Maintenance Decision-Making Optimization for Quality Degrading Production Systems through Deep Reinforcement Learning

Chiye Ma[a], Wei Wang[ab*]
[a]Department of Mechanical Engineering, City University of Hong Kong, Kowloon, Hong Kong, China
[b]Shenzhen Research Institute of City University of Hong Kong, Shenzhen 518057, China

Abstract: Considering that serial systems based on small-lot production operate in a customized mode, we investigate the maintenance decision-making problem of quality degrading machines in these serial systems. To obtain the optimal maintenance policies, intelligent optimization algorithms are only applicable for small systems which have a few degrading machines. However, these algorithms cannot be directly implemented as the search space of a large system with many degrading machines exponentially increases. Therefore, in this work, value-based deep reinforcement learning algorithm QMIX, is adopted to obtain real-time optimal maintenance policies by regarding each degrading machine as an agent. Specifically, system states are defined, and three types of actions are introduced to maintain degrading machines. A reward function is established based on the maintenance action cost along with the system states. Our conducted simulation study demonstrates that QMIX method is valid to optimize maintenance policies for serial production systems with degrading machines. Taking completion time as an indicator, a contrast experiment indicates that maintenance policies learned by QMIX can shorten the completion time.

Keywords: Stochastic Modeling, Degrading Machines, Maintenance Decision-Making, Multi-Agent Systems

1. INTRODUCTION

Product quality problems are the focus of production systems engineering. To ensure that production systems operate efficiently, sustainably and steadily, it is of critical significance to maintain degrading machines appropriately [1]. Degrading machines need to be maintained in a real-time manner during machining processes. To obtain optimal maintenance policies, maintenance cost, time and completion time are the main indicators which need to be optimized [2]. Previous works are resorted to intelligent optimization algorithms for optimizing maintenance policies [3]. However, because there are many degrading machines in practical production systems and the search space of maintenance actions is large, intelligent optimization algorithms are not applicable anymore. Therefore, to extend production systems with limited number of degrading machines to general ones, it is important to study optimal maintenance policies through deep multi-agent reinforcement learning [4].

In real production systems, there are two facets of reasons contributing to product problems. On one hand, products are perishable which means that they cannot stay in production lines for long time [5,6]. On the other hand, due to the characteristics of the machine itself, some work-in-process cannot meet quality standards [7,8]. In the latter case, to reduce defective parts, inspection machines are introduced to remove defective parts in production systems timely. These production systems operate in a customized and small-lot mode for fulfilling product quality requirement. In this production mode, systems are dynamic and different from steady systems. They cannot be analyzed using steady system analysis methods [9]. Aiming to address this problem, a model with geometric machines in production systems is established and approximate algorithm is proposed [10]. Based on small-lot production assumptions, Bernoulli serial production lines are established, and transient performance is analyzed [11].

However, in actual production processes, the life and processing conditions of degrading machines can degrade to a failed state as the running time increases [2,12]. To ensure that production systems operate efficiently, taking maintenance actions to degrading machines becomes critical. The common maintenance actions include corrective maintenance (CM) and preventive maintenance (PM). For example, based on degrading machines and batch production assumptions, a production-driven opportunistic maintenance policy is proposed [13]. In semiconductor manufacturing, mixed-integer programming models are formulated for scheduling all due PM tasks [14]. Based on the proposed discrete-time Markov decision model, optimal maintenance policies are obtained by considering average cost [15]. However, these works do not consider action space explosion problems as the number of degrading machines increases.

Therefore, reinforcement learning (RL) shed lights on dealing with this problem [16,17]. Considering aging machines in serial production lines, a reward function is formulated by taking advantage of double deep Q-network (DDQN) to obtain optimal maintenance policies [18]. As an extended work, an improved multi-agent reinforcement learning (MARL) named value decomposed actor-critic (VDAC) is proposed to solve maintenance decision-making problems [19,20]. For flow line systems, condition-oriented maintenance scheduling is investigated based on reinforcement learning [21]. Nguyen et al. propose a weighted-QMIX-based optimization method to carry out maintenance activities for multi-component systems [22]. Although the above literature considers multiple machines maintenance problems from a system level, few work involves with product quality problems.

