Proceedings of the 2024 Winter Simulation Conference
H. Lam, E. Azar, D. Batur, S. Gao, W. Xie, S. R. Hunter, and M. D. Rossetti, eds.

DEEP REINFORCEMENT LEARNING FOR SETUP AND TARDINESS MINIMIZATION IN PARALLEL MACHINE SCHEDULING

SoHyun Nam[1], Jiwon Baek[1], Young-In Cho[1], and Jong Hun Woo[1,2]

[1]Dept. of Naval Architecture and Ocean Engineering, Seoul National University, Seoul, KOREA
[2]Research Institute of Marine Systems Engineering, Seoul National University, Seoul, KOREA

ABSTRACT

This paper introduces a novel deep reinforcement learning algorithm for the identical parallel machine scheduling problem, with a focus on minimizing setup time and tardiness. In modern manufacturing environments, accommodating diverse consumer demands has emerged as a primary challenge, shifting focus towards small-batch, multi-product production schemes. However, traditional optimization techniques pose challenges due to the inherent complexity of production environments and uncertainties. Recently, reinforcement learning has emerged as a promising alternative for scheduling in such dynamic environments. We formulate the scheduling problem as a Markov decision process, with states designed to capture the key performance indicators related to setup and tardiness. The algorithm's actions correspond to selecting one heuristic rules among SSPT, ATCS, MDD, or COVERT. During the learning phase, we employ the Proximal Policy Optimization (PPO) algorithm. Experimental results on a custom dataset demonstrate superior performance compared to individual applications of existing heuristics.

1 INTRODUCTION

The current manufacturing environment is undergoing rapid changes. Consumer needs have shifted from mass production to small-batch production of multiple varieties, leading to increased complexity in production environments. Additionally, environmental concerns have emerged as major issues, and the previous strategy of mass production and stockpiling inventory is found no longer sustainable. Under these circumstances, modern flexible manufacturing systems (FMS) have emerged. FMS aims to enable the production of various types of products on a single type of equipment with minimal setup changes, even when facing various product specification changes. However, the increased diversity of products often poses additional challenges compared to traditional mass production systems. Fortunately, advancements in computing technology have facilitated the tracking and optimization of production and logistics facilities, enabling the management and control of production events at the level of individual products.

Scheduling can contribute to achieving the ultimate goal of modern manufacturing system, by reducing waste, waiting times, and inefficiencies in inventory and logistics, and maximizing the advantages of flexible manufacturing systems. Owing to these potential contributions, scheduling has gained an enormous interest and has been studied for decades. Mokotoff (2001) classified Scheduling problem-solving methodologies into three categories: exact methods, where the entire solution space is explored to prove that no better solution exists; enumerative methods, where the computational complexity is bounded by an exponential function of the input size; and approximation algorithms, which offer suboptimal approximated solutions within the limitation of polynomial computational resources.

One of the most basic types of scheduling problems is the Parallel Machine Scheduling Problem (PMSP), where there are n jobs and m machines, and each job requires specified processing time to be completed. This paper aims to solve the Identical Parallel Machine Scheduling Problem (IPMSP), where each job has the same processing time on each machine. IPMSP represents a fundamental case in manufacturing environments, where multiple machines are placed in a single workstation to avoid bottlenecks and increase

efficiency. Scheduling in IPMSP can be interpreted in two ways: as a routing problem, determining the assignment of jobs to machines, or as a sequencing problem, deciding the order in which jobs are processed on each machine. It is noteworthy that the Identical Parallel Machine Scheduling Problem is NP-complete.

The most popular objective function in PMSP is the makespan, which means the time when all jobs are completed. In real-world manufacturing systems, however, considering other objective functions is often required. This study focuses on setup time and tardiness as objective functions. Minimizing setup time is crucial to maximize the advantage of having different jobs processed on the same machine in flexible manufacturing systems. Moreover, in project industries where various components assemble into a larger product through multiple consecutive assembly processes, timeliness is crucial due to tight assembly schedules and strict due dates, since delays can significantly impact overall progress. This study develops a robust scheduling algorithm by combining discrete event simulation environment and reinforcement learning algorithm to dynamically react to changes in setup and tardiness at each decision time step.

The remainder of this paper proceeds as follows: relevant literature review will be conducted in Section 2, and the IPMSP problem will be described in Section 3. In Section 4, the proposed reinforcement learning framework will be introduced, with the detailed descriptions of the elements such as the Markov Decision Process(MDP) and learning agent that constitute the algorithm. Comprehensive experimental results and framework evaluation will follow in Section 5. Finally, conclusions will be drawn in Section 6.

2 LITERATURE SURVEY

The PMSP is a combinatorial optimization challenge that has garnered significant attention from researchers over several decades. Classical PMSP primarily is aimed at minimizing the maximum completion time of jobs, known as makespan. However, recent research has expanded to multi-objective optimization considering objectives such as tardiness, earliness, and cost. Moreover, to better simulate real production environments, studies have begun to incorporate various constraints such as job preemption, machine breakdowns, and jobs with assigned release and due dates simultaneously.

