into a maze exploration game, where a path to the exit evaluated by stepwise rewards corresponds to a valid solution evaluated by option-wise costs. Such an equivalent representation enables RL-based frameworks to try and learn optimal remanufacturing decisions in the preferred area and direction. The core of the frameworks consists of the DeSoRVA rule and Espertate-indexed...
knowledge dictionary, where the DeSoRVA rule guides the manufacturer to take better actions, and Espertate ideally represents the state space of the decision-making process and indexes the sophisticated knowledge.

From the numerical studies organized according to the industrial practice, it is found that the proposed algorithms with HCR strategy can learn to choose proper recovery options and balance the current optimality with the overall goal. Specifically, the manufacturer has 10% ∼ 20% and 6% ∼ 8% cost-down in Cat.2 and Cat.3 problems, respectively, where the accelerated convergence comes from the design of the DeSoRVA rule and Espertate. The two techniques are our important innovations and contributions, by which the utilization of knowledge (in Q-learning) and the discrimination accuracy of the neural networks (in DDQN) are improved. In comparison, the DDQN-based algorithm performs better for large-scale problems, where the neural networks realize the scale-independently transformation from the state to Q values. Even if the quality status and the problem scale in production are different from the training stage, the DDQN-based algorithm is still capable of handling, showing its robustness in the pandemic. Moreover, an appropriate discretization configuration of Espertate contributes to a significant performance improvement.

Since the failure rate of certain vulnerable modules increases due to the lifecycle, the changing quality statuses of collected products can be considered in future research. It is also possible to allow disordered sequences in remanufacturing, where multiple collected products are
Algorithm 4 DDQN framework for HCR problem.

1: initialize current Q network, target Q network, and replay bank
2: for episode in range (1, EPISODE) do
3: initialize \( t = 1 \), \( Z_t = \{S_0, \ldots, S_t, 0, \ldots, 0, 0, \ldots, 0\} \), \( E_t = \{0, 0, 0, 0, 0\} \), \( Q_t = \{0, 0, 0, 0, 0\} \), \( \text{fin} = \text{False} \)
4: repeat
5: calculate \( E_t \) by (7)
6: generate random number \( \epsilon_{sp}, \epsilon_{sp} \in (0, 1) \)
7: \( a_t \leftarrow \text{Get Current Action}(Z_t, Q^R_t, \epsilon_{sp}, \text{strategy}) \)
8: if \( Q_t = \emptyset \) then
9: \( \text{fin} \leftarrow \text{True} \)
10: else if \( a_t = y_m \) then //module procurement does not change \( t \)
11: obtain \( Z'_t \) by (4) and record corresponding cost \( C_{t,a_t} \)
12: \( Z_t \leftarrow Z'_t \)
13: continue //skip updating Q table
14: else
15: obtain \( Z_{t+1} \) by (4) and record corresponding cost \( C_{t,a_t} \)
16: obtain \( r_t \) and \( E_{t+1} \) from \( C_{t,a_t} \) and \( Z_{t+1} \), respectively
17: add \( \{E_t, a_t, E_{t+1}, r_t, \text{fin}\} \) to replay bank
18: pick \( M \) samples from replay bank
19: for \( m \) in range \((1, M)\) do
20: calculate \( Y_m \) by (10)
21: end for
22: update \( w^R \) by (11)
23: \( t \leftarrow t + 1 \), \( Z_t \leftarrow Z_{t+1} \) and \( E_t \leftarrow E_{t+1} \)
24: end if
25: until \( \text{fin} \) is \text{True}
26: if episode \% \( P = 0 \) then //periodically update the target Q network
27: \( \mu^G \leftarrow \mu^R \)
28: end if
29: for each \( i \) and \( j \) do
30: proceed remaining products and jobs and calculate total cost \( C_T \) at \( T \)
31: end for
32: end for

processed at the same time. These extensions will bring challenges to knowledge presentation and exploration strategies.

CRediT authorship contribution statement

You Shang: Formal analysis, Programming validation, Writing – original draft. Sijie Li: Conceptualization, Methodology, Supervision, Writing – review & editing, Funding acquisition.

Data availability

Data will be made available on request.

