Article

Injection Mold Production Sustainable Scheduling Using Deep Reinforcement Learning

Seunghoon Lee ¹ , Yongju Cho ² and Young Hoon Lee ^{1,*}

¹ Department of Industrial Engineering, Yonsei University, 50 Yonsei-ro, Seodaemun-gu, Seoul 03722, Korea; shbrandonlee@yonsei.ac.kr

² Korea Institute of Industrial Technology, 89 Yangdaegiro-gil, Seobuk-gu, Cheonan-si 31056, Korea; yjcho@kitech.re.kr

* Correspondence: youngh@yonsei.ac.kr; Tel.: +82-2-2123-4813

Received: 31 August 2020; Accepted: 19 October 2020; Published: 21 October 2020

Abstract: In the injection mold industry, it is important for manufacturers to satisfy the delivery date for the products that customers order. The mold products are diverse, and each product has a different manufacturing process. Owing to the nature of mold, mold manufacturing is a complex and dynamic environment. To meet the delivery date of the customers, the scheduling of mold production is important and is required to be sustainable and intelligent even in the complicated system and dynamic situation. To address this, in this paper, deep reinforcement learning (RL) is proposed for injection mold production scheduling. Before presenting the RL algorithm, a mathematical model for the mold scheduling problem is presented, and a Markov decision process framework is proposed for RL. The deep Q-network, which is an algorithm for RL, is employed to find the scheduling policy to minimize the total weighted tardiness. The results of experiments demonstrate that the proposed deep RL method outperforms the dispatching rules that are presented for minimizing the total weighted tardiness.

Keywords: deep reinforcement learning; sustainable scheduling; mold manufacturing; DQN

1. Introduction

In Industry 4.0, manufacturing companies consider technical issues [1] in addition to economic, environmental, and social issues [2] for sustainability. By considering the technical impacts on manufacturing, the manufacturers try to satisfy customer demand and to improve the production efficiency for enhancing sustainability. Among the diverse factors for achieving their endeavor, the agility or the flexibility of manufacturing plays an important role. If the manufacturers do not cope with a dynamic manufacturing environment with agility, it may lead to customers' dissatisfaction and a decrease in the competitiveness of the manufacturers in the market. In Industry 4.0, the core of decision-making in manufacturing operations is based on information accessibility at the right time, which assists and enables the maintenance of manufacturing agility. With an increased utilization of the information, gathering and inferring industrial data, evolved intelligent production planning, scheduling, and control systems are required. Combining new technologies in Industry 4.0 with manufacturing processes, the performance of the systems is enhanced through real-time, data-driven, and continuous learning from a more varied range of data sources [3].

Among them, scheduling is a core process for manufacturing companies to maximize profits and to reduce costs at the same time. In particular, in a complex system and a dynamic manufacturing environment, more sustainable and intelligent scheduling that can respond to them is needed because poor scheduling has higher costs, a longer production time, and higher tardiness. Thus, to deal with a

manufacturing site's complexity and to improve its effectiveness, scheduling needs to be evolved and improved for sustainability and intelligence.

Injection mold (hereafter mold) manufacturing is a complex production system. Molds are used as the semifinal or final parts that can produce products for most firms. This is a special case where producing products are used in manufacturing processes for other industries such as automobiles, shoes, and electronics. As mold manufacturing plays an important role in the industry, mold manufacturing has been the focus of many studies in terms of the design process, manufacturing, testing the molds, and tuning the parameters of the molding machines [4]. These studies have focused on manufacturing to improve the quality of the manufacturing molds. Although it is important to look after the quality of the products for customer satisfaction, delivering the products on time is a matter of satisfaction for the customers as well.

The complexity of the mold manufacturing production system comes from a multi-product production system that is based on the customers. The products can be of many different types, and their processes vary from product to product. In addition, the processing time is dependent on the manufacturing line and generally ranges from a minute to hours. Therefore, scheduling to determine the allocation of jobs for machines is crucial. The demand and type of products are dependent on the customers, and satisfying the due date plays an important role in determining competitiveness in the mold industry. The mold industry is a typical multi-time, small-sized production. Some of the typical issues include frequent setup time occurrences, complicated processes, and planning for delays due to an urgent order and a dynamic environment. Therefore, developing effective and intelligent scheduling that can counteract flexibly while considering the products' due date and the diverse product types in a dynamic environment is necessary for the mold industry.

In this study, to address the complexity and dynamic environment of mold manufacturing, deep reinforcement learning (RL) is employed for the mold scheduling problem. This study developed a scheduling method that uses the deep Q-network (DQN) as the algorithm for deep RL, which is applied to the scheduling for mold manufacturing systems. The state, action, and reward are defined properly for the mold scheduling problem. In the training section, the DQN is learned during the episodes, which are iterations for the mold scheduling problem. Once the training is finished, the performance of the proposed algorithm is evaluated by using dispatching rules for minimizing the total weighted tardiness.

The remaining sections of this study are as follows. Section 2 is the literature review of the mold manufacturing industry and deep RL. In Section 3, the mathematical model is presented to explain the mold scheduling problem, the Markov decision process (MDP) framework is proposed for RL, and the algorithm of deep RL is described. Section 4 presents the result of experiments, which are implemented through a case study. The discussion and conclusion are presented in Sections 5 and 6, respectively.

2. Literature Review

2.1. Mold Manufacturing

As the mold is a core part of the diverse manufacturing industry, numerous studies have been conducted to improve production efficiency and enhance sustainability in the market. Low and Lee [5] focused on the cavity layout design system for the injection mold. The cavity is the main one among diverse parts that constitute the mold. As time-to-market for plastic products is getting shorter, the available time for mold manufacturing is decreasing. By making a template for designing the cavity layout, miscommunication was avoided between product and mold designers, and time was saved. Hu and Masood [6] exploited the intelligent cavity layout design system to assist mold designers in the cavity design step. Fu et al. [7] dealt with the parting surface, core, and cavity blocks in the computer-aided mold design system, which is the bottleneck. The architecture of the mold design system and the methodology to generate them were proposed.

Other studies addressed the layout design of the cooling system for the mold as the cooling step takes a large portion of the mold manufacturing process. The cooling system is important for the productivity of the process and the quality of the mold. Li et al. [8] developed a graph traversal algorithm to make cooling circuits from the graph structure that was devised to capture a given initial design. The heuristic search and fuzzy framework were used for developing the cooling circuits into the layout designs and evaluating the designs. Liang [9] derived the optimization model to achieve the highest transfer efficiency of the cooling system. Li et al. [10] used a topology optimization to design the conformal cooling system for the mold and simplified the analysis of the cooling process by the cycle averaged approach. The boundary element method was employed to find solutions and to calculate the sensitivities.