Therefore, in this work, based on small-lot production mode and finite buffer capacity assumptions, we establish Bernoulli reliability machines with machining quality degradation. Given that the stochasticity and complexity of production systems, one single degrading machine is regarded as an agent. Maintenance actions are specified. According to the current states of agents and all kinds of maintenance cost, a reward function is formulated. Optimized maintenance policies are obtained through QMIX [23]. The numerical study demonstrates that production systems after applying optimal maintenance policies can shorten the completion time.

2. MODELS

2.1. Assumptions

In this work, serial production lines consisting of Bernoulli machines, degrading machines, inspection machines and finite buffer capacity are modelled in Figure 1. In machining processes, there are a variety of reasons leading machines to degrading such as flexible fixture location errors and cutting tool wear. Defective parts can be produced in this way and be transferred to the downstream buffer. Once they are inspected by inspection machines, they will be removed from serial production lines.

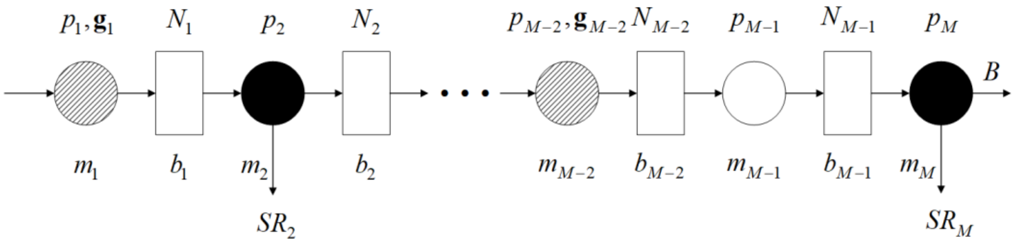

Figure 1. Serial Production Lines with Degrading Machines

Assumptions for this model in Figure 1 are specified below.
(1) This serial production line consists of M machines $(m_1, \dots, m_M)$ and $M-1$ buffers $(b_1, \dots, b_{M-1})$.
(2) Machines have identical cycle, and work at the beginning of each cycle.
(3) Buffers have limited capacity, i.e., $1 \leq N_i < \infty, i = 1, \dots, M-1$. Parts in buffers can be transferred randomly.
(4) Machines follow Bernoulli reliability model. Specifically, in one cycle, machine $m_i, i = 1, \dots, M$ can produce a part with probability p_i, cannot work with probability $1 - p_i$, if and only if the machine is not either starved or blocked. p_i is called the efficiency of machine m_i. For ease of expression, we introduced I_q and I_{insp} to represent index sets of degrading machines and inspection machines.
(5) Degrading machines $m_i, i \in I_q$ follow degradation model. Quality efficiency $\boldsymbol{g}_i = [g_{i,1}, \dots, g_{i,F}]$ can be degraded through matrix $\boldsymbol{D}_i$ (see equation (1)). Under (4), m_i can produce a non-defective part with $g_{i,j}$, and produce a defective part with $1 - g_{i,j}$.
(6) During one cycle, if machine m_i is up and buffer b_i has N_i parts, and the downstream of them cannot process parts, machine m_i is blocked. If machine m_i is up and buffer $b_i - 1$ is empty, machine m_i is starved. Machine m_1 is never starved and m_M is never blocked.
(7) The serial production system needs to finish one batch non-defective parts. Time to complete one batch is called completion time, denoted by CT.

$$\boldsymbol{D}_i = \begin{bmatrix} d_{1,1} & d_{1,2} & 0 & \cdots & 0 & 0 \\ 0 & d_{2,2} & d_{2,3} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & d_{F-1,F-1} & d_{F-1,F} \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix}. \tag{1}$$

2.2. Maintenance Rules

As shown in Figure 1, machine m_1 and m_{M-2} are degrading machines. The degradation processes and maintenance rules can be described in Figure 2. The transition probabilities among these states are defined by the matrix $\boldsymbol{D}_i$.

