CC BY NC

Int j simul model 23 (2024) 4, 692-703
Original scientific paper

OPTIMIZING PRODUCTION WITH DEEP REINFORCEMENT LEARNING

Wei, Z. H.[*,#]; Yan, L.[*] & Yan, X.[**]

[*] School of Economics and Management, China University of Geosciences, Wuhan 430074, China

[**] School of Business Administration / Cantonese Merchant School / Innovation and Entrepreneurship School, Guangdong University of Finance and Economics, Guangzhou 510320, China

E-Mail: weizhehuan@cug.edu.cn, ylyzb@cug.edu.cn, yanxiao@gdufe.edu.cn

([#] Corresponding author)

Abstract

The optimization management of complex production processes is the key to enhancing the competitiveness of enterprises in modern manufacturing. Traditional optimization methods often struggle to manage the dynamic and intricate nature of production environments, highlighting the need for more intelligent and efficient approaches. As an advanced technique in artificial intelligence, Deep Reinforcement Learning (DRL) has demonstrated significant potential in addressing high-dimensional, nonlinear dynamic system optimization through interactive learning with the environment. However, current research approaches face limitations in handling the diversity and dynamism inherent in complex production workflows. In this study, a simulation model was constructed to accurately represent real-world production processes, which was subsequently employed as the basis for DRL-driven optimization. Results indicate that this method effectively enhances the overall performance and adaptability of production systems, providing robust support for advancements in smart manufacturing and Industry 4.0.

(Received in August 2024, accepted in October 2024. This paper was with the authors 3 weeks for 1 revision.)

Key Words: Complex Production Processes, Deep Reinforcement Learning, Simulation Model, Optimization Design, Smart Manufacturing

1. INTRODUCTION

In modern manufacturing, the management and optimization of complex production processes are critical for enhancing the competitiveness of enterprises [1-4]. As production techniques advance and demands diversify, traditional manual experience and simple optimization methods struggle to meet the challenges posed by increasingly intricate production environments [5, 6]. Accurate and efficient simulation and optimization of production processes can significantly improve productivity, reduce costs, and increase the system's flexibility and adaptability [7-10]. Consequently, exploring advanced techniques for simulating and optimizing complex production processes holds significant practical relevance.

In recent years, DRL, an emerging technology in artificial intelligence, has demonstrated considerable potential in solving high-dimensional, nonlinear dynamic system optimization problems [11-14]. Through interactive learning with the environment, DRL incrementally optimizes decision-making processes under conditions of incomplete information and high uncertainty, offering novel insights and methods for production process optimization [15, 16]. Applying DRL to the simulation and optimization of complex production processes not only promises to overcome the limitations of traditional methods but also provides robust technical support for smart manufacturing and Industry 4.0.

Despite DRL's notable achievements across various fields, challenges remain in its application to the simulation and optimization of complex production processes [17-19]. Existing research methods often fall short in addressing the diversity and dynamism of complex production processes, making it difficult to capture the myriad uncertainties present in real-world production environments [20, 21]. Moreover, constructing efficient and accurate

simulation models for production processes remains a primary research difficulty. Developing effective integration between simulation technologies and DRL algorithms to achieve precise modelling and optimization of complex production processes is a pressing issue.

This study can be divided into two parts. First, a simulation model of complex production processes was constructed to accurately replicate various workflows and dynamic behaviours observed in actual production. Second, based on DRL methodologies, an optimization framework for complex production processes was designed and implemented. By combining DRL with the simulation model, this study aims to achieve efficient optimization of complex production processes, thereby enhancing overall system performance and adaptability. This study not only advances theoretical research on production process optimization but also provides a solid foundation for the practical application of intelligent optimization techniques in real-world manufacturing, contributing valuable academic insights and practical potential.

2. SIMULATION MODEL OF COMPLEX PRODUCTION PROCESSES

2.1 Directed Acyclic Graph (DAG)

In the simulation of complex production processes, the DAG model was introduced to effectively represent and manage individual production tasks and their dependencies. Production workflows typically involve multiple interdependent tasks, with dependency relationships determining the sequence of execution. By utilizing a DAG model, the sequence of tasks within the entire production process can be clearly illustrated, ensuring that each task begins only after all preceding tasks have been completed. This approach avoids logical inconsistencies and execution blocks caused by cyclical dependencies. Furthermore, the DAG model aids in identifying critical paths and bottleneck tasks within the production process, thereby providing an intuitive reference for optimizing production scheduling and resource allocation.

In simulating complex production processes, nodes within the DAG model represent various production task computational units. These computational units encompass a range of tasks, including data transformation, data filtering, processing operations, equipment maintenance, and quality inspection, among others. The execution of each node signifies an independent production step or operational unit, while the directed edges between nodes specify the dependencies and direction of data flow among tasks. In this simulation, topological sorting ensures that each production task is executed only after all preceding tasks have been completed, thus avoiding cyclical dependencies and task blockages. Nodes with zero in-degree represent independent tasks that do not depend on others and can be prioritized for execution, while nodes with zero out-degree indicate terminal tasks that, upon completion, do not trigger further tasks. Topological traversal is a critical step in applying the DAG model within production process simulation. This process initiates from nodes with zero in-degree, executing one task at a time, and subsequently updating the in-degree of each successive node. The detailed steps are as follows:

Step 1: Identify all nodes with zero in-degree and add them to the initial execution queue. **Step 2:** Remove a node from the queue and execute its associated production task. **Step 3:** Update the in-degrees of all successor nodes; if any successor node's in-degree becomes zero, add it to the execution queue. **Step 4:** Repeat steps 2 and 3 until the queue is empty.