Some authors have addressed the perspective of energy saving, as energy consumption is related to the company's cost as well as environmental impacts. Li et al. [11] figured out the relationship between energy consumption and process parameters on mold machine tools via experimental observations in the research. With the observations, the throughput rate was found to be a critical factor for the energy and eco-efficiency of mold processes. Another study discussed the guideline to characterize the energy consumption of the mold process [12]. Five steps were presented for the guideline to calculate the estimated energy consumption. With the guideline, it could estimate and benchmark a variety of manufacturing processes and products using mold process by considering theoretical minimum energy that was computed with part design, material, and process planning, the estimation of energy consumption that was calculated with manufacturing resource information. Thus, many studies have been conducted from a variety of perspectives to improve the efficiency and sustainability of the company. However, as mentioned, as the mold products are used in other manufacturing as parts, the customer-oriented operational view of studies needs to be discussed. Based on the customer-oriented operation, the scheduling is a core process as it decides the number and the type of the product to manufacture, which leads to the demand of the customers.

In the literature, a few studies have approached mold scheduling with a variety of objectives. The mold scheduling problem has been formulated diversely, such as parallel machine scheduling, flexible job shop scheduling, and resource-constraint project scheduling problems.

Past studies [13,14] have addressed scheduling injection molding with production planning. The production plan was used as the input for the scheduling, and the heuristic method was developed for a scheduling problem by minimizing the makespan. Some authors have employed a meta-heuristic algorithm for mold scheduling. Oztemel et al. [15] developed a bee algorithm for multi-mode resource-constrained project scheduling in the mold industry to minimize the mold project duration. Wang et al. [16] used an ant colony optimization algorithm for the selection of machines that are needed in each working procedure, and the job sequence was determined by the heuristic algorithm. Choy et al. [17] introduced a genetic algorithm with optimization to minimize the job tardiness and applied it to the mold manufacturing environment.

Others employed a Petri net to solve the mold scheduling problem. The Petri net is a basic process model in process mining that expresses the concurrent situations and synchronizes the events as the model. With the Petri net, Wu et al. [18] solved the mold project schedule, which is expressed as the resource-constrained multiple project scheduling problem that uses a timed-colored Petri net. Caballero-Villalobos et al. [19] combined the genetic algorithm and the Petri net to solve the parallel machine scheduling problem for mold manufacturing by using every advantage.

This study considers the scheduling for mold manufacturing with the importance of products and their tardiness simultaneously as the total weighted tardiness. As the mold manufacturing industry is customer-oriented more than other industries due to the use purpose of the mold, the delivery date is important, and customers receive updates from the manufacturer as per a certain period of time. Therefore, the more practical and realistic objective that is the total weighted tardiness is considered. In addition, in this study, the processing time for manufacturing mold is stochastic, which means the average processing time is known when scheduling, and the actual one is informed

later. This characteristic is essential for the real case, which is the dynamic manufacturing environment. The methodologies that have been presented by previous studies may not be flexible for changing the manufacturing environment. In this study, as the proposed methodology is based on a learning basis, the change of the manufacturing environment is perceived, and it may respond to the change flexibly.

2.2. Reinforcement Learning

Deep RL is a machine learning category and it refers to the combination of RL and deep learning. RL has been widely adopted in the decision-making process, and it can be formulated as a Markov decision process. RL consists of an agent and the environment, and the agent finds a global optimal policy as follows. Once the agent observes the state of the environment, the agent selects the action from action candidates based on the policy that the agent has. Then, the state of the environment is changed by the action, and the agent takes action in the new state. The reward comes from the environment, and the policy of the agent is improved by periodically calculating the long-term reward. The difference between RL and deep RL is that in deep RL, a neural network is used to choose an action from the candidates of the action when the action is being selected. The primary advantage of deep RL is the expression of the complicated model in a relatively simple manner, and the factors for decision making can be considered diversely. Additionally, the agent finds the optimal policy by learning and enhancing the policy through trial and error, and the decision can be made by the optimal policy in a dynamic environment.

Deep RL has been spotlighted after demonstrating an extraordinary performance for problems and applied scheduling problems in a variety of industries [20–22]. Among the algorithms for deep RL, the deep Q-network (DQN) has been employed by many researchers. Atallah et al. [23] applied the DQN to find the energy-efficient scheduling policy considering the characteristics of vehicles in a RoadSide Unit's communication range. Wang et al. [24] addressed the scheduling of multi-workflows for infrastructure-as-a-service clouds using DQN. Xu et al. [25] used the DQN and deep neural network to find the optimal link and for allocating power into the schedule.

DQN has been developed from Q-learning introduced by Watkins [26]. Q-learning is based on a Q-value that calculates the pairing of the state and action for the expected value of the future reward. Mnih et al. [27] developed a DQN that combines Q-learning and neural networks. Through Q-learning, the DQN approximates the Q-value through the neural network. The RL has a problem that cannot converge due to the correlations between the target value and Q-value, which occurred by the order of observations. To eliminate the correlations, the DQN uses replay memory that stores the experienced data for selecting the data randomly when learning. Besides, the target network that is employed to prevent the target value is changed continuously. The weight of the target network is updated iteratively for stable convergence of expected Q-value.

In this study, the DQN is employed for mold production scheduling that has not been addressed previously and is designed to minimize the total weighted tardiness.

3. Mold Production Scheduling

3.1. Problem Definition

This section describes the environment for mold manufacturing and explains some assumptions that are used to model the mold scheduling system. Generally, mold manufacturers receive the order of customers with the delivery date. With the order, the mold that is to be produced is passed through the main workflow, which is divided into three parts: design, manufacturing, and testing. Once the design is discussed with the customer, the processes of the mold are determined, and the materials that are needed for the mold are ordered. The mold production system has diverse specifications for the products and small production orders, each of which is unique; thus, the processes of the molds can be different.

The mold is grouped by size (small, middle, and large) and is named based on the number of parts: a two-plate or a three-plate mold. The mold consists of many parts, and the parts have operations that need to be processed. The mold is completed when the operations of all the parts are finished. Figure 1 shows the configuration of the mold. The processing time of the operations is dependent on the type and size of the mold. In this study, it is assumed that if the size of the mold is large, the operation processing time is longer. For example, if there are operations for large and small molds, respectively, and they are the same, the processing time of the operation for a large mold is longer than the small mold. Additional assumptions are as follows:

- The moving time between the operations is excluded.
- The machines that have the same type are considered to have equal abilities.
- One operation can be processed on a machine at a time.
- The number of operations and parts are known and are not changed until the mold is completed.
- If the machine is idle, the operation of parts that are waiting for assigning is allocated to the spot unless the number of waiting operations is empty.
- The machine is dedicated to certain types of operations.
- When the setup of the machine is changed, the setup time occurs.

