Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 93 (2020) 383–388

www.elsevier.com/locate/procedia

53rd CIRP Conference on Manufacturing Systems

Deep reinforcement learning-based dynamic scheduling in smart manufacturing

Longfei Zhou[a], Lin Zhang[b,*], Berthold K. P. Horn[a]

[a]*Massachusetts Institute of Technology, 32 Vassar St, Cambridge, MA 02139, USA*
[b]*Beihang University, Xueyuan Rd, Haidian District, Beijing 100191, China*

Abstract

Scheduling problems are a classic type of optimization problems in the manufacturing domain, such as job shop scheduling, flexible job shop scheduling, and distributed job shop scheduling problems. Especially, the dynamic task scheduling problem is closer to the requirements of real manufacturing systems than the static scheduling problem. In recent years, with the deeper application of the Internet of Things, big data, and cloud platform technologies, manufacturing systems have evolved from job shops to networked, collaborative and intelligent manufacturing systems. Smart manufacturing scheduling has some characteristics compared with job shop scheduling not only because of the larger number of tasks and services, but also because of the dynamic state of services and uncertainties. In this paper, we analyze the smart manufacturing service scheduling problem and give its mathematical description. Then a deep reinforcement learning-based method is proposed to minimize the maximum completion time of all tasks. In the system framework of the proposed method, the agent, environment, as well as the interaction between them, are designed. The queue times of all candidate services are considered as the system state, and the maximum queue time at the current moment is considered as the target value. Besides, we use two networks the prediction network and the target network to learn the prediction value and the target value, respectively. Two case studies are used to show the efficiency of the considered problem and the proposed method.

Keywords: Smart manufacturing; dynamic scheduling; deep reinforcement learning; dispatching rule

1. Introduction

Manufacturing task scheduling plays an important role in the manufacturing industry. Especially, the dynamic task scheduling problem is closer to the requirements of real manufacturing systems than the static scheduling problem [1, 2]. As manufacturing systems develop from basic workshops to flexible manufacturing, distributed manufacturing, and then networked service-oriented manufacturing, and even today's smart manufacturing systems, the complexity of manufacturing systems becomes higher and higher, and the uncertainty of systems becomes more and more serious. This also leads to more difficulties to solve the dynamic scheduling problem of complex manufacturing systems.

* Corresponding author. Tel.: +86-1-8233-8577; fax: +86-1-8233-8577.
E-mail address: zhanglin@buaa.edu.cn (Lin Zhang).

Nomenclature

T_i	the i-th arriving task
A_i	arrival time of T_i
$P(t)$	probability distribution of task time interval
k_i	task type of T_i
N	number of task types
S_i	the i-th manufacturing service
D	earliest available time of S_i at time t
M	number of manufacturing services
C	service capability matrix
c_{ij}	service time of task type j on service S_i
t_s	start time of the considered scheduling process
t_e	end time of the considered scheduling process
π_i	the j-th scheduling policy
a_i	action the agent takes at i-th iteration
s_i	system state at i-th iteration
r	reward the environment feedbacks to the agent

2212-8271
10.1016/j.procir.2020.05.163

λ	decay rate of the agent
Q	Q value of action a at state s
θ	parameters in neural networks

Typical task scheduling methods can be divided into two categories: precise scheduling methods and approximate scheduling methods. The precise methods search the whole solution space and then obtain the global optimal solutions. Hence, precise methods are not so efficient at solving large-scale scheduling problems because of its high computational complexity. Compared with precise scheduling methods, the approximate methods do not search the whole solution space but explore multiple directions based on particular strategies [3]. So, the approximate methods have lower computational complexity and obtain feasible solutions more quickly, while has more advantages in solving large-scale scheduling problems. But the approximate methods cannot guarantee the optimal solution of the considered scheduling problem. Approaches for the dynamic job shop scheduling problem include variable neighborhood search [4], agent-based approach [5], real-time process information-based method [6], etc. The dynamic scheduling problem in flexible manufacturing systems is more complex than job shops because of larger solution spaces and flexibility of manufacturing resources. Approaches for the dynamic scheduling problem in flexible manufacturing systems include learning-based approach [7], heuristic algorithms [8], agent-based method [9], gravitational emulation local search algorithm [10], multi-objective evolutionary algorithms [11], etc.