Schutten and Leussink (1996) conducted research on minimizing lateness under situations with different release and due dates using branch and bound algorithms. Mokotoff (2004) proposed a linear programming-based exact algorithm that relaxes the problem into a partial linear description from the problem's topological structure, obtaining an integer solution to minimize makespan. Additionally, Sun and Wang (2003) developed a dynamic programming algorithm and proposed two heuristics to minimize total weighted earliness and tardiness. These researches attempted to find an exact solution to the PMSP. However, Hiraishi et al. (2002) demonstrated that while PMSP instances with exact due dates for all jobs can be solved in polynomial time, the problem becomes NP-hard even without setup if due date tolerance exists. Due to the inherent difficulty of these problems, several approximation algorithms have been studied. Lee (2018) developed an effective dispatching heuristic and used an iterated greedy-based metaheuristic to improve solutions obtained, minimizing total tardiness. The aforementioned studies focused on IPMSP, where job processing times are independent of machines. The Unrelated Parallel Machine Scheduling Problem (UPMSP) is an extension of PMSP where job processing times on each machine are not identical and has also been extensively studied. Kim et al. (2002) used simulated annealing to minimize maximum tardiness in UPMSP with sequence-dependent setup times. Chen and Chen (2009) applied a hybrid metaheuristic, combining variable neighborhood descent and tabu search, to minimize the weighted number of tardy jobs in UPMSP.

In recent years, there has been recognition of the need for planning in dynamic environments where new jobs are continuously released. The emergence of Reinforcement Learning (RL) has enabled approximation algorithms to address such challenges. Yuan et al. (2013) attempted Q-learning to minimize lateness and the ratio of tardy jobs. Li et al. (2023) utilized RNN and PPO algorithms to minimize total tardiness in IPMSP problems with due dates and family setups by designing an efficient RNN-based two-stage learning process.

Moreover, research combining unrelated machine scheduling problems with reinforcement learning has also seen considerable progress. Zhang et al. (2012) attempted off-policy function approximation-based RL to minimize weighted tardiness, representing scheduling problems as semi-MDPs and training RL agents to

select actions among various dispatching heuristics. Paeng et al. (2021) studied DQN algorithms to minimize total tardiness in UPMSP problems with sequence-dependent family setups, considering sequence-dependent setup times between different types of jobs as a key constraint. Julaiti et al. (2022) developed a DDPG approach to minimize weighted tardiness, considering setup due to job type changes and stochastic machine breakdowns and improving learning efficiency through a sampling approach. One industrial application is found in Nam et al. (2023), where Deep Reinforcement Learning(DRL) was implemented to optimize the input sequence of steel bars in shipyard. The welding line for steel section bars was modeled as parallel machine shop, and both the setup time and tardiness were considered.

To the best of our knowledge, it can be concluded that there is a lack of research of general IPMSP with multiple objectives and the implementation of reinforcement learning simultaneously. Therefore, based on the previous research of Nam et al. (2023), we aim to suggest a generalized methodology for IPMSP with sequence dependent family setup time and tardiness objectives. The main contributions of this paper can be summarized as follows:

- Two objective functions, setup and minimizing tardiness are considered.
- A new feature representation to achieve multi-objective optimization is proposed.
- A dynamic manufacturing environment with release and due dates is implemented using discrete event simulation for learning.

3 PROBLEM DESCRIPTION

3.1 Assumptions

The formulation of the Identical Parallel Machine Scheduling Problem (IPMSP) with distinct job release dates and due dates is as follows:

- n independent jobs $J = \{J_1, J_2, \cdots, J_n\}$ are processed across m machines $M = \{m_1, m_2, \cdots, m_m\}$.
- Each machine can process only one job at a time, and once a job has started, it cannot be interrupted.
- Each job is completed once it is processed on a single machine.
- All jobs have different release dates and due dates.
- Each job, denoted as J_i, has a processing time of p_i, which remains consistent regardless of the machine it is executed on.
- Each job J_i is assigned a feature θ_j within l different features, and a setup is required for a machine to switch between jobs of different features.
- The setup time is determined by the difference between job features in consecutive order.
- The objective is to minimize both setup time and total tardiness.

3.2 Objectives

In practical scenarios, setup time may be either fixed or determined based on the similarity between different products. In this problem, the setup time $\sigma_{p,q}$ between any two consecutive jobs of job feature θ_p and θ_q is defined as the difference in their feature values, generating the following equation (1):

$$\text{Setup time } \sigma_{p,q} = |\theta_p - \theta_q| \tag{1}$$

The tardiness of a job J_i is defined as the difference between the completion time C_i and the predetermined due date d_i, only if the completion time exceeds the due date as shown in (2).

$$\text{Tardiness } T_i = \max((C_i - d_i), 0) \tag{2}$$

Considering the weighted sum, the objective of the proposed problem can be described as (3).

$$\text{minimize } \sum_{i=1}^{n} w_t \times \max((C_i - d_i), 0) + w_s \times |\theta_{J_i} - \theta_{J_{i-1}}| \tag{3}$$

3.3 Notations

Indices	
i	Index of jobs
j	Index of job features
k	Index of machines
Variables	
J_i	ith job in the set of jobs $J = \{J_1, J_2, \cdots, J_n\}$
θ_j	jth feature of the set of job features $F = \{\theta_1, \theta_2, \cdots, \theta_l\}$
θ_{J_i}	Feature value of job J_i
m_k	kth machine in the set of machines $M = \{m_1, m_2, \cdots, m_m\}$
C_i	Completion time of job J_i
d_i	Due date of job J_i
N_q	Number of jobs waiting in queue
$\bar{p}_i$	Average processing time of job J_i
$\sigma_{p,q}$	Setup time required for machine converting from job feature θ_p to θ_q
$\sigma(J_i, J_j)$	Setup time required when machine m_k converting from job J_i to job J_j
T_i	Tardiness of job J_i
G_i	Tardiness level of job J_i, assigned among 4 values in $G = \{1,2,3,4\}$

4 METHODOLOGY

In the following section, the process of applying DRL to optimize the suggested problem will be presented. To implement DRL, the first step is to define the Markov Decision Process (MDP). Subsequently, it is crucial to design the actions and rewards appropriately to achieve the objectives. Finally, an efficient learning algorithm for training is necessary.