Figure 1. Injection mold configuration.

3.2. Mathematical Modeling

In this section, a mathematical model is presented for the mold scheduling problem that has been previously described. This model shows the decision-making point with constraints for mold scheduling.

$$\min \sum_i w_i T_i \quad (1)$$

Subject to

$$T_i \geq C_i - d_i, \quad \forall i \in I \quad (2)$$

$$st_{ip0} \geq st_{ip(o-1)} + \sum_k pt_{ip(o-1)} \cdot x_{ip(o-1)k}, \quad \forall i \in I, p \in \mathcal{P}_i, o \in \mathcal{O}_p^i, o > 1, k \in \mathcal{M}_{p(o-1)}^i \quad (3)$$

$$st_{ip0} \geq st_{ip'0} + pt_{ip'0} + su_{ip0ip'0k} - (2 - x_{ip0k} - x_{ip'0k} + y_{ip0ip'0k}) \cdot H, \quad \forall i, i' \in I, \\ p \in \mathcal{P}_i, p' \in \mathcal{P}_{i'}, k \in \mathcal{M}_p^i \cap \mathcal{M}_{p'}^{i'}, o \in \mathcal{O}_p^i, o' \in \mathcal{O}_{p'}^{i'}, |o \neq o'| \quad (4)$$

$$st_{ip'0} \geq st_{ip0} + pt_{ip0} + su_{ip0ip'0k} - (3 - x_{ip0k} - x_{ip'0k} + y_{ip0ip'0k}) \cdot H, \quad \forall i, i' \in I, \\ p \in \mathcal{P}_i, p' \in \mathcal{P}_{i'}, k \in \mathcal{M}_p^i \cap \mathcal{M}_{p'}^{i'}, o \in \mathcal{O}_p^i, o' \in \mathcal{O}_{p'}^{i'}, |o \neq o'| \quad (5)$$

$$st_{ip0} \geq r_i, \quad \forall i \in I, p \in \mathcal{P}_i, o \in \mathcal{O}_p^i, o = 1 \quad (6)$$

$$C_{ip} \geq st_{ip0} + \sum_k pt_{ip0} \cdot x_{ip0k}, \quad \forall i \in I, p \in \mathcal{P}_i, o \in \mathcal{O}_p^i, o \in |\mathcal{O}_p^i|, k \in \mathcal{M}_p^i \quad (7)$$

$$C_i = \max_{p \in \mathcal{P}_i} C_{ip}, \quad \forall i \in I \quad (8)$$

$$\sum_k x_{ipok} = 1, \forall i \in I, p \in \mathcal{P}_i, o \in \mathcal{O}_p^i, k \in \mathcal{M}_p^i \quad (9)$$

$$T_i, C_i, st_{ipo}, C_{ip} \geq 0, x_{ipok}, y_{ijp'i'j'p'} \in \{0, 1\} \quad (10)$$

Equation (1) relates to the objective that minimizes the total weighted tardiness of the jobs. Equation (2) calculates the tardiness of each job. Equation (3) ensures the precedence relations between the consecutive operations of a part for the job. Equations (4) and (5) guarantee that the overlapping of the operations on the machine is restrained. Equation (6) means that the job cannot be started before the release time itself. Equations (7) and (8) calculate the completion time of each part of the job and the finished time for the job, respectively. Equation (9) guarantees that only one operation can be processed on a machine.

3.3. Markov Decision Process Framework

For deep RL, the MDP is proposed in this section, which can mathematically model the decision process. In deep RL, the agent observes the state of the environment at a certain time t , s_t , and an action, a_t , which is selected by the policy of the agent among the actions that are given to the environment. The state is changed to the next state, s_{t+1} , through the transition probability, and the reward, r_t , is received from the environment. The same procedure occurs at every discrete time. In the next section, the state, action, and reward are described, and the transition probability is excluded because it is not necessarily required for deep RL [28,29].

3.3.1. State

In terms of the environment of the mold production scheduling system, the main information is largely divided by two factors: the machine and the job. In this system, the state is meant to be a set of idle machine statuses. Let $\bar{K}_t$ be a set of idle machines and ty_{kt} denotes the setup type of the machine $k \in \bar{K}_t$ at time t . Machines are dedicated to certain operation types, $sy = \{1, 2, \dots, SY\}$, and additional setup time occurs if a different type of operation is allocated to the machine that already has the setup type.

For job information, let $\bar{\mathcal{O}}_t^k$ be a set of waiting operations at time t for machine k and SO_t^{sy} is the sum of the operations that have operation type sy in $\bar{\mathcal{O}}_t^k$. As mentioned in Section 2.2, the job has the weight $jw = \{1, 2, \dots, JW\}$ and let OW_t^{jw} be the sum of the operations that have jw at time t . Based on the explanation above, the state of each machine k at time t , s_t^k , is expressed as follows.

$$s_t^k = \left\{ ty_{kt}, \left(\frac{OW_t^1}{|\bar{\mathcal{O}}_t^k|}, \frac{OW_t^2}{|\bar{\mathcal{O}}_t^k|}, \dots, \frac{OW_t^{JW}}{|\bar{\mathcal{O}}_t^k|} \right), \left(\frac{SO_t^1}{|\bar{\mathcal{O}}_t^k|}, \frac{SO_t^2}{|\bar{\mathcal{O}}_t^k|}, \dots, \frac{SO_t^{SY}}{|\bar{\mathcal{O}}_t^k|} \right) \right\}, \forall k, t \quad (11)$$

In Equation (11), by dividing the total number of operations, $|\bar{\mathcal{O}}_t^k|$, the state vector is standardized as a set of vector numbers that are expressed in the range of $[0, 1]$. By doing this, the agent is able to visit all the states [23]. Finally, s_t can be expressed as follows.

$$s_t = \{s_t^1, s_t^2, \dots, s_t^K\}, \forall t \quad (12)$$

3.3.2. Action

The action refers to the selection of the operation for assigning an idle machine k . Hence, the candidates of the action, which is the action space, is the total number of waiting operations for machine k . Equation (13) shows the action space for s_t^k .

$$a_t^k \in \{\bar{\mathcal{O}}_t^k\}, \forall k, t \quad (13)$$

Based on Equation (13), a_t can be described as follows.

$$a_t = \{a_t^1, a_t^2, \dots, a_t^K\}, \quad \forall t \quad (14)$$

3.3.3. Reward

The reward is meant to be the objective of the mold production system, which minimizes the total weighted tardiness. To calculate the tardiness of the job, the completion time is required, which is difficult to calculate at the decision point. In addition, even if the completion time of the job is expected at the decision point, it is important to consider the possibility that the job might be finished earlier or later than the expected time owing to other jobs' schedules.