2.2 Workflow model

In the simulation of complex production processes, the core task of workflow production scheduling involves systematically assigning different production stages and tasks to appropriate computational resources, ensuring the production system operates efficiently and

reliably. Workflow production scheduling refers to the automated division, allocation, and execution of production tasks within a cloud computing platform or data centre, based on the demands of the production process. During this process, each production task is treated as a subtask, and dependencies among tasks form a DAG model, ensuring that the execution order aligns with the logical sequence of the production process. With effective scheduling strategies, tasks can be completed in the shortest possible time while maximizing the utilization of computational resources. Specifically, in the simulation of complex production processes, tasks may include material handling, equipment scheduling, quality inspection, and more. The execution of each task may depend on the completion of prior tasks, and workflow scheduling effectively organizes these tasks, ensuring that dependencies are executed correctly. Fig. 1 shows an example of a workflow task model containing six tasks.

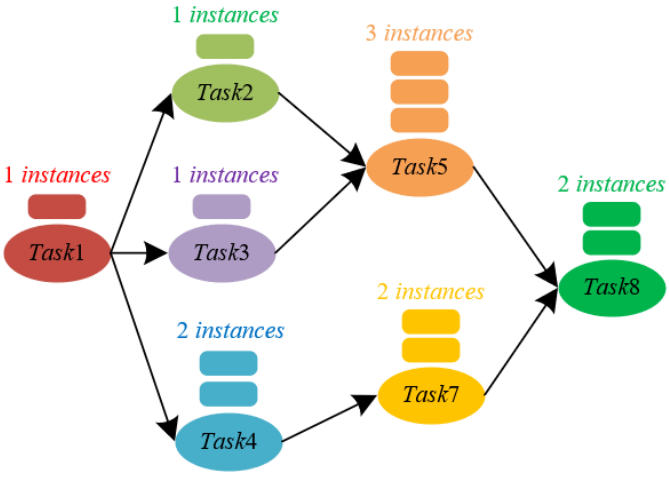

Figure 1: Example of a workflow task model containing eight tasks.

The workflow scheduling process of production tasks in complex production processes generally involves three main steps: a) Subdivision of workflow subtasks. The complex production process is divided into multiple interrelated and dependent subtasks. Each subtask is represented as a node in the DAG model, with directed edges indicating dependencies between tasks. b) Subtask scheduling. In this phase, production tasks are allocated to virtual machines or computational resources in a cloud platform or data centre based on various constraints, including priority, resource requirements, execution time, and dependencies. Resource consumption, execution order, and dynamic changes in the production process must all be considered to optimize scheduling order and resource allocation. c) Production task execution monitoring and management. During task execution, real-time monitoring of each subtask's status is essential to assess progress and identify potential bottlenecks. Data collected during task execution allows for dynamic adjustment of scheduling strategies and optimization of resource allocation, thus enhancing the efficiency and reliability of the entire production process simulation. Through this systematic workflow scheduling process, complex production process simulations can achieve not only greater computational efficiency but also ensure accurate task execution and smooth workflow progression in highly dynamic production environments.

Complex production processes typically involve multiple subtasks and workflows, with intricate dependency relationships and prioritization orders among them. By constructing a DAG model, these dependencies among subtasks can be clearly represented, providing a scientific basis for scheduling. In actual complex production process simulations, task execution rules involve not only dependency relationships but also considerations of resource allocation and scheduling strategies. For example, the resource requirements and number of instances for each task may vary, necessitating a scheduling system that dynamically adjusts

based on task demands and resource availability. For tasks such as *Task*2, *Task*3, and *Task*4, which depend on *Task*1, execution must be delayed until *Task*1 is complete to maintain task sequence integrity. Furthermore, during task execution, the DAG model should optimize resource allocation according to factors such as resource requirements, execution time, and instance count, thereby preventing excessive or insufficient resource use and enabling efficient production process simulation.

2.3 Data centre model for complex production processes

In the simulation model of complex production processes, virtual machine resource modelling is a core component for achieving efficient task scheduling and resource allocation. Virtual machine resources in the data centre can be represented as a set $N = \{n_1, n_2, ..., n_l\}$, where l denotes the total number of virtual machines. Each virtual machine $n_u = (MI_u, CP_u, ME_u, DI_u)$ has a unique resource configuration, which includes computational capacity MI_u, number of CPU cores CP_u, memory size ME_u, and disk capacity DI_u. These configurations enable virtual machines to support various types of computational tasks. To optimize task scheduling, it is necessary to consider the compatibility between the computational requirements of tasks and the resources of the virtual machines. For instance, some tasks may require significant memory or computational power, while others may demand high disk capacity or network bandwidth. Thus, the key to virtual machine resource modelling lies in describing resource configurations accurately to ensure that tasks can be executed on appropriate virtual machines, maximizing resource utilization. Assuming that a task s_u within the complex production process simulation is assigned to a virtual machine n_j, the execution time of task s_u can be calculated by the following equation:

