A Survey of AI-enabled Dynamic Manufacturing Scheduling: From Directed Heuristics to Autonomous Learning

JIEPIN DING, MINGSONG CHEN, and TING WANG, East China Normal University, China
JUNLONG ZHOU, Nanjing University of Science and Technology, China
XIN FU, University of Houston, United States
KEQIN LI, State University of New York, United States

As one of the most complex parts in manufacturing systems, scheduling plays an important role in the efficient allocation of resources to meet individual customization requirements. However, due to the uncertain disruptions (e.g., task arrival time, service breakdown duration) of manufacturing processes, how to respond to various dynamics in manufacturing to keep the scheduling process moving forward smoothly and efficiently is becoming a major challenge in dynamic manufacturing scheduling. To solve such a problem, a wide spectrum of artificial intelligence techniques have been developed to (1) accurately construct dynamic scheduling models that can represent both personalized customer needs and uncertain provider capabilities and (2) efficiently obtain a qualified schedule within a limited time. From these two perspectives, this article systemically makes a state-of-the-art literature survey on the application of these artificial intelligence techniques in dynamic manufacturing modeling and scheduling. It first introduces two types of dynamic scheduling problems that consider service- and task-related disruptions in the manufacturing process, respectively, followed by a bibliometric analysis of artificial intelligence techniques for dynamic manufacturing scheduling. Next, various kinds of artificial-intelligence-enabled schedulers for solving dynamic scheduling problems including both directed heuristics and autonomous learning methods are reviewed, which strive not only to quickly obtain optimized solutions but also to effectively achieve the adaption to dynamics. Finally, this article further elaborates on the future opportunities and challenges of using artificial-intelligence-enabled schedulers to solve complex dynamic scheduling problems. In summary, this survey aims to present a thorough and organized overview of artificial-intelligence-enabled dynamic manufacturing scheduling and shed light on some related research directions that are worth studying in the future.

CCS Concepts: • **Computer systems organization** → **Embedded software**; • **Information systems** → **Enterprise resource planning;**

Additional Key Words and Phrases: Artificial intelligence, dynamic scheduling, directed heuristic, autonomous learning, manufacturing system

This work was supported by the Natural Science Foundation of China (62272170), Shanghai Trusted Industry Internet Software Collaborative Innovation Center, Shanghai Gaofeng & Gaoyuan Project for University Academic Program Development, and "Digital Silk Road" Shanghai International Joint Lab of Trustworthy Intelligent Software (22510750100).

Authors' addresses: J. Ding, M. Chen (corresponding author), and T. Wang, East China Normal University, MoE Engineering Research Center of Software/Hardware Co-design Technology and Application, Shanghai, China; emails: 52205902006@stu.ecnu.edu.cn, mschen@sei.ecnu.edu.cn, twang@sei.ecnu.edu.cn; J. Zhou, Nanjing University of Science and Technology, School of Computer Science and Engineering, Nanjing, China; email: jlzhou@njust.edu.cn; X. Fu, University of Houston, Department of Electrical & Computer Engineering, Houston, United States; email: xfu8@central.uh.edu; K. Li, State University of New York, Department of Computer Science, New Paltz, New York, United States; email: lik@newpaltz.edu.

0360-0300/2023/07-ART307 $15.00
https://doi.org/10.1145/3590163

ACM Reference format:
Jiepin Ding, Mingsong Chen, Ting Wang, Junlong Zhou, Xin Fu, and Keqin Li. 2023. A Survey of AI-enabled Dynamic Manufacturing Scheduling: From Directed Heuristics to Autonomous Learning. *ACM Comput. Surv.* 55, 14s, Article 307 (July 2023), 36 pages.
https://doi.org/10.1145/3590163

1 INTRODUCTION

Along with the prosperity of **artificial intelligence (AI)**, **industrial internet of things (IIoT)**, and **human-cyber-physical systems (HCPSs)** techniques, we are witnessing an industrial revolution toward industry 5.0, aiming to create a human-centric and highly autonomous manufacturing mode for individualized and digitalized products [1]. Unlike industry 4.0, industry 5.0 substantially improves the intelligence levels of traditional manufacturing systems for better flexibility, mass customization, and higher productivity. Especially, industry 5.0 emphasizes that manufacturing should raise awareness of its contribution to society and develop toward sustainability and resilience [2]. To achieve such grand visions, manufacturing systems are required to promote their capability of autonomous learning. In this situation, scheduling as a key part of manufacturing systems has naturally pivoted to be data driven, where dynamic scheduling decisions are made precisely based on the learning from both historical system data and real-time communication between humans, machines, and sensors.

The manufacturing scheduling problem mainly involves two kinds of entities, i.e., tasks and services [3]. Manufacturing tasks refer to individualized requirements submitted by consumers, while manufacturing services denote various kinds of resources (e.g., machines, raw materials, human resources) encapsulated by specific modern encapsulation technologies [4]. To facilitate the task-to-service mapping, manufacturing tasks can be decomposed into subtasks with finer granularities to fit their target manufacturing services. As a classic NP-hard problem, static scheduling tries to figure out an optimal solution to task allocation on services under various **quality of service (QoS)** constraints (e.g., makespan, cost), assuming that the settings of manufacturing tasks and services are all known a priori. However, such an assumption is not always true due to ubiquitous uncertain disruptions (e.g., task arrival time, task requirements, service breakdown duration) in a real-world manufacturing environment. As a result, solutions obtained by existing static scheduling methods may often violate the specified QoS constraints during practical manufacturing processes, resulting in inestimable losses. In order to cope with such uncertain disruptions, more and more dynamic scheduling methods [3] are investigated to keep scheduling processes moving forward, which further complicates the manufacturing scheduling problem. Due to the problem of "state space explosion," when dealing with large-scale manufacturing scheduling problems, it is extremely time-consuming to achieve optimal solutions. Therefore, traditional dynamic scheduling methods are not suitable for large-scale manufacturing, especially when manufacturing systems require real-time adjustments to uncertain disruptions to ensure manufacturing progress.

To enable large-scale dynamic scheduling, AI-based methods have gained increasing attention due to their ability to quickly and efficiently approximate optimal solutions. As shown in Figure 1, existing AI-based dynamic manufacturing algorithms can be classified into two categories, i.e., directed heuristic-based methods and autonomous learning-based methods. Based on various iterative evolution schemes, the directed heuristic-based methods can quickly achieve near-optimal solutions. However, such methods typically require complete information on both tasks and services to make scheduling decisions, which is difficult to obtain in practice within an uncertain environment. Generally, when encountering uncertain disruptions during manufacturing, directed heuristic-based methods need to figure out optimal task-to-service mappings promptly

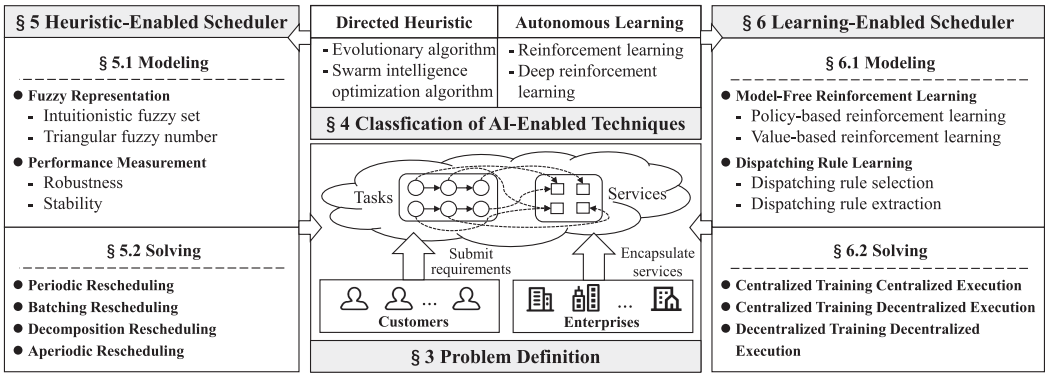

Fig. 1. Overview and organization of this article.

based on the estimated information of remaining tasks and services, and conduct the rescheduling if necessary by reallocating subsequent unfinished tasks to their newly assigned services. Although the directed heuristic-based methods are easy to implement, frequent rescheduling tasks may easily result in the notorious "schedule nervousness" problem, which strongly limits their applicability for mass production. Therefore, how to reasonably adjust rescheduling frequency becomes one of the most notable problems in the design of directed heuristic-based methods. As an alternative, autonomous learning-based methods enable decision-making based on various effective scheduling policies, which are autonomously learned by agents from their frequent interactions with the scheduling environment. Unlike directed heuristic-based methods, autonomous learning-based methods are based on sequential decision-making, which dispatches one subtask at each decision point based on the observed state of the scheduling environment in real time. Although autonomous learning-based methods are powerful in addressing **dynamic scheduling problems (DSPs)** without taking rescheduling frequency into account, they suffer from the problems of low training efficiency to achieve expected policies. The goal of this article is to detail various kinds of solutions (i.e., heuristic-enabled schedulers in Section 5 and learning-enabled schedulers in Section 6) to show how the above two problems (i.e., schedule nervousness and low training efficiency) are addressed, respectively.

Figure 1 shows the overview and organization of this article. Based on the two categories of disruptions (i.e., service-related disruptions and task-related disruptions) caused by uncertain disruptions in the manufacturing process, this article first introduces a general formalization for different dynamic scheduling of problems (e.g., job shop scheduling, flow shop scheduling, parallel scheduling, and production planning). Next, a systematic survey of two mainstream AI-based dynamic manufacturing techniques (i.e., directed heuristics and autonomous learning) is presented. Finally, this article details existing effective solutions (i.e., heuristic-enabled schedulers and learning-enabled schedulers) to handle various uncertain disruptions during the manufacturing process from the perspectives of both modeling and solving. Compared with existing related surveys of manufacturing scheduling problems, this article makes the following three major contributions:

(1) A comprehensive view of AI-enabled techniques for dynamic scheduling (see Section 4). Aiming to reflect the application status of AI-enabled techniques in the manufacturing field, this article conducts a bibliometric analysis using VOSviewer. Two kinds of nature-inspired AI-enabled techniques are explored for DSPs, revealing why DSP solutions evolve from directed heuristics to autonomous learning.
(2) A purposeful elaboration and comparison of AI-enabled schedulers for dynamic scheduling (see Sections 5 and 6). From the perspectives of both modeling and solving, this article

Table 1. Comparison of Related Surveys

Reference	Problem	Environment	Method
Giret et al. [5]	Job shop scheduling	Static	Hybrid heuristics
Rossit et al. [6]	Flow shop scheduling	Static	Objective-based analysis
Edis et al. [7]	Parallel machine scheduling	Static	Exact/approximation methods
Ojstersek & Brezocnik [8]	Production scheduling	Static	Evolutionary computation
Robert et al. [9]	Resource-constrained scheduling	Static	Hybrid metaheuristics
Singh et al. [10]	Manufacturing scheduling	Static	Metaheuristic techniques
Priore et al. [11]	Manufacturing scheduling	Dynamic	Machine learning
Zarandi et al. [3]	Manufacturing scheduling	Dynamic	Intelligent approaches
Liu et al. [12]	Manufacturing scheduling	Dynamic	DRL framework
Our survey	Manufacturing scheduling	Dynamic	Artificial intelligence

elaborates on the pros and cons of two kinds of AI-based dynamic schedulers (i.e., heuristic-enabled schedulers and learning-enabled schedulers) and gives examples to illustrate which dynamic scheduling scenarios they apply to.

(3) Identification of critical challenges and future opportunities (see Section 7). The end of this article makes a deep discussion of various research challenges confronting modeling and solving DSPs caused by ubiquitous uncertain disruptions in the real-world manufacturing environment. Meanwhile, corresponding research opportunities are explored to inspire future work, which promotes the effective and sustainable development of modern manufacturing systems.

The rest of this article is organized as follows. Section 2 makes a comparison between related surveys on manufacturing scheduling problems. Section 3 defines the DSPs and presents the motivation of AI-enabled techniques for dynamic scheduling. Section 4 identifies the categories of AI-enabled techniques for dynamic scheduling. Section 5 discusses the application of heuristic-enabled schedulers on dynamic scheduling. Section 6 introduces the application of learning-enabled schedulers on dynamic scheduling. Finally, Section 7 concludes the article and presents future opportunities.

2 RELATED WORK

Common scheduling problems in manufacturing systems include the **job shop scheduling problem (JSSP)** [5], **flow shop scheduling problem (FSSP)** [6], parallel machine scheduling problem [7], production planning [8], and resource-constrained project scheduling [9], which can be collectively referred to as manufacturing scheduling problems. Table 1 makes a comparison between related surveys on manufacturing scheduling problems. In this table, the first five surveys focused on specific categories of manufacturing scheduling problems, while the remaining five works conducted comprehensive investigations on various manufacturing scheduling problems. Note that the first six surveys do not take uncertain disruptions into account. Since the first six works investigated techniques for solving static scheduling problems, none of them can achieve optimal solutions in a dynamic scheduling environment. To achieve effective dynamic adaption to ubiquitous uncertain disruptions in real-world manufacturing environments, researchers focus on studying emerging AI-enabled techniques.

As shown in the table, three surveys (i.e., [3, 11, 12]) investigated several AI-enabled techniques for solving DSPs. For example, Priore et al. [11] provided a systematic review of a real-time scheduling system that uses different machine learning approaches to modify dispatching rules to achieve rapid response in a dynamic manufacturing environment, such as inductive learning, **support vector machine (SVM)**, and **reinforcement learning (RL)**. In [3], Zarandi et al.

investigated five major AI-enabled techniques for dynamic scheduling, which are fuzzy logic, expert systems, machine learning, stochastic local search optimization algorithms, and constraint programming. For more complex DSPs in cloud manufacturing, Liu et al. [12] summarized a general **deep reinforcement learning (DRL)** framework, under which machines can automatically extract high-level features of tasks and services to learn inherent scheduling patterns.

Unlike the surveys in [3, 11, 12], this article comprehensively considers AI-enabled techniques, which are classified into two categories: (1) directed heuristic-based methods that include evolutionary computation and swarm intelligence optimization and (2) autonomous learning-based methods that contain RL and DRL. Instead of introducing a general AI framework, this article focuses on elaborating how AI-enabled techniques can achieve dynamic control for different scheduling problems from the perspectives of both modeling and solving.

3 PROBLEM DEFINITION AND MOTIVATION

In this section, we first introduce the preliminaries of DSPs in manufacturing (see Section 3.1), including problem description (see Section 3.1.1), disruption categories (see Section 3.1.2), and solution strategies (see Section 3.1.3). Next, we describe the motivation of existing AI-enabled techniques for dynamic scheduling in Section 3.2.

3.1 Dynamic Scheduling in Manufacturing

3.1.1 Problem Description. According to the ever-changing manufacturing environment, dynamic scheduling arranges tasks for candidate services to allocate resources rationally and update solutions consecutively. Typically, the scheduling procedure is decomposed into two sub-problems: (1) service assignment that refers to the process of selecting appropriate services for each subtask based on non-functional requirements, i.e., QoS [4], and (2) task sequencing that focuses on minimizing idle service time by reasonably arranging congested tasks on a service, since task congestion may occur when a large number of tasks pile up on one service at the same time.