The dynamic scheduling in smart manufacturing (DSSM) problem is one of the key technologies to achieve resource integration and service on-demand usage, involving virtual resource management, task distribution in a cross-enterprise environment, and shop floor scheduling problems. The multitask scheduling problem of cloud 3D printing services was studied from aspects of model matching and service scheduling [12]. Logistics is also an important factor in the scheduling problem in collaborative manufacturing environment [13, 14]. Besides, simulation technology is an important approach for dynamic scheduling problems [15]. With the recent development of sensing and communication technologies, real-time data becomes more available. In real-time simulation, a simulation system is continually affected by the real-time data of the real system. A dynamic data-based optimization heuristic was proposed for the dynamic shop floor scheduling problem [16]. An event-triggered method was proposed to deal with the randomly arriving tasks to cloud manufacturing system [17].

In recent years, artificial intelligence technologies have brought much further development to different fields such as transportation, manufacturing, energy, economy, etc. Machine learning is one of the most important methods in artificial intelligence domain to develop intelligent algorithms and systems. In machine learning, there are three kinds of algorithms including supervised learning, unsupervised learning and reinforcement learning [18]. Among them, both supervised learning and unsupervised learning train their machine learning models based on pre-collected data, but reinforcement learning does not rely on a dataset. Therefore, reinforcement learning is more suitable for some decision-making problems such as adventure games [19] in which there is no data available for training the model. The dynamic task scheduling problem in a smart manufacturing environment can also be considered as a dynamic decision-making problem because tasks come randomly at any time and manufacturing services become available and unavailable over time. The scheduling agent needs to decide which manufacturing service is the optimal one for the current arrival task. Deep reinforcement learning is an improvement of typical reinforcement learning in which deep neural network is used to construct the intelligent policy [20].

Some researchers have proposed reinforcement learning algorithms for task scheduling in a service-oriented environment. To solve the scheduling problem in distributed systems, paper [21] proposed a reinforcement learning algorithm considering the heterogeneity and disposition of nodes within the grid to obtain a shorter execution time. In order to enable the agent to modify their actions based on experiences gained in interacting with the other agents and the environment, a multi-agent framework for optimization using metaheuristics is proposed in which all agents share information and collaborate with each other [22]. To guide the scheduling of multi-workflows in cloud infrastructure, paper [23] used a multi-agent deep-Q-network in which the number of workflow applications and virtual machines was used as state inputs and the maximum completion time and cost as rewards. In paper [24], the deep Q-network was extended to address the job shop scheduling decision of multiple edge devices in a smart manufacturing factory considering edge computing.

One of the critical issues of scheduling in an actual smart manufacturing environment is that there are a lot of uncertain events and emergencies possibly occurring randomly over time. Reinforcement learning is a powerful tool to deal with this kind of decision-making problems with random natures. It is interesting to design and implement an intelligent scheduling agent for dealing with the DSSM problem in which customers' tasks arrive randomly with different types and requirements. In this paper, we firstly give the mathematical description of the DSSM problem, and then a deep reinforcement learning-based framework for the DSSM problem is presented in order to minimize the makespan of all tasks over time. The main contributions of this paper include the mathematical model of the considered DSSM problem and the proposed deep reinforcement learning-based system framework for the DSSM problem.

The rest of this paper is organized as follows. In Section 2, we analyze the considered DSSM problem and build its mathematical model including tasks, services, service capability matrix, optimization objectives, assumptions, and constraints. In Section 3, we propose a dynamic scheduling method based on the deep reinforcement learning algorithm for the DSSM problem. In Section 4, two case study are applied to show the details of the considered problem and the proposed method. Finally, conclusions and future works are discussed in Section 5.

2. Problem formulation

The definition of the DSSM problem is given in terms of assumptions, tasks, services, service capability, optimization objectives, and constraints.

2.1. Assumptions

There are four assumptions for tasks and services as well as their relationship in this paper.

- The tasks arrive at the smart manufacturing system randomly, and the time interval of task arrival obeys a certain probability distribution function;
- Each service can perform one or more types of tasks;
- Each task type can be performed by one or more services;
- The service time for each service to perform a certain task type is constant and does not change over time.