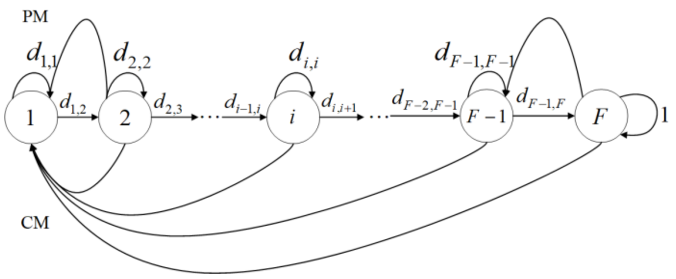

Figure 2. Machines Degrading States and Maintenance Rules Diagram

In Figure 2, PM operation can make states of degrading machines restore to the former states, i.e., quality efficiency $g_{i,j}$ to $g_{i,j-1}$. For machine m_i, the cost and time of PM are pd_i and pc_i, respectively. The machine under PM is idle. CM operation can make states of degrading machines restore to completely new, i.e., quality efficiency $g_{i,j}$ to $g_{i,1}$. For machine m_i, the cost and time of CM are cd_i and cc_i, respectively. Machine under CM is idle. If no action exerts on degrading machines, it is called do nothing (DoN).

3. MULTI-AGENT MAINTENANCE DECISION-MAKING

In this section, to overcome maintenance action space explosion, we take advantage of MARL to search optimal maintenance decisions over the planning horizon. The main task is to formulate multiple degrading machines maintenance decision-making as an MARL problem.

3.1. Markov Game

Simulation-based methods can describe the production environment specified exactly [24]. Time t indicates the simulation time step. At each simulation episode, degrading machines start to work without degradation. Then, they degrade following their own transition probability matrix $\boldsymbol{D}_i$ as shown in Figure 2. In this simulation-based production environment, Markov decision processes (MDP) can be utilized to describe the system operation. When the system produces B non-defective parts, the simulation of this production system is finished. Each agent means a degrading machine $m_i, i \in I_q$. The partial observation o_t^i of agent m_i needs to consider the current degrading state $sd_t^i \in \{1, \ldots, F_i\}$ and the remaining maintenance time $rd_t^i \in \{0,1, \ldots, cd_i\}$. When $rd_t^i > 0$, degrading machines are idle and cannot process parts from their upstream. The system state denoted by $S_t \in S$. At each time step, an agent is taken an action u_t^i and the action set of all agents is $\boldsymbol{u} \in \boldsymbol{U}$. A transition on this environment is generated by state transition function $P(s'|s,u): S \times \boldsymbol{U} \times S \rightarrow [0,1]$. A reward function for all agents is expressed by $r(s,\boldsymbol{u}): S \times \boldsymbol{U} \rightarrow \mathbb{R}$ and discount factor is denoted by $\gamma \in [0,1)$.

In this environment, each agent can obtain its own partial observation $o \in O$. During simulation, each agent can obtain an action-observation trajectory $\tau \in T$. Based on this, a stochastic policy $\pi^i(u^i|\tau^i): T \times U \rightarrow [0,1]$. Following a joint policy π, the action-value function can be given: $Q^{\pi}(s_t,u_t) = E_{s_{t+1:\infty},u_{t+1:\infty}}[R_t|s_t,u_t]$. R_t indicates the discounted return: $R_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$.