References

Dittirch, M.-A., & Fohlenmeister, S. (2020). Cooperative multi-agent system for production control using reinforcement learning. CIRP Annals, 69(1), 389-392.
Hasselt, H. V., Guez, A., & Silver, D. (2016). Deep Q-learning with double Q-learning. In Proceedings of the AAAI conference on artificial intelligence (pp. 2094-20100). AAAI Press.
Hennig, M., Graffinger, M., Hofmann, R., Gerhard, D., Dumas, S., & Rosenberger, P. (2021). Introduction of a time series machine learning methodology for the application in a production system. Advanced Engineering Informatics, 47, Article 101197.
Huang, J., Chang, Q., & Arinez, J. (2020). Deep reinforcement learning based preventive maintenance policy for serial production lines. Expert Systems with Applications, 160, Article 113701.
Kardos, C., Laframme, C., Gallina, V., & Sihn, W. (2020). Dynamic scheduling in a job-shop production system with reinforcement learning. Procedia CIRP, 97, 104-109.
Khader, N., & Voon, S. W. (2021). Adaptive optimal control of stencil printing process using reinforcement learning. Robotics and Computer-Integrated Manufacturing, 71, Article 102132.
Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In Proceedings of the international conference on learning representations (pp. 1-11). ICLR Press.
Li, Y., & Ou, J. (2020). Optimal ordering policy for complementary components with partial backordering and emergency replenishment under spectral risk measure. European Journal of Operational Research, 288(2), 538-549.
Li, S., & Shang, Y. (2021). A quality status encoding scheme for PCB-based products in IoT-enabled remanufacturing. Frontiers of Computer Science, 15(5), Article 155615.
Liang, H., Wen, X., Liu, Y., Zhang, H., Zhang, L., & Wang, L. (2021). Logistics-involved QoS-aware service composition in cloud manufacturing with deep reinforcement learning. Robotics and Computer-Integrated Manufacturing, 67, Article 101991.
Liu, Y., Chen, Y., & Jiang, T. (2020). Dynamic selective maintenance optimization for multi-state systems over a finite horizon: A deep reinforcement learning approach. European Journal of Operational Research, 283(1), 166-181.
Luo, S., Zhang, L., & Fan, Y. (2021). Dynamic multi-objective scheduling for flexible job shop by deep reinforcement learning. Computers & Industrial Engineering, 159, Article 107489.
Mehrotra, S., Rahimian, H., Barah, M., Luo, F., & Schantz, E. (2020). A model of supply-chain decisions for resource sharing with an application to ventilator allocation to combat COVID-19. Naval Research Logistics, 67(5), 303-320.
Minh, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. Nature, 518(7540), 529-533.
Mutha, A., Bansal, S., & Guide, V. D. R. (2016). Managing demand uncertainty through core acquisition in reinforcement learning. Production and Operations Management, 25(8), 1440-1464.
Qi, Q., Tao, F., Cheng, Y., Cheng, J., & Nee, A. Y. C. (2021). New IT driven rapid manufacturing for emergency response. Journal of Manufacturing Systems, 60, 928-935.
Shin, K., & Lee, T. (2020). Emergency medical service resource allocation in a mass casualty incident by integrating patient prioritization and hospital selection problems. IIE Transactions, 52(10), 1141-1155.
Vanhooydonck, A., Van Goethem, S., Van Loon, J., Vandormael, R., Vleugels, J., Peeters, T., Smets, S., Stokhuijzen, D., Van Camp, M., Vleerbeek, L., Verlinden, L., Verwulgen, S., & Watts, R. (2021). Case study into the successful emergency production and certification of a filtering facepiece respirator for Belgian hospitals during the COVID-19 pandemic. Journal of Manufacturing Systems, 60, 876-892.
Wang, L., Hu, X., Wang, Y., Xu, S., Ma, S., Yang, K., Liu, Z., & Wang, W. (2021). Dynamic job-shop scheduling in smart manufacturing using deep reinforcement learning. Computer Networks, 190, Article 107969.
Wang, Q., & Nie, X. (2019). A stochastic programming model for emergency supply chain planning considering traffic congestion. IIE Transactions, 51(8), 910-920.
Wu, C., Liao, M., Karatas, M., Chen, S., & Zheng, Y. (2020). Real-time neural network scheduling of emergency medical mask production during COVID-19. Applied Soft Computing, 97, Article 106790.
Zou, H., Tao, J., Elayed, S. K., Elattar, E. E., Almalaq, A., & Mohamed, M. A. (2021). Stochastic multi-carrier energy management in the smart grid using reinforcement learning and unscented transform. International Journal of Electrical Power & Energy Systems, 130, Article 106988.