4.1 Markov Decision Process

The fundamental concept of RL is based on the idea that decisions made within an environment can be modeled as a Markov Decision Process (MDP). At any arbitrary time step t, the state of the environment can be represented as s_t. After the agent takes action a_t, the environment transitions to a new state s_{t+1}, according to the transition function. The agent receives a reward r_t depending on how desirable the state is. The probability of the agent taking a specific action a_t at state s_t is represented by the probability distribution $\pi\{a_t|s_t\}$. The objective of Reinforcement Learning, based on policy gradient methods, is to express the degree of desirability of states as rewards in order to achieve the objective for a given problem, and to identify an appropriate $\pi\{a_t|s_t\}$ that maximizes this reward. Properly modeling states, actions, and rewards is essential for optimization. In our research, the state, action, and reward representations are designed based on the MDP formulations proposed in the earlier industrial application research by Nam et al. (2023).

4.1.1 Action

When scheduling n jobs across m machines, the solution space becomes excessively large due to the need to determine both the job order and machine assignments. To efficiently reduce the action space, this paper proposes selecting one of four priority rules for actions: Shortest Processing Time and Shortest Setup Time (SSPT), Apparent Tardiness Cost with Setup (ATCS), Modified Due Date (MDD), and Cost OVER Time (COVERT). When any machine becomes idle after completing a job, a calling event occurs, requesting the next job to be processed, and one of the four actions is chosen.

SSPT is a priority rule that considers both the processing time and the setup time. It prioritizes jobs with shorter average processing time $\bar{p}_i$ and shorter setup time $\sigma(J_{i-1},J_i)$. For all jobs remaining in the queue at time t(denoted as $Q(t)$), the job J_i is assigned to machine m_k based on equation (4).

$$J_i = \text{ argmin } \{\bar{p}_i + \sigma(J_{i-1},J_i)\}, \forall J_i \in Q(t) \tag{4}$$

The ATCS is an adaptation of the Apparent Tardiness Cost (ATC) rule initially proposed by Lee and Pinedo (1997) for minimizing total weighted tardiness. The rule integrates the goal of reducing setup times alongside with the goal of minimizing tardiness. This rule grants higher priority to jobs with shorter processing times, reduced slack time, and shorter setup times. Selection of job J_i for machine m_k after processing job J_j follows the equation (5). The parameters, k_1 and k_2 are calculated by the method proposed by Lee and Pinedo (1997).

$$\begin{aligned} J_i &= \text{ argmax } \{\frac{1}{\bar{p}_i} \times \exp\left(-\frac{\max(d_i-\bar{p}_i-t,0)}{k_1\bar{p}}\right)\exp\left(-\frac{\sigma(J_j,J_i)}{k_2\bar{\sigma}}\right), \forall J_i \in Q(t) \\ \bar{p} &= \frac{1}{N_q}\sum_{i=1}^{N_q}\bar{p}_i, \\ \bar{\sigma} &= \frac{1}{N_q}\sum_{i=1}^{N_q}\sigma(J_j,J_i) \end{aligned} \tag{5}$$

The MDD rule is a combination of the Earliest Due Date (EDD) rule and the Shortest Remaining Processing Time (SRPT) rule. This rule determines the modified due date by selecting the larger value between the original due date and the expected completion time if the job were to start immediately. Then it selects the job that minimizes this modified due date, as illustrated in equation (6).

$$J_i = \text{ argmin } \left(\max\left(d_i, t+\bar{p}_i\right)\right), \forall J_i \in Q(t) \tag{6}$$

The COVERT rule is a priority rule that considers both the processing time and the associated cost. In the COVERT rule, jobs are ranked according to a cost-to-time ratio, where the cost represents the importance or urgency associated with each job. The rule aims to select the job that maximizes the value of cost divided by processing time, thereby prioritizing jobs based on their value or importance relative to their processing time. In practice, it is calculated as shown in equation (7), with the assumption that a smaller ratio of slack time to the processing time of the job indicates a better schedule. Tardy jobs are given the highest priority to ensure they are assigned to machines first. The parameter k was selected after a case study in which the candidates varied in the range of $(0.1,20)$.

$$J_i = \text{ argmax } \left(\frac{1}{\bar{p}_i}\max\left(1-\frac{\max(d_i-\bar{p}_i-t,0)}{k\bar{p}_i},0\right),0\right), \forall J_i \in Q(t) \tag{7}$$

4.1.2 State

The state serves as an numerical interpretation of the current environment, enabling the agent to utilize it for decision-making while representing how desirable the current state is. To ensure efficient convergence of the algorithm, it is recommended to leverage domain knowledge and represent states as combinations of relevant features. A concatenation of S_1, S_2, S_3, and S_4, which represent various features related to the objectives of setup and tardiness. Detailed descriptions and mathematical formulations are as follows:

$S_1 = \{S_{1,1}, \cdots, S_{1,m}\}$ is a vector of size m, where $S_{1,k}$ represents the proportion of jobs in the current queue that can be processed on machine m_k without any additional setup changes. It is calculated as the ratio of the number of jobs with the same job feature of formerly processed job on machine m_k (denoted as $N_{1,k}$) to the total number of jobs in the queue $N_{1,q}$, as illustrated in equation (8).

$$S_{1,k} = \begin{cases} \frac{N_{1,k}}{N_{1,q}} & \text{on initial state where there are no jobs in the queue} \\ 1 & \text{otherwise} \end{cases} \tag{8}$$

S_2 and S_3 refer to the number of jobs corresponding to a specific tardiness level. These values are based on the tightness metric proposed by Zhang et al. (2012). S_2 focuses on jobs in the queue with same job feature as the calling machine m_k, while S_3 focuses on jobs in the queue with the different job features. The jobs in the queue are divided into two categories, one having the same setup feature as the formerly processed job on m_k, and the other having a different setup feature from the former job on m_k. The number of jobs in the queue that belongs to each category are denoted as $N_{2,k}$ and $N_{3,k}$, respectively.

$$\begin{aligned} N_{2,k} &: \text{\# of jobs having the job feature same as the former job on } m_k \\ N_{3,k} &: \text{\# of jobs having job features different from the former job on } m_k \end{aligned} \tag{9}$$

Then, each category splits the jobs into four levels according to the tardiness level $g \in \{G_l | l = 1,2,3,4\} = \{G_1, G_2, G_3, G_4\}$, thereby generating total 8 values in the form of $N_{j,k,l} (j = 2 \text{ or } 3)$. The criteria for division is the maximum and minimum estimated processing time of the jobs considering the variance factor δ_{pt}, and thus calculated according to equation (10). The mathematical formulation of the classification process is demonstrated in equation (11). G_1 refers to the level where the remaining time until the due date from the current time t exceeds the maximum processing time, and G_2 corresponds to cases where the remaining time until the due date is less than the maximum processing time but greater than the minimum processing time. G_3 indicates situations where the remaining time is equal to or less than the minimum processing time, meaning that even if the job starts immediately, it is unavoidable to be tardy. G_4 denotes jobs that have already passed their due dates at the current time point.

$$\begin{aligned} &\text{Maximum estimated processing time } r_{i,\max} = (1+\delta_{pt}) \times \bar{p}_i \\ &\text{Minimum estimated processing time } r_{i,\min} = (1-\delta_{pt}) \times \bar{p}_i \end{aligned} \tag{10}$$

$$\text{Tardiness level of job } J_i = G(J_i) = \begin{cases} G_1 & \text{if } (d_i - t) \in (r_{i,\max}, +\infty) \\ G_2 & \text{if } (d_i - t) \in (r_{i,\min}, r_{i,\max}] \\ G_3 & \text{if } (d_i - t) \in (0, r_{i,\min}] \\ G_4 & \text{if } (d_i - t) \in (-\infty, 0] \end{cases} \tag{11}$$

Let $N_{j,k,l} (j = 2 \text{ or } 3)$ be the number of jobs with the tardiness level g and setup status k. The state feature $S_{2,k,l}$ and $S_{3,k,l}$ is calculated by equation (12).

$$\begin{aligned} N_{2,k,l} &= \text{\# of jobs with the same setup status as the former job on } m_k \text{ \& tardiness level } G_l \\ N_{3,k,l} &= \text{\# of jobs with different setup statuses from the former job on } m_k \text{ \& tardiness level } G_l \\ S_{j,k,l} &= \frac{N_{j,k,l}}{N_{j,k,l}} \quad (G_l \in \{G_1, G_2, G_3, G_4\}) \end{aligned} \tag{12}$$

S_4 is a vector of size m, each element $S_{4,k}$ representing the progress ratio of the job on machine m_k. It refers to the remaining processing time compared to the total processing time of the job currently being processed on the machine m_k, normalized to a value between 0 and 1, as in equation (13).

$$S_{4,k} = \begin{cases} \frac{(t_s)_i - p_i - t}{\bar{p}_i} & \text{If } m_k \text{ is working job } J_i, \text{ where } (t_s)_i = \text{starting time of job } J_i \\ 0 & \text{If } m_k \text{ is idle} \end{cases} \tag{13}$$

The number of features is thus given as $2m+8$. The state features are concatenated into a 1-dimensional vector and given as input for the reinforcement learning agent.

4.1.3 Reward

In reinforcement learning, the reward serves as a direct and immediate reinforcement for the agent's actions, acting as a proxy for indirectly estimating the desirability of a state and providing strong incentives for the agent to take good actions. In this study, the reward was designed based on the formulation of Nam et al. (2023) to achieve multi-objective optimization of minimizing both tardiness and setup time. Since each action involves deciding which job to process on an idle machine, the reward is intended to be the sum of the tardiness reward r_t and the setup time reward r_s, calculated as in equation (14). To prevent from a single objective disproportionately affecting the overall reward, r_t and r_s are appropriately scaled. The tardiness reward is transformed according to an exponential equation, where 1 indicates not being tardy, and values close to 0 indicate increasing tardiness. The setup time reward indicates how much setup time is incurred by selecting a particular job compared to the possible maximum setup time case, where 0 indicates having no setup time while -1 represents having maximum setup time.