2.2. Tasks

Let T_i denote the i-th arriving task, and the arrival time of T_i is A_i. Assume that the time interval of task arrival obeys probability distribution $P(t)$, then:

$$| A_{i+1} - A_i | \sim P(t) \tag{1}$$

Besides, different service demanders have individualized requirements, resulting in task diversity. And we use task types to reflect task diversity between different customers. Let k_i denote the task type of T_i, then we can match an arrival task T_i to a set of services based on the task type of T_i. And we assume that there are N task types in the smart manufacturing system in total.

2.3. Services

An important difference between the typical scheduling problem and dynamic scheduling problem is the dynamic change of system states over time. In DSSM, the status of each service changes over time because the processing task will be completed, and new tasks are possibly scheduled to each service. We assume that there are M services in the smart manufacturing platform. Let S_i donate the i-th manufacturing service, and let $D(S_i, t)$ donate the earliest available time of service S_i at time t. Obviously, the earliest available time of service S_i at time t can not be earlier than t, which is expressed mathematically by (2).

$$D(S_i, t) \geqslant t \tag{2}$$

Particularly, the earliest available time of S_i at time t is equal to t if service S_i is available at time t. Besides, the states of manufacturing services are uncertain. One reason is that services join and quit dynamically in smart manufacturing systems because service providers decide when and what services to be provided to the platform. The manufacturing platform just shares the information on manufacturing resources but does not control them for service providers.

2.4. Service capability

The same service type is provided by multiple distributed service providers with different service capabilities. Besides, each service can perform one or more task types and each task type can be performed by one or more services. So, the relationship between services and task types is a many-to-many relationship. We define a $M \times N$ matrix $C = \{c_{ij}\}_{M\times N}$ to be the service capability matrix which is used to describe the many-to-many relationship between services and task types. In the service capability matrix, c_{ij} denotes the service time of task type j on service S_i.

2.5. Objective

In the static scheduling problem, there is a start time 0 and a set of jobs to be scheduled. In this considered dynamic scheduling problem, however, there are always new tasks arriving at the system over time and the system runs until we stop it. So, the only way to evaluate the schedule is by selecting a specific time window to consider. Therefore, in order to compare the performance of different scheduling policies, we consider the time window from t_s to time t_e in each iteration. The optimization objective of the considered DSSM model is to minimize the makespan of all tasks over time. Mathematically, the optimization objective is defined by Equation (3) as below.

$$\min_{\pi_j \in \Phi} \max_{1 \leq i \leq M} (t_e + D(S_i, t_e)) \mid_{\pi_j} \tag{3}$$

The *max* represents the worst situation which is the latest available time among all services after the decision by the policy π_j. And then the *min* represents the performance of the best policy π chosen from the set *Phi* of all candidate policies. As defined, π_j is the j-th scheduling policy and is a mapping from task type k and service status D to services S, as given in Equation 4.

$$\pi_j : k \times D \rightarrow S \tag{4}$$

From Equation 2, we can see that we need to firstly find the makespan of the schedule by each scheduling policy at time t. The makespan is the maximum queue time during all M services at time t. Then we compare the makespans of schedules by all different policies, and choose the policy π_j with a minimum makespan at time t.

2.6. Constraints

There are two constraints in the DSSM model. One constraint is that the start time of Task Ti should be larger that the arrival time of Ti, which is expressed mathematically by Equation 5.

$$A_i \le D(\pi(k_i, D(S_1, A_i), D(S_2, A_i), \cdots, D(S_M, A_i)), A_i) \quad (5)$$

Another constraint is that each task T_i should be distributed to one of these services which are able to execute task T_i, which means the service time of T_i on these candidate services should less than infinite. For task T_i and candidate service Sj, the second constraint can be expressed mathematically as $c_{j,k_i} < \infty$.

3. Reinforcement learning method

3.1. System framework

Generally, we need to find one or several good scheduling policies to be used in the whole scheduling period in order to generate a satisfactory real-time schedule for all coming tasks over time. Typical scheduling rules include Shortest Processing Time (SPT), Longest Processing Time (LPT), Shortest Queue Time (SQT), and Longest Queue Time (LQT). In this paper, we design an efficient scheduling agent to choose a suitable service during all candidate services for each arriving task. The deep reinforcement learning framework [18] is applied to our scheduling method. Particularly, the deep Q-learning method is applied to solve the considered DSSM problem, as shown in Figure 1.