3.2. Model Formulation

As discussed in section 3.1, the local observation for agent m_i is: $o^i = [sd^i, rd^i]$. The global state of this system is defined as: $s = [o^1, o^2, \ldots, o^{|I_q|}, K]$, where K is the number of non-defective parts produced by the last machine. The maintenance actions can be formulated by Equation (2).

$$u_t^i = \begin{cases} 0, & \text{take PM on machine } m_i, \\ 1, & \text{take CM on machine } m_i, \\ 2, & \text{leave machine } m_i \text{ as it is.} \end{cases} \tag{2}$$

During time step t, the joint action can be described as: $u_t = [u_t^1, \dots, u_t^{|I_q|}]$. In this setting, there are three types of actions. For centralize learning process, the dimension of actions of all agents is $3^{|I_q|}$. When increasing the number of action types, the dimension of actions space will grow exponentially. In our work, we consider decentralized policies that allow each agent to select its own action independently based on its own action-observation trajectories. To design a reasonable reward function, some conflicting factors need to be balanced. On one hand, frequent maintenance actions can lead machines to producing non-defective parts with high probabilities. On the other hand, redundant maintenance action can delay the whole production processes and increase total cost. Therefore, the reward function designed in this work consists of states of degrading machines, PM cost, CM cost. At time step t, the reward function is defined as follows:

$$r_t = \omega \sum_{i=1}^{|I_q|} \frac{1}{sd_t^i} - \sum_{i=1}^{|I_q|} C_i^{PM} \mathbb{1}_i^{PM}(t) - \sum_{i=1}^{|I_q|} C_i^{CM} \mathbb{1}_i^{CM}(t), \tag{3}$$

where each item is explained below:
ω is the assigned weight for degrading states.
sd_t^i is the current state of machine m_i at time step t.
Therefore, $w \sum_{i=1}^{|I_q|} \frac{1}{sd_t^i}$ denotes the total benefits of at time step t.
C_i^{PM} is the cost of exerting PM on machine m_i, i.e., $C_i^{PM} = pc_i + pd_i$.
$\mathbb{1}_i^{PM}(t)$ is an indicator function, expressing whether PM is taken on machine m_i. If $\mathbb{1}_i^{PM}(t) = 1$, PM is taken on machine m_i. Otherwise, $\mathbb{1}_i^{PM}(t) = 0$.
C_i^{CM} is the cost of exerting CM on machine m_i, i.e., $C_i^{CM} = cc_i + cd_i$.
$\sum_{i=1}^{|I_q|} C_i^{PM} \mathbb{1}_i^{PM}(t)$ means the total cost of PM on all degrading machines.
Similarly, $\sum_{i=1}^{|I_q|} C_i^{CM} \mathbb{1}_i^{CM}(t)$ means the total cost of CM on all degrading machines.

4. QMIX THEORY AND NETWORK

For multi-agent cooperative problems, decentralized policies can overcome the difficulty of joint action spaces explosion. MARL in the centralized training and decentralized execution fashion is adopted for this work. However, this method also exists some challenges. The action-value function Q_{tot} is dependent on the global state and joint action. This function is hard to learn.

4.1. QMIX Theory

Among these MARL algorithms, one easy way is to forgo the centralized action-value function as Q_{tot} is hard to represent and use. Alternatively, independent Q-learning is proposed to assign each agent to an individual action-value function Q_i [25]. But this method cannot obtain the global optimum because of forgoing Q_{tot}. Based on the global action-value function Q_{tot}, Rashid et al. proposed an MARL algorithm QMIX [23]. As shown in Equation (4), the result of a global argmax for Q_{tot} is the same as a set of individual argmax for each Q_i. According to this equivalency, each agent can select its own action based on Q_i.

$$\underset{u}{\operatorname{argmax}}\, Q_{tot}(\tau, u) = \begin{pmatrix} \underset{u^1}{\operatorname{argmax}}\, Q_1(\tau^1, u^1) \\ \vdots \\ \underset{u^n}{\operatorname{argmax}}\, Q_n(\tau^n, u^n) \end{pmatrix}. \tag{4}$$