Assuming that i, j, and k are the indices of tasks, subtasks, and services, $S_{i,j}$ and $E_{i,j}$ represent the start time and completion time of each subtask, respectively. Based on these notations, the definition of a general DSP is mathematically described as minimizing makespan $C_{\max}$ under a set of constraints:

$$\min: \ C_{\max} = \max_{\forall i,j} E_{i,j} \tag{1}$$

$$s.t. \begin{cases} \sum_{k=1}^{K} \lambda_{i,j,k} = 1 \\ S_{i,j} = \begin{cases} A_i & \text{if} \quad j = 1 \\ \max(E_{i,j-1}, E_{JI_{k,x},OI_{k,x}}) & otherwise \end{cases} \\ E_{i,j} = S_{i,j} + \sum_{k=1}^{K} \lambda_{i,j,k} t_{i,j,k} \\ other\ constraints\ on\ tasks\ or\ services, \end{cases} \tag{2}$$

where A_i indicates the arrival time of task J_i and $t_{i,j,k}$ is the processing time of subtask $O_{i,j}$ performed on service M_k. Here, $JI_{k,x}$ and $OI_{k,x}$ are the task index and subtask index of the x^{th} subtask performed on service M_k, respectively. The notation $\lambda_{i,j,k}$ is an indicator variable, indicating subtask $O_{i,j}$ is processed by service M_k if $\lambda_{i,j,k} = 1$. The first constraint specifies that each subtask is performed on only one service. The second constraint on $S_{i,j}$ indicates that the start time of each subtask depends on the completion time of adjacent subtasks of the same task or the completion time of adjacent subtasks performed on the same service. The third constraint means that each subtask must be completed before executing the next subtask. Other constraints depend on the

specific non-functional requirements of tasks or services for scheduling problems, such as the energy constraints of services and subtask sequence constraints within different tasks. For example, energy-efficient scheduling focuses on rationally allocating resources to complete all the manufacturing tasks within the scope of the limited energy consumption. For FSSPs, it is necessary to add a constraint (i.e., the execution sequence of subtasks in different tasks is exactly the same), which is suitable for batch production. Due to the unpredictable disruptions in real-world manufacturing processes, it is hard to figure out exact parameters (e.g., A_i and $t_{i,j,k}$) in practice. For example, we cannot accurately predict the arrival time A_i of future tasks in advance. Furthermore, services may be interrupted due to sudden machine breakdown or material shortage, which may result in extended processing time $t_{i,j,k}$.

Due to the promising capability to approximate optimal solutions, AI-enabled techniques can achieve reasonable resource allocation solutions for complex DSPs in a quick and efficient manner. To evaluate the effectiveness and efficiency of such techniques from different perspectives, various kinds of benchmark datasets have been designed and open-sourced. However, due to privacy concerns, most of these datasets are of small scale [13–17], which cannot fully reflect the performance of state-of-the-art AI-enabled techniques for their quantitative comparison. To address this issue, various data augmentation methods have been studied to extend small-scale benchmarks. By enlarging the sizes of datasets or dataset elements according to specific distributions, e.g., Poisson distribution and Uniform distribution [18, 19], datasets for large-scale scheduling can be generated to accommodate complex DSPs.

3.1.2 Disruption Categories in Manufacturing Process. According to the definitions of manufacturing entities, the uncertainty of the parameters aforementioned can be mainly classified into the following two categories [20], i.e., service-related disruptions and task-related disruptions.

Service-related Disruptions. Service-related disruptions are caused by imprecise QoS or sudden interruption of services. Since a solution is optimized based on the QoS (e.g., processing time) predicted by analyzing historical data, whether the manufacturing processes of tasks follow the solution depends on the accuracy of predictions. Specifically, if a service is unavailable for any task as expected, the subsequent tasks will fail to execute as scheduled. Moreover, severe weather such as blizzards and earthquakes may cause the completed subtasks to be blocked during transportation to the subsequent subtasks on time.

Task-related Disruptions. Task-related disruptions are typically caused by urgent tasks, the arrival of new tasks, due date changes, or task cancellation. Note that the arrival of new tasks is one of the most common forms of dynamics. Since the arrival time of a manufacturing task depends on its submission time, it is difficult to predict it in advance. In this case, once an urgent manufacturing task arrives, the scheduling performance will be affected inevitably.

3.1.3 Solution Strategies. Based on the time points of responses to uncertain disruptions, solution strategies are classified into two categories: (1) proactive scheduling that provides early warning measures before the execution of a schedule and (2) reactive scheduling that conducts real-time adjustment during the manufacturing process. Typically, proactive scheduling requires the interrupted time of a service and its interruption duration in advance. Based on the analysis of historical data, the distribution of interrupted information (i.e., corresponding interrupted time and duration) follows a Uniform distribution.

Table 2 comprehensively compares the AI-enabled techniques (i.e., directed heuristics and autonomous learning) used for solving DSPs with service- and task-related disruptions. From this table, we can find that directed heuristic-enabled methods are widely used to solve DSPs with task-related and service-related disruptions. This is mainly because directed heuristic-enabled methods can be easily implemented to obtain superior results by predicting interruptions in advance or

Table 2. AI-enabled Techniques for DSPs with Service- and Task-related Disruptions

Service-related Disruptions			
Cause	**Strategy**	**Problem**	**Algorithm**
Machine breakdown	P	JSSP	GA [21]
	P	FJSSP	GA [22]
	P	FJSSP	NSGA [23]
	P	FJSSP	NSGA [19]
	R	FSSP	ABC [24]
	P	MPSP	GA [18]
Stochastic processing time	P	JSSP	SA + TS [25]
	P	FMS	GA + SA [26]
	P	FMS	GA [27]
	P	SMSP	SA [28]
	P	FJSSP	VNS + Q-learning [29]
	P	FSSP	K-means + GA [30]
Task-related Disruptions			
Cause	**Strategy**	**Problem**	**Algorithm**
Dynamic task arrivals	R	JSSP	GEP [31]
	R	FJSSP	double DQN [32]
	R	JSSP	double loop DQN [33]
	R	FSSP	VNS [34]
	R	JSSP	dueling double DQN [35]
	R	BSP	VNS + Q-learning [36]
	R	SMSP	NSGA [37]
	PR	FSSP	GA [38]
	PR	MPSP	PG [39]
Dynamic requests for various products	PR	FJSSP	DQN [40]
	PR	FMS	DQN [41]
	PR	BSP	Q-learning [42]
Combination of Service-related and Task-related Disruptions			
Cause	**Strategy**	**Problem**	**Algorithm**
Random task arrivals and machine breakdown	R	FSSP	GA [43]
	PR	JSSP	GA + TS [44]
	PR	FJSSP	NSGA [45]
	PR	FSSP	NSGA [46]

[1]We use the symbols P, R, and PR to indicate proactive strategy, reactive strategy, and a combination of the two, respectively.
[2]Flexible job shop scheduling problem (FJSSP), flexible manufacturing system (FMS), single machine scheduling problem (SMSP), multi-project scheduling problem (MPSP), batch scheduling problem (BSP).
[3]Genetic algorithm (GA), nondominated sorting genetic algorithm (NSGA), artificial bee colony (ABC), simulated annealing (SA), tabu search (TS), variable neighborhood search (VNS), gene expression programming (GEP), deep Q-network (DQN), policy gradient (PG).

triggering rescheduling once an uncertain disruption occurs. Since sequential decision-making is suitable to achieve real-time scheduling of consecutive arrival tasks in manufacturing systems, autonomous learning-enabled methods mainly focus on addressing task-related disruptions rather than service-related disruptions. Specifically, a proactive strategy is often used to deal with service-related disruptions, since interrupted information (e.g., interrupted time of a service and

its interruption duration) can be predicted based on historical data. Unlike uncertain disruptions caused by services, personalized task information (e.g., arrival time and manufacturing requirements) submitted by customers cannot be easily pre-determined in advance. Therefore, reactive strategies (i.e., triggering rescheduling once a task arrives) are often used to address task-related disruptions.

3.2 Motivation Behind Intelligent Dynamic Scheduling

To solve scheduling problems, exact algorithms can quickly figure out optimal solutions for small-scale manufacturing, including dynamic programming [47], branch-and-bound approach [48, 49], and constraint programming [50]. For example, Pang and Le [47] formulated a scheduling problem as a non-convex mixed-integer nonlinear program. Based on a reachable graph, they presented a dynamic programming approach to optimize energy consumption. Experiments on datasets demonstrate that the proposed method achieves significantly less deviation and computational time compared to baseline approaches. Cheng et al. [51] developed an extended branch-and-bound algorithm to obtain an optimal solution to a permutation FSSP, which takes advantage of the sufficient precondition-based dominance relation to reduce search load efficiently. Later, to improve the performance of branch-and-bound algorithms, they designed a special construct based on dominance properties (i.e., upper bounds or lower bounds) to enforce a sequence of non-overlapping activities [52, 53]. In [50], Zeballos presented a constraint programming methodology that consists of both a model and a search strategy to solve flexible scheduling problems. In this approach, different features in industrial environments are considered, including the capacity of tool magazines, limits on set-up, the due date of tasks, and constraints on machine tools. Moreover, by assigning machines and tool types to part subtask procedures, this method uses a domain-specific search strategy to guide the movement through the search space. To clarify the relationship between the computation time of this approach and the scale of scheduling problems (i.e., the number of machines and the number of parts), they developed comparative experiments. From the experimental results applied to various test problems, the formulation performance of this approach is sensitive to the increasing number of both machines and parts. Furthermore, the formulation performance is more influenced by augment in the number of machines than in the number of parts. To improve solution efficiency, Google OR-tools serve as a set of open-source and fast operational tools for solving optimization problems, providing a unified interface to call common solvers such as GLPK, Gurobi, and CPLEX [54].

Since scheduling problems are NP-hard, along with the increasing scale of manufacturing, skyrocketing computation complexity hinders exact algorithms from effectively achieving large-scale manufacturing. To search for near-optimal solutions within an acceptable computation time, Li et al. [55] presented a modified dynamic programming algorithm, which designs a transition sequence heuristic function and evaluation function to explore potential states. Note that each potential state in a problem is treated as a node. Since the maximum number of nodes per stage is strictly controlled within a reasonable value, this algorithm explores only the most promising states, which greatly reduces search space. Moreover, by formulating online printing shop scheduling problems as mixed-integer linear programming formulations, Lunardi et al. [56] designed a constraint programming optimization solver to fully explore the structure of underlying correlations through modeling language.

Although these approximate algorithms can accelerate to tackle complex scheduling problems based on a series of static assumptions, they suffer from poor adaption to real-world manufacturing environments. As a promising way, AI-enabled techniques are considered to be effective as well for dynamic scheduling due to the following three reasons. First, instead of pursuing exact solutions, AI-enabled technologies aim to obtain near-optimal solutions within a reasonable

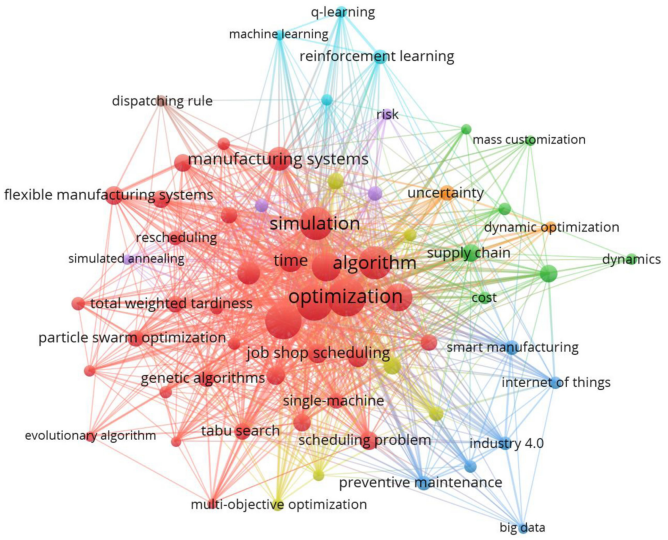

Fig. 2. Keyword co-occurrence network related to dynamic scheduling in the manufacturing field.

time. Second, although AI-enabled techniques (e.g., directed heuristic-enabled methods) focus on solving static scheduling problems, they can adjust solutions by combining rescheduling strategies. Third, due to sequential decision-making, AI-enabled techniques such as RL assign subtasks to machines incrementally to achieve dynamic adaption. Therefore, the following sections will focus on AI-enabled techniques for DSPs.

4 CLASSIFICATION OF AI-ENABLED TECHNIQUES FOR DYNAMIC SCHEDULING

This section first presents an overview of AI-enabled techniques for solving DSPs (see Section 4.1). Then, it details two kinds of AI-enabled techniques for dynamic scheduling, i.e., directed heuristics (see Section 4.2) and autonomous learning (see Section 4.3).

4.1 Overview of AI-enabled Techniques for Dynamic Scheduling

To comprehensively understand dynamic scheduling in the manufacturing field, we conducted a bibliometric analysis of relevant works using VOSviewer. For the topic of *dynamic scheduling*, we filtered out irrelevant works and collected a total of 891 papers, where all the papers are extracted from the core collection of Web of Science dating from 2012 to 2021. By setting the threshold for the minimum number of occurrences of a keyword to 15, Figure 2 shows the keyword co-occurrence network related to dynamic scheduling in the manufacturing field, where each circle indicates a keyword and each line indicates the strength of relationships between two keywords. Note that the larger the circle size is, the more representative the keyword is. Based on the occurrence relationship of different keywords, there are seven clusters indicated by different colors. As shown in the figure, we can find that the red one is the largest cluster, while the remaining six clusters are all smaller, where the red one mainly contains keywords such as *optimization*, *algorithm*, *simulation*, *particle swarm optimization*, and *genetic algorithm*. In other words, directed heuristic-related techniques represented by particle swarm optimization algorithms and genetic algorithms have received great attention in solving DSPs in the manufacturing field. The keywords in the bright blue cluster mainly involve autonomous learning-related techniques such as *machine learning*, *reinforcement learning*, and *Q-learning*. Moreover, keywords such as *dynamics*, *uncertainty*, and

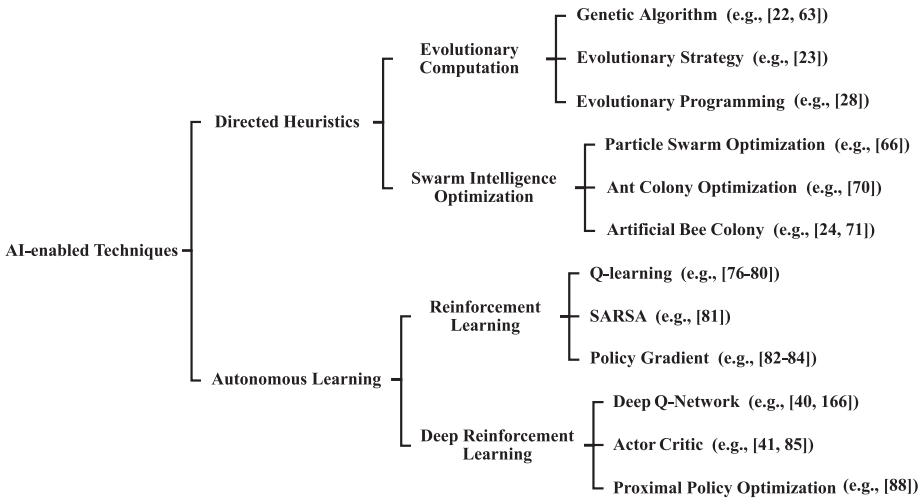

Fig. 3. Classification tree of AI-enabled techniques for dynamic scheduling in the manufacturing field.

dynamic optimization show that researchers are not limited to static scheduling problems but take into account uncertain disruptions in manufacturing processes.

Similar to the work [57] where Telikani et al. regarded the evolutionary computation approaches as a kind of AI technique to improve the performance of machine learning, in this article, we define the notation "AI-enabled techniques" as mainstream AI techniques including heuristic search, reinforcement learning, and deep learning [58]. To visualize the classes of AI-enabled techniques in the domain of dynamic scheduling, Figure 3 gives the classification tree of dynamic scheduling in the manufacturing field, where AI-enabled techniques are classified into two categories, i.e., directed heuristics and autonomous learning. As shown in the figure, the directed heuristics mainly consist of evolutionary algorithms (see Section 4.2.1) and swarm intelligence optimization algorithms (see Section 4.2.2), while autonomous learning involves reinforcement learning (see Section 4.3.1) and deep reinforcement learning (see Section 4.3.2).