$$\begin{aligned} &\text{Tardiness reward for job } J_i : r_{t,i} = \exp\{-(C_i - d_i)\} \\ &\text{Setup time reward for job } J_i : r_{s,i} = \frac{\sigma(J_{i-1}, J_i)}{\max\sigma(J_i, J_j)}, \quad \forall J_j \in J = \{J_1, J_2, \cdots, J_n\} \\ &\text{Weighted reward } r_i = 0.6 \times r_{t,i} + 0.4 \times r_{s,i} \end{aligned} \tag{14}$$

4.1.4 State Transition

To effectively capture the essential features of a flexible manufacturing system, this study employed Discrete Event Simulation (DES) using the open-source Python library SimPy. Figure 1 illustrates the overall learning process. Whenever a job is completed processing on the machine, a calling event occurs and the agent receives s_t as input. The deep neural network of the agent generates a probability distribution for selecting each action. When one of the four priority rules is chosen, the state transition through the next state occurs automatically according to the internal logic of the DES environment.

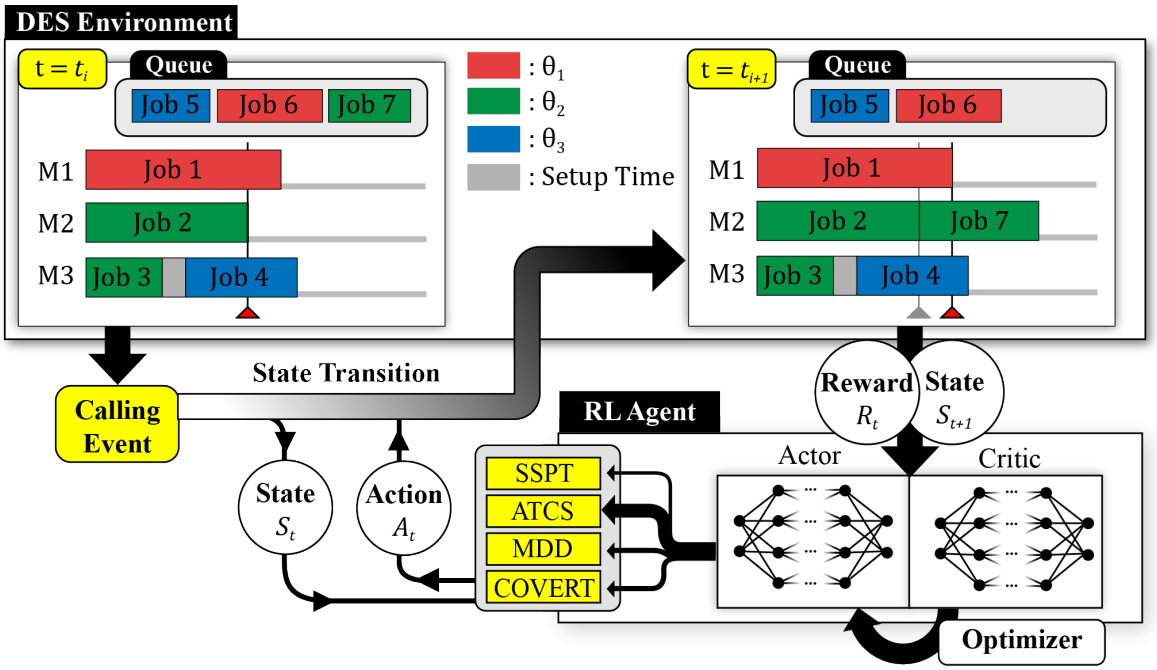

Figure 1: Overview of the learning process.

In the illustrated example, the calling event occurs after Job 2 completes processing on M2. Job 7 is assigned the highest priority rank in the queue by the selected action, and thus assigned to M2. The next timestep is determined by the earliest occurrence of another machine becoming idle, as calculated by the internal logic of the DES environment. State s_{t+1} and reward r_t are provided to the agent through forwarding to the next step. Throughout this process, a set of $\{s_t, a_t, r_t, s_{t+1}\}$ accumulates as experience. Finally, the agent computes the loss function using sampled experience sets and the learning progresses through AdaHessian optimizer.

4.2 Learning Algorithm

For the learning algorithm, the Proximal Policy Optimization (PPO) learning algorithm proposed by Schulman et al. (2017) was adopted. The PPO algorithm was selected for showing its effectiveness in many scheduling problem researches, including the work of Li et al. (2023), Cho et al. (2022), and Oh et al. (2022).

The PPO algorithm belongs to the policy gradient family of algorithms, which takes state features and directly outputs the probability of each action using a deep neural network. This network is optimized using the surrogate loss function L. The expression for L is given in equation (15), where A is the advantage providing a pure marginal reward gained from the agent's action, regardless of the previous state.