$$RS(s_u) = \frac{m_u}{MIt_j} \tag{1}$$

In this simulation model, certain assumptions regarding virtual machine resources are made to simplify the scheduling process and enhance simulation efficiency. It is initially assumed that all virtual machines are located within the same data centre and that they share similar resource configurations. This assumption simplifies the resource allocation problem within the complex production process, avoiding additional complexity and delays associated with cross-data-centre scheduling. Another important assumption is that the workflow scheduling considers only the execution time of the tasks, excluding other factors that may impact the total duration, such as inter-task communication, data transfer, or storage operations. This assumption further simplifies the task execution model in the simulation of complex production processes, allowing researchers to directly analyse and optimize task sequencing and resource scheduling strategies without the computational overhead introduced by these additional factors. Specifically, the completion time $RZS(s_u)$ of task s_u can be represented as follows:

$$RZS(s_u) = \begin{cases} RS(s_u), D(s_u) = \theta \\ MAX_{s_j \in D(s_u)}\{RS(s_j))\} + RS(s_u), D(s_u) \neq \theta \end{cases} \tag{2}$$

The maximum completion time of the workflow in the complex production process simulation model is defined by the following equation:

$$LT_{JO_{uf}} = MAX\{RZS(s_{JS})\} \tag{3}$$

This equation represents the completion time of a production task s_{JS} with an out-degree of zero. The maximum execution time for the task s_u was determined by considering the possibility that s_{JS} may have multiple parallel subtasks.

In the simulation model of complex production processes, the response time of the workflow is a key indicator of overall workflow efficiency. It is typically defined as the difference between the workflow's completion time and its submission time. Due to the

dependencies among tasks within the workflow, the overall completion of the workflow is marked by the completion of all subtasks, rather than the completion of any single task. Consequently, the calculation of response time for the workflow must be based on the completion status of the entire workflow, rather than the completion time of an individual task. In the simulation model of this chapter, the submission time of the workflow is typically taken as the submission time t_{SJ} of the first production task s_{KS}, with the assumption that all subtasks have the same submission time. This assumption simplifies the scheduling of workflows and the calculation of response time, as, in real-world production processes, multiple production tasks within a workflow often start simultaneously. This is particularly true in data centre resource scheduling, where task submission times are typically synchronized. For a multi-workflow environment, the average response time can be calculated using the following equations:

$$ES_{JO_{uf}} = LT_{JO_{uf}} - t_{SJ} \tag{4}$$

$$ES_{AV} = \frac{\sum_{uf=1}^{v} ES_{JO_{uf}}}{v} \tag{5}$$

In the simulation model of complex production processes, the makespan model serves to accurately measure the total time required for all tasks to be completed within the data centre. The makespan represents the maximum completion time across all virtual machines, marking the latest moment by which all tasks have been completed. It is commonly used to evaluate the overall scheduling efficiency of the production process. To construct this model, it is necessary to define the completion time ZS_j of each virtual machine ε_j, representing the time required for virtual machine ε_j to complete all assigned tasks. The formula for calculating the completion time of virtual machine ε_j is expressed as $ZS_j = \varepsilon_j - KS(\varepsilon_j)$, where ε_j denotes the execution completion time for all tasks allocated to the j^{th} virtual machine, and $KS(\varepsilon_j)$ is the start time of task execution on that virtual machine. By applying this equation, the model is able to accurately track the time taken by each virtual machine from the start of execution until task completion, thereby determining the makespan for each virtual machine.

$$LT = MAX(\{ZS_0, ZS_1, \ldots, ZS_l\}) \tag{6}$$

3. OPTIMIZATION DESIGN OF COMPLEX PRODUCTION PROCESSES BASED ON DRL

In the optimization of complex production processes, the parallel partitioning of subtasks within a workflow is a critical step. The primary goal is to enhance overall scheduling efficiency by partitioning production tasks and processing them in parallel. A workflow typically consists of multiple interdependent subtasks, each of which may have distinct computational requirements and resource constraints. Parallelizing these subtasks not only accelerates task completion and reduces the overall execution time of the production process but also minimizes task conflicts and resource waste through appropriate resource allocation, thus further optimizing the scheduling performance of the workflow. The introduction of Quality of Service (QoS) constraints into DRL-based task scheduling algorithms ensures that workflow scheduling takes into account not only efficiency but also the quality requirements of task execution. In complex production processes, many tasks may have time-sensitive or resource-sensitive characteristics. QoS constraints ensure that, while scheduling efficiency is maintained, specific requirements such as maximum response time, minimum resource consumption, or predefined priority levels are met. Through DRL algorithms, the agent can make real-time adjustments and optimizations in response to different QoS constraints. By combining these aspects, a complex production process optimization system can be designed that adapts to changing environments and

balances both efficiency and quality, thereby improving resource management and task scheduling performance.

Regarding the parallel partitioning of subtasks, production tasks within a workflow must first be divided based on their execution phases, taking into account the dependencies between them. Within the same execution phase, production tasks have no dependencies and can therefore be executed in parallel. The key to this phase partitioning lies in identifying which tasks can be executed simultaneously without violating the dependency order of the production tasks, thus maximizing resource utilization and shortening the workflow's execution time. The parallelization of tasks in workflow production is divided into two main steps:

Step 1: The parallel partitioning of production tasks within a single workflow is the first step, followed by the parallel scheduling optimization across multiple workflows. The primary objective of this partitioning is to maximize resource utilization and minimize task waiting times. Within a single workflow, the production tasks are first analysed using the DAG model of the workflow, with each subtask's parent node set $D(s_u)$ and other task characteristics such as resource requirements and execution time being extracted. These characteristics are then used to define the execution phases of the subtasks. Subtasks that share the same parent task are grouped into the same parallel execution phase, ensuring parallel execution while maintaining the dependency relationships between tasks.