Since manufacturing scheduling focuses on dynamic response timeliness in practical manufacturing scenarios, computation complexity analysis plays an important role in measuring the performance of AI-enabled techniques for solving DSPs. Here, the time complexity of directed heuristic-enabled methods is typically $O(gmn)$, where g is the number of generations, n is population size, and m is the size of individuals. As an alternative, the training complexity for autonomous learning-enabled methods (e.g., Q-learning, PG, DQN) is generally defined as $O(p|s|^2|a|)$, where p is the number of iterations, $|s|$ is the number of states, and $|a|$ is the number of actions. Note that the parameters m, $|s|$, and $|a|$ are strongly affected by the size of scheduling problems themselves. Therefore, when the size of a dynamic scheduling problem increases, the complexity of both types of AI-enabled techniques will increase dramatically. To reduce the solving/training time of AI-enabled techniques, there are two classes of widely used solutions: (1) improve the optimization/learning efficiency of each generation/iteration to accelerate convergence speed or (2) reduce the optimization/learning time of each generation/iteration.

To visualize the development trends of these two kinds of AI-enabled techniques for manufacturing, Figure 4 shows the number of papers related to dynamic scheduling over the past decade. As illustrated in the figure, the total number of papers using directed heuristics and autonomous learning for dynamic scheduling has steadily increased from 2012 to 2018, and has grown rapidly

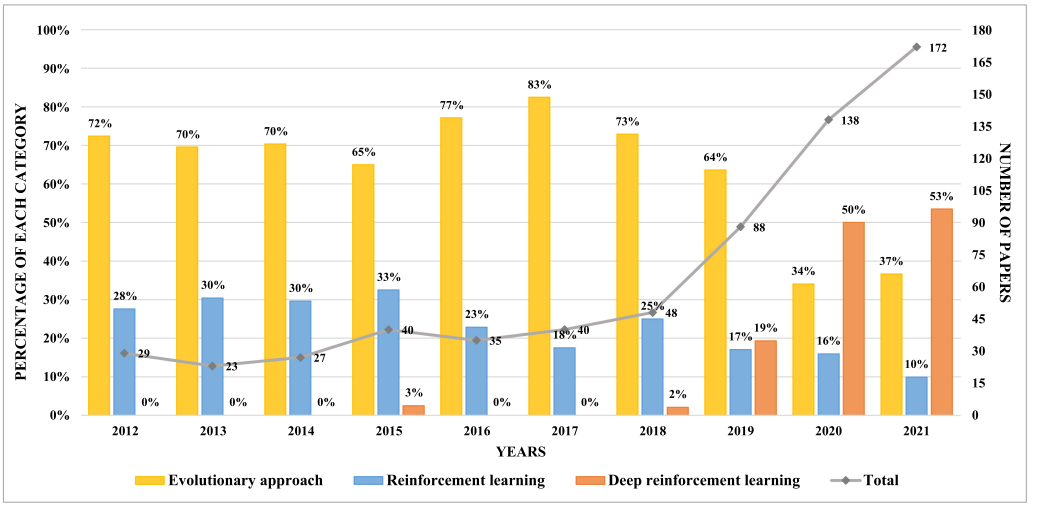

Fig. 4. Number of papers related to AI-enabled techniques for dynamic scheduling.

since 2019. These AI-enabled technologies aforementioned have received increasing attention due to the following two reasons:

(1) To achieve better flexibility, mass customization, and higher productivity, manufacturing systems are pushed to improve their intelligence levels. As an essential part of manufacturing systems, intelligent scheduling can improve the utilization rate of idle resources and alleviate the pressure of resource shortage.
(2) Since DSPs are NP-hard problems, traditional approaches cannot search for large-scale scheduling within an acceptable time. To achieve efficient large-scale scheduling, AI-enabled techniques quickly search for near-optimal solutions rather than optimal solutions.

As shown in the figure, directed heuristics for dynamic scheduling attracted great attention from 2012 to 2021 due to easy implementation and high search efficiency. However, mass customization requirements make it difficult for directed heuristics to achieve fully dynamic adaptation by adjusting solutions through rescheduling. Unlike directed heuristics, RL is suitable for DSPs due to sequential decision-making. Suffering from the issue of "state space explosion," RL had not been widely used to solve DSPs until rapid growth occurred in 2019, and it even surpassed directed heuristics in 2020. This is because DRL uses the powerful representation ability of deep learning to fit Q-tables or directly fit policies to effectively alleviate the "state space explosion" issue. Furthermore, the blockbuster Alpha Go from 2016 to 2017 has greatly promoted the development of DRL, which can adjust actions through interacting with the environment.

4.2 Directed Heuristics

Based on the theory of "survival of the fittest" [22], directed heuristics continuously improve the quality of a population through the guidance of well-behaved individuals (i.e., solutions). Figure 5 shows the procedure of directed heuristic-based methods, which mainly consists of five steps, i.e., encoding, initialization, evolution, updating population, and decoding. For the encoding step, one-dimensional representation and two-dimensional representation are the most common forms for a solution, where two-dimensional representation shows the service assignment and task sequencing information more intuitively, while one-dimensional representation encodes the information of both dimensions into one [59]. For the initialization step, the population is generally composed

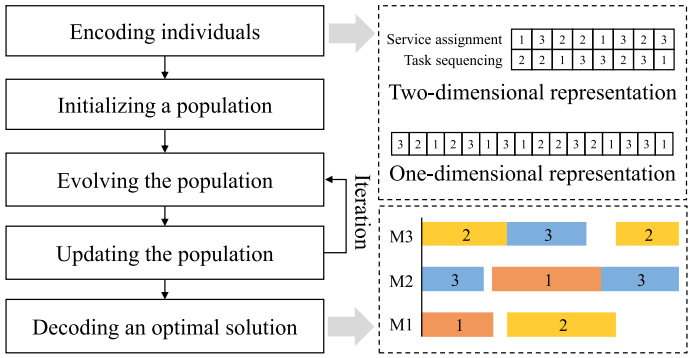

Fig. 5. A general procedure of directed heuristic-based methods.

of random individuals based on an encoding operator. Since the quality of an initial population directly affects convergence speed and final solutions, existing methods designed problem-specific rules to improve the quality of the initial population [60, 61]. For each iteration, individuals evolve according to information sharing in the population to generate offspring individuals. To effectively update the population, the original individual will be replaced by its better-behaved offspring individual. Directed heuristics repeat evolving and updating the population until algorithms converge, and then decode an optimal solution. Based on the start and end time of each subtask calculated in the decoding step, the solutions can be visualized by a Gantt chart. In this article, we mainly introduce two typical classes of directed heuristic-based methods, i.e., evolution computation and swarm intelligence optimization, as follows.

4.2.1 Evolutionary Computation. By simulating the collective learning process of a population composed of individuals, evolutionary computation algorithms effectively update the population. Due to easy implementation and high search efficiency, evolutionary computation algorithms are widely used for scheduling problems. To find a near-optimal solution, a population evolves under guidance from well-behaved individuals by performing selection, crossover, and mutation operators. Here, selection operators give well-adapted individuals more opportunities to be retained in the next generation population than poorly adapted individuals. Crossover operators generate offspring individuals by changing the information of individuals, and mutation operators produce new individuals randomly to the population. Research on evolutionary computation algorithms is classified into three categories, i.e., **genetic algorithms (GAs)** [18, 22, 62–64], evolution strategies [23], and evolutionary programming [28].

Since GA is the most common evolutionary computation algorithm, this subsection takes it as an example to detail evolutionary processes. In GA, an individual represents a candidate solution, which contains the information of both service assignment and task sequencing. Based on selection, crossover, and mutation operators, a population covering multiple individuals evolves continuously to produce offspring individuals. Since GA follows the "survival of the fittest" to update individuals in the population, it ensures that the population can continue to approach an optimal solution. In other words, individuals with good behaviors stand out from the candidates and are retained in the next population.

4.2.2 Swarm Intelligence Optimization. Similar to evolutionary computation algorithms, swarm intelligence optimization algorithms mainly simulate the population behavior of insects, herds, birds, and fish. Typically, the evolutionary process of a population can be regarded as a swarm intelligence only if it satisfies two conditions [65]: (1) self-organization that allows an

initial disordered population to determine the movement trajectories of food search through local interactions among individuals and (2) the division of labor that helps different professional individuals (well behaved) perform different tasks at the same time.

Based on the simulation of different animal behaviors, swarm intelligence optimization algorithms mainly include **particle swarm optimization (PSO)** algorithms [66–69], **ant colony optimization (ACO)** algorithms [70], **artificial bee colony (ABC)** algorithms [24, 71, 72], **flower pollination algorithms (FPAs)** [73], **biogeography-based optimization (BBO)** algorithms [74], and whale optimization algorithms [75]. In these algorithms, individuals search for food in a cooperative way and constantly improve their search directions by learning from their own experience and the experience of other individuals. More details on how these algorithms solve the DSPs will be presented in Section 5.

4.3 Autonomous Learning

4.3.1 Reinforcement Learning. The learning process of RL is to update a policy by successively executing actions to maximize cumulative rewards. Common RL algorithms mainly include Q-learning [8, 76–80], SARSA [81], **policy gradient (PG)** [82–84], and **actor-critic (AC)** [41, 85]. In RL, a state S_t has Markov property if and only if state S_{t+1} depends only on state S_t at time t. In other words, the future is independent of the past given the present. When a sequence of random states in a process has Markov property, the process is a **Markov decision process (MDP)**. Based on the description of MDP, RL modeling mainly involves the following four scheduling components: states, action, policy, and reward. Specifically, an agent performs different actions at each state and chooses one of these actions. According to the action executed by the agent, the environment will change to another state and give feedback on a reward. Since dynamic scheduling is the process of assigning tasks to appropriate services one by one based on the current state of a system, and each task assignment causes a change in the state of a system, almost all the DSPs can be formalized into MDPs.

4.3.2 Deep Reinforcement Learning. Inspired by neural networks that simulate the human brain for analysis and learning, deep learning mimics the mechanism of the human brain to interpret data [86]. By combining low-level features, a distributed feature representation of the data can form more abstract high-level representations, attribute categories, or features. Although RL is good at DSPs due to sequential decision-making, it suffers from the problem of state space explosion for large-scale scheduling. Especially in a real-world manufacturing environment, large state space severely deteriorates the performance of training. To effectively solve complex scheduling problems, the DQN-based methods introduce deep learning to learn scheduling policies, whose procedures are presented in Figure 6. As shown in the figure, an agent observes a state s_t of the current manufacturing environment, which is input to a neural network as a feature vector. Based on the feature vector, the neural network outputs Q-values, which represent the scores obtained by the interactions between the agent and the environment. Based on the Q-values, the agent selects an action, represented by a candidate operation or a dispatching rule. Finally, the manufacturing environment returns a reward to the agent and changes to another state. We can find that DQN does not require massive storage space to learn an optimal scheduling policy. Besides DQN, more DRL methods have been investigated to solve DSPs, such as AC [87] and **proximal policy optimization (PPO)** [88], where the learning procedures of these algorithms are similar to DQN.

5 HEURISTIC-ENABLED SCHEDULER

This section reviews the research works of heuristic-enabled schedulers on dynamic scheduling. These works focus on modeling and solving DSPs. We first introduce the heuristic-enabled

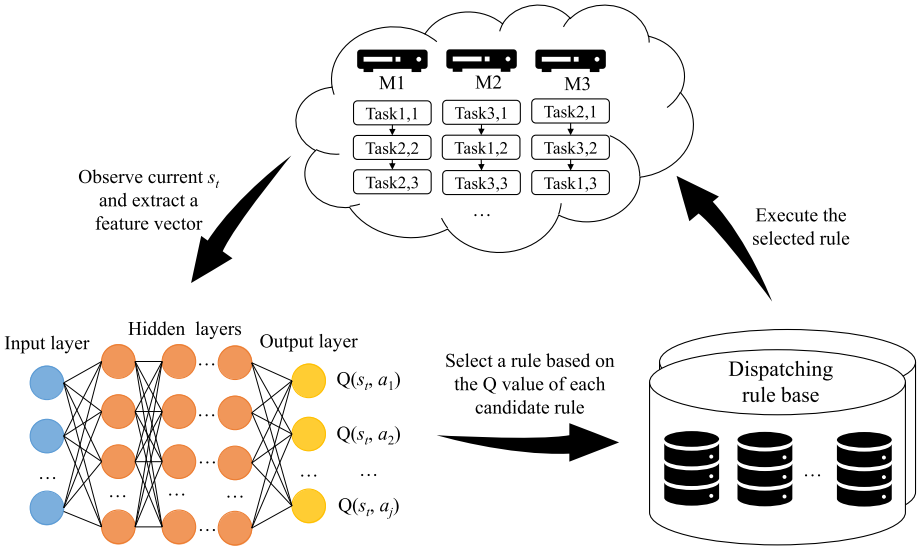

Fig. 6. Procedures of DQN-based methods [40].

modelers in Section 5.1, including fuzzy representation (see Section 5.1.1) and performance measurement (see Section 5.1.2). Then, we describe heuristic-enabled solvers in Section 5.2, including periodic rescheduling (see Section 5.2.1), batch rescheduling (see Section 5.2.2), decomposition-based rescheduling (see Section 5.2.3), and aperiodic rescheduling (see Section 5.2.4). Finally, we summarize heuristic-enabled schedulers in Section 5.3.

5.1 Heuristic-enabled Modeler

5.1.1 Fuzzy Representation. To enable reasonable allocation of manufacturing resources, most of the existing works focus on building optimization models with precise QoS. However, in real-world manufacturing scenarios, the manufacturing ability of resources cannot be accurately measured. This is because the manufacturing process is influenced by various factors, such as insufficient workforce and unreliable machines. To address this problem, these works [63, 89, 90] used fuzzy representation instead of precise QoS to offset the delay caused by uncertain disruptions. Table 3 summarizes the classification of fuzzy representation-based methods in solving DSPs, including intuitionistic fuzzy set, triangular fuzzy number, type-2 fuzzy set, left-right fuzzy set, and interval-based fuzzy set.

Intuitionistic Fuzzy Set. An **intuitionistic fuzzy set (IFS)** needs to define a membership function based on historical data, making it more capable of expressing QoS fluctuations when processing uncertain information [97, 98]. For example, Vishwakarma and Sharma [89] designed a vague lambda-tau method based on IFS theory to analyze the dynamics in the operational behavior of an industrial system. In this approach, they defined various reliability parameters to evaluate the system performance, such as system failure rates, mean time to repair, mean time between failures, expected number of failures, system reliability, and machine availability. To solve multi-objective optimization problems under the optimistic and pessimistic viewpoint, Rani et al. [90] presented a linear membership function and a non-linear membership function corresponding to optimistic and pessimistic optimization objectives, respectively. In [63], Zhang et al. used interval-valued intuitionistic fuzzy entropy weight to represent inaccurate QoS attribute values, which can obtain a better preference of QoS attributes and task priority for further calculation.

Table 3. Classification of Fuzzy Representation-based Methods in Solving DSPs

Fuzzy Representation	Problem	Approach	Solution
Intuitionistic fuzzy set	Manufacturing plant [89]	Vague lambda-tau method	A vague lambda-tau method based on IFS theory is used to evaluate the performance of a dynamic system.
	Manufacturing system [90]	Linear and non-linear membership functions	Linear and non-linear membership functions correspond to optimistic and pessimistic optimization objectives, respectively.
	Multi-task scheduling [63]	Interval-valued intuitionistic fuzzy entropy	Interval-valued intuitionistic fuzzy entropy weight is used to represent inaccurate QoS attribute values.
Triangular fuzzy number	FJSSP [72]	TFN-based ABC	TFN-based ABC represents the uncertainty of processing time as a TFN.
	FJSSP [71]	TFN-based two-stage ABC	TFN-based two-stage ABC triggers a rescheduling operator once a new task is inserted.
	Production planning and scheduling [91]	Fuzzy bi-stage decision	A fuzzy bi-stage decision model is used to handle the uncertain time scales of machine startup, task processing, and transformation.
	FJSSP [74]	TFN-based BBO	TFN-based BBO minimizes fuzzy completion time under fuzzy due date.
	FJSSP [68]	TFN-based HPSO	TFN-based HPSO assigns resources for each subtask while considering the uncertainty of processing time.
	JSSP [92]	Fuzzy framework	A fuzzy framework is defined for calculating non-processing energy.
Type-2 fuzzy set	Real-time embedded system [93]	Interval type-2 trapezoidal membership function	An interval type-2 trapezoidal membership function is proposed to increase the interpretability of dynamics.
Left-right fuzzy set	Open-station assembly line [94]	State-space model	A state-space model uses active time left-right (LR) fuzzy numbers to represent uncertain processing time.
Interval-based fuzzy set	SMSP [95]	Global dominance relation	A global dominance relation is developed for DSPs with unknown information.
	Make-to-order manufacturing system [96]	Fuzzy analytic network process	A fuzzy analytic network process method is used to identify top suppliers, which are considered as the last objective.