Since it is impossible to directly compute the value function within the deep neural network, the generalized advantage estimation $\hat{A}$ was employed, as shown in equation (16) where T, δ_t, γ, and λ indicate the time horizon, the TD advantage estimate for time step t, the discount factor, and the exponential weight discount, respectively. To prevent excessive parameter fluctuations during training, clipping is introduced into the loss function, preventing abrupt updates beyond a certain threshold and thus ensuring a more stable convergence. The pseudocode of the entire algorithm is presented below, including updates for both the actor and critic.

$$L(\theta) = \hat{E}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}\left(r_t(\theta), 1-\varepsilon, 1+\varepsilon\right)\hat{A}_t \right) \right] \tag{15}$$

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + (\gamma\lambda)^{T-t+1}\delta_{T-1} \tag{16}$$

5 EXPERIMENTS

5.1 Problem Instances

The problem instances generated in this study represent a flexible manufacturing system consisting of 100 jobs and 5 machines. The processing time of each job follows a uniform distribution in the range of [10, 20], and the job type feature is assigned as a random integer ranging from 0 to 5. The interarrival time of jobs in the DES environment was set as the mean processing time divided by the number of machines in order to prevent the system explosion. The due date tightness τ, indicating that the job's due date margin with respect to its processing time, varied by problem instances. The processing time variability factor, denoted as δ_{pt}, set to values between 0.2 and 0.5, allowed the job processing time to be stochastically determined within a range of $(1-\delta_{pt}, 1+\delta_{pt})$. The parameters of the DES were selected arbitrarily, as the objective of this study is to demonstrate the generalization ability of the suggested methodology through its application to a mathematical exercise, rather than solving a specific industrial problem.

5.2 Network Structures

The actor and critic networks of the PPO agent utilized in this study consist of a DNN structure with a total of 5 layers. The first layer accepts $2m+8$ input nodes. The subsequent 3 hidden layers have 512, 512, and 256 nodes, respectively, employing the Rectified Linear Unit (ReLU) activation function. The final layer comprises 4 nodes, each outputting the probability distribution for selecting one of the four heuristics. Additionally, the hyperparameters for PPO were set to $K = 1, \alpha = 0.0001, \varepsilon = 0.2$, and $\gamma = 0.98$, determined through a grid search process during training to achieve optimal performance.

Algorithm 1 Pseudocode for Reinforcement Learning with PPO

Initialize weights θ of the actor network π_θ and weights θ_v of the critic network V_{θ_v}
for episode $= 1, \ldots, E$ **do**

[Data Sampling]
  get initial state s_0
  repeat
    calculate the probabilities $\pi_{\theta_{\text{old}}}(\cdot|s_t)$ for all actions
    select an action a_t according to $\pi_{\theta_{\text{old}}}(\cdot|s_t)$
    execute action a_t in DES simulator and observe reward r_t and next state s_{t+1}
    put sample (s_t, a_t, r_t, s_{t+1}) into trajectory set D
    $t \leftarrow t+1$ and $s_t \leftarrow s_{t+1}$
  until s_{t+1} is terminal

[Policy Optimization]
  for epoch $= 1, \ldots, K$ **do**
    calculate the generalized advantage estimates $\hat{A}_i$ for each sample in D

$$\hat{A}_i = \delta_i + (\gamma\lambda)\delta_{i+1} + \cdots + (\gamma\lambda)^{T-i+1}\delta_{T-1}$$

$$\text{where } \delta_i = \begin{cases} r_i + \gamma V_{\theta_v}(s_{i+1}) - V_{\theta_v}(s_i) & \text{if } s_{i+1} \text{ is non-terminal} \\ r_i - V_{\theta_v}(s_i) & \text{otherwise} \end{cases}$$

    calculate loss function L_p for the actor network

$$L_p = -\frac{1}{|D|}\sum_{i=1}^{|D|} \min(r_i(\theta)\hat{A}_i, \text{clip}(r_i(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_i) \text{ where } r_i(\theta) = \frac{\pi_\theta(a_i|s_i)}{\pi_{\theta_{\text{old}}}(a_i|s_i)}$$

    calculate loss function L_v for the critic network

$$L_v = \frac{1}{|D|}\sum_{i=1}^{|D|} \delta_i^2$$

    optimize θ, θ_v using Adam optimizer with loss function $L = L_p + \beta L_v$ and learning rate α
  end for
  empty the trajectory set D
end for

5.3 Result

After about 500 epochs of training, the reward converged to a certain level, and a comparison test was conducted on 100 test instances. The overall performance of the RL agent compared to four other heuristics is shown in Table 1 and Figure 2. RL demonstrated superior performance compared to other heuristic methods. Although the fluctuation of the test instances resulted in RL showing slightly worse performance than the ATCS heuristic in terms of mean tardiness, the overall tardiness level improved across the test instances when considering the median value of mean tardiness. RL exhibited approximately a 5% reduction in mean setup time compared to the next best performing heuristic, ATCS. After analyzing the stochastic policy over a single problem instance, it was revealed that the probability of agent choosing SSPT and ATCS significantly differed based on the number of tardy jobs currently occupying the queue. The stochastic policy over the entire time horizon is illustrated in Figure 3. This suggests that the agent takes the current status of tardiness into consideration when calculating the stochastic policy, preventing a single superior heuristic from dominating the policy regardless of the state features.

Table 1: Performance comparison between RL agent and heuristic rules.