Compared with value decomposition networks (VDNs) [26], QMIX can be used by a larger family of monotonic functions. Monotonicity needs to be ensured by a constraint as follows:

$$\frac{\partial Q_{tot}}{\partial Q_i} \geq 0, \forall i \in n. \tag{5}$$

4.2. QMIX Network

To meet this constraint, QMIX expresses Q_{tot} through a set of networks. As shown in Figure 3, this architecture consists of agent networks, a mixing network and hypernetworks [23]. Agent networks have same architecture to output their own value functions $Q_i(\tau^i, u^i)$. They use deep recurrent Q-networks (DRQN) inputting their observations o_t^i and last action u_{t-1}^i [27]. To satisfy this constraint, the weights in this mixing network are set to be non-negative. Each hypernetwork takes the global state as input and generates the weights of each layer. These hypernetworks are composed of a single linear layer followed by an absolute value activation function to ensure that the weights are non-negative. The biases can be generated in the same way, but it does not have to be guaranteed to be non-negative. The global state is used as an input of the hypernetworks so that the full information of the system can be integrated into the joint action value estimate.

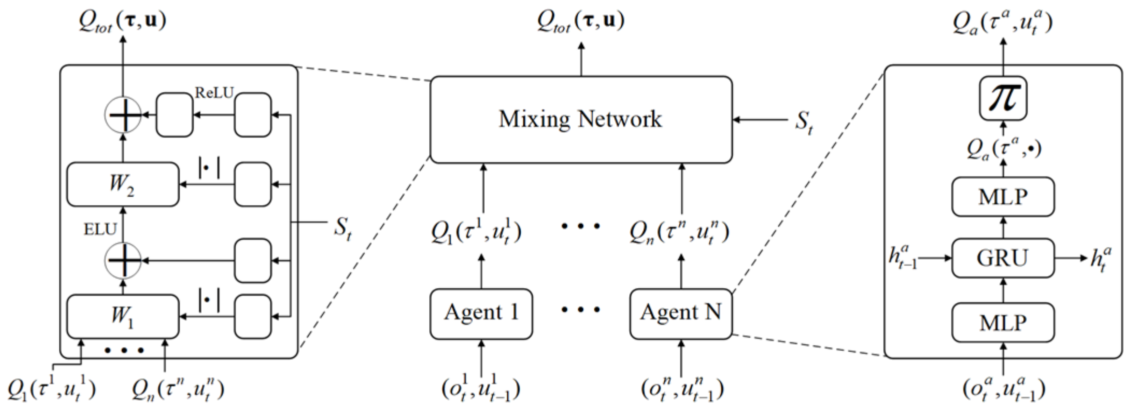

Figure 3. QMIX Network Architecture [23]

To minimize loss function of QMIX, we use end-to-end way to train:

$$\mathcal{L}(\theta) = \sum_{i=1}^{b}[(y_i^{tot} - Q_{tot}(\boldsymbol{\tau}, \boldsymbol{u}, s; \theta))^2], \tag{6}$$

where b is the batch size of defined replay buffer, and $y^{tot} = r + \gamma \max_{\boldsymbol{u'}} Q_{tot}(\boldsymbol{\tau}', \boldsymbol{u}', s'; \theta^-)$ and θ^- are the parameters of a target network as in deep Q-network (DQN) [28]. Since Equation (4) holds, Q_{tot} can be maximized. It has linear relationships with Q_i, thus avoiding the dimension explosion problems.

5. NUMERICAL STUDY

To verify the QMIX-based maintenance policies, QMIX-based experiments are carried out and explained. Then, comparing with baselines, it is shown that QMIX-based policies are better.

5.1. Experiment Description

We firstly verify the convergence of this algorithm during training course. Specifically, QMIX is applied to a six-machine five-buffer serial production system through training to obtain optimal maintenance policies. Next, the optimal policies are compared with two baselines. Run-to-Failure (R2F) policies indicate that machines are taken CM when they are in the worst state. The other one is that we take no actions on degrading machines, i.e., DoN policies. Parameters of this system are listed below including machine parameters (Table 1), buffer parameters (Table 2), degrading quality efficiencies and degrading transition matrices (Equation (7)-(8)).