Triangular Fuzzy Number. Unlike IFS, which needs to define an accurate member function, the member function of a **triangular fuzzy number (TFN)** is reduced to a triangle determined by three values (i.e., upper bound, most probable value, and lower bound), which more intuitively represent the probability of each attribute value in the interval [99–102]. Therefore, TFN is suitable for scheduling problems that cannot obtain large amounts of historical data to define appropriate member functions. For example, Gao et al. [72] proposed an improved ABC algorithm to solve a realistic FJSSP in a remanufacturing system with the goal of minimizing fuzzy completion time and fuzzy machine workload, where the uncertainty of processing time is represented by a TFN. To further address real-world FJSSPs with new task insertions, Gao et al. [71] presented a two-stage ABC algorithm with THN. In this approach, the first stage is to optimize fuzzy processing time and obtain a high-quality solution, while the second stage triggers a rescheduling operator once a new task is inserted. In [91], Han et al. constructed a fuzzy bi-stage decision model to deal with a DSP, where TFN represents the uncertain time scales of machine startup, task processing, and transformation. To solve this decision model, they proposed a PSO-based method together with TFN to update solutions and finally obtained an optimal solution, which achieves better near-optimal solutions and improves computational efficiency.

Table 4. Performance Metrics and Measurements for Dynamic Scheduling

Metric	Method	Measurement
Robustness	Scenario-based stochastic programming [19, 70, 79, 107–109]	Robustness measurement evaluates the overall performance deviation (e.g., makespan, tardiness) under different scenarios.
	Robust counterpart optimization [80, 106, 110, 111]	Robustness measurement is minimizing the expected optimization objectives (e.g., total completion time, total flow time) under the worst-case scenario.
Stability	Partial performance deviation [23, 103, 112]	Stability metric is used to obtain a solution with a small deviation in partial performance of each subtask (e.g., completion time and sequence).

Others. Besides IFS and TFN, other fuzzy representations also apply to dynamic scheduling, such as left-right fuzzy sets and interval-based fuzzy sets. For example, to address ambiguous characteristics of tasks caused by the prediction of timing constraints in the initial design stage, Shurla et al. [93] presented an interval type-2 trapezoidal membership function (i.e., a lower membership function and an upper membership function) to increase the interpretability of real-time embedded systems, which involves two functions. Compared to models with crisp timing parameters and fuzzy type-1, it is found that the model with fuzzy type-2 consumes the least system energy by ensuring task completion with maximum earliness. To address manual mixed-model assembly lines, Ruppert et al. [94] proposed a state-space model based on a **left-right (LR)** fuzzy set to represent the active time of each module in modular products. Similar to IFS, the LR fuzzy set requires both the efficient integration of data-driven information and expert knowledge of process engineers into the model. For more complex manufacturing environments, where known information has only the lower and upper bounds for processing times of each task, Aydilek et al. [95] developed a global dominance relation, which describes processing times as the weighted average of an interval. Computational analysis reveals that the overall average error of this method is only 1.34% worse than the optimal solution.

5.1.2 Performance Measurement. Although makespan can effectively reflect the time performance of a solution, it cannot be used to measure the ability of a solution to cope with uncertain disruptions in a real-world manufacturing environment [103–105]. According to Section 3.1.2, uncertain disruptions in the manufacturing field can be mainly classified as service-related disruptions and task-related disruptions. Once a disruption occurs, a solution will suffer from the issue of being inapplicable to real-time manufacturing scenarios, resulting in the delay of tasks. To enable the on-time completion of manufacturing tasks, real-time adjustment is necessary to be applied to a solution to accommodate manufacturing scenarios with different uncertain disruptions. Since the size adjustment of a solution may easily result in the notorious "schedule nervousness" problem, performance measurements are used to judge whether the solution is resilient to uncertain disruptions. To better evaluate the ability of a solution to deal with uncertain disruptions, Table 4 introduces two typical metrics of DSPs, i.e., robustness and stability, which are detailed as follows.

Robustness. The metric robustness is used to measure solutions with a small deviation in overall performance. Based on the number of uncertain scenarios, Table 4 presents two classes of robustness methods, i.e., **scenario-based stochastic programming (SSP)** and **robust counterpart optimization (RCO)** [106]. Specifically, SSP tries to obtain an optimal solution that is as close as possible to all the given scenarios, while RCO is derived from the corresponding deterministic

model by considering the worst-case values of uncertain parameters. Due to the comprehensive consideration of different scenarios, a robust solution obtained based on an SSP measurement can better deal with different uncertain disruptions. For example, in the manufacturing field, robustness measurement is used to evaluate the deviation in overall performance (e.g., makespan, tardiness) between proactive solutions and realized solutions [70, 79, 107–109], where a proactive solution refers to a solution before an uncertain disruption occurs, and a realized solution is a newly optimized solution once triggering rescheduling. To further accurately evaluate the robustness of machines, Yang et al. [19] developed a measurement method called RMc for FJSSP, where the effect of float time on solutions is determined by considering both machine failure probability and fluctuation range.

Compared to SSP, RCO is easier to estimate in practice since it only requires clear upper and lower bounds of uncertain parameters. For example, to handle DSPs with uncertain processing time, Ying [106], Wang et al. [80], Lu et al. [111], and Chang et al. [110] defined robustness as the expected optimization objectives (e.g., total completion time, total flow time) under various worst-case scenarios. Specifically, Ying [106] formulated a two-machine FSSP with uncertain processing times as an RCO model, which introduces an asymmetric and bounded uncertain dataset to find a robust solution. In [110] Chang et al. presented a min-max distributional robust model that is universal for complex DSPs, since it does not require exact probability distributions. To address SMSPs with interval processing times, Lu et al. [111] proposed a mixed integer linear program together with an iterative improvement heuristic to obtain a robust solution with minimum worst-case total flow time. Moreover, Wang et al. [80] developed a united-scenario neighborhood structure based on a bad scenario set, where a threshold of performance is given as a standard to evaluate system performance rather than the optimal performance.

Stability. Unlike robustness, stability measurement considers partial performance deviation, where a stable solution refers to a solution that has a very small deviation in performance of each subtask (e.g., completion time and executing sequence) between a proactive solution and a realized solution. For example, to address SMSPs with machine breakdown, Liu et al. [112] calculated the sum of the absolute deviations of task completion time between a realized solution and a proactive solution to evaluate the stability of the proactive solution. In [103], Shen and Yao presented a multi-objective dynamic scheduling optimization model for FJSSP, where each rescheduling point evaluates three measurements related to shop stability: (1) deviation from early execution of subtasks, (2) deviation from the delay of subtasks, and (3) extension of the completion time of subtasks. To handle this model, they designed a multi-objective evolutionary algorithm based on a proactive-reactive method, which can obtain an optimal solution according to the interrupted information (e.g., interrupted time and duration) of machines and the QoS values of arrived tasks. If a random event occurs, the algorithm coupled with a proactive-reactive strategy will trigger rescheduling operators at each rescheduling point to re-optimize the original solution. To further improve the stability of solutions for FJSSPs, Ahmadi et al. [23] proposed two evolutionary algorithms to optimize makespan and stability simultaneously, which can effectively prevent a tremendous change caused by machine breakdown.

5.2 Heuristic-enabled Solver

By combining rescheduling strategies, heuristic-enabled solvers can adjust solutions in time to handle uncertain disruptions during manufacturing. Since frequent rescheduling operations will increase the chance of "schedule nervousness," how often a rescheduling should be triggered is becoming a critical issue in real-time manufacturing systems. To address this problem, many rescheduling policies have been developed, including periodic rescheduling, batch rescheduling, decomposition-based rescheduling, and aperiodic rescheduling [113], as introduced below.

5.2.1 Periodic Rescheduling. By periodically triggering rescheduling operations, periodic rescheduling-based methods can effectively address most uncertain disruptions that do not require immediate resolution. For example, to handle the problem of continuous arrival of tasks in a hybrid flow shop, Chen et al. [38] presented a periodic scheduling policy by decomposing a DSP into successive static sub-problems to reduce the computational difficulty, where each static sub-problem is regarded as a decision interval. Within each decision interval, they designed a modified GA to minimize the **mean longest waiting duration (MLWD)** of all the tasks. By investigating the sensitivity of decision intervals, they concluded that as the decision interval increases, the decreasing trend in MLWD becomes more significant. In [114], Angel-Bello et al. proposed a hybrid heuristic method based on a periodic rescheduling approach to minimize makespan in a dynamic SMSP with sequence-dependent setup times. Specifically, the production horizon is equally divided into time intervals. To obtain an optimal solution that is more suitable for the current scheduling environment, they presented two rescheduling strategies, where the first rescheduling strategy is to schedule all the available subtasks at the beginning of each interval to minimize makespan, and the second strategy focuses on scheduling future tasks that can be completed within the current interval. Experimental results show that scheduling future tasks in advance can improve resource utilization by reducing the idle time of machines at the current time interval.

Unlike the studies in [38, 114] assuming that machines are fully automatic, Delgoshaei et al. [115] took the flexibility of workers into account to achieve a human-centric manufacturing mode. Since human resource proficiency can be promoted by work experience and training, they took workers with different skill levels into account [116]. Note that the skill rates can be measured by conducting studies of workers before and after promotion. To achieve short-term period scheduling of dynamic cellular manufacturing systems in a dual resource-constrained environment, Delgoshaei et al. [115] developed a multi-period scheduling algorithm to find a scheduling strategy for in-house manufacturing using worker assignment (both skilled and temporary workers) and outsourcing. At the beginning of each scheduling period, periodic training programs are used to improve the skill level of workers. The algorithm allocates manufacturing tasks to in-house machines or outsourced machines according to their capacities. To address the state space explosion issue due to the increasing scheduling scale, Delgoshaei and Ali [117] proposed a hybrid TS and ACO algorithm. In this approach, a short-term memory structure and long-term memory structure enable the TS algorithm to obtain more chances to find potential solutions, where a short-term memory is used to prohibit the TS algorithm from revisiting those previous rejected solutions, and a long-term memory stores those solutions that are permanently banned. Moreover, a problem-specific strategy is used to avoid the proposed hybrid algorithm falling into local optima. Although the aforementioned periodic rescheduling-based methods can handle uncertain disruptions by regular rescheduling, they cannot adjust solutions in time once an urgent uncertain disruptions occurs.

5.2.2 Batch Rescheduling. Batch rescheduling refers to the regular arrangement of tasks belonging to the same batch. Similar to periodic rescheduling, batch rescheduling-based methods trigger rescheduling operations until all the tasks belonging to the same batch arrive at manufacturing systems. For the case of new task arrivals, Chen et al. [118] and Zhou et al. [119] selected tasks into batches for processing. In [118], Chen et al. proposed two algorithms (i.e., GA and ACO), which combine the earliest ready time and longest processing time heuristic to solve parallel batch processing machine scheduling problems. To alleviate the problem of electric supply-demand mismatch, Zhou et al. [119] designed an energy-efficient single-batch scheduling model with non-identical release time and task sizes. Furthermore, two heuristics are presented to address the model, where the first heuristic effectively groups tasks into batches, and the second heuristic is used to evaluate solutions and obtain a Pareto front. Finally, they proved that the proposed heuristics can effectively solve the single-batch scheduling model to balance sustainability and

productivity, which is practical for the actual production environment. Moreover, Cao et al. [120] presented a surrogate-assisted symbolic organized search algorithm to deal with parallel batch processor scheduling problems with dynamic task arrivals. First, they built a two-stage algorithmic framework to optimize batch sequencing to guide the batch allocation. Second, to estimate the fitness of batch sequencing, they designed a surrogate model, which can speed up the search by avoiding moving in the wrong search directions. Finally, SARSA is used to build a self-adaptive parameter control system to balance the global and local searches of the algorithm.

Taking both service-related disruptions and task-related disruptions, Pei et al. [121] investigated a series-batching scheduling problem with machine breakdown and dynamic task arrivals, where processing time and task size vary by types, assuming that each subtask with a different type requires setup time before processing. Based on these definitions, they formulated the series-batching scheduling problem as a mixed inter-programming model with the goal of minimizing the makespan. To solve this model, they modified a gravitational search algorithm with a batching mechanism. For this batching mechanism, once all tasks are assigned to a manufacturing system, they are split into batches and then scheduled on all services to optimize all objectives. To further handle a dynamic serial batch scheduling problem, Pei et al. [122] divide the problem into two stages: (1) the first stage is composed of two factories producing tasks submitted by customers, where the uncertain disruptions caused by machine breakdown and dynamic tasks' arrival are considered simultaneously, and (2) in the second stage, vehicles carry the completed tasks from factories to customers. For each stage, they presented two structural properties, where the first property is that a solution remains unchanged when any two tasks in a batch are swapped, and for the second one, there exists an optimal solution that all tasks in each batch are processed in the non-decreasing order of their arrival time to machines. Based on these two properties, they proposed a heuristic algorithm to analyze the performance of worst-case scenarios, which yields near-optimal solutions to large-scale problems in a shorter time.

5.2.3 Decomposition-based Rescheduling. Decomposition-based rescheduling focuses on decomposing a complex problem into multiple sub-problems to reduce solving complexity. For example, Gao et al. [123] decomposed an FJSSP with new task arrivals into two stages, i.e., initialization and rescheduling. During the initialization stage, Cao et al. regarded the problem as a static FJSSP with the same available time of machines, where the available time of machines in the rescheduling stage relies on the completion time of subtasks. To simplify flexible FSSPs with machine breakdown, Wang and Choi [124] developed a neighboring K-means clustering algorithm, which can group machines into several clusters. Note that traditional K-means clustering algorithms must solve the following two issues to decompose the flexible FSSPs: (1) how to group the machines into machine clusters and (2) how to choose a suitable machine cluster number. To address the first issue, Wang and Choi proposed a machine allocation algorithm, which allocates machines to the nearest neighboring machine cluster centers by calculating the distances of machines' stochastic vectors. Aiming to choose an appropriate machine cluster number, they proposed a clustering algorithm based on weighted cluster validity indices, which measure the intra-cluster distances and the inter-cluster distances between machine clusters. Specifically, the intra-cluster distance measures the distances of objects within a cluster to represent its compactness, while the inter-cluster distance computes the distance between two different clusters. Based on the decomposition, this algorithm can quickly generate an optimal solution for each machine cluster.

Unlike the previous works that ignore earliness and tardiness penalties for the completion time of each tasks, Wang and Li [125] designed a JIT JSSP, where subtasks that are completed earlier/later than their due dates will incur a penalty. To cope with this problem, they proposed a **variable neighborhood search (VNS)** procedure, which can explore the huge feasible solution space efficiently by alternating the swap and insertion neighborhood structures. To further ad-

dress the JIT JSSP, Ahmadian et al. [126] decomposed the problem into two smaller problems, i.e., assignment and sequencing. Pinedo and Hadavi pointed out that optimal scheduling based on a given sequence can be obtained in polynomial time if a feasible sequence can be provided [127]. Inspired by this observation, Ahmadian et al. designed a VNS algorithm with novel relation neighborhoods to obtain near-optimal subtask sequences for those smaller problems. Here, the relaxation neighborhoods partially destruct subtask sequences and then re-construct certain subtasks. Experimental results demonstrate that the proposed VNS algorithm obtains new best solutions for 57% of benchmark instances.