		RL	ATCS	MDD	SSPT	COVERT
Tardiness $\bar{T}$	Mean	16.71	15.79	22.95	18.99	20.88
	Median	15.00	15.40	21.52	17.91	19.91
Setup $\bar{\sigma}$	Mean	0.93	0.97	1.97	1.37	1.94
	Median	0.92	0.96	1.95	1.35	1.95

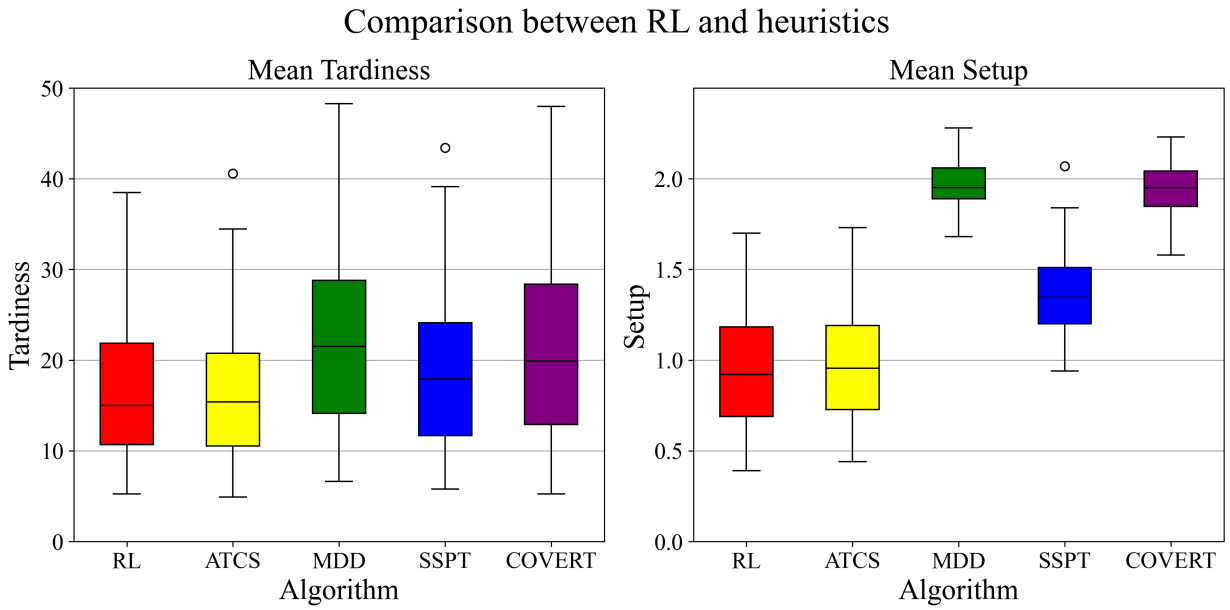

Figure 2: Mean tardiness and setup of RL and heuristics after 100 instances.

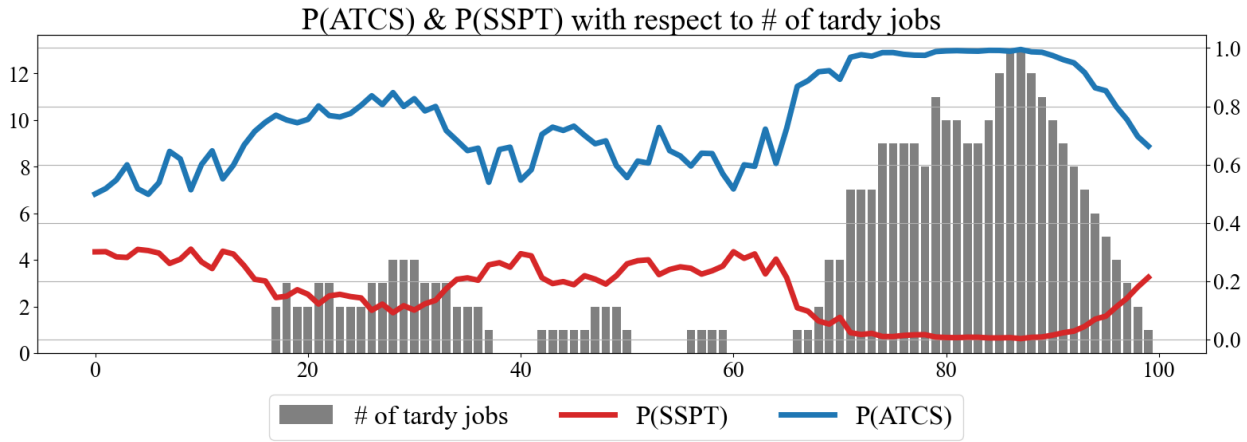

Figure 3: Probability Difference between heuristics with respect to number of tardy jobs.

6 CONCLUSION

This study introduces a reinforcement learning framework based on discrete event simulation for the identical parallel machine scheduling problem. Results comparing the proposed framework with four existing

heuristics demonstrate that the RL agent outperforms the others. The significance of this study lies in training an RL agent for scheduling in dynamic manufacturing systems which can promptly respond at any time when given the state features. Additionally, it is important to note that the proposed method effectively trained an RL agent that considers both setup time and tardiness as simultaneous objectives in scheduling. Lastly, by constructing a discrete event simulation model, this study enabled immediate tracking of KPIs (key performance indicators) in the production environment, thereby facilitating the learning process and allowing for easy expansion of the algorithm through the collection of state features for learning.

The result of this study is applicable to various types of problem instances and modified FMS environments. By creating environments where parameters such as job and machine numbers, due date tightness, and processing time variability factor are determined stochastically, a more realistic simulation of FMS environments can be achieved. Given the stochastic characteristics as the state feature, it is expected that the agent is able to proactively utilize the information for scheduling. Additionally, considering setup time determined as a more complex function of job features presents a possible avenue for future research, and the impact of such modified environments on scheduling algorithms needs to be evaluated further.