Table 1. Machine Parameters

Parameters	$\boldsymbol{m_1}$	$\boldsymbol{m_2}$	$\boldsymbol{m_3}$	$\boldsymbol{m_4}$	$\boldsymbol{m_5}$	$\boldsymbol{m_6}$
Efficiency $\boldsymbol{p_i}$	0.9	0.96	0.97	0.93	0.91	0.92
PM cost $\boldsymbol{pc_i}$	2	-	2	2	-	-
PM time $\boldsymbol{pd_i}$	1	-	1	1	-	-
CM cost $\boldsymbol{cc_i}$	8	-	8	8	-	-
CM time $\boldsymbol{cd_i}$	2	-	2	2	-	-

Table 2. Buffer Parameters

Parameters	$\boldsymbol{b_1}$	$\boldsymbol{b_2}$	$\boldsymbol{b_3}$	$\boldsymbol{b_4}$	$\boldsymbol{b_5}$
Buffer capacity $\boldsymbol{N_i}$	3	5	5	3	3

$$\boldsymbol{g}_i = \begin{cases} [0.9, 0.7, 0.5, 0.25], & i = 1, \\ [0.9, 0.7, 0.5, 0.25], & i = 3, \\ [0.9, 0.7, 0.6, 0.5, 0.25], & i = 4. \end{cases} \tag{7}$$

$$\boldsymbol{D}_1 = \begin{bmatrix} 0.8 & 0.2 & 0 & 0 \\ 0 & 0.8 & 0.2 & 0 \\ 0 & 0 & 0.7 & 0.3 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \boldsymbol{D}_2 = \begin{bmatrix} 0.9 & 0.1 & 0 & 0 \\ 0 & 0.7 & 0.3 & 0 \\ 0 & 0 & 0.6 & 0.4 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \boldsymbol{D}_3 = \begin{bmatrix} 0.9 & 0.1 & 0 & 0 & 0 \\ 0 & 0.7 & 0.3 & 0 & 0 \\ 0 & 0 & 0.6 & 0.4 & 0 \\ 0 & 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{8}$$

For each simulation episode of serial production systems, the number of non-defective parts required is set to: $B = 10$. In this experiment, the weight of the defined reward function is specified: ω is equal to 0 when DoN is applied to degrading machines, and ω is equal to 10 when degrading machines are under maintenance. Each agent network has a gated recurrent unit (GRU) with a 64-dimensional hidden state and is connected by fully connected layers. The optimizer during training course is root mean square propagation (RMSprop) with a learning rate $\alpha = 5 \times 10^{-4}$. The discounted factor γ is 0.98. The mixing network includes a hidden layer of 32 units with parameters given by hypernetworks. An exponential linear unit (ELU) activation function acts on the output of the hidden layer. Each hypernetwork consists of a feedforward network, a 64-unit hidden layer and a rectified linear unit (ReLU) activation function sequentially. This model is trained by 6.5×10^4 epochs. In the first 5000 epochs, actions are selected by ϵ-greedy. ϵ decrease from 1 to 0.05 linearly. A mini-batch size is set to 32 sampling from replay buffer with size 100.

5.2. Experiment Results

The training results can be shown in Figure 4. As one can see, the reward of this system is convergent within 40, 000 training epochs. Evidently, it can reach a higher value. To display the effectiveness of QMIX-based policies, we compare with R2F and DoN policies. These two policies are run by 6.5×10^4 epochs and show the average value of reward and CT. As shown in Figure 5, there are some fluctuations of CT using QMIX-based policies because of the interaction between the stochastic nature of production systems and the generated policies. In general, this method can obviously shorten CT. As to reward, it is higher compared to average reward of R2F. It should be noted that the reward is always 0 under DoN. The results provide some insights that R2F policies are not efficient for industrial maintenance.