5.2.4 Aperiodic Rescheduling. In real-time systems, time-triggered mechanisms may cause significant losses due to the over-provisioning of manufacturing resources, which seriously affects the efficiency of rescheduling [128]. To address this issue, aperiodic rescheduling updates the solution once an uncertain disruption occurs. For example, aiming to solve an FJSSP with random task arrivals, Zhang et al. [129] designed an efficient memetic algorithm that minimizes both makespan and mean tardiness at each rescheduling point (i.e., once a new task arrives) to optimize solutions. Moreover, Syed et al. [130] proposed an extended task-shifting algorithm for scheduling non-preemptive tasks, which can utilize both online and offline strategies to ensure the successful execution of aperiodic tasks. To manage idle resources, they developed an offline strategy, which can make efficient use of processors in partition-free scenarios. Experimental results demonstrate that the extended task-shifting algorithm can achieve higher bandwidth utilization with less scheduling overhead. To further solve DSPs with aperiodic task arrivals, Zhou et al. [131] proposed an **event-triggered dynamic task scheduling (EDS)** method, which triggers scheduling operations on two types of subtasks: (1) the first subtask of a new task that has just arrived and (2) adjacent intermediate subtasks whose predecessor subtasks have completed, where intermediate subtasks refer to subtasks except for the first and last subtasks of tasks. Once receiving an aperiodic subtask to be triggered, the DES method will select one of the subtasks to be triggered according to the QoS values of all the candidate services. By prioritizing the earliest executable subtasks, they proposed a subtask-oriented scheduling strategy that can not only effectively reduce the idle time of services but also avoid the problem of prolonging the execution time caused by service preemption.

5.3 Discussions

Affected by dynamics in real-world manufacturing environments, the uncertain disruptions of DSPs cannot be accurately modeled. To address this problem, heuristic-enabled schedulers use fuzzy representations instead of precise QoS that are difficult to obtain, making it possible to offset the prolonged processing time caused by uncertain disruptions. Moreover, robustness and stability are two important performance metrics to evaluate whether a solution can deal with uncertain disruptions effectively. Besides robustness and stability, other performance metrics related to uncertain disruptions (e.g., reliability and risk) have raised great concerns in various fields, such as product distribution scheduling [132], logistics configuration [133], financial risk control [134], and delayed delivery [135]. Since the effectiveness of the above two methods depends on the accuracy of predictions of uncertain disruptions, heuristic-enabled schedulers need to reschedule solutions to achieve dynamic adaption, where the time points of rescheduling are important to determine. Typically, heuristic-enabled schedulers trigger rescheduling once an uncertain disruption occurs, which is called aperiodic rescheduling. Obviously, aperiodic rescheduling is easily prone to "schedule nervousness" due to the frequent adjustment of solutions. Unlike aperiodic rescheduling, heuristic-enabled schedulers make regular adjustments to solutions by dividing time horizons or grouping tasks, which is applicable to DSPs that do not have emergency situations. Therefore, to better deal with complex DSPs, heuristic-enabled schedulers need to adjust the rescheduling frequency in time to improve scheduling efficiency.

6 LEARNING-ENABLED SCHEDULER

Unlike heuristic-enabled schedulers, which have to combine rescheduling strategies to deal with most of the uncertain disruptions, learning-enabled schedulers arrange each subtask based on its current state step by step rather than making all decisions at once. From the perspectives of modeling and solving DSPs, this section reviews the research efforts of learning-enabled schedulers on scheduling control. Below, this section first introduces the learning-enabled modelers in Section 6.1, including model-free reinforcement learning (see Section 6.1.1) and dispatching rule learning (see Section 6.1.2). Then, based on different multi-agent RL architectures, Section 6.2 introduces three learning-enabled solvers, i.e., centralized training centralized execution (see Section 6.2.1), centralized training decentralized execution (see Section 6.2.2), and decentralized training decentralized execution (see Section 6.2.3). Finally, we summarize learning-enabled schedulers in Section 6.3.

6.1 Learning-enabled Modeler

6.1.1 Model-free Reinforcement Learning. Depending on whether the environment is modeled or not, the RL approach can be classified into two categories, i.e., model-based RL and model-free RL. Then model-based RL approaches rely on the information of an entire scheduling environment to build its scheduling model, which requires multiple model reconstructions to adapt to a dynamic scheduling environment. However, since the information of dynamic scheduling environments cannot be fully obtained, the accuracy of the scheduling model cannot be guaranteed. Unlike model-based RL approaches, model-free RL approaches can figure out an optimal policy without modeling the scheduling environment. Based on the different value outputs of actions, Table 5 classifies model-free RL approaches for dynamic scheduling into two categories, i.e., policy-based and value-based RL, which are detailed as follows.

Value-based RL. Value-based RL designs scheduling policies by modeling and estimating value functions, whose representative algorithms are mainly Q-learning and DQN. For each decision of Q-learning, an agent needs to retrieve a Q-table to obtain its Q-value for a given state and action. As the state space increases, the search speed of an agent will be greatly reduced due to the ever-increasing Q-table. To solve this issue, DQN uses neural networks to approximate value functions, which improves training efficiency.

To understand how the value-based RL solves DSPs, Table 5 introduces different scheduling components of value-based RL. As presented in Table 5, value-based RL mainly includes Q-learning and DQN. For example, to handle a dynamic JSSP with new task insertions and uncertain processing time, Xanthopoulos et al. [136] designed an intelligent controller to dynamically select dispatching rules with limited human expert interference. Specifically, by taking uncertain disruptions into account, this controller uses fuzzy schedulers based on both fuzzy logic and multi-objective evolutionary optimization to store its knowledge in a set of linguistic rules, where the knowledge reflects how the controlled system operates under various circumstances. Note that different fuzzy schedulers here correspond to different tradeoffs between minimizing mean earliness and mean tardiness. Furthermore, they proposed an RL-based approach named RLS, which applies a simple transformation routine to map the current system condition to a discrete state. At each decision point, the controller selects a dispatching rule from a set of rules and receives a reward. Based on experiments, the intelligent controller in all simulation cases can significantly improve both mean earliness and mean tardiness. Similar to this work [136], Shiue et al. [137] formulated the shop floor scheduling as MDPs with the reward of maximum throughput. To improve the adaptation of **multiple dispatching rules (MDRs)** responding to changes, Shiue et al. [137] established a **real-time scheduling (RTS)**-based knowledge base, which can quickly yield acceptable solutions to enable the system to make decisions in real time.

Table 5. Classification of Model-free RL Approaches for Dynamic Scheduling

Value-based Reinforcement Learning		
Approach	**Problem**	**Definition of Key RL Component**
Q-learning	SMSP [136]	**State**: system information; **Action**: selecting a dispatching rule; **Reward**: mean earliness and mean tardiness
	Shop floor control [137]	**State**: system information; **Action**: choosing a dispatching rule; **Reward**: minimum throughput
	Stochastic manufacturing system [138]	**State**: both machine and task information; **Action**: deciding the next stage of a system; **Reward**: the expected profits of the overall first-class products
	Make-to-order system [139]	**State**: task information and production capacity; **Action**: accepting or rejecting arrival tasks; **Reward**: averaged profit
	Hybrid FSSP [140]	**State**: workpiece information; **Action**: selecting a machine; **Reward**: the machining time of all workpieces
DQN	Smart manufacturing scheduling [141]	**State**: both task and system features; **Action**: selecting a dispatching rule for a machine; **Reward**: the utilization of machines
	Smart manufacturing scheduling [142]	**State**: information of candidate services; **Action**: selecting a dispatching rule; **Reward**: the maximum completion time of all tasks
	Shoemaking production [143]	**State**: production information; **Action**: the joint angle of manipulator coordinates; **Reward**: total discount
Policy-based Reinforcement Learning		
Approach	**Problem**	**Definition of Key RL Component**
PG	Workflow management [144]	**State**: both machine and task features; **Action**: selecting an execution site; **Reward**: total workflow workload
	MPSP [39]	**State**: environment information; **Action**: selecting a machine; **Reward**: total makespan and logistical distance
	Resource scheduling problem [145]	**State**: the status of both assigned and waiting tasks in a factory; **Action**: assigning factories to each task; **Reward**: averaged completion time, averaged manufacturing cost, and averaged resource usage
	Production control system [146]	**State**: information of production system; **Action**: selecting a dispatching rule; **Reward**: weighted sum of both dense and sparse rewards
AC	Resource scheduling problem [87]	**State**: resource state; **Action**: choosing an operation; **Reward**: total cost for finishing the part processing

Although Q-learning is promising in solving DSPs, it suffers from the problem of state space explosion. To address this problem, Lin et al. [141] developed a DQN-based edge computing framework to reduce the response time of making production decisions. In this work, the framework is deployed in a smart semiconductor manufacturing factory, consisting of numerous edge devices and a cloud center, where each edge device monitors and controls a machine to observe both machine and job information in real time. Since the cloud center is connected to all the edge devices, it can determine proper dispatching rules for each machine based on the monitored machine and job information. Unlike traditional DQN-based methods that can only select one dispatching rule at each decision point, Lin et al. proposed a multi-class DQN with multiple output neurons, which can determine multiple dispatching rules simultaneously. Experimental results show that the model trained by the multi-class DQN performed better than the traditional DQN in dynamic scheduling environments.

Policy-based RL. Policy-based RL directly models and estimates policies by maximizing feedback. Since policy-based RL does not require modeling based on state-value functions, it outputs action probabilities instead of specific values, avoiding the state space explosion due to the increasing size of scheduling problems. Furthermore, policy-based RL can easily converge because each gradient descent is a direct improvement of the strategy. For example, Kintsakis et al. [144] designed a machine learning model to schedule distributed resources in a workflow management system. To handle this model, they designed a PG algorithm based on a sequence-to-sequence neural structure, which is used for estimating the gradients of policy networks as function approximators. To maximize the total future expected rewards, PG trains the parameters of a policy network by simulating the probabilities of actions. Aiming to schedule massive interrelated tasks with uncertain arrivals of tasks in a timely and effective way, Chen et al. [39] established a dual-objective optimization model with the goal of minimizing both total makespan and logistical distances. Since too dense system information will affect training performance, they designed a dynamic state clustering algorithm, which ensures positions in the same state are closer to each other than those in other states. Moreover, they developed an **RL-based assigning policy (RLAP)** approach to obtain non-dominated solution sets. Compared with NSGA-II and Q-learning, the quality of obtained solutions by the RLAP approach can be improved up to 32.1% and 5.7%, respectively.

Aiming to further reduce response time by leveraging the reusability and extensibility of historical experience, Wu et al. [87] designed an AC algorithm for computer-aided process planning. By training two networks, i.e., policy network and evaluation network, the policy network can determine which combination to be chosen next, while the evaluation network can directly estimate the cost based on a resource state. To eliminate the influence of different input orders of operating states on network outputs, the policy network uses a one-dimensional convolution layer. Furthermore, based on a long short-term memory unit, the policy network calculates the probability distribution of actions and gets rid of long-distance dependence on encoding and decoding. Based on the aforementioned improvements, the AC algorithm can not only significantly reduce the response time but also cope with the dynamic changes in resource availability.

6.1.2 Dispatching Rule Learning. Since combined actions of DSPs greatly increase action space, traditional RL approaches suffer from the problem of poor convergence [147–149]. For example, unlike JSSPs that only sequence subtasks, distributed JSSPs need to assign tasks to factories before sequencing subtasks in a factory [150]. To accelerate convergence, designing dispatching rules to reduce the action space as a promising way has been gradually investigated in manufacturing systems [151–154]. According to different construction methods of dispatching rules, Table 6 classifies dispatching rules-based approaches into two categories, i.e., dispatching rules selection and dispatching rules extraction, described as follows.

Dispatching Rule Selection. Based on a set of dispatching rules, dispatching rule selection tries to replace combined actions as dispatching rules to avoid poor convergence problems caused by large combined action space. For example, to investigate a distributed JSSP considering technical precedence constraints, Xiong et al. [155] designed four dispatching rules, where each rule involves two sub-problems, i.e., assigning jobs to factories and determining the sequence of operations at each factory. To reduce the action space size, they designed four dispatching rules: (1) SOP rule, where an operation with the least slack should be given priority to load for processing; (2) MSOP rule, which prioritizes operations with lower tardiness; (3) RR+SOP rule, where RR shows dynamic and global characteristics; and (4) RR+MSOP rule. Since model parameters significantly affect the performance of dispatching rules, they analyzed the performance of four proposed dispatching rules. Experimental results illustrate that four dispatching rules perform well when the due dates of tasks are relatively loose. To further solve a dynamic JSSP, Aydin and Öztemel [157] constructed an autonomous dynamic scheduling system, where

Table 6. Classification of Dispatching Rules-based Approaches for Dynamic Scheduling

Dispatching Rule Selection		
Approach	**Problem**	**Description**
DQN	JSSP [155]	Four dispatching rules are designed and analyzed in different scenarios.
	FJSSP [40]	Double DQN and soft target weight are used to enhance the training efficiency of DQN.
Neural network approach	Shop floor control [156]	An intelligent multi-controller approach is designed based on self-organizing maps to achieve dispatching rules selection in real time.
	Semiconductor wafer fabrication system [153]	A multi-objective semiconductor fab scheduler is proposed to satisfy both customers and semiconductor companies.
	FSSP [152]	A real-time scheduling rule selection method based on a neural network is proposed.
Q-learning	JSSP [157]	An agent trained by Q-III makes decisions in real time to choose the most appropriate scheduling rule.
	SMSP [154]	A Q-learning algorithm takes three given dispatching rules as actions based on the current state to minimize the averaged delay.
	Production scheduling [148]	A single machine agent can learn common dispatch rules for different example scenarios.
GA+SVM	FSSP [149]	The task of data transformation is completed by a GA-based feature selection mechanism, and SVM is used to select the categories of dynamic dispatching rules.
Decision tree	Stochastic and dynamic job shop problem [158]	Based on the latent knowledge extracted from the data of the manufacturing environment, dispatching rules are selected according to the decision tree at each scheduling stage.
Dispatching Rule Extraction		
Approach	**Problem**	**Description**
GA	Industrial scheduling [159]	Dispatching rules are extracted by combining the ideas of push and pull.
Genetic programming	Production scheduling [160]	Genetic programming coupled with an evolutionary algorithm is proposed to update complex rules by analyzing system information.
GEP	FJSSP [161]	A GEP method designs terminal sets to extract dispatching rules with different lengths and functions.

scheduling strategies can be adjusted through self-learning based on a trial-and-error manner. In this approach, an improved RL algorithm called Q-III constantly updates the knowledge for future actions and gains experience in the process of updating knowledge, aiming to learn how to interpret current states and decide a dispatching rule.

To solve DSPs in real-time scheduling environments, Guh et al. [156] developed an appropriate dispatching rule selection strategy named the intelligent multi-controller, which includes three mechanisms, i.e., the training instance generation mechanism based on simulation, data pre-processing mechanism, and real-time MDRs selection mechanism based on self-organizing maps. Specifically, the data pre-processing mechanism involves a Las Vegas filter-based feature selection, which is carried out independently by learning algorithms before applying the classifier to a selected feature subset. Based on online simulation experiment verification, the intelligent multi-controller can minimize throughput, mean cycle time, and the number of tardy parts in solving real-time DSPs.

Dispatching Rule Extraction. Unlike dispatching rule selection, which learns how to select a suitable rule from a large number of candidate rules, dispatching rule extraction focuses on extracting more potentially efficient dispatching rules. This is because limited dispatching rules have difficulty covering various dynamic scheduling scenarios for real-world applications [160]. To solve this problem, Geiger et al. [159] presented a novel method to automatically discover efficient scheduling rules in the production environment. First, by designing a reasoning mechanism in the evaluation procedure and the numerical output of the performance evaluator, this method can effectively measure and modify the next-generation solutions. Second, since the performance measurements for all candidate rules are passed to the reasoning mechanism, a new set of rules can be constructed from the current performance rule using an evolutionary search operation. Third, once a set of rules is passed to the problem domain, the performance of the new rules is evaluated. This method repeats the above evaluations until an acceptable general performance level is reached. Experiments prove the superiority of the proposed method in various independent environments.