In this study, the processing time of the operation that is selected as the action is regarded as the reward. At the decision point, the processing time of the operation is not known; hence, the decision is made by using the average processing time. By multiplying the weight of the operation of the job, the weight of the job is reflected in the reward.

If the processing time of the last operation is long, this may have the impact that the tardiness is larger. The reward formulation considers that it prevents larger tardiness by the long processing time of the last operation and selects the operation that belongs to the job that has a high weight first.

From the above, the reward consequential to the action for each machine is formulated in Equation (15), and r_t is expressed in Equation (16).

$$r_t^k = w_i \cdot p_{ipo}, \quad (15)$$

$$r_t = \{r_t^1, r_t^2, \dots, r_t^K\}, \quad \forall t \quad (16)$$

3.4. Deep RL for the Mold Production Scheduling System

Based on the aforementioned details, in this study, the agents and the environment are structured and interact, as illustrated in Figure 2. There are as many agents as the number of machines. Once a machine is idle, the agent corresponding to the machine observes the machine state. When observing, the setup type of the machine and the waiting operations to be allocated for the machine are considered. Then, the agent selects the operation while waiting, according to the policy of the neural network, which is the brain of the agents. After taking the action, the agent obtains the reward from the environment. As suggested in [30], while each agent acts in a decentralized fashion, the neural network is shared to take advantage of centralized learning. Therefore, when the elements in the environment change, for example, the number of machines change, the DQN does not need to be retrained.

Figure 2. Interaction between the agents and the environment.

Algorithm 1 shows the mold scheduling procedure that is based on the DQN algorithm for training. At time t , when $\overline{\mathcal{K}}_t$ and $\overline{\mathcal{O}}_t^k$ exist at the same time, scheduling is implemented on the basis of the DQN algorithm until one of them is empty. When the agent selects the action, there are two ways to achieve this: random selection and Q -value selection. Random selection is ϵ greedy and the procedure is as follows. A random variable is generated, and if it is less than the ϵ value, then the action is selected randomly. Otherwise, the action is chosen following the maximum Q -value. To calculate the Q -value of each action, the operation's average processing time, the type of the operation, and the staying time of the job that the operation belongs to are considered.

Algorithm 1. DQN for mold scheduling

Input: scheduling problem
 Output: Parameters θ of a Q -Network

- 1: Initialize replay memory U
- 2: Initialize Q -Network with weights θ
- 3: Initialize target network Q with weights θ^-
- 4: for $ep = 1, T$ do
- 5: Get $\overline{\mathcal{K}}_t$ and $\overline{\mathcal{O}}_t^k$
- 6: if $\overline{\mathcal{K}}_t \neq \emptyset$ and $\overline{\mathcal{O}}_t^k \neq \emptyset$ then
- 7: repeat
- 8: Observe s_t
- 9: Select a random action a_t with probability ϵ greedy policy.
 Otherwise, $a_t = \max_a Q(s_t, a; \theta)$
- 10: Execute a_t (select $o \in \overline{\mathcal{O}}_{kt}$, which is assigned to machine k)
- 11: Observe s_{t+1} , and r_t is given to the agent
- 12: Save transition (s_t, a_t, r_t, s_{t+1}) in U
- 13: $\overline{\mathcal{K}}_t \leftarrow \overline{\mathcal{K}}_t \setminus \{k\}$, $\overline{\mathcal{O}}_t^k \leftarrow \overline{\mathcal{O}}_t^k \setminus \{o\}$
- 14: until $\overline{\mathcal{K}}_t = \emptyset$ or $\overline{\mathcal{O}}_t^k = \emptyset$
- 15: end if
- 16: if $|U| \geq L$ then
- 17: Sampling random minibatch (s_q, a_q, r_q, s_{q+1}) from U
- 18: Calculate loss
- 19: Perform a gradient decent regarding weights θ
- 20: end if
- 21: if $|U| \geq L$ and $ep \% N = 0$
- 22: update $\theta^- = \theta$
- 23: end if
- 24: end for

The waiting time is calculated by the current time minus the release time of the job. Algorithm 2 explains the procedure for selecting the action. The transition is stored in the memory replay U . When the transitions are more than half of the size of U , the loss is calculated with the random minibatch from U . Lastly, in every N episode after calculating the loss, the target weight is updated. Figure 3 shows the framework for the steps of the proposed method.

Algorithm 2. Action selection

- 1: random value $\leftarrow$ random (ϵ):
- 2: if random value $\leq \epsilon$ then
- 3: action $\leftarrow$ random ($\overline{\mathcal{O}}_t^k$)
- 4: else
- 5: action $\leftarrow \max Q(s_t^k, \overline{\mathcal{O}}_t^k)$
- 6: end if

Figure 3. Framework of steps of the proposed method.

The proposed scheduling method consists of training and testing phases. In the training phase, the same scheduling problem is solved iteratively, and the scheduling problems with the other condition are solved in the testing phase. Figure 4 shows the framework of the DQN training for this study. After training, in the testing phase, the proposed method with the trained weight of the Q-network is tested to compare the other dispatching rules in the new scheduling problems to check the performance.

Figure 4. Framework of the deep Q-network (DQN) for training.

4. Experimental Results

This section describes the results of the proposed method for the mold scheduling problem and it provides a comparison of the dispatching rules. The proposed method was coded in Python with the Keras application programming interface and tested on a 3.2 GHz Intel i7 with 16 GB of RAM. The neural network that was used for the method is fully connected, and there are four hidden layers. Each node has 64, 32, 16, and 8 layers, respectively, and the leaky rectified linear unit is used as an activation function for all the layers. For the testing phase, the hyperparameters that are used for the test are listed in Table 1.

Table 1. Hyperparameters.

Hyperparameter	Value
Learning rate	0.001
Discount factor γ	0.9
Epsilon to select action randomly ϵ	0.1
Update target point N	10
Minimum training point L	1000
Size of memory U	20,000
Iteration	3000
Batch size	64
Optimizer	RMSprop

After applying the hyperparameters, the method's performance is influenced; however, the optimal value of the hyperparameters is difficult to determine because of the large search space. In this study, a random search [31] was conducted to find the best value.