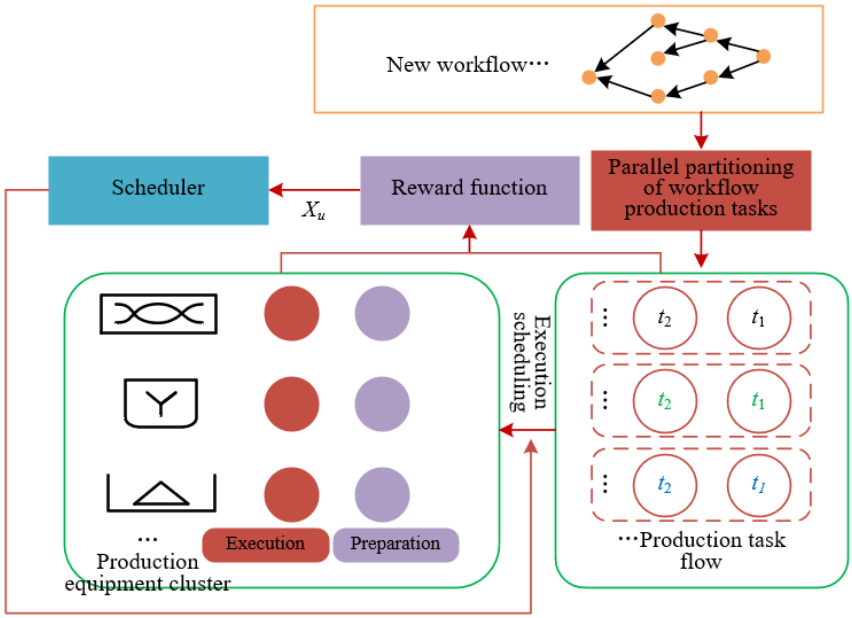

Figure 2: Framework for optimization design of complex production processes based on DRL.

Step 2: When scheduling across multiple workflows, in addition to considering the parallel partitioning of tasks within individual workflows, the resource sharing and scheduling optimization between workflows must also be taken into account. First, the internal parallelization of each arriving workflow is carried out, identifying the parent-child relationships between tasks and determining the execution phases of each task. All workflows are then encapsulated into a scheduling list, which is sorted according to the workflow submission times. During scheduling, the order of task execution must respect both the workflow submission time and the dependency relationships between tasks. Furthermore, to further enhance resource utilization, the scheduler can perform dynamic scheduling based on the resource requirements and execution times of the tasks across multiple workflows. Fig. 2 shows the framework of cloud data centre workflow scheduling based on DRL.

In this study, the scheduling of workflow production tasks not only aims to optimize makespan but also considers the response time of the workflow, which is defined as the total

time from the workflow submission to its completion. The response time is a key indicator for measuring the scheduling performance of workflows, influenced by factors such as task dependencies and resource contention. Therefore, Double Deep Q-Network (DoubleDQN) was selected as the DRL model for this study, as it demonstrates clear advantages in solving high-dimensional, complex state-space problems. The DoubleDQN model, through its refined *Q*-value estimation capability, continuously optimizes the scheduling strategy during the training process, reducing response time and task waiting times. By designing an appropriate reward function that incorporates response time as a critical factor, the agent is enabled to dynamically adjust its scheduling strategy based on the current task states, balancing both efficiency and QoS constraints. In multi-workflow scheduling scenarios, DoubleDQN can balance the scheduling order and resource allocation of production tasks through intelligent exploration and exploitation strategies, ensuring that the system maximizes resource utilization and scheduling efficiency while meeting QoS requirements.

In a complex production process, the state information parameters of the DRL model need to comprehensively account for various key factors in the scheduling process to enable the agent to make reasonable decisions within the environment. First, the state information should include the current status of virtual machine resources and the execution status of each pending production task's parent tasks. In this model, the state (L_{ST}) of a virtual machine can be represented as ($n_1, n_2, n_3, ..., n_l$), where each n_i denotes the resource configuration of the virtual machine and the completion time of the tasks already assigned to it. This allows the agent to understand the availability of each virtual machine and the resource requirements of tasks, providing a basis for scheduling decisions. Simultaneously, the workflow's state information (S_{WA}) includes the topological order of the pending sub-tasks and the execution status of each sub-task's parent tasks. Specifically, the state S_{WA} can be represented by the set $\{T_1, T_2, T_3, ...\}$, where each $S_u = (s_u, D(s_u))$ includes the current sub-task to be scheduled (s_u) and its parent task set ($D(s_u)$), which is used to determine whether the sub-task can begin execution. Regarding the definition of the action space, the agent's task is to select the appropriate virtual machine resources to schedule the pending production tasks. The action space $X = \{1, 2, 3, \ldots, l\}$ represents the number of virtual machines, where each action corresponds to a specific virtual machine choice. Specifically, when the agent observes the current environment state $T_u = (L_{ST}, S_{WA})$, it assesses the production task's dependency relationship $D(s_u)$ to determine whether the task s_u can be scheduled. If all of s_u's parent tasks have been completed (i.e., $D(s_u) = \phi$), the task can begin scheduling. The agent then selects an action $X_u = u$ to schedule the task and applies this decision to the current workflow scheduling. After execution, the agent receives a reward from the environment, indicating the quality of the scheduling decision. The reward is typically based on scheduling efficiency, task completion time, and resource utilization. The agent adjusts its strategy based on the reward value, favouring higher-reward actions in subsequent training iterations.