To solve dynamic FJSSP with limited buffers, Teymourifar et al. [161] proposed a useful approach for extracting an applicable and effective dispatching rule, taking both the irregular arrival of tasks and stochastic machine breakdown into account. In the proposed approach, a gene expression programming method is proposed to construct rules and undertake further improvement in a short time. Then, the generated rules are evaluated via a simulation model. The simulation case proves that the obtained rules are not only robust to different dynamic scenarios but also scalable to be applied to similar scheduling problems.

6.2 Learning-enabled Solver

Based on trajectories (including states, actions, and rewards), learning-enabled solvers train models through the continuous interactions between agents and their surrounding environments. According to multi-agent RL architectures, Table 7 introduces three classes of learning-enabled solvers for dynamic scheduling, i.e., **centralized training centralized execution (CTCE)**, **centralized training decentralized execution (CTDE)**, and **decentralized training decentralized execution (DTDE)**. For centralized approaches, the observed information is shared among agents and a central controller to facilitate efficient model training and execution. Inevitably, frequent communication between the agents and the central controller greatly prolongs both the training and execution time of models. The decentralized approach is the opposite of the centralized approach. Although a decentralized approach can effectively avoid the communication cost between the agents and the central controller by training models separately, the training performance of these models will deteriorate. More details of multi-agent RL architectures are presented as below.

6.2.1 Centralized Training Centralized Execution. Agents in a CTCE architecture send their partial observations to a central controller, which makes decisions for all the agents. In this way, agents can make better decisions, which are decided based on the fully observed environmental information. For example, to achieve the optimal scheduling of multi-stage processes in a manufacturing system, Qu et al. [42] formulated an MDP model for DSPs. To address this model, they proposed a centralized Q-learning approach to enable real-time cooperation of each processing machine inside the system. In this approach, a distributed weighted vector is designed to capture the cooperative pattern of massive action space. Aiming to minimize makespan, power, and resource cost simultaneously, this work used three agents for each objective to speed up the convergence of learning processes, respectively. To achieve an online dynamic multiple workflow scheduling, Asghari et al. [162] proposed a Q-learning approach to train the scheduling model in a centralized manner, which involves two phases: (1) each agent in the first phase explores the workflow to get

Table 7. Classification of Multi-agent Learning-enabled Solvers for Dynamic Scheduling

Architecture	Problem	Objective	RL	Characteristic
CTCE	Production scheduling [42]	Makespan	Q-learning	Distributed weighted vector
	FSSP [162]	Makespan, power, and resource cost	Q-learning	Two-phase training
	JSSP [163]	Mean tardiness of the finished tasks	Q-learning	Negotiation and cooperation
	Production scheduling problem [164]	Time-weighted function	Q-learning	Sliding window protocol
	JSSP [88]	Makespan	PPO	Stochastic gradient ascent
	JSSP [165]	Makespan	Actor-critic	Petri net representation
CTDE	Semiconductor manufacturing [166]	Makespan	DQN	Sharing parameters among agents
	Discrete manufacturing system [167]	Profit	MADDPG	Partially observable Markov model
	Demand-driven manufacturing [168]	Revenue and penalty	PG	Stochastic processing network
DTDE	JSSP [169]	Makespan	PG	Lightweight communication mechanism
	Parallel production system [170]	Cost	PPO	Opportunistic maintenance
	Distributed system [171]	Makespan	Q-learning	Machine learning box
	Routing and scheduling [172]	Cost	Q-learning	Variable neighborhood descent
	Tight-coupled assembly system [173]	Makespan	Q-learning	Decentralized dynamic interaction architecture
	Cloud resource management [174]	Makespan and cost	Q-learning	Cloud resource provisioning
	Workflow scheduling [175]	Makespan and cost	DQN	Multi-criteria interaction Markov game
	MPSP [176]	Makespan and logistical distance	RLAP	Dynamic state representing
	Smart manufacturing system [177]	Makespan	DQN	Negotiation among agents

the highest access cost of a critical path, and (2) the second phase is to allocate resources before task deadlines. Before the resource allocation process, they conducted a proper initial clustering of resources depending on the characteristics of each workflow, which speeds up the training process of allocating resources to tasks. Experimental results show that the trained scheduling model can not only reduce cost but also increase the utilization of resources.

Since Q-learning as a value-based RL algorithm suffers from the issue of state space explosion caused by the increasing problem scales, Wang et al. [88] developed a modified PPO algorithm to find an optimal scheduling policy, which involves two parts by integrating the advantages of value-based RL and policy-based RL, i.e., interaction part and training part. In the interaction part, agents interact with their environment to sample trajectories based on an old policy network $\pi(\theta')$. In the training part, the algorithm samples mini-batches from a buffer to update the policy network $\pi(\theta)$, where parameter θ is updated according to a stochastic gradient ascent method. Experimental results illustrate that the proposed method can effectively realize adaptive real-time scheduling compared with traditional scheduling methods. Note that although the central controller can make better decisions due to knowing all the observations, its training and execution speed will be greatly affected.

6.2.2 Centralized Training Decentralized Execution. During the training process using a CTDE architecture, a central controller can get the observations of all the agents, which ensures the performance of models. Unlike agents in the CTCE architecture, which execute models in a centralized manner, each agent in the CTDE architecture only outputs actions according to its own local observations to reduce execution time. For example, to solve sequence-dependent semiconductor production scheduling problems, Park et al. [166] presented a multi-agent model. In this model, each production stage corresponds to an agent, which allocates tasks to machines based on observations. Furthermore, a set of transitions that consist of state, action, reward, and next state are stored in a replay buffer and extracted randomly. To improve the training performance of this model, they developed a DQN algorithm by sharing parameters among all the agents. To achieve energy management of discrete manufacturing systems, Lu et al. [167] designed a **multi-agent deep deterministic policy gradient (MADDPG)** algorithm, which learns a critic network for each agent based on local rewards and derives a decentralized actor network using the centralized critic network. By formulating a **demand response (DR)** problem as a partially observable Markov model, each agent maintains a separate policy network and takes an action based on its observation of corresponding machines' energy consumption and buffer storage. Once all the agents take actions, multi-agents cooperate to obtain shared rewards. For the complex industrial DR problem, the MADDPG algorithm can reduce the total electricity cost by 9.8% in a lithium-ion battery assembly process.

6.2.3 Decentralized Training Decentralized Execution. In a DTDE architecture, each agent makes decisions independently during the training and execution process based on their own observed state, which will deteriorate the training performance of model. To improve training performance, Gabel and Riedmiller [169] proposed a distributed approach by integrating multiple learning agents, where independent agents try to improve their local policies with respect to a common goal. To further enhance the ability of agents, they developed a lightweight communication mechanism to partially obtain the information of future arrival tasks in advance, rather than just reactive task scheduling. Experiments prove that the proposed approach can effectively handle both static and dynamic JSSPs. Moreover, Orhean et al. [171] proposed an RL algorithm to address distributed scheduling problems. To support an agent in learning optimal scheduling policies under different machine setups, they proposed a **machine learning box (MBOX)** using a BURLAP library, which offers a simple and configurable interface for the implementation of various planning and learning algorithms. To respond to manufacturing requests, MBOX designs a task scheduling method that in a timely manner notifies the system to update the processing time of tasks. Based on comparison experiments, application models based on MBOX can be easily mapped onto parallel scheduling systems to improve their overall efficiency.

6.3 Discussions

The dynamics of scheduling problems make it difficult for learning-enabled schedulers to obtain the complete transition probabilities of complex manufacturing environments. To handle this problem, learning-enabled schedulers use model-free RL rather than model-based RL for DSPs, which can make full use of the state information explored for the environment [85]. Since DSPs including service assignment and task sequencing select proper combined actions rather than single actions at each decision point, the rapid increase in combined action space greatly affects the convergence of learning-enabled schedulers. To accelerate convergence, learning-enabled schedulers achieve effective scheduling control by designing multiple composite dispatching rules as actions. However, limited composite dispatching rules cannot comprehensively cover the relations between tasks and services, resulting in poor performance of scheduling policies.

To cope with this issue, effective decision-making should pay more attention to end-to-end policy network training to prevent excessive dependence on the quality of composite dispatching rules. Moreover, during the training processes of policy networks, information (e.g., network parameters, observed states) sharing between agents can effectively improve training efficiency, while their communication will inevitably increase training time. In the future, more effective learning-enabled schedulers are expected to be designed for complex DSPs, making it possible to train a well-performed model with less training time.

7 CONCLUSIONS AND FUTURE OPPORTUNITIES

Due to ubiquitous uncertain disruptions in the real-world manufacturing environment, it is difficult for traditional scheduling methods to effectively solve complex DSPs, which are classic NP-hard problems. Unlike traditional scheduling methods that spend a great deal of computation time in pursuit of exact optimal solutions, AI-based methods can quickly achieve near-optimal solutions. To understand the effectiveness of existing AI-based dynamic scheduling methods within the uncertain manufacturing environment, this article introduced both their pros and cons in an evolutionary manner, starting from directed heuristics to state-of-the-art autonomous learning methods. The heuristic-enabled schedulers iteratively update solutions in combination with dynamic scheduling strategies to achieve dynamic adaptation to various uncertain disruptions, while learning-enabled schedulers can perfectly adapt to dynamics based on their sequential decision-making. Although both of these AI-enabled schedulers can address most complex DSPs in practice, it is still difficult for them to reach safety-critical and environment-friendly massive manufacturing, which poses stringent requirements for accuracy, responsiveness, and energy efficiency.

Along with the fast development of emerging AI techniques, manufacturing industries continue to evolve rapidly, which in turn puts forward more unknown disruptions and corresponding requirements for dynamic scheduling in modern manufacturing. To accommodate to such a new manufacturing environment, the following topics are worth exploring further in the near future:

- **Data-driven Scheduling.** As a promising way to alleviate the "schedule nervousness" problem caused by notable scheduling fluctuations, data-driven predictive scheduling has been increasingly studied at each decision point to enable the accurate prediction of various uncertain information. Assuming that uncertainty information follows specific data distributions, traditional dynamic scheduling methods are too idealistic to achieve optimal scheduling solutions in practice. This is mainly because an inaccurate decision-making process will inevitably result in the degradation of scheduling efficacy. To address this issue, more and more emerging dynamic scheduling methods resort to AI-enabled techniques, which can naturally figure out accurate uncertainty information from some given historical manufacturing data, thus substantially improving the scheduling performance.
- **Human-centric Scheduling.** How to quickly and accurately adapt to various uncertain manufacturing dynamics (e.g., the changes in customer needs or manufacturing capabilities) in real time is becoming an important issue of human-centric scheduling. Since AI-controlled intelligent manufacturing systems cannot fully handle unrecognized disruptions, researchers should refocus on the interactions between humans and intelligent manufacturing systems during scheduling processes. By leveraging human intelligence to intervene in the identification of disruptions that cannot be recognized by machines, intelligent manufacturing systems will improve their ability to respond in a timely manner to various disturbances throughout the scheduling process.
- **On-demand Scheduling.** Due to the uncertain number of manufacturing tasks in different scheduling periods, resources in manufacturing systems usually retain a large redundancy

to cope with the sudden increase of manufacturing tasks, resulting in a waste of numerous idle manufacturing resources. To improve the utilization of such manufacturing resources and achieve the on-demand use of manufacturing resources, modern manufacturing systems should support dynamic assessment of both the number and urgency of current manufacturing tasks to achieve an optimal allocation of manufacturing resources.

- **Low-carbon Scheduling.** Due to the increasing concerns on "carbon neutrality" and "peak carbon dioxide emissions," how to achieve energy-efficient schedules to support low-carbon manufacturing is becoming an urgent issue in the field of manufacturing. Since carbon dioxide emissions are involved in different parts of manufacturing systems and different stages of scheduling processes, there is a strong demand for appropriate dynamic service allocation of manufacturing tasks to minimize the overall carbon dioxide emissions, while other optimization objectives should still be ensured.

Although the above topics (but not limited to) can substantially benefit the evolution of industry 5.0, so far there is no work that takes all of them into account simultaneously. We believe that dynamic schedulers will play a more important role in industry 5.0 to enable optimal decisions during the scheduling process in a low-carbon manufacturing environment, which will significantly contribute to the sustainability and resilience of our society's development.

REFERENCES

[1] X. Xu, Y. Lu, B. Vogel-Heuser, and L. Wang. 2021. Industry 4.0 and industry 5.0 - Inception, conception and perception.*Journal of Manufacturing Systems* 61 (2021), 530–535.

[2] M. D. Nardo and H. Yu. 2021. Industry 5.0: The prelude to the sixth industrial revolution.*Applied System Innovation* 4, 3 (2021), 45.

[3] M. Zarandi, A. Asl, S. Sotudian, and O. Castillo. 2020. A state of the art review of intelligent scheduling. *Artificial Intelligence Review* 53 (2020), 501–593.

[4] Y. Liu, L. Wang, X. Wang, X. Xu, and L. Zhang. 2019. Scheduling in cloud manufacturing: State-of-the art and research challenges.*International Journal of Production Research* 57, 15–16 (2019), 4854–4879.

[5] A. Giret, D. Trentesaux, and V. Prabhu. 2015. Sustainability in manufacturing operations scheduling: A state of the art review. *Journal of Manufacturing Systems* 37, 1 (2015), 126–140.

[6] D. Rossit, F. Tohmé, and M. Frutos. 2018. The non-permutation flow-shop scheduling problem: A literature review. *Omega* 77 (2018), 143–153.

[7] E. Edis, C. Oguz, and I. Ozkarahan. 2013. Parallel machine scheduling with additional resources: Notation, classification, models and solution methods. *European Journal of Operational Research* 230, 3 (2013), 449–463.

[8] R. Ojstersek and M. Brezocnik. 2020. Multi-objective optimization of production scheduling with evolutionary computation: A review. *International Journal of Industrial Engineering Computations* 11 (2020), 359–376.

[9] P. Robert, P. Nathalie, and B. Francois. 2020. A survey of hybrid metaheuristics for the resource-constrained project scheduling problem. *European Journal of Operation Research* 280, 2 (2020), 395–416.

[10] H. Singh, S. Tyagi, and P. Kumar. 2019. Scheduling in cloud computing environment using metaheuristic techniques: A survey. *Emerging Technology in Modelling and Graphics* 937 (2019), 753–763.

[11] P. Priore, A. Gómez, R. Pino, and R. Rosillo. 2014. Dynamic scheduling of manufacturing systems using machine learning: An updated review. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 28, 1 (2014), 83–97.

[12] Y. Liu, L. Zhang, L. Wang, Y. Xiao, X. Xu, and M. Wang. 2019. A framework for scheduling in cloud manufacturing with deep reinforcement learning. In *Proceedings of the 17th IEEE International Conference on Industrial Informatics (INDIN'19),* 1775–1780.

[13] E. TAillard. 1993. Benchmarks for basic scheduling problems. *European Journal of Operational Research* 64, 2 (1993), 278–285.

[14] E. Demirkol, S. Mehta, and R. Uzsoy. 1998. Benchmarks for shop scheduling problems. *European Journal of Operational Research* 109, 1 (1998), 137–141.

[15] I. Kacem, S. Hammadi, and P. Borne. 2002. Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems. *IEEE Transactions on Systems, Man and Cybernetics* 32, 1 (2002), 1–13.

[16] D. Lee and F. DiCesare. 1994. Scheduling flexible manufacturing systems using petri nets and heuristic search. *IEEE Transactions on Robotics and Automation* 10, 2 (1994), 123–132.