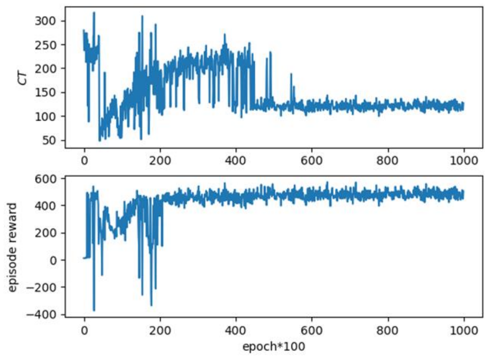

Figure 4. Training Results

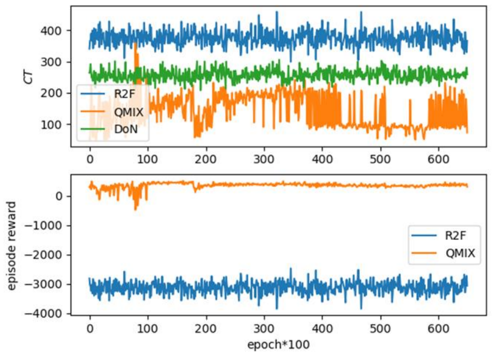

Figure 5. Comparison Results

6. CONCLUSION

In this work, maintenance decision-making problems for the serial production lines with degrading machines are solved through MARL. We firstly established the serial production lines model with degrading machines and defined maintenance actions. Then, maintenance decision-making problems for this model can be formulated by MARL. In the small-lot production mode, the completion time is very important. Therefore, it is key to balance the completion time and maintenance cost. Therefore, an MARL algorithm which has good performance in cooperative problems is adopted and is suitable for this decision-making problems in this paper. Finally, the convergence of this model and comparison experiments are verified.

In future work, for the maintenance decision-making problems of production systems, degraded machines subject to different degradation models and different production systems structures will be studied. Due to the complexity and nonlinear characteristics of the system, novel multi-agent reinforcement learning algorithms will be studied for optimizing maintenance decisions.

Acknowledgements

The authors gratefully acknowledge the financial supports of this work by the Research Grants Council of Hong Kong under the grant No. 11215323, and the National Natural Science Foundation of China under the grant No. 72101221.

References

[1] L. Pinciroli, P. Baraldi, and E. Zio, “Maintenance optimization in industry 4.0,” Reliability Engineering & System Safety, vol. 234, p. 109204, Jun. 2023.
[2] H. Wang, “A survey of maintenance policies of deteriorating systems,” European Journal of Operational Research, vol. 139, no. 3, pp. 469–489, Jun. 2002.
[3] L. Li, Y. Wang, and K.-Y. Lin, “Preventive maintenance scheduling optimization based on opportunistic production-maintenance synchronization,” Journal of Intelligent Manufacturing, vol. 32, no. 2, pp. 545–558, Feb. 2021.
[4] H. Yang, W. Li, and B. Wang, “Joint optimization of preventive maintenance and production scheduling for multi-state production systems based on reinforcement learning,” Reliability Engineering & System Safety, vol. 214, p. 107713, Oct. 2021.
[5] J. Wang, Y. Hu, and J. Li, “Transient Analysis to Design Buffer Capacity in Dairy Filling and Packing Production Lines,” Journal of Food Engineering, vol. 98, no. 1, pp. 1–12, 2010.
[6] F. Ju, J. Li, and J. A. Horst, “Transient Analysis of Serial Production Lines With Perishable Products: Bernoulli Reliability Model,” IEEE Transactions on Automatic Control, vol. 62, no. 2, pp. 694–707, Feb. 2017.
[7] S. M. Meerkov and L. Zhang, “Product Quality Inspection in Bernoulli Lines: Analysis, Bottlenecks, and Design,” International Journal of Production Research, vol. 48, no. 16, pp. 4745–4766, 2010.
[8] M. Colledani et al., “Design and Management of Manufacturing Systems for Production Quality,” CIRP Annals, vol. 63, no. 2, pp. 773–796, 2014.
[9] J. Li and S. M. Meerkov, Production Systems Engineering. New York, NY: Springer, 2009.
[10] G. Chen et al., “Transient Performance Analysis of Serial Production Lines with Geometric Machines,” IEEE Transactions on Automatic Control, vol. 61, no. 4, pp. 877–891, 2016.
[11] Z. Jia et al., “Finite Production Run-Based Serial Lines with Bernoulli Machines: Performance Analysis, Bottleneck, and Case Study,” IEEE Transactions on Automation Science and Engineering, vol. 13, no. 1, pp. 134–148, 2016.
[12] M. Colledani and T. Tolio, “Integrated quality, production logistics and maintenance analysis of multi-stage asynchronous manufacturing systems with degrading machines,” CIRP Annals, vol. 61, no. 1, pp. 455–458, 2012.
[13] T. Xia et al., “Production-driven opportunistic maintenance for batch production based on MAM–APB scheduling,” European Journal of Operational Research, vol. 240, no. 3, pp. 781–790, Feb. 2015.
[14] X. Yao, et al., “Optimal preventive maintenance scheduling in semiconductor manufacturing,” IEEE Transactions on Semiconductor Manufacturing, vol. 17, no. 3, pp. 345–356, Aug. 2004.