For the testing data, the data were received from a small Korean company that made molds that customers order. Some of the missing information was gained from the staff that worked for the company. The molds were divided into three categories: small, medium, and large. The number of parts that are needed for making the molds is dependent on the order of the customers and their size. In this study, the main parts, the main and base cavity, and the core, which are the most critical parts, were considered. Depending on the size, additional parts were considered. According to the size, the number of small, medium, and large parts is three, four, and five. In addition, the processing time of the operations is dependent on the size. The increasing ratio of the average processing time for small, medium, and large is 1, 1.3, and 1.6, respectively. The operations of each part are different, and nine patterns were extracted from them. Table 2 shows the operating patterns. The arrow sign is meant for the direction of the operations, and the next operation can be commenced when the predecessor operation is finished.

Table 2. Operation patterns.

Pattern	Description
1	$A \rightarrow C \rightarrow D$
2	$A \rightarrow C \rightarrow D \rightarrow E$
3	$A \rightarrow C \rightarrow D \rightarrow B \rightarrow G \rightarrow E$
4	$H \rightarrow C \rightarrow D \rightarrow E$
5	$A \rightarrow C \rightarrow B$
6	$A \rightarrow B \rightarrow C \rightarrow F \rightarrow D \rightarrow E$
7	$A \rightarrow C$
8	$A \rightarrow F \rightarrow C \rightarrow E$
9	$A \rightarrow F \rightarrow D \rightarrow E$

→: Operational sequence direction.

The due date of the jobs is 1.3 times the average sum of the processing time of the operations, and the weight w_i is assigned in accordance to the size: 1 for small, 2 for medium, and 3 for large. To compare the performance of the proposed method, four dispatching rules were designed. In order to determine whether an operation should be allocated to an idle machine, two steps need to be performed.

As the job contains parts and the parts have operations, one of the parts for the available jobs can be determined. In other words, (1) a job is selected among the available jobs, and (2) the operation of the available parts for the job is chosen. Because the objective of the problem is to minimize the total weighted tardiness, they consider the earliest due date (EDD) rule for the job selection. The EDD rule selects the shortest remaining time for the job and it is expressed as $\min_{i \in \bar{I}_t} \{d_i - t\}$ where $\bar{I}_t$ is a set of available jobs at time t . Three out of four dispatching rules (ES, EL, EC) use the EDD rule for the job selection. The last rule (WEL) considers the waiting time as well as the EDD, which is $\min_{i \in \bar{I}_t} \{wa \cdot (d_i - t) + (1 - wa) \cdot (t - r_i)\}$ where wa is the weight for the EDD and the waiting time is set to 0.8.

Next, the second step for the four dispatching rules is explained. The first rule (ES) considers the shortest remaining average processing time. The operation of the part that is the smallest sum of the remaining average processing times for the available parts of the selected job is selected. Let O_{pt}^i be a set of operations for part p of job i at time t and the second decision for the first rule can be expressed as $\min_{p \in \mathcal{P}_i} \left\{ \sum_{o=1}^{|O_{pt}^i|} p_{ipo} \right\}$. The second rule (EL) is used to contemplate the longest average processing time. The operation of the part that is the largest sum of the remaining average processing times among the available parts for the selected job is chosen, $\max_{p \in \mathcal{P}_i} \left\{ \sum_{o=1}^{|O_{pt}^i|} p_{ipo} \right\}$. The critical ratio is considered in the

third rule (EC) for the part selection. It selects the operation of the part that has the smallest value, which is calculated using the due date of the job. This is followed by performing a subtraction from the current time, and then it is divided by the sum of the remaining average processing times for the

job's part, $\min_{p \in \mathcal{P}_i} \left\{ \frac{(d_i - t)}{\sum_{o=1}^{|O_{pt}^i|} p_{ipo}} \right\}$. In the last rule (WEL), the method of selecting the part for the operation is the same as that of EL. Table 3 shows the composite dispatching rules at a glance.

Table 3. Composite dispatching rules.

Dispatching Rule	Job Selection	Part Selection
Earliest due date with the shortest remaining average processing time (ES)	$\min_{i \in \bar{I}_t} \{(d_i - t)\}$	$\min_{p \in \mathcal{P}_i} \left\{ \sum_{o=1}^{ O_{pt}^i } p_{ipo} \right\}$
Earliest due date with the longest remaining average processing time (EL)		$\max_{p \in \mathcal{P}_i} \left\{ \sum_{o=1}^{ O_{pt}^i } p_{ipo} \right\}$
Earliest due date with the critical ratio (EC)	$\min_{i \in \bar{I}_t} \{(wa \cdot (d_i - t) + (1 - wa) \cdot (t - r_i))\}$	$\min_{p \in \mathcal{P}_i} \left\{ \frac{(d_i - t)}{\sum_{o=1}^{ O_{pt}^i } p_{ipo}} \right\}$
Earliest due date and waiting time with the longest remaining average processing time (WEL)		$\max_{p \in \mathcal{P}_i} \left\{ \sum_{o=1}^{ O_{pt}^i } p_{ipo} \right\}$

To evaluate the performance of the proposed method, five scenarios are assumed with different occurrence ratios of the job's size. As the size of the job takes up a large proportion of the objective function because the larger size of the job can have more operations, each scenario has a different occurrence ratio, which is shown in Table 4 as a cumulative value.

Table 4. Occurrence ratio for the size of the mold.

Scenario	Small	Medium	Large
1	0.3	0.6	1
2	0.5	0.8	1
3	0.2	0.7	1
4	0.3	0.5	1
5	0.1	0.2	1

In this study, four types of machines were considered. Each type of machine is dedicated to certain operations as follows: type 1 for operations A and C, type 2 for operations B and D, type 3 for operations E and F, and type 4 for operations G and H. There are 11 machines and {3, 5, 2, 2} from types 1 to 4.

For the training phase, with an instance of 10 jobs in scenario 1, the neural network was trained for 3000 episodes. During the training, the selection of an operation was based on the average processing time, and the actual processing time was the same for all the episodes. Therefore, the neural network can learn about the static environment first. Figure 5 shows the results of the total weighted tardiness that was obtained by the DQN in 3000 episodes. As shown, the curve of the graph gradually decreases with an increase in the episode, which indicates that the proposed DQN learned the proper policy for the scheduling problem.