[17] P. Brandimarte. 1993. Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations Research* 41 (1993), 157–183.
[18] J. Ding, Y. Wang, S. Zhang, W. Zhang, and Z. Xiong. 2019. Robust and stable multi-task manufacturing scheduling with uncertainties using a two-stage extended genetic algorithm. *Enterprise Information Systems* 13, 10 (2019), 1442–1470.
[19] Y. Yang, M. Huang, Z. Wang, and Q. Zhu. 2020. Robust scheduling based on extreme learning machine for bi-objective flexible job-shop problems with machine breakdowns. *Expert Systems with Applications* 158, 15 (2020), 113545.
[20] D. Ouelhadj and S. Petrovic. 2009. A survey of dynamic scheduling in manufacturing systems.*Journal of Scheduling* 12 (2009), 417–431.
[21] Z. Wu, S. Sun, and S. Xiao. 2018. Risk measure of job shop scheduling with random machine breakdowns. *Computers & Operations Research* 99 (2018), 1–12.
[22] N. Al-Hinai and T. ElMekkawy. 2011. Robust and stable flexible job shop scheduling with random machine breakdowns using a hybrid genetic algorithm. *International Journal of Production Economics* 132, 2 (2011), 279–291.
[23] E. Ahmadi, M. Zandieh, M. Farrokh, and S. Emami. 2016. A multi objective optimization approach for flexible job shop scheduling problem under random machine breakdown by evolutionary algorithms. *Computers & Operations Research* 73 (2016), 56–66.
[24] K. Peng, Q. Pan, L. Gao, B. Zhang, and X. Pang. 2018. An improved artificial bee colony algorithm for real-world hybrid flowshop rescheduling in steelmaking-refining-continuous casting process. *Computers & Industrial Engineering* 122 (2018), 235–250.
[25] M. Haddadzade, M. Razfar, and M. Zarandi. 2014. Integration of process planning and job shop scheduling with stochastic processing time. *International Journal of Advanced Manufacturing Technology* 71 (2014), 241–252.
[26] V. Ghezavati and M. Saidi-Mehrabad. 2010. Designing integrated cellular manufacturing systems with scheduling considering stochastic processing time. *International Journal of Advanced Manufacturing Technology* 48 (2010), 701–717.
[27] J. Lin, C. Chiu, and Y. Chang. 2019. Simulation-based optimization approach for simultaneous scheduling of vehicles and machines with processing time uncertainty in FMS. *Flexible Services and Manufacturing Journal* 31 (2019), 104–141.
[28] C. Lu, S. Lin, and K. Ying. 2012. Robust scheduling on a single machine to minimize total flow time.*Computers & Operations Research* 39, 7 (2012), 1682–1691.
[29] R. Li, W. Gong, and C. Lu. 2022. A reinforcement learning based RMOEA/D for bi-objective fuzzy flexible job shop scheduling. *Expert Systems With Applications* 203 (2022), 117380.
[30] K. Wang and S. Choi. 2014. A holonic approach to flexible flow shop scheduling under stochastic processing times. *Computers & Operations Research* 43 (2014), 157–168.
[31] L. Nie, L. Gao, P. Li, and X. Shao. 2013. Reactive scheduling in a job shop where jobs arrive over time. *Computers & Industrial Engineering* 66, 2 (2013), 389–405.
[32] D. Johnson, G. Chen, and Y. Lu. 2022. Multi-agent reinforcement learning for real-time dynamic production scheduling in a robot assembly cell. *IEEE Robotics and Automation Letters* 7, 2 (2022), 7684–7691.
[33] B. Luo, S. Wang, B. Yang, and L. Yi. 2021. An improved deep reinforcement learning approach for the dynamic job shop scheduling problem with random job arrivals. *Journal of Physics: Conference Series* 1848, 1 (2021), 012029.
[34] D. Rahmani and R. Ramezanian. 2016. A stable reactive approach in dynamic flexible flow shop scheduling with unexpected disruptions: A case study. *Computers & Industrial Engineering* 98 (2016), 360–372.
[35] B. Han and J. Yang. 2020. Research on adaptive job shop scheduling problems based on dueling double DQN. *IEEE Access* 8 (2020), 186474–186495.
[36] J. Shahrabi, M. Adibi, and M. Mahootchi. 2017. A reinforcement learning approach to parameter estimation in dynamic job shop scheduling. *IEEE Robotics and Automation Letters* 110 (2017), 75–82.
[37] D. Wang, F. J. Wang, and Y. Wang. 2016. Integrated rescheduling and preventive maintenance for arrival of new jobs through evolutionary multi-objective optimization. *Soft Computing* 20 (2016), 1635–1652.
[38] J. Chen, M. Wang, X. Kong, G. Huang, Q. Dai, and G. Shi. 2019. Manufacturing synchronization in a hybrid flowshop with dynamic order arrivals. *Journal of Intelligent Manufacturing* 30 (2019), 2659–2668.
[39] S. Chen, S. Fang, and R. Tang. 2018. A reinforcement learning based approach for multi-projects scheduling in cloud manufacturing. *International Journal of Production Research* 57, 10 (2018), 3080–3098.
[40] S. Luo. 2020. Dynamic scheduling for flexible job shop with job insertions by deep reinforcement learning. *Applied Soft Computing* 91 (2020), 106208.
[41] L. Hu, Z. Liu, W. Hu, Y. Wang, J. Tan, and F. Wu. 2020. Petri-net-based dynamic scheduling of flexible manufacturing system via deep reinforcement learning with graph convolutional network. *Journal of Manufacturing Systems* 55 (2020), 1–14.

[42] S. Qu, T. Chu, J. Wang, J. Leckie, and W. Jian. 2015. A centralized reinforcement learning approach for proactive scheduling in manufacturing. In *Proceedings of the 20th IEEE Conference on Emerging Technologies & Factory Automation (ETFA'15)*, (1–8).
[43] H. Rahman, R. Sarker, and D. Essam. 2018. Multiple-order permutation flow shop scheduling under process interruptions. *International Journal of Advanced Manufacturing Technology* 97 (2018), 2781–2808.
[44] L. Zhang, L. Gao, and X. Li. 2013. A hybrid intelligent algorithm and rescheduling technique for job shop scheduling problems with disruptions. *International Journal of Advanced Manufacturing Technology* 65 (2013), 1141–1156.
[45] Y. Li, Y. He, Y. Wang, F. Tao, and J. Sutherland. 2020. An optimization method for energy-conscious production in flexible machining job shops with dynamic job arrivals and machine breakdowns. *Journal of Cleaner Production* 254, 1 (2020), 120009.
[46] F. Liu, S. Wang, Y. Hong, and X. Yue. 2017. On the robust and stable flowshop scheduling under stochastic and dynamic disruptions. *IEEE Transactions on Engineering Management* 64, 4 (2017), 539–553.
[47] C. Pang and C. Le. 2014. Optimization of total energy consumption in flexible manufacturing systems using weighted p-timed petri nets and dynamic programming. *IEEE Transactions on Automation Science and Engineering* 11, 1 (2014), 1083–1096.
[48] S. Shim and Y. Kim. 2008. A branch and bound algorithm for an identical parallel machine scheduling problem with a job splitting property. *Computers & Operations Research* 35, 3 (2008), 863–875.
[49] J. Pereira and M. Vilà. 2015. An exact algorithm for the mixed-model level scheduling problem. *International Journal of Production Research* 53, 19 (2015), 5809–5825.
[50] L. Zeballos. 2010. A constraint programming approach to tool allocation and production scheduling in flexible manufacturing systems. *Robotics and Computer-Integrated Manufacturing* 26, 6 (2010), 725–743.
[51] J. Cheng, H. Kise, and H. Matsumoto. 1997. A branch-and-bound algorithm with fuzzy inference for a permutation flowshop scheduling problem. *European Journal of Operational Research* 96, 3 (1997), 578–590.
[52] A. Yenipazarli, H. Benson, and S. Erenguc. 2016. A branch-and-bound algorithm for the concave cost supply problem.*International Journal of Production Research* 54, 13 (2016), 3943–3961.
[53] J. Liou. 2020. Dominance conditions determination based on machine idle times for the permutation flowshop scheduling problem. *Computers & Operations Research* 122 (2020), 104964.
[54] L. Perron and V. Furnon. 2019. Or-tools. https://developers.google.cn/optimization?hl=zh-cn.
[55] X. Li, K. Xing, M. Zhou, X. Wang, and Y. Wu. 2019. Modified dynamic programming algorithm for optimization of total energy consumption in flexible manufacturing systems. *IEEE Transactions on Automation Science and Engineering* 16, 2 (2019), 691–705.
[56] W. Lunardi, E. Birgin, D. Ronconi, and H. Voos. 2021. Metaheuristics for the online printing shop scheduling problem. *European Journal of Operational Research* 293, 2 (2021), 419–441.
[57] A. Telikani, A. Tahmassebi, W. Banzhaf, and A. H. Gandomi. 2021. Evolutionary machine learning: A survey. *Comput. Surveys* 54, 8 (2021), 1–35.
[58] P. Wang. 2019. On defining artificial intelligence. *Journal of Artificial General Intelligence* 10, 2 (2019), 1–37.
[59] J. Zhang, G. Ding, Y. Zou, S. Qin, and J. Fu. 2019. Review of job shop scheduling research and its new perspectives under Industry 4.0. *Journal of Intelligent Manufacturing* 30 (2019), 1809–1830.
[60] C. Zhang, X. Dong, X. Wang, X. Li, and Q. Liu. 2010. Improved NSGA-II for the multiobjective flexible job-shop scheduling problem.*Journal of Engineering Mechanics* 46, 11 (2010), 159–163.
[61] X. Jiang, Z. Tian, W. Liu, Y. Suo, K. Chen, X. Xu, and Z. Li. 2022. Energy-efficient scheduling of flexible job shops with complex processes: A case study for the aerospace industry complex components in China. *Journal of Industrial Information Integration* 27 (2022), 100293.
[62] D. Lei. 2010. Solving fuzzy job shop scheduling problems using random key genetic algorithm. *International Journal of Advanced Manufacturing Technology* 49, 1 (2010), 253–262.
[63] W. Zhang, J. Ding, Y. Wang, S. Zhang, and Z. Xiong. 2019. Multi-perspective collaborative scheduling using extended genetic algorithm with interval-valued intuitionistic fuzzy entropy weight method. *Journal of Manufacturing Systems* 53 (2019), 249–260.
[64] G. Rey, A. Bekrar, V. Prabhu, and D. Trentesaux. 2014. Coupling a genetic algorithm with the distributed arrival-time control for the JIT dynamic scheduling of flexible job-shops. *International Journal of Production Research* 52, 12 (2014), 3688–3709.
[65] J. Kennedy and R. Eberhart. 1995. Particle swarm optimization. *International Conference on Neural Networks* 4 (1995), 1942–1948.
[66] S. Saeedi, R. Khorsand, S. Bidgoli, and M. Ramezanpour. 2020. Improved many-objective particle swarm optimization algorithm for scientific workflow scheduling in cloud computing.*Computers & Industrial Engineering* 147 (2020), 106649.

[67] J. Jian, L. Wang, H. Chen, and X. Nie. 2020. Scheduling optimization of time-triggered cyber-physical systems based on fuzzy-controlled QPSO and SMT solver. *Energies* 13, 3 (2020), 668.
[68] T. Jamrus, C. Chien, M. Gen, and K. Sethanan. 2018. Hybrid particle swarm optimization combined with genetic operators for flexible job-shop scheduling under uncertain processing time for semiconductor manufacturing. *IEEE Transactions on Semiconductor Manufacturing* 31, 1 (2018), 32–41.
[69] Z. Wang, J. Zhang, and S. Yang. 2019. An improved particle swarm optimization algorithm for dynamic job shop scheduling problems with random job arrivals. *Swarm and Evolutionary Computation* 51 (2019), 100594.
[70] R. Zhang, S. Song, and C. Wu. 2020. Robust scheduling of hot rolling production by local search enhanced ant colony optimization algorithm. *IEEE Transactions on Industrial Informatics* 16, 4 (2020), 2809–2819.
[71] K. Gao, P. Suganthan, Q. Pan, M. Tasgetiren, and A. Sadollah. 2016. Artificial bee colony algorithm for scheduling and rescheduling fuzzy flexible job shop problem with new job insertion. *Knowledge-Based Systems* 109, 1 (2016), 1–16.
[72] K. Gao, P. Suganthan, Q. Pan, T. Chua, C. Chong, and T. Cai. 2016. An improved artificial bee colony algorithm for flexible job-shop scheduling problem with fuzzy processing time.*Expert Systems with Applications* 65, 15 (2016), 52–67.
[73] W. Xu, Z. Ji, and Y. Wang. 2018. A flower pollination algorithm for flexible job shop scheduling with fuzzy processing time. *Modern Physics Letters B* 32 (2018), 1840113.
[74] J. Lin. 2015. A hybrid biogeography-based optimization for the fuzzy flexible job-shop scheduling problem. *Knowledge-Based Systems* 78 (2015), 59–74.
[75] I. Strumberger, N. Bacanin, M. Tuba, and E. Tuba. 2019. Resource scheduling in cloud computing based on a hybridized whale optimization algorithm. *Applied Sciences* 9, 22 (2019), 4893.
[76] Z. Zhang, L. Zheng, and M. Weng. 2007. Dynamic parallel machine scheduling with mean weighted tardiness objective by Q-Learning.*International Journal of Advanced Manufacturing Technology* 34 (2007), 968–980.
[77] Y. Wang. 2018. Adaptive job shop scheduling strategy based on weighted Q-learning algorithm. *Journal of Intelligent Manufacturing* 31 (2018), 417–432.
[78] F. Zhao, L. Zhang, J. Cao, and J. Tang. 2021. A cooperative water wave optimization algorithm with reinforcement learning for the distributed assembly no-idle flowshop scheduling problem. *Computers & Industrial Engineering* 153 (2021), 107082.
[79] X. Shen, Y. Han, and J. Fu. 2017. Robustness measures and robust scheduling for multi-objective stochastic flexible job shop scheduling problems. *Soft Computing* 21 (2017), 6531–6554.
[80] B. Wang, X. Wang, and H. Xie. 2018. Bad-scenario-set robust scheduling for a job shop to hedge against processing time uncertainty. *International Journal of Production Research* 57, 10 (2018), 3168–3185.
[81] Z. Cao, C. Lin, and M. Zhou. 2015. A knowledge-based cuckoo search algorithm to schedule a flexible job shop with sequencing flexibility. *IEEE Transactions on Automation Science and Engineering* 53, 19 (2015), 5809–5825.
[82] J. Park, J. Chun, S. Kim, Y. Kim, and J. Park. 2021. Learning to schedule job-shop problems: Representation and policy learning using graph neural network and reinforcement learning.*International Journal of Production Research* 59, 11 (2021), 3360–3377.
[83] P. Zheng, P. Zhang, J. Wang, J. Zhang, C. Yang, and Y. Jin. 2020. A data-driven robust optimization method for the assembly job-shop scheduling problem under uncertainty. *International Journal of Computer Integrated Manufacturing* 35, 10–11 (2020), 1043–1058.
[84] K. Dooley and F. Mahmoodi. 1992. Identification of robust scheduling heuristics: Application of Taguchi methods in simulation studies. *Computers & Industrial Engineering* 22, 4 (1992), 359–368.
[85] C. Liu, C. Chang, and C. Tseng. 2020. Actor-critic deep reinforcement learning for solving job shop scheduling problems. *IEEE Access* 8 (2020), 71752–71762.
[86] G. Hinton and R. Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *Science* 313 (2006), 504–507.
[87] W. Wu, Z. Huang, J. Zeng, and K. Fan. 2021. A fast decision-making method for process planning with dynamic machining resources via deep reinforcement learning. *Journal of Manufacturing Systems* 58 (2021), 392–411.
[88] L. Wang, X. Hu, Y. Wang, S. Xu, S. Ma, K. Yang, Z. Liu, and W. Wang. 2021. Dynamic job-shop scheduling in smart manufacturing using deep reinforcement learning.*Computer Networks* 190, 8 (2021), 107969.
[89] Y. Vishwakarma and S. Sharma. 2016. Uncertainty analysis of an industrial system using intuitionistic fuzzy set theory. *International Journal of System Assurance Engineering and Management* 7, 1 (2016), 73–83.
[90] D. Rani, T. Gulati, and H. Garg. 2016. Multi-objective non-linear programming problem in intuitionistic fuzzy environment: Optimistic and pessimistic view point. *Expert Systems with Applications* 64 (2016), 228–238.
[91] J. Han, Y. Liu, L. Luo, and M. Mao. 2020. Integrated production planning and scheduling under uncertainty: A fuzzy bi-level decision-making approach. *Knowledge-Based Systems* 201–202, 9 (2020), 106056.