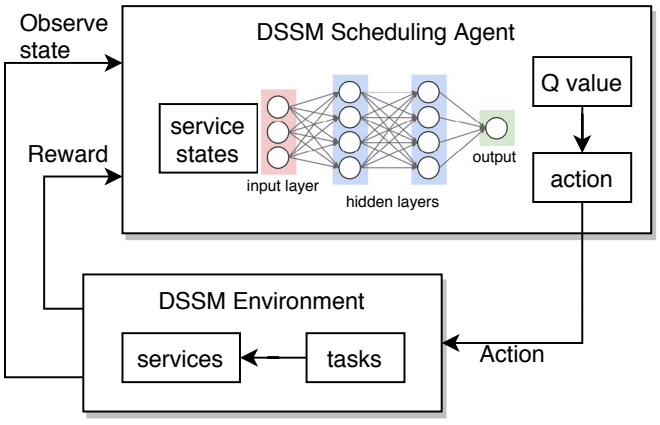

Fig. 1. System framework of the DRL-based scheduling method.

3.2. Scheduling agent

In order to represent the agent logically and train it, we need to represent it as a formula that can be optimized. The loss function is a value that shows how much our predictions differ from our actual goals. For example, a model's prediction may indicate that it sees more value when sending a task to one of its candidate services, and in fact, it can get more rewards by sending this task to another service. We want to reduce the gap between the predicted value and the target value. We define the loss function as Equation (6).

$$L = \left(r + \lambda \max_{a_{i+1}} \hat{Q}(s_{i+1}, a_{i+1}; \theta') - Q(s_i, a_i; \theta)\right)^2 \quad (6)$$

We first perform an action that is to choose a service for the current task and add this task to the queue of the chosen service. Then we see the reward and the new system state. Based on the results, we calculate the maximum target Q value and then discount it. At last, we add the current reward to the discounted target Q value, and obtain the target value.

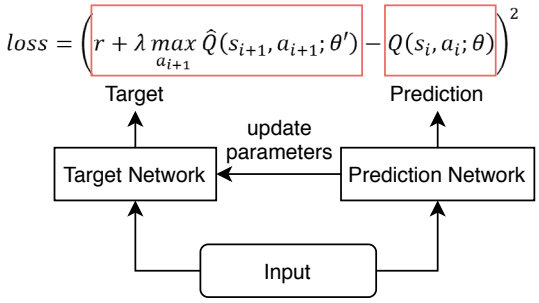

Fig. 2. Target network of the scheduling agent.

It is risky to use the same network to calculate both of the prediction value and the target value because there may be a lot of divergence between them. Hence, we apply the framework including two networks for learning during the training process, including Target Network and Prediction Network, as shown in Figure 2. The target network has the same structure as the prediction network but with some frozen parameters. For every iteration, parameters are copied from the prediction network to the target network, which results in more stable training by making the target network fixed.

4. Case study

4.1. Service capability matrix

In this section, we applied the proposed method to a real manufacturing system to show its usefulness and efficiency in details. Assume there are three task types ($N = 3$) and five services ($M = 5$). Therefore, the size of the service capability matrix C is 5×3, as shown in Table 1. From Table 1, we can see that services S_1, S_2, S_4, S_5 support two task types, and service S_3 supports one task type. Besides, each task type is supported by three different services.

4.2. Task arrivals and scheduling

We applied the proposed method in two cases. One case is with different task intervals, and another case is with the same task intervals.

Table 1. Service capability matrix.

Service Time	Task Type 1	Task Type 2	Task Type 3
S_1	*	5.3	4.7
S_2	4.1	3.6	*
S_3	*	*	5.3
S_4	4.5	4.4	*
S_5	3.2	*	4.8

4.2.1. Case 1: different task intervals

In this case, we consider all tasks arriving at different intervals, as shown in Figure 3. The right side of Figure 3 shows

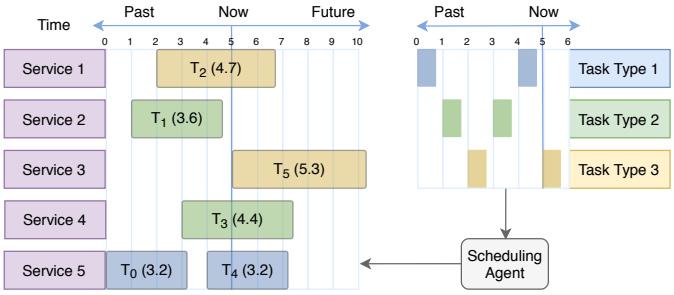

Fig. 3. Scheduling results of case 1 with different task intervals.

the randomly arriving tasks. We can see different types of tasks arrive over time. The execution process of the scheduling algorithm is listed below in multiple steps.