To achieve the optimization objective, the reward function must accurately reflect the balance between scheduling efficiency and QoS. In this study, the reward function needs to consider not only the makespan during the scheduling process but also the response time of the workflow. Specifically, the reward function can be defined as a composite performance metric, which incorporates the following aspects: a) The shorter the time required to complete the scheduling of production tasks, the higher the overall scheduling efficiency of the system. Therefore, scheduling decisions with shorter makespan will be rewarded with higher values. b) Response time, as a key indicator of workflow performance, must also be included in the reward function. Overall, the design of the reward function can combine both makespan and response time. The specific equation can be expressed as follows:

$$E = \psi_1 * \frac{1}{LT} + \psi_2 * \frac{1}{ES_{AVE}} \tag{7}$$

where, $\psi_1 + \psi_2 = 1$. The DRL model operates primarily through the interaction between the agent and the environment, continuously adjusting its scheduling strategy to minimize the maximum makespan and reduce the response time of the workflow, thereby improving the overall service quality of data centre scheduling. The agent first selects an action based on the current state information. This action typically corresponds to the scheduling decision of a pending production task on a specific virtual machine. The state information \(*Mstate* \) and \(*Twait* \) reflect the current utilization of resources and the dependency relationships between tasks, while the action space \(*A* \) represents the possible scheduling operations. After the agent selects an action, the environment provides a reward, which is influenced not only by the current makespan and response time but also by multidimensional factors such as resource utilization. The agent, by continuously observing the environmental feedback, adjusts its decision-making strategy in order to maximize the long-term cumulative reward. This is achieved by minimizing overall scheduling time and response time through each scheduling decision.

4. EXPERIMENTAL RESULTS AND ANALYSIS

As shown in Fig. 3, under different numbers of tasks in the workflow, the optimization method proposed in this study demonstrates significant advantages in terms of makespan (the maximum time required to complete the execution of all tasks). As the number of tasks increases, the makespan value for the proposed optimization method gradually increases. However, compared to other optimization methods, it remains at a relatively lower level. For instance, when the number of tasks in the workflow is 2, the makespan of the proposed method is 550, while the First-fit method results in a makespan of 600, the Random method yields 600, the Tabu Search method provides 580, and the Bayesian Optimization method gives 575, clearly indicating the superior performance of the proposed method when the number of tasks is small. As the number of tasks increases to 7, the makespan of the proposed method reaches 1750, which is still lower than First-fit (1900), Random (1920), Tabu Search (1800), and Bayesian Optimization (1770). This demonstrates that the DRL-based optimization method can effectively reduce the maximum completion time of production tasks in large-scale workflows compared to traditional optimization methods, highlighting its potential in complex production environments.

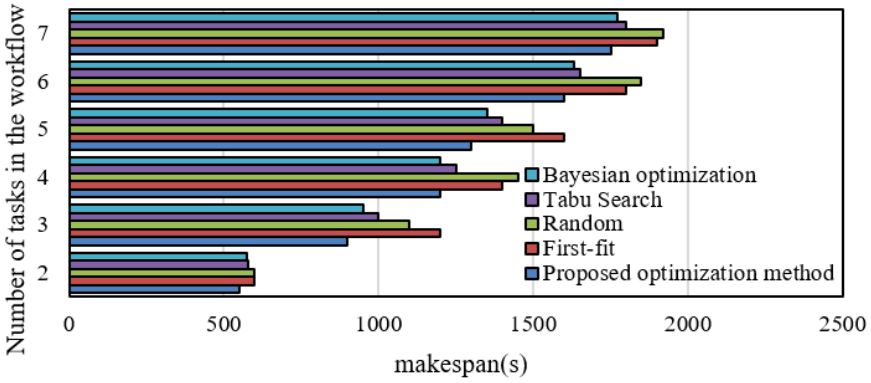

Figure 3: Changes in makespan with varying numbers of workflow tasks.

Based on the data shown in Fig. 4, it can be observed that, as the number of tasks in the workflow increases, the average response time of all optimization methods generally increases. However, the DRL optimization method proposed in this study consistently outperforms other traditional optimization algorithms across various workflow task quantities. Specifically, when the number of tasks is 2, the response time for the proposed method is 480, while the response times for First-fit, Random, Tabu Search, and Bayesian Optimization are 560, 540, 520, and 500, respectively, indicating a clear advantage of the proposed method with fewer tasks. As the

number of tasks increases, the response time for the proposed method continues to show the smallest rate of increase. When the number of tasks reaches 7, the response time for the proposed method is 880, which is lower than that of First-fit (940), Random (1080), Tabu Search (920), and Bayesian Optimization (900). This demonstrates that the DRL optimization method effectively reduces response time, even as the number of workflow tasks increases, and remains superior to other optimization methods in such dynamic production task environments.