The total number of jobs in the testing phase was 10, 30, 50, and 70 for each scenario; thus, a total of 20 cases were used for testing the proposed method. Each case was tested with 100 independent instances, and the total number of operations that had to be processed for each job was approximately 150, 500, 800, and 1000. The trained Q network was selected for testing during the 2633rd episode, which showed the best performance during the episodes. The results are represented by the average values for 100 instances, which are shown in Table 5. In all the scenarios, the performance of deep RL for minimizing the total weighted tardiness outperformed the other dispatching rules. Even in

an extreme situation, such as scenario 5, the deep RL showed a better performance than the other dispatching rules, and it was observed that the deep RL worked to schedule the operations in harsh conditions. As the number of jobs increased, the standard deviation of the deep RL was larger than the other rules, as shown in Figure 6.

Figure 5. Total weighted tardiness that was obtained by the DQN during the episode.

Table 5. Results for the mean of the total weighted tardiness.

Jobs	Rule	Scenario 1 (10 ²)	Scenario 2 (10 ²)	Scenario 3 (10 ²)	Scenario 4 (10 ²)	Scenario 5 (10 ²)
10	Deep RL	8.7	7.1	10.1	9.9	14.3
	ES	40.9	30.9	43.3	48.7	69.8
	EL	32.9	25.2	35.3	37.1	55.8
	EC	38.0	29.3	40.9	42.8	60.7
	WEL	22.7	18.2	24.6	26.2	35.3
30	Deep RL	260.2	188.7	279.2	284.9	438.0
	ES	570.3	434.4	611.1	650.7	929.4
	EL	518.3	394.9	565.0	592.2	838.8
	EC	519.9	398.1	560.1	583.4	864.5
	WEL	469.1	363.5	494.4	535.1	720.5
50	Deep RL	863.2	648.2	945.9	978.1	1391.5
	ES	1595.6	1327.3	1831.4	1984.6	2758.3
	EL	1597.9	1271.7	1734.6	1880.5	2595.0
	EC	1595.6	1243.6	1755.7	1856.0	2586.9
	WEL	1428.2	1142.8	1546.7	1689.5	2266.7
70	Deep RL	1797.7	1295.5	1953.3	2042.5	2875.6
	ES	3372.4	2538.9	3652.1	3863.5	5479.5
	EL	3233.3	2423.5	3493.9	3700.1	5214.2
	EC	3304.3	2541.7	3601.4	3754.7	5382.6
	WEL	2991.4	2271.6	3224.4	3415.0	4707.1

Specifically, in the case of 50 and 70 jobs, the standard deviation of the deep RL was largest, which showed that there was a large difference between the best and the worst performance. Another interesting thing is that WEL had a better performance than the rest of the rules, because the rule considered the combination of the waiting time and the due date of the job. This means that the waiting time may be necessary to prevent certain jobs from waiting for too long.

Figure 7 shows the results of the total weighted tardiness for each instance for 30 jobs in scenario 1 and the average, minimum, and maximum of the total weighted tardiness in the box plot. The orange bar in the rectangular box indicates the median value, the green dot is the mean value, and the asterisk in blue is an outlier. As shown in Figure 7, in many instances, it is difficult to determine the superiority of the inferior performance of the dispatching rules. However, deep RL showed a stable performance

in most instances. This observation shows that deep RL is larger than the other dispatching rules depending on the uncertainty of the processing time.

Figure 6. Standard deviation for each job in scenario 1.

Figure 7. Graph of the total weighted tardiness for each instance and the box plot of the 30 jobs in scenario 1.

5. Discussion

In a manufacturing environment, it is difficult to make a decision to schedule jobs under a dynamic situation, and it is important to provide the decision-maker with a certain rule or policy to achieve efficient scheduling. In a variety of studies, rule-based methods have been employed for scheduling jobs; however, this method has some disadvantages. One of the disadvantages is that a single rule cannot outperform the other rules for different shop configurations [32]. The results that are presented in this study show that the deep RL method for scheduling jobs to minimize the total weighted tardiness can perform better than the other suggested rules in diverse situations. In most instances, the performance of deep RL was higher than the other rules, which is stable in a dynamic situation. In all the scenarios, the processing time was not the same, which generates different processing times. The information known before the scheduling jobs was the average processing time and the rules that were selected to consider it. In the situation that the processing time was not static, deep RL had a robust performance. Likewise, in the practical manufacturing environment, the exact processing time cannot be expected and the change in the number of machines occurs frequently due to machine break, failure, and so forth. In the situation, the proposed method is able to show better performance and provide a robust scheduling policy.

To reduce the complexity of the state, a novel definition state is proposed to recognize the environment. In addition, generally, the reward is equivalent to the objective function. However, it is highly difficult to anticipate the tardiness because the mold product consists of several parts and the parts have operations to be processed on dedicated machines. To complete the mold product, the operations of all parts are finished. In addition, as the operations of the parts of a mold product are affected by machine status and other parts of other products, selecting the operation of a part from the waiting parts to be processed on an idle machine by calculating the tardiness in the middle of the manufacturing the mold is complex and difficult. Even if the expected tardiness is calculated, the accuracy of the calculation may not be guaranteed due to the dynamic environment. In this study, the reward for considering the objective that minimizes the total weighted tardiness is designed in the view of the processing time of the operation. The agent tends to maximize the sum of the rewards and, the processing time of the operation and the weight of the product that the operation belongs to are considered to the reward. This is equivalent to that the operation that has the longest processing time is allocated first among others in consideration of the weight of the product that includes the operation. By doing this, although this is not related to the tardiness of the product directly, it prevents the tardiness is larger by the last operation of the product.

To apply the proposed method practically, the manufacturing scheduling system is required with computer support for solving scheduling problems. Basically, all data is collected from all machines and the information of customers' orders is stored in the database system. Using historical data, the neural network is learned with the proposed state, action, and reward. After enough learning, the neural network is used to schedule the operations on idle machines. At that time, the transmission between the neural network and the idle machines should be possible. Depending on manufacturing sites, the instruction to select the operation from the neural network is given to workers in front of machines or machines directly. In addition, learning the neural network that is the weight needs to be done periodically to accommodate the change of the manufacturing environment, as the weight is a key factor that decides the action based on the state. When using the suggested scheduling model, the assumptions for the model are addressed to fit the manufacturing field. For example, if the distance between machines is far, the moving time should be considered to add a new decision variable. Furthermore, the assumption that the same type of machine has the same ability is handled as the actual ability. For instance, average processing time may be different between the same type of machines so that the ability is considered by the field data.

The proposed algorithm provides benefits in the view of the management for the manufacturing process. As the algorithm is based on the learning with historical data, the proposed algorithm can confront manufacturing processes flexibly in an unpredictable situation so that the cumbersome works can be reduced such as reconfiguration of scheduling policy due to unexpected situations. In addition, even if new processes are added, the algorithm is updated through simulation before applying to the production system so that the efficiency of the production is sustained.