Step1: Task T_0 arrives at time 0 when five services are all available which means $D(S_i, 0) = 0$. Note that the type of T_0 is 1 ($k_0 = 1$), and task type 1 is supported by service S_2, S_4 and S_5. So, one service needs to be chosen from S_2, S_4 and S_5 according to the current scheduling policy. The decision-making for task T_0 is based on the following computations (7):

$$\begin{cases} D(S_2, 0^+) = D(S_2, 0) + c_{2,1} = 4.1 & \text{if } a_0(T_0) = S_2 \\ D(S_4, 0^+) = D(S_4, 0) + c_{4,1} = 4.5 & \text{if } a_0(T_0) = S_4 \\ D(S_5, 0^+) = D(S_5, 0) + c_{5,1} = 3.2 & \text{if } a_0(T_0) = S_5 \end{cases} \tag{7}$$

From the computations, we can see that S_5 obtains earlier available time than S_2 and S_4 after the action for T_0. Therefore, the scheduling agent assigns S_5 to task T_0, as shown in the left side of Figure 3.

Step2: Task T_1 arrives at time 1 and the task type of task T_1 is 2, which means $k_1 = 2$. From Table 1, we can see that services S_1, S_2 and S_4 support task type 2 with service time 5.3, 3.6 and 4.4, respectively. Hence, the earliest available times of these candidate services after the action for task T_1 are computed by (8).

$$\begin{cases} D(S_1, 1^+) = D(S_1, 1) + c_{1,2} = 6.3 & \text{if } a_1(T_1) = S_1 \\ D(S_2, 1^+) = D(S_2, 1) + c_{2,2} = 4.6 & \text{if } a_1(T_1) = S_2 \\ D(S_4, 1^+) = D(S_4, 1) + c_{4,2} = 5.4 & \text{if } a_1(T_1) = S_4 \end{cases} \tag{8}$$

Therefore, task T_1 chooses service S_2, and the earliest available time of S_2 is updated to 4.6.

Step3: Task T_2 arrives at time 2 and the type of T_2 is 3 ($k_2 = 3$). From Table 1 we can see that the candidate services for T_2 are S_1, S_3 and S_5. The earliest available times of these candidate services after the action for task T_2 are computed by (9).

$$\begin{cases} D(S_1, 2^+) = D(S_1, 2) + c_{1,3} = 6.7 & \text{if } a_2(T_2) = S_1 \\ D(S_3, 2^+) = D(S_3, 2) + c_{3,3} = 9.9 & \text{if } a_2(T_2) = S_3 \\ D(S_5, 2^+) = D(S_5, 2) + c_{5,3} = 8.0 & \text{if } a_2(T_2) = S_5 \end{cases} \tag{9}$$

Therefore, task T_2 chooses service S_1, and the earliest available time of S_1 is updated to 6.7.

Step4: At time 3, task T_3 arrives with type 2 ($k_3 = 2$) as shown in Figure 3. And the computations for task T_3 are given by (10).

$$\begin{cases} D(S_1, 3^+) = D(S_1, 3) + c_{1,2} = 12.0 & \text{if } a_3(T_3) = S_1 \\ D(S_2, 3^+) = D(S_2, 3) + c_{2,2} = 8.2 & \text{if } a_3(T_3) = S_2 \\ D(S_4, 3^+) = D(S_4, 3) + c_{4,2} = 7.4 & \text{if } a_3(T_3) = S_4 \end{cases} \tag{10}$$

Therefore, task T_3 chooses service S_4, and the earliest available time of S_4 is updated to 7.4.