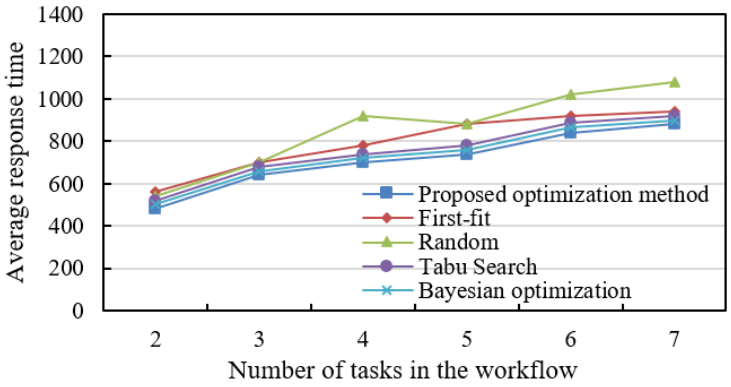

Figure 4: Changes in average response time with varying numbers of workflow tasks.

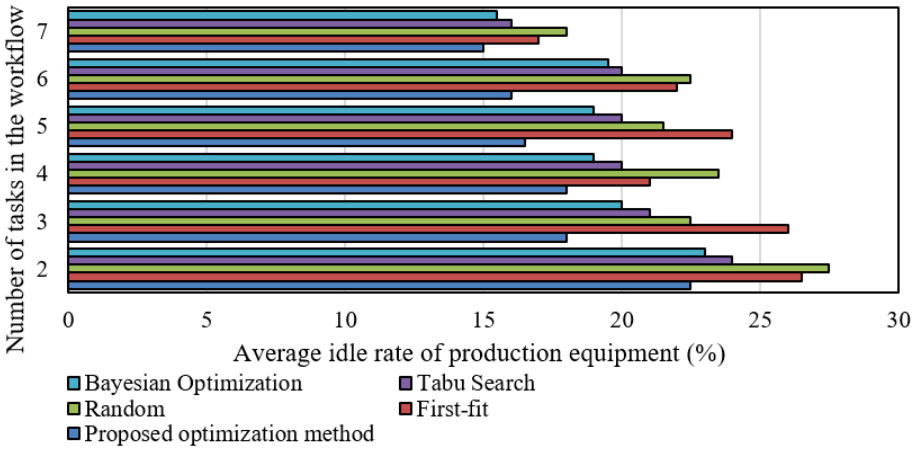

Figure 5: Average idle rate of production equipment in clusters with varying numbers of workflow tasks.

Based on the data presented in Fig. 5, it can be observed that the proposed optimization method demonstrates a significant advantage in terms of the average idle rate of production equipment across different workflow task quantities. Specifically, when the number of tasks is 2, the equipment idle rate for the proposed method is 22.5 %, which is notably lower than the rates for First-fit (26.5 %), Random (27.5 %), Tabu Search (24 %), and Bayesian Optimization (23 %). As the number of tasks increases, the idle rate for the proposed method continues to decrease gradually. When the number of tasks reaches 7, the idle rate further drops to 15 %, which is significantly lower than First-fit (17 %), Random (18 %), Tabu Search (16 %), and Bayesian Optimization (15.5 %). Overall, the proposed method maintains a consistently low equipment idle rate at all task quantities, indicating its effectiveness in improving equipment utilization during production process optimization.

According to the data presented in Fig. 6, as the number of virtual machines increases, the proposed optimization method consistently maintains the lowest makespan, demonstrating a clear advantage over traditional optimization algorithms. Specifically, when the number of virtual machines is 3, the makespan of the proposed method is 2150, which is significantly lower than the values for First-fit (2300), Random (2250), Tabu Search (2220), and Bayesian Optimization (2200). As the number of virtual machines increases to 6, the makespan of the

proposed method further decreases to 1450, still outperforming First-fit (1550), Random (1520), Tabu Search (1500), and Bayesian Optimization (1470). Overall, the optimization method proposed in this study achieves the lowest makespan for each virtual machine configuration, with the most noticeable advantage observed when fewer resources are available. This approach effectively reduces task completion time and enhances production efficiency.

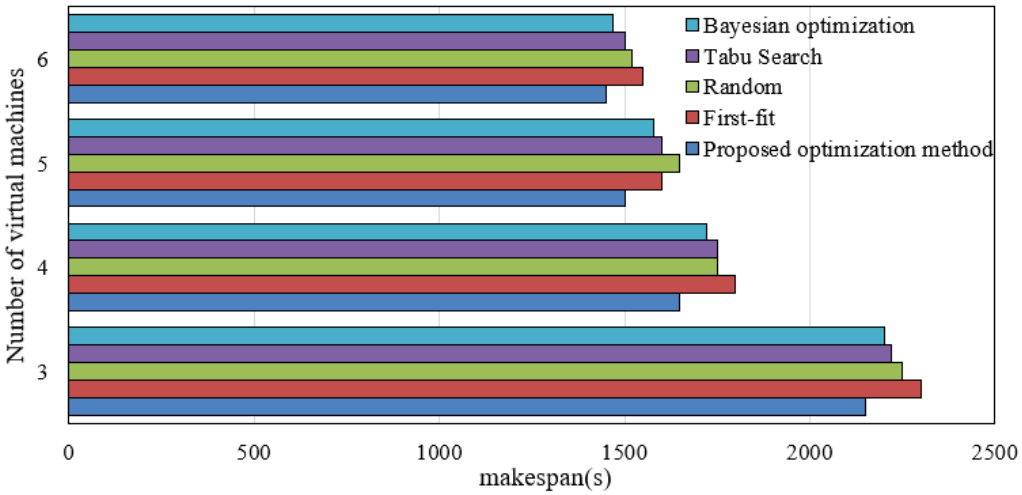

Figure 6: Makespan values of workflow scheduling with varying numbers of virtual machines.