In the proposed method, the *Q*-networks were trained with a small-scale problem, and then the trained *Q*-network was used for large-scale problems. As a result, the deep RL showed a good performance in comparison to the other rules without retraining, which shows the robustness of the trained *Q*-network. Additionally, in the proposed method, the centralized neural network is designed, so that the neural network is not needed to retrain when the number of the machine is changed.

However, when the type of operations, machines, and the sequence of the operations are changed or are added, the *Q*-network needs to be retrained, which is one of the limitations of this study. Another limitation is that the model has some assumptions that do not embrace all the factors in manufacturing situations. For example, the method of manufacturing molds is different from the specification of what customers order. To embrace this, the order of the customers was largely divided by the size of the mold: small, medium, and large. In addition, all the processes are not included, but the core processes are considered to represent mold manufacturing. In this study, diverse machines were not considered, and it is assumed that the machines have the same performance. In the real

world, the performance can differ depending on the machine year. However, the difference between the machines was included in the processing time as the processing time of the operations was not the same for every instance.

In terms of future research, a state is needed to develop Q-networks that are not necessary for retraining, even though operation types, machine types, and so forth are changed. In addition, more uncertain factors, such as the breakdown of machines, can be considered. It can be extended as a multi-objective, such as minimizing the makespan and tardiness as they are conflicted. As another future research, the performance of the proposed algorithm can be compared with other heuristic algorithms that were presented in other studies [33].

6. Conclusions

In Industry 4.0, as manufacturing firms consider technical issues for sustainability as well as economic, environmental, and social issues, they try to enhance the production efficiency and meet customers' demand with the technical introduction to the manufacturing. To fulfill the firms' exertion, enhancing the agility of manufacturing is key. Because the manufacturing environment is getting dynamic due to an increase in the number of products to be produced, not responding to the environment with agility causes customers' dissatisfaction and has a negative influence on the companies' competitiveness in the market as a result. Therefore, intelligent systems have been required to cope with the dynamic environment for complex production systems to enhance sustainability.

In complex production systems, scheduling is a core process that can increase profits and reduce costs. Manufacturing molds consist of complex production systems since many parts are combined and the specification of each mold is different. Moreover, the parts have different processes, and those factors induce the dynamic manufacturing environment. Due to the complexity and the dynamic environment, the importance of scheduling for mold manufacturing has emerged and sustainable and intelligent scheduling has been required to handle them with agility. However, few studies have been conducted on scheduling in mold manufacturing. In addition, the literature has proposed heuristic rules that are intractable in the real world. To address the complexity and the dynamic environment as well as practicality, in this study, deep RL was designed for mold scheduling problems. Before applying deep RL, a mathematical model was developed for the mold scheduling problem, and the MDP was presented for the objective of minimizing the total weighted tardiness. The algorithm of deep RL, DQN, was used to find the optimal scheduling policy for the mold scheduling problem.

To evaluate the performance of deep RL, some dispatching rules were suggested as a comparison with deep RL. As a result, deep RL outperformed the other dispatching rules under diverse scenarios. In the case that has large operations and a large number of high-weighted jobs, deep RL showed a lower total weighted tardiness than the others. Moreover, even in the situations in which the processing time for each instance was not deterministic, the proposed method assures good performance in a dynamic situation.

Author Contributions: Conceptualization, S.L. and Y.H.L.; methodology, S.L.; software, S.L.; validation, S.L. and Y.H.L.; formal analysis, S.L.; investigation, S.L.; data curation, Y.C.; writing—original draft preparation, S.L.; writing—review and editing, S.L.; visualization, S.L.; supervision, Y.H.L.; project administration, Y.C.; funding acquisition, Y.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the regional industry base organization support project (P0001955, Support project to innovate IoT and big data-based mold manufacturing value chain), funded by the Ministry of Trade, Industry and Energy (MOTIE).

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Glossary/Nomenclature/Abbreviations

I	Jobs
$\mathcal{P}_i$	Parts of job i
$\mathcal{O}_p^i$	Operations of part p for job i
$\mathcal{M}_p^i$	Machines for dedicating $\mathcal{O}_p^i$
i	Job
k	Machine
p	Part
o	Operation
d_i	Due date of job i
r_i	Release time of job i
w_i	Weight of job i
p_{ipo}	Average processing time of operation o of part p for job i
$su_{ipoi'p'o'k}$	Setup time for the changing operation o of part p for job i to operation o' of part p' for job i' on machine k
H	Huge number
T_i	Tardiness of job i
C_i	Completion time of job i
C_{ip}	Completion time of part p for job i
st_{ipo}	Start time for operation o of part p for job i
x_{ipok}	1 if the operation o of part p for job i is assigned to machine k ; otherwise, it is 0
$y_{ipoi'p'o'}$	1 if operation o of part p of job i is scheduled before the operation o' of part p' for job i' ; otherwise, it is 0

References

1. Waibel, M.; Steenkamp, L.; Moloko, N.; Oosthuizen, G. Investigating the effects of smart production systems on sustainability Elem. *Procedia Manuf.* **2017**, *8*, 731–737. [\[CrossRef\]](#)
2. Lee, S.; Lee, Y.H.; Choi, Y. Project Portfolio Selection Considering Total Cost of Ownership in the Automobile Industry. *Sustainability* **2019**, *11*, 4586. [\[CrossRef\]](#)
3. Oluyisola, O.E.; Sgarbossa, F.; Strandhagen, J.O. Smart Production Planning and Control: Concept, Use-Cases and Sustainability Implications. *Sustainability* **2020**, *12*, 3791. [\[CrossRef\]](#)
4. Low, M.L.H.; Lee, K.S. Mould data management in plastic injection mould industries. *Int. J. Prod. Res.* **2008**, *46*, 6269–6304. [\[CrossRef\]](#)
5. Low, M.; Lee, K. A parametric-controlled cavity layout design system for a plastic injection mould. *Int. J. Adv. Manuf. Technol.* **2003**, *21*, 807–819. [\[CrossRef\]](#)
6. Hu, W.; Masood, S. An intelligent cavity layout design system for injection moulds. *Int. J. CAD/CAM* **2002**, *2*, 69–75.
7. Fu, M.; Fuh, J.; Nee, A. Core and cavity generation method in injection mould design. *Int. J. Prod. Res.* **2001**, *39*, 121–138. [\[CrossRef\]](#)
8. Li, C.; Li, C.; Mok, A. Automatic layout design of plastic injection mould cooling system. *Comput.-Aided Des.* **2005**, *37*, 645–662. [\[CrossRef\]](#)
9. Liang, J.-Z. An optimal design of cooling system for injection mold. *Polym.-Plastics Technol. Eng.* **2002**, *41*, 261–271. [\[CrossRef\]](#)
10. Li, Z.; Wang, X.; Gu, J.; Ruan, S.; Shen, C.; Lyu, Y.; Zhao, Y. Topology optimization for the design of conformal cooling system in thin-wall injection molding based on BEM. *Int. J. Adv. Manuf. Technol.* **2018**, *94*, 1041–1059. [\[CrossRef\]](#)
11. Li, W.; Kara, S.; Qureshi, F. Characterising energy and eco-efficiency of injection moulding processes. *Int. J. Sustain. Eng.* **2015**, *8*, 55–65. [\[CrossRef\]](#)
12. Madan, J.; Mani, M.; Lyons, K.W. Characterizing energy consumption of the injection molding process. In *International Manufacturing Science and Engineering Conference*; American Society of Mechanical Engineers: Madison, WI, USA, 2013; p. V002T04A015.
13. Nagarur, N.; Vrat, P.; Duongsuwan, W. Production planning and scheduling for injection moulding of pipe fittings: A case study. *Int. J. Prod. Econ.* **1997**, *53*, 157–170. [\[CrossRef\]](#)
14. Lin, C.; Wong, C.; Yeung, Y. Heuristic approaches for a scheduling problem in the plastic molding department of an audio company. *J. Heuristics* **2002**, *8*, 515–540. [\[CrossRef\]](#)