Step5: At time 4, task T_4 arrives with type 1 ($k_4 = 1$). The computations for task T_4 are given by (11).

$$\begin{cases} D(S_2, 4^+) = D(S_2, 4) + c_{2,1} = 8.7 & \text{if } a_4(T_4) = S_2 \\ D(S_4, 4^+) = D(S_4, 4) + c_{4,1} = 11.9 & \text{if } a_4(T_4) = S_4 \\ D(S_5, 4^+) = D(S_5, 4) + c_{5,1} = 7.2 & \text{if } a_4(T_4) = S_5 \end{cases} \tag{11}$$

Therefore, task T_4 chooses service S_5, and the earliest available time of S_5 is updated to 7.2.

Step6: At time 5, task T_5 arrives with type 3 ($k_5 = 3$). The computations for task T_5 are given by (12).

$$\begin{cases} D(S_1, 5^+) = D(S_1, 5) + c_{1,3} = 11.4 & \text{if } a_4(T_5) = S_1 \\ D(S_3, 5^+) = D(S_3, 5) + c_{3,3} = 10.3 & \text{if } a_4(T_5) = S_3 \\ D(S_5, 5^+) = D(S_5, 5) + c_{5,3} = 12.0 & \text{if } a_4(T_5) = S_5 \end{cases} \tag{12}$$

Therefore, task T_5 chooses service S_3, and the earliest available time of S_3 is updated to 10.3. The training progress of deep network converges after around 55 iterations and then keeps stable. The final scheduling results of case 1 are shown on the left side of Figure 3.

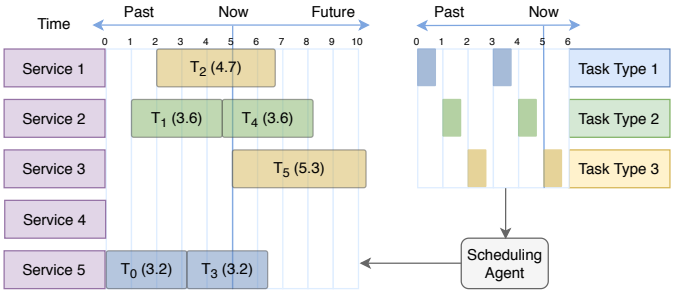

Fig. 4. Scheduling results of case 2 with the same task intervals.

4.2.2. *Case 2: same task intervals*

In order to further test the proposed approach, we applied it to another case, in which the task intervals are the same between different task types. The scheduling results are shown in Figure 4. It can be seen that the scheduling results of case 2, as shown in Figure 4, are different from the results of case 1, as shown in Figure 3. In these two cases, the manufacturing services are the same and the service capabilities are also the same. The only difference between them is the task arrival.

5. Conclusions

Dynamic task scheduling is a critical problem in a smart manufacturing environment. This paper considers the dynamic service scheduling problem in smart manufacturing and builds its mathematical model including assumptions, tasks, services, service capability, optimization objectives, and constraints. Then we propose a deep reinforcement learning-based method for this problem. In the system framework of the proposed method, the agent, environment, as well as the interaction between them, are designed. The queue times of all candidate services are considered as the system state, and the maximum queue time at the current moment is considered as the target value. Besides, we use two networks the prediction network and the target network to learn the prediction value and the target value, respectively. Two case studies with different task interval probabilities are used to illustrate the usefulness and efficiency of the proposed method. There are some details needed to be further studied such as more complex structures of tasks and a more complex relationship between services and task types. Another possible future work is to consider more optimization objectives such as manufacturing costs and service quality.