[15] C. C. Karamatsoukis and E. G. Kyriakidis, “Optimal maintenance of two stochastically deteriorating machines with an intermediate buffer,” European Journal of Operational Research, vol. 207, no. 1, pp. 297–308, Nov. 2010.

[16] S. Gronauer and K. Diepold, “Multi-agent deep reinforcement learning: a survey,” Artificial Intelligence Review, vol. 55, no. 2, pp. 895–943, Feb. 2022.

[17] A. Oroojlooy and D. Hajinezhad, “A review of cooperative multi-agent deep reinforcement learning,” Applied Intelligence, vol. 53, no. 11, pp. 13677–13722, Jun. 2023.

[18] J. Huang, Q. Chang, and J. Arinez, “Deep reinforcement learning based preventive maintenance policy for serial production lines,” Expert Systems with Applications, vol. 160, p. 113701, Dec. 2020.

[19] J. Su, S. Adams, and P. Beling, “Value-Decomposition Multi-Agent Actor-Critics,” AAAI, vol. 35, no. 13, pp. 11352–11360, May 2021.

[20] J. Su et al., “Deep multi-agent reinforcement learning for multi-level preventive maintenance in manufacturing systems,” Expert Systems with Applications, vol. 192, p. 116323, Apr. 2022.

[21] R. Lamprecht, F. Wurst, and M. F. Huber, “Reinforcement learning based condition-oriented maintenance scheduling for flow line systems,” in 2021 IEEE 19th International Conference on Industrial Informatics (INDIN), Palma de Mallorca, Spain: IEEE, Jul. 2021.

[22] V.-T. Nguyen et al., “Weighted-QMIX-based Optimization for Maintenance Decision-making of Multi-component Systems,” in PHM Society European Conference, pp. 360–367, Jun. 2022.

[23] T. Rashid et al., “QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning,” Proceedings of the 35th International Conference on Machine Learning, PMLR 80:4295-4304, Jun. 2018.

[24] A. Alfieri and A. Matta, “Mathematical programming formulations for approximate simulation of multistage production systems,” European Journal of Operational Research, vol. 219, no. 3, pp. 773–783, Jun. 2012.

[25] M. Tan, “Multi-agent reinforcement learning: Independent vs. cooperative agents,” in Proceedings of the tenth international conference on machine learning, pp. 330–337, 1993.

[26] P. Sunehag et al., “Value-decomposition networks for cooperative multi-agent learning based on team reward,” in Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, in AAMAS ’18. Richland, pp. 2085–2087, 2018.

[27] V Mnih et al., “Human-level control through deep reinforcement learning,” Nature, 518 (7540):529–533, 2015.

[28] M. Hausknecht and P. Stone, “Deep Recurrent Q-Learning for Partially Observable MDPs,” In AAAI Fall Symposium on Sequential Decision Making for Intelligent Agents, 2015.