REFERENCES

Chen, C.-L., and C.-L. Chen. 2009. "Hybrid Metaheuristics for Unrelated Parallel Machine Scheduling with Sequence-Dependent Setup Times". *The International Journal of Advanced Manufacturing Technology* 43:161–169.

Cho, Y. I., S. H. Nam, K. Y. Cho, H. C. Yoon, and J. H. Woo. 2022. "Minimize Makespan of Permutation Flowshop Using Pointer Network". *Journal of Computational Design and Engineering* 9(1):51–67.

Hiraishi, K., E. Levner, and M. Vlach. 2002. "Scheduling of Parallel Identical Machines to Maximize the Weighted Number of Just-in-Time Jobs". *Computers & Operations Research* 29(7):841–848.

Julaiti, J., S.-C. Oh, D. Das, and S. Kumara. 2022. "Stochastic Parallel Machine Scheduling Using Reinforcement Learning". *Journal of Advanced Manufacturing and Processing* 4(4):e10119.

Kim, D.-W., K.-H. Kim, W. Jang, and F. F. Chen. 2002. "Unrelated Parallel Machine Scheduling with Setup Times Using Simulated Annealing". *Robotics and Computer-Integrated Manufacturing* 18(3-4):223–231.

Lee, C.-H. 2018. "A Dispatching Rule and a Random Iterated Greedy Metaheuristic for Identical Parallel Machine Scheduling to Minimize Total Tardiness". *International Journal of Production Research* 56(6):2292–2308.

Lee, Y. H., and M. Pinedo. 1997. "Scheduling Jobs on Parallel Machines with Sequence-Dependent Setup Times". *European Journal of Operational Research* 100(3):464–474.

Li, F., S. Lang, B. Hong, and T. Reggelin. 2023. "A Two-Stage RNN-Based Deep Reinforcement Learning Approach for Solving the Parallel Machine Scheduling Problem with Due Dates and Family Setups". *Journal of Intelligent Manufacturing*:1–34.

Mokotoff, E. 2001. "Parallel Machine Scheduling Problems: A Survey". *Asia-Pacific Journal of Operational Research* 18(2):193.

Mokotoff, E. 2004. "An Exact Algorithm for the Identical Parallel Machine Scheduling Problem". *European Journal of Operational Research* 152(3):758–769.

Nam, S.-H., Y.-I. Cho, and J. H. Woo. 2023. "Reinforcement Learning for Minimizing Tardiness and Set-Up Change in Parallel Machine Scheduling Problems for Profile Shops in Shipyard". *Journal of the Society of Naval Architects of Korea* 60(3):202–211.

Oh, S. H., Y. I. Cho, and J. H. Woo. 2022. "Distributional Reinforcement Learning with the Independent Learners for Flexible Job Shop Scheduling Problem with High Variability". *Journal of Computational Design and Engineering* 9(4):1157–1174.

Paeng, B., I.-B. Park, and J. Park. 2021. "Deep Reinforcement Learning for Minimizing Tardiness in Parallel Machine Scheduling with Sequence Dependent Family Setups". *IEEE Access* 9:101390–101401.

Schulman, J., F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. 2017. "Proximal Policy Optimization Algorithms". *arXiv Preprint arXiv:1707.06347*.

Schutten, J. M., and R. Leussink. 1996. "Parallel Machine Scheduling with Release Dates, Due Dates and Family Setup Times". *International Journal of Production Economics* 46:119–125.

Sun, H., and G. Wang. 2003. "Parallel Machine Earliness and Tardiness Scheduling with Proportional Weights". *Computers & Operations Research* 30(5):801–808.

Yuan, B., L. Wang, and Z. Jiang. 2013. "Dynamic Parallel Machine Scheduling Using the Learning Agent". In *2013 IEEE International Conference on Industrial Engineering and Engineering Management*, 1565–1569. IEEE.

Zhang, Z., L. Zheng, N. Li, W. Wang, S. Zhong, and K. Hu. 2012. "Minimizing Mean Weighted Tardiness in Unrelated Parallel Machine Scheduling with Reinforcement Learning". *Computers & Operations Research* 39(7):1315–1324.

AUTHOR BIOGRAPHIES

SOHYUN NAM earned Master from Department of Naval Architecture and Ocean Engineering at Seoul National University. Her research interest is in Discrete Event Simulation and Reinforcement learning. Her email address is sohyon525@snu.ac.kr.

JIWON BAEK is a graduate school student in the Department of Naval Architecture and Ocean Engineering at Seoul National University. She has a Bachelor of Engineering in Naval Architecture and Ocean Engineering. Her research interests include DES simulation and production scheduling optimzation. Her email address is baekjiwon@snu.ac.kr.

YOUNG-IN CHO received his B.S. degree in Naval Architecture and Ocean Engineering (NAOE) from Seoul National University (SNU) in 2020. He is currently working toward the Ph.D. degree in NAOE from SNU. His research interests include production scheduling based on deep reinforcement learning. His email address is whduddlsi@snu.ac.kr.

JONG HUN WOO is a full professor in the Department of Naval Architecture and Ocean Engineering at Seoul National University. He holds a Ph.D in Naval Architecture and Ocean Engineering from Seoul National University. His research interest is in deep reinforcement learning, DES(discrete event simulation), APS(Advanced Planning and Scheduling) and other knowledge related with industrial engineering. He has R&D experiences mainly in shipbuilding industry. His email address is j.woo@snu.ac.kr.