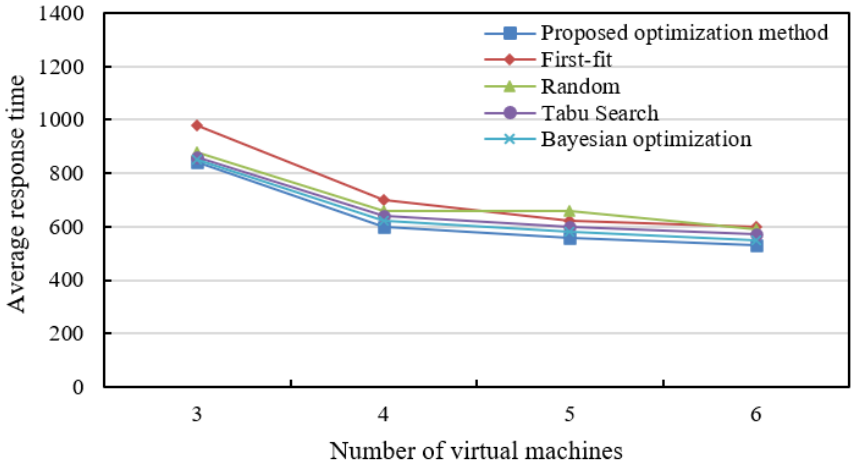

Figure 7: Average response time of workflow with increasing numbers of virtual machines.

According to the data presented in Fig. 7, as the number of virtual machines increases, the proposed optimization method consistently demonstrates a significant advantage in terms of average response time. Specifically, when the number of virtual machines is 3, the average response time of the proposed method is 840, lower than that of First-fit (980), Random (880), Tabu Search (860), and Bayesian Optimization (850). When the number of virtual machines increases to 4, the response time of the proposed method further decreases to 600, significantly outperforming other algorithms with response times of 700 (First-fit), 660 (Random), 640 (Tabu Search), and 620 (Bayesian Optimization). As the number of virtual machines increases to 5 and 6, the response times of the proposed method are 560 and 530, respectively, maintaining the lowest values and clearly outperforming First-fit (620, 600), Random (660, 590), Tabu Search (600, 570), and Bayesian Optimization (580, 550). Overall, as the number of virtual machines increases, the average response time of the proposed method continues to decrease, consistently maintaining the lowest levels and demonstrating its ability to effectively optimize production processes across different scales.

5. CONCLUSION

This study focuses on the optimization of complex production processes and proposes an optimization method that integrates DRL with the simulation model, aiming to improve the overall efficiency and adaptability of the production system. The research primarily covers two aspects: first, the development of a simulation model for complex production processes to accurately replicate various workflows and dynamic behaviours in production; and second, the design and implementation of an optimization scheme for production processes based on DRL methods. This approach seeks to address the limitations of traditional optimization methods when dealing with large-scale production tasks, such as low computational efficiency and poor adaptability. The experimental section evaluates several key performance indicators, including makespan, average response time, and equipment idle rate under varying numbers of tasks in the workflow, as well as the scheduling performance of production tasks with different numbers of virtual machines. The experimental results demonstrate that the DRL method outperforms traditional optimization algorithms (e.g., First-fit, Random, Tabu Search, and Bayesian Optimization) across all indicators. It effectively reduces the makespan and average response time of production tasks while optimizing the idle rate of cluster equipment, thus significantly enhancing the efficiency and resource utilization of the production system.

In conclusion, the DRL optimization method proposed in this study shows significant advantages in addressing the scheduling challenges of complex production processes. By combining with the simulation model, DRL can effectively perform decision optimization in dynamic and complex production environments, improving the overall performance and adaptability of the production system. The experimental results indicate that the method excels in reducing makespan, improving response time, and decreasing equipment idle rate, particularly in large-scale production tasks and complex production processes. It can better adapt to changes in resource configuration and task scheduling requirements, thus providing a solution with broad application potential.

REFERENCES

[1] Pang, J. L. (2023). Adaptive fault prediction and maintenance in production lines using deep learning, *International Journal of Simulation Modelling*, Vol. 22, No. 4, 734-745, doi:10.2507/IJSIMM22-4-CO20

[2] Ai, T.; Huang, L.; Song, R. J.; Jiao, F.; Ma, W. G. (2023). An improved deep reinforcement learning approach: a case study for optimisation of berth and yard scheduling for bulk cargo terminal, *Advances in Production Engineering & Management*, Vol. 18, No. 3, 303-316, doi:10.14743/apem2023.3.474

[3] Stevanov, B.; Sremcev, N.; Lazarevic, M.; Anderla, A.; Sladojevic, S.; Vidicki, P. (2022). Optimization of the subassembly production process using simulation, *International Journal of Simulation Modelling*, Vol. 21, No. 4, 663-674, doi:10.2507/IJSIMM21-4-633