15. Oztemel, E.; Selam, A.A. Bees Algorithm for multi-mode, resource-constrained project scheduling in molding industry. *Comput. Ind. Eng.* **2017**, *112*, 187–196. [\[CrossRef\]](#)
16. Wang, Y.-B.; Wang, G.; Zhao, L.-Z.; GAO, G.-A. Dynamic scheduling of mold manufacturing based on ant colony optimization. *Comput. Integr. Manuf. Syst.* **2006**, *12*, 1028.
17. Choy, K.L.; Leung, Y.; Chow, H.K.; Poon, T.; Kwong, C.; Ho, G.T.; Kwok, S. A hybrid scheduling decision support model for minimizing job tardiness in a make-to-order based mould manufacturing environment. *Expert Syst. Appl.* **2011**, *38*, 1931–1941. [\[CrossRef\]](#)
18. Wu, Y.; Zhuang, X.-C.; Song, G.-H.; Xu, X.-D.; Li, C.-X. Solving resource-constrained multiple project scheduling problem using timed colored Petri nets. *J. Shanghai Jiaotong Univ. (Sci.)* **2009**, *14*, 713. [\[CrossRef\]](#)
19. Caballero-Villalobos, J.P.; Mejía-Delgadillo, G.E.; García-Cáceres, R.G. Scheduling of complex manufacturing systems with Petri nets and genetic algorithms: A case on plastic injection moulds. *Int. J. Adv. Manuf. Technol.* **2013**, *69*, 2773–2786. [\[CrossRef\]](#)
20. Hubbs, C.D.; Li, C.; Sahinidis, N.V.; Grossmann, I.E.; Wassick, J.M. A deep reinforcement learning approach for chemical production scheduling. *Comput. Chem. Eng.* **2020**, *141*, 106982. [\[CrossRef\]](#)
21. Lee, S.; Lee, Y.H. Improving Emergency Department Efficiency by Patient Scheduling Using Deep Reinforcement Learning. *Healthcare* **2020**, *8*, 77. [\[CrossRef\]](#)
22. Waschneck, B.; Reichstaller, A.; Belzner, L.; Altenmüller, T.; Bauernhansl, T.; Knapp, A.; Kyek, A. Optimization of global production scheduling with deep reinforcement learning. *Procedia CIRP* **2018**, *72*, 1264–1269. [\[CrossRef\]](#)
23. Atallah, R.; Assi, C.; Khabbaz, M. Deep reinforcement learning-based scheduling for roadside communication networks. In Proceedings of the 2017 15th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt), Paris, France, 15–19 May 2017; pp. 1–8.
24. Wang, Y.; Liu, H.; Zheng, W.; Xia, Y.; Li, Y.; Chen, P.; Guo, K.; Xie, H. Multi-objective workflow scheduling with deep-Q-network-based multi-agent reinforcement learning. *IEEE Access* **2019**, *7*, 39974–39982. [\[CrossRef\]](#)
25. Xu, S.; Liu, P.; Wang, R.; Panwar, S.S. Realtime scheduling and power allocation using deep neural networks. In Proceedings of the 2019 IEEE Wireless Communications and Networking Conference (WCNC), Marrakesh, Morocco, 15–18 April 2019; pp. 1–5.
26. Watkins, C.J.C.H. Learning from Delayed Rewards. Ph.D. Thesis, King's College, London, UK, 1989.
27. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529. [\[CrossRef\]](#) [\[PubMed\]](#)
28. He, Y.; Liang, C.; Yu, F.R.; Han, Z. Trust-Based Social Networks with Computing, Caching and Communications: A Deep Reinforcement Learning Approach. *IEEE Trans. Netw. Sci. Eng.* **2020**, *7*, 66–79. [\[CrossRef\]](#)
29. Ong, H.Y.; Chavez, K.; Hong, A. Distributed deep Q-learning. *arXiv* **2015**, arXiv:1508.04186. Available online: <https://arxiv.org/pdf/1508.04186.pdf> (accessed on 10 June 2020).
30. Foerster, J.; Assael, I.A.; De Freitas, N.; Whiteson, S. Learning to communicate with deep multi-agent reinforcement learning. In Proceedings of the Advances in Neural Information Processing Systems 29 (NIPS 2016), Barcelona, Spain, 5–10 December 2016; pp. 2137–2145.
31. Bergstra, J.; Bengio, Y. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.* **2012**, *13*, 281–305.
32. Pickardt, C.; Branke, J.; Hildebrandt, T.; Heger, J.; Scholz-Reiter, B. Generating dispatching rules for semiconductor manufacturing to minimize weighted tardiness. In Proceedings of the IEEE 2010 Winter Simulation Conference, Baltimore, MD, USA, 5–8 December 2010; pp. 2504–2515.
33. Bottani, E.; Centobelli, P.; Cerchione, R.; Gaudio, L.; Murino, T. Solving machine loading problem of flexible manufacturing systems using a modified discrete firefly algorithm. *Int. J. Ind. Eng. Comput.* **2017**, *8*, 363–372. [\[CrossRef\]](#)

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).