References

[1] L. Zhou, L. Zhang, A dynamic task scheduling method based on simulation in cloud manufacturing, in: Theory, Methodology, Tools and Applications for Modeling and Simulation of Complex Systems, Springer, 2016, pp. 20–24.
[2] L. Zhou, L. Zhang, C. Zhao, Y. Laili, L. Xu, Diverse task scheduling for individualized requirements in cloud manufacturing, Enterprise Information Systems 12 (3) (2018) 300–318.
[3] F. Pezzella, G. Morganti, G. Ciaschetti, A genetic algorithm for the flexible job-shop scheduling problem, Computers & Operations Research 35 (10) (2008) 3202–3212.
[4] M. Adibi, M. Zandieh, M. Amiri, Multi-objective scheduling of dynamic job shop using variable neighborhood search, Expert Systems with Applications 37 (1) (2010) 282–287.
[5] A. Rajabinasab, S. Mansour, Dynamic flexible job shop scheduling with alternative process plans: an agent-based approach, The International Journal of Advanced Manufacturing Technology 54 (9-12) (2011) 1091–1107.
[6] L. Zhou, L. Zhang, L. Ren, J. Wang, Real-time scheduling of cloud manufacturing services based on dynamic data-driven simulation, IEEE Transactions on Industrial Informatics 15 (9) (2019) 5042–5051.
[7] C. Chiu, Y. Yih, A learning-based methodology for dynamic scheduling in distributed manufacturing systems, International Journal of Production Research 33 (11) (1995) 3217–3232.
[8] P. Fattahi, A. Fallahi, Dynamic scheduling in flexible job shop systems by considering simultaneously efficiency and stability, CIRP Journal of Manufacturing Science and Technology 2 (2) (2010) 114–123.
[9] C. Sahin, M. Demirtas, R. Erol, A. Baykasoğlu, V. Kaplanoğlu, A multi-agent based approach to dynamic scheduling with flexible processing capabilities, Journal of Intelligent Manufacturing 28 (8) (2017) 1827–1845.
[10] A. A. R. Hosseinabadi, H. Siar, S. Shamshirband, M. Shojafar, M. H. N. M. Nasir, Using the gravitational emulation local search algorithm to solve the multi-objective flexible dynamic job shop scheduling problem in small and medium enterprises, Annals of Operations Research 229 (1) (2015) 451–474.
[11] X. Shen, X. Yao, Mathematical modeling and multi-objective evolutionary algorithms applied to dynamic flexible job shop scheduling problems, Information Sciences 298 (2015) 198–224.
[12] L. Zhou, L. Zhang, Y. Laili, C. Zhao, Y. Xiao, Multi-task scheduling of distributed 3d printing services in cloud manufacturing, The International Journal of Advanced Manufacturing Technology 96 (9-12) (2018) 3003–3017.
[13] L. Zhou, L. Zhang, L. Ren, Modelling and simulation of logistics service selection in cloud manufacturing, Procedia CIRP 72 (2018) 916–921.
[14] L. Zhou, L. Zhang, Y. Fang, Logistics service scheduling with manufacturing provider selection in cloud manufacturing, Robotics and Computer-Integrated Manufacturing 65 (2020) 101914.
[15] L. Zhang, L. Zhou, L. Ren, Y. Laili, Modeling and simulation in intelligent manufacturing, Computers in Industry 112 (2019) 103123.
[16] M. Kück, J. Ehm, T. Hildebrandt, M. Freitag, E. M. Frazzon, Potential of data-driven simulation-based optimization for adaptive scheduling and control of dynamic manufacturing systems, in: 2016 Winter Simulation Conference (WSC), IEEE, 2016, pp. 2820–2831.
[17] L. Zhou, L. Zhang, B. R. Sarker, Y. Laili, L. Ren, An event-triggered dynamic scheduling method for randomly arriving tasks in cloud manufacturing, International Journal of Computer Integrated Manufacturing 31 (3) (2018) 318–333.
[18] M. I. Jordan, T. M. Mitchell, Machine learning: Trends, perspectives, and prospects, Science 349 (6245) (2015) 255–260.
[19] G. Lample, D. S. Chaplot, Playing fps games with deep reinforcement learning, in: Thirty-First AAAI Conference on Artificial Intelligence, 2017.
[20] M. Volodymyr, K. Koray, S. David, A. R. Andrei, V. Joel, Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529–533.
[21] A. I. Orhean, F. Pop, I. Raicu, New scheduling approach using reinforcement learning for heterogeneous distributed systems, Journal of Parallel and Distributed Computing 117 (2018) 292–302.
[22] M. A. L. Silva, S. R. de Souza, M. J. F. Souza, A. L. C. Bazzan, A reinforcement learning-based multi-agent framework applied for solving routing and scheduling problems, Expert Systems with Applications 131 (2019) 148–171.
[23] Y. Wang, H. Liu, W. Zheng, Y. Xia, Y. Li, P. Chen, K. Guo, H. Xie, Multi-objective workflow scheduling with deep-q-network-based multi-agent reinforcement learning, IEEE Access 7 (2019) 39974–39982.
[24] C.-C. Lin, D.-J. Deng, Y.-L. Chih, H.-T. Chiu, Smart manufacturing scheduling with edge computing using multi-class deep q network, IEEE Transactions on Industrial Informatics (2019).