[4] Sun, Z. Y.; Han, W. M.; Gao, L. L. (2023). Real-time scheduling for dynamic workshops with random new job insertions by using deep reinforcement learning, *Advances in Production Engineering & Management*, Vol. 18, No. 2, 137-151, doi:10.14743/apem2023.2.462

[5] Song, G.; Wang, X.; Wu, J.; Li, S.; Liu, Z.; Wang, X. (2021). Analysis of the information management system in the manufacturing process of cigarette enterprises using fuzzy AHP, *Journal of Intelligent & Fuzzy Systems*, Vol. 40, No. 4, 8257-8267, doi:10.3233/JIFS-189648

[6] Starczewska, W.; Uhl, T. (2020). Queuing models for production lines on a selected example, *Scientific Journals of the Maritime University of Szczecin*, Vol. 63, No. 135, 104-109, doi:10.17402/445

[7] Bildanov, R. G. (2021). Parametric model of the production process of radiopharmaceuticals, *Atomic Energy*, Vol. 131, No. 2, 93-96, doi:10.1007/s10512-022-00844-w

[8] Jing, Z.; Hu, N.; Song, Y.; Song, B.; Gu, C.; Pan, L. (2022). On the design and implementation of a blockchain-based data management system for ETO manufacturing, *Applied Sciences*, Vol. 12. No. 18, Paper 9184, 17 pages, doi:10.3390/app12189184
[9] Esteso, A.; Peidro, D.; Mula, J.; Díaz-Madroñero, M. (2023). Reinforcement learning applied to production planning and control, *International Journal of Production Research*, Vol. 61, No. 16, 5772-5789, doi:10.1080/00207543.2022.2104180
[10] Stranieri, F.; Fadda, E.; Stella, F. (2024). Combining deep reinforcement learning and multi-stage stochastic programming to address the supply chain inventory management problem, *International Journal of Production Economics*, Vol. 268, Paper 109099, 13 pages, doi:10.1016/j.ijpe.2023.109099
[11] Bharadwaj, D.; Dutt, D. (2022). Simulation of reinforcement learning algorithm for motion control of an autonomous humanoid, *Journal Européen des Systèmes Automatisés*, Vol. 55, No. 5, 679-685, doi:10.18280/jesa.550514
[12] Arishi, A.; Krishnan, K. (2023). A multi-agent deep reinforcement learning approach for solving the multi-depot vehicle routing problem, *Journal of Management Analytics*, Vol. 10, No. 3, 493-515, doi:10.1080/23270012.2023.2229842
[13] Sakr, A. H.; Aboelhassan, A.; Yacout, S.; Bassetto, S. (2023). Simulation and deep reinforcement learning for adaptive dispatching in semiconductor manufacturing systems, *Journal of Intelligent Manufacturing*, Vol. 34, No. 3, 1311-1324, doi:10.1007/s10845-021-01851-7
[14] Sharma, R.; Singh, I.; Prateek, M.; Pasricha, A. (2020). Comparative study of learning and execution of bipedal by using forgetting mechanism in reinforcement learning algorithm, *Journal Européen des Systèmes Automatisés*, Vol. 53, No. 3, 335-343, doi:10.18280/jesa.530304
[15] Panzer, M.; Bender, B. (2022). Deep reinforcement learning in production systems: a systematic literature review, *International Journal of Production Research*, Vol. 60, No. 13, 4316-4341, doi:10.1080/00207543.2021.1973138
[16] Dreher, A.; Bexten, T.; Sieker, T.; Lehna, M.; Schütt, J.; Scholz, C.; Wirsum, M. (2022). AI agents envisioning the future: forecast-based operation of renewable energy storage systems using hydrogen with deep reinforcement learning, *Energy Conversion and Management*, Vol. 258, Paper 115401, 19 pages, doi:10.1016/j.enconman.2022.115401
[17] Yun, L.; Wang, D.; Li, L. (2023). Explainable multi-agent deep reinforcement learning for real-time demand response towards sustainable manufacturing, *Applied Energy*, Vol. 347, Paper 121324, 12 pages, doi:10.1016/j.apenergy.2023.121324
[18] Chang, J.; Yu, D.; Hu, Y.; He, W.; Yu, H. (2022). Deep reinforcement learning for dynamic flexible job shop scheduling with random job arrival, *Processes*, Vol. 10, No. 4, Paper 760, 20 pages, doi:10.3390/pr10040760
[19] Zhang, M.; Lu, Y.; Hu, Y.; Amaitik, N.; Xu, Y. (2022). Dynamic scheduling method for job-shop manufacturing systems by deep reinforcement learning with proximal policy optimization, *Sustainability*, Vol. 14, No. 9, Paper 5177, 16 pages, doi:10.3390/su14095177
[20] Huang, H.; Tan, X. (2021). Application of reinforcement learning algorithm in delivery order system under supply chain environment, *Mobile Information Systems*, Vol. 2021, Paper 5880795, 11 pages, doi:10.1155/2021/5880795
[21] Yang, L.; Sathishkumar, V. E.; Manickam, A. (2023). Information retrieval and optimization in distribution and logistics management using deep reinforcement learning, *International Journal of Information Systems and Supply Chain Management*, Vol. 16, No. 1, 1-19, doi:10.4018/IJISSCM.316166