[92] I. González-Rodríguez, J. Puente, J. Palacios, and C. Vela. 2020. Multi-objective evolutionary algorithm for solving energy-aware fuzzy job shop problems. *Soft Computing* 24 (2020), 16291–16302.
[93] A. Shukla, R. Nath and P. Muhuri. 2015. Energy efficient task scheduling with Type-2 fuzzy uncertainty. *2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, Istanbul, 2015, 1–8. DOI: 10.1109/FUZZ-IEEE.2015.7338103
[94] T. Ruppert, G. Dorgo, and J. Abonyi. 2020. Fuzzy activity time-based model predictive control of open-station assembly lines.*Journal of Manufacturing Systems* 54 (2020), 12–23.
[95] A. Aydilek, H. Aydilek, and A. Allahverdi. 2017. Algorithms for minimizing the number of tardy jobs for reducing production cost with uncertain processing times.*Applied Mathematical Modelling* 45 (2017), 982–996.
[96] G. Bodaghi, F. Jolai, and M. Rabbani. 2018. An integrated weighted fuzzy multi-objective model for supplier selection and order scheduling in a supply chain. *International Journal of Production Research* 56, 10 (2018), 3590–3614.
[97] X. Zhang, Y. Deng, L. Chan, P. Xu, M. Sankaran, and Y. Hu. 2013. IFSJSP: A novel methodology for the job-shop scheduling problem based on intuitionistic fuzzy sets. *International Journal of Production Research* 51, 17 (2013), 5100–5119.
[98] S. Zhang and S. Wang. 2018. Flexible assembly job-shop scheduling with sequence-dependent setup times and part sharing in a dynamic environment: Constraint programming model, mixed-integer programming model, and dispatching rules. *IEEE Transactions on Engineering Management* 65, 3 (2018), 487–504.
[99] J. Zheng, L. Wang, and J. Wang. 2020. A cooperative coevolution algorithm for multi-objective fuzzy distributed hybrid flow shop. *Knowledge-Based Systems* 194, 22 (2020), 105536.
[100] J. Lin. 2020. Backtracking search based hyper-heuristic for the flexible job-shop scheduling problem with fuzzy processing time.*Engineering Applications of Artificial Intelligence* 77 (2020), 186–196.
[101] B. Liu, Y. Fan, and Y. Liu. 2015. A fast estization of distribution algorithm for dynamic fuzzy flexible job-shop scheduling problem. *Computers & Industrial Engineering* 87 (2015), 193–201.
[102] V. Sonmez, M. Testik, and O. Testik. 2018. Overall equipment effectiveness when production speeds and stoppage durations are uncertain. *International Journal of Advanced Manufacturing Technology* 95 (2018), 121–130.
[103] X. Shen and X. Yao. 2015. Mathematical modeling and multi-objective evolutionary algorithms applied to dynamic flexible job shop scheduling problems. *Information Sciences* 298, 20 (2015), 198–224.
[104] S. Goren, I. Sabuncuoglu, and U. Koc. 2012. Optimization of schedule stability and efficiency under processing time variability and random machine breakdowns in a job shop environment. *Naval Research Logistics* 59, 1 (2012), 26–38.
[105] M. Al-Behadili, D. Ouelhadj, and D. Jones. 2020. Multi-objective biased randomised iterated greedy for robust permutation flow shop scheduling problem under disturbances. *Journal of the Operational Research Society* 71, 11 (2020), 1847–1859.
[106] K. Ying. 2015. Scheduling the two-machine flowshop to hedge against processing time uncertainty. *Journal of Operational Research Society* 66 (2015), 1413–1425.
[107] V. Leon, S. Wu, and R. Storer. 1994. Robustness measures and robust scheduling for job shops. *IIE Transactions* 26, 5 (1994), 32–43.
[108] D. Rahmani and M. Heydari. 2014. Robust and stable flow shop scheduling with unexpected arrivals of new jobs and uncertain processing times. *Journal of Manufacturing Systems* 33, 1 (2014), 84–92.
[109] A. Zanchettin. 2021. Robust scheduling and dispatching rules for high-mix collaborative manufacturing systems. *Flexible Services and Manufacturing Journal* (2021), 1–24.
[110] Z. Chang, J. Ding, and S. Song. 2018. Distributionally robust scheduling on parallel machines under moment uncertainty. *European Journal of Operational Research* 272, 3 (2018), 832–846.
[111] C. Lu, K. Ying, and S. Lin. 2014. Robust single machine scheduling for minimizing total flow time in the presence of uncertain processing times. *Computers & Industrial Engineering* 74 (2014), 102–110.
[112] L. Liu, H. Gu, and Y. Xi. 2007. Robust and stable scheduling of a single machine with random machine breakdowns. *International Journal of Advanced Manufacturing Technology* 31, 7–8 (2007), 645–654.
[113] R. Larsen and M. Pranzo. 2019. A framework for dynamic rescheduling problems. *International Journal of Production Research* 57, 1–2 (2019), 16–33.
[114] F. Angel-Bello, J. Vallikavungal, and A. Alvarez. 2021. Fast and efficient algorithms to handle the dynamism in a single machine scheduling problem with sequence-dependent setup times. *Computers & Industrial Engineering* 152 (2021), 106984.
[115] A. Delgoshaei, M. Ariffin, and A. Ali. 2016. A multi-period scheduling method for trading-off between skilled-workers allocation and outsource service usage in dynamic CMS. *International Journal of Production Research* 55, 4 (2016), 997–1039.
[116] M. Aryanezhad, V. Deljoo, and S. Al e hashem. 2009. Dynamic cell formation and the worker assignment problem: A new model. *International Journal of Advanced Manufacturing Technology* 3–4 (2009), 329–342.
[117] A. Delgoshaei and A. Ali. 2017. An applicable method for scheduling temporary and skilled-workers in dynamic cellular manufacturing systems using hybrid ant colony optimization and tabu search algorithms. *Journal of Industrial and Production Engineering* 34, 6 (2017), 425–449.

[118] H. Chen, B. Du, and G. Huang. 2010. Metaheuristics to minimise makespan on parallel batch processing machines with dynamic job arrivals. *International Journal of Computer Integrated Manufacturing* 23, 10 (2010), 942–956.
[119] S. Zhou, M. Jin, and N. Du. 2020. Energy-efficient scheduling of a single batch processing machine with dynamic job arrival times. *Energy* 209, 15 (2020), 118420.
[120] Z. Cao, C. Lin, M. Zhou, and J. Zhang. 2020. Surrogate-assisted symbiotic organisms search algorithm for parallel batch processor scheduling. *IEEE-ASME Transactions on Mechatronics* 25, 5 (2020), 2155–2166.
[121] J. Pei, X. Liu, P. Pardalos, W. Fan, S. Yang, and L. Wang. 2014. Application of an effective modified gravitational search algorithm for the coordinated scheduling problem in a two-stage supply chain. *International Journal of Production Research* 70 (2014), 335–348.
[122] J. Pei, X. Liu, W. Fan, P. Pardalos, A. Migdalas, B. Goldengorin, and S. Yang. 2016. Minimizing the makespan for a serial-batching scheduling problem with arbitrary machine breakdown and dynamic job arrival. *International Journal of Advanced Manufacturing Technology* 86 (2016), 3315–3331.
[123] K. Gao, P. Suganthan, M. Tasgetiren, Q. Pan, and Q. Sun. 2015. Effective ensembles of heuristics for scheduling flexible job shop problem with new job insertion. *Computers & Industrial Engineering* 90 (2015), 107–117.
[124] K. Wang and S. Choi. 2012. A decomposition-based approach to flexible flow shop scheduling under machine breakdown. *International Journal of Production Research* 50, 1 (2012), 215–234.
[125] S. Wang and Y. Li. 2014. Variable neighbourhood search and mathematical programming for just-in-time job-shop scheduling problem. *Mathematical Problems in Engineering* 2014 (2014), 431325.
[126] M. Ahmadian, A. Salehipour, and T. Cheng. 2021. A meta-heuristic to solve the just-in-time job-shop scheduling problem. *European Journal of Operational Research* 288 (2021), 14–29.
[127] M. Pinedo and K Hadavi. *Scheduling: Theory, Algorithms and Systems Development.* Heidelberg: Spinger.
[128] A. Lackorzyński, A. Warg, M. Völp, and H. Härtig. 2012. Flattening hierarchical scheduling. In *Proceedings of the 10th ACM International Conference on Embedded Software,* 93–102.
[129] L. Zhang, X. Li, L. Wen, and G. Zhang. 2013. An efficient memetic algorithm for dynamic flexible job shop scheduling with random job arrivals. *International Journal of Software Science & Computational Intelligence* 5, 1 (2013), 63–77.
[130] A. Syed, D. Pérez, and G. Fohler. 2018. Job-shifting: An algorithm for online admission of non-preemptive aperiodic tasks in safety critical systems. *Journal of Systems Architecture* 85–86 (2018), 14–27.
[131] L. Zhou, L. Zhang, B. Sarker, Y. Laili, and L. Ren. 2018. An event-triggered dynamic scheduling method for randomly arriving tasks in cloud manufacturing .*International Journal of Computer Integrated Manufacturing* 31, 3 (2018), 318–333.
[132] X. Sun, S. Chung, F. Chan, and Z. Wang. 2018. The impact of liner shipping unreliability on the production-distribution scheduling of a decentralized manufacturing system.*Transportation Research Part E* 114 (2018), 242–269.
[133] P. Hsu, M. Aurisicchio, and P. Angeloudis. 2019. Risk-averse supply chain for modular construction projects. *Automation in Construction* 106 (2019), 102898.
[134] M. Vieira, H. Paulo, T. Pinto-Varela, and A. Barbosa-Póvoa. 2020. Assessment of financial risk in the design and scheduling of multipurpose plants under demand uncertainty.*International Journal of Production Research* 5 (2020), 1–21.
[135] Z. Wang, T. Ng, and C. Pang. 2021. Due-date quotation model for manufacturing system scheduling under uncertainty. *Discrete Event Dynamic Systems* 4 (2021), 1–23.
[136] A. Xanthopoulos, D. Koulouriotis, V. Tourassis, and D. Emiris. 2013. Intelligent controllers for bi-objective dynamic scheduling on a single machine with sequence-dependent setups. *Applied Soft Computing* 13, 12 (2013), 4704–4717.
[137] Y. Shiue, K. Lee, and C. Su. 2018. Real-time scheduling for a smart factory using a reinforcement learning approach. *Computers & Industrial Engineering* 125 (2018), 604–614.
[138] P. Paraschos, G. Koulinas, and D. Koulouriotis. 2020. Reinforcement learning for combined production-maintenance and quality control of a manufacturing system with deterioration failures. *Journal of Manufacturing Systems* 56 (2020), 470–483.
[139] X. Li, J. Wang, and R. Sawhney. 2012. Reinforcement learning for joint pricing, lead-time and scheduling decisions in make-to-order systems.*European Journal of Operational Research* 221, 1 (2012), 99–109.
[140] W. Han, F. Guo, and X. Su. 2019. A reinforcement learning method for a hybrid flow-shop scheduling problem. *Algorithms* 12, 11 (2019), 222.
[141] C. Lin, D. Deng, Y. Chih, and H. Chiu. 2019. Smart manufacturing scheduling with edge computing using multiclass deep Q network. *IEEE Transactions on Industrial Informatics* 15, 7 (2019), 4276–4284.
[142] L. Zhou, L. Zhang, and B. Horn. 2020. Deep reinforcement learning-based dynamic scheduling in smart manufacturing.*Procedia CIRP* 93 (2020), 383–388.
[143] Y. Tsai, C. Lee, T. Liu, T. Chang, C. Wang, S. Pawar, P. Huang, and J. Huang. 2020. Utilization of a reinforcement learning algorithm for the accurate alignment of a robotic arm in a complete soft fabric shoe tongues automation process. *Journal of Manufacturing Systems* 56 (2020), 501–513.

[144] A. Kintsakis, F. Psomopoulos, and P. Mitkas. 2019. Reinforcement learning based scheduling in a workflow management system. *Engineering Applications of Artificial Intelligence* 81 (2019), 94–106.
[145] H. Zhu, M. Li, Y. Tang, and Y. Sun. 2020. A deep-reinforcement-learning-based optimization approach for real-time scheduling in cloud manufacturing. *IEEE Access* 8 (2020), 9987–9997.
[146] A. Kuhnle, J. Kaiser, F. Theiß, N. Stricker, and G. Lanza. 2021. Designing an adaptive production control system using reinforcement learning. *Journal of Intelligent Manufacturing* 32 (2021), 855–876.
[147] F. Chan, H. Chan, H. Lau, and R. Ip. 2003. Analysis of dynamic dispatching rules for a flexible manufacturing system.*Journal of Materials Processing Technology* 138, 1–3 (2003), 325–331.
[148] Y. Wang and J. Usher. 2005. Application of reinforcement learning for agent-based production scheduling. *Engineering Applications of Artificial Intelligence* 18, 1 (2005), 73–82.
[149] Y. Shiue. 2009. Data-mining-based dynamic dispatching rule selection mechanism for shop floor control systems using a support vector machine approach. *International Journal of Production Research* 47, 13 (2009), 3669–3690.
[150] I. Chaouch, O. Driss, and K. Ghedira. 2019. A novel dynamic assignment rule for the distributed job shop scheduling problem using a hybrid ant-based algorithm. *Applied Intelligence* 49 (2019), 1903–1924.
[151] E. Silva, M. Costa, M. Silva, and F. Pereira. 2014. Simulation study of dispatching rules in stochastic job shop dynamic scheduling. *World Journal of Modelling and Simulation* 10, 3 (2014), 231–240.
[152] W. Mouelhi-Chibani and H. Pierreval. 2010. Training a neural network to select dispatching rules in real time.*Computers & Industrial Engineering* 58, 2 (2010), 249–256.
[153] H. Min and Y. Yih. 2003. Selection of dispatching rules on multiple dispatching decision points in real-time scheduling of a semiconductor wafer fabrication system. *International Journal of Production Research* 41, 16 (2003), 3921–3941.
[154] Y. Wang and J. Usher. 2004. Learning policies for single machine job dispatching. *Robotics and Computer-Integrated Manufacturing* 20, 6 (2004), 553–562.
[155] H. Xiong, H. Fan, G. Jiang, and G. Li. 2017. A simulation-based study of dispatching rules in a dynamic job shop scheduling problem with batch release and extended technical precedence constraints. *European Journal of Operational Research* 257, 1 (2017), 13–24.
[156] R. Guh, Y. Shiue, and T. Tseng. 2011. The study of real time scheduling by an intelligent multi-controller approach. *International Journal of Production Research* 49, 10 (2011), 2977–2997.
[157] M. Aydin and E. Öztemel. 2000. Dynamic job-shop scheduling using reinforcement learning agents. *Robotics and Autonomous Systems* 33 (2000), 169–178.
[158] G. Metan, I. Sabuncuoglu, and H. Pierreval. 2010. Real time selection of scheduling rules and knowledge extraction via dynamically controlled data mining. *International Journal of Production Research* 48, 23 (2010), 6909–6938.
[159] C. Geiger, R. Uzsoy, and H. Aytuğ. 2006. Rapid modeling and discovery of priority dispatching rules: An autonomous learning approach. *Journal of Scheduling* 9 (2006), 7–34.
[160] C. Pickardt, T. Hildebrandt, J. Branke, J. Heger, and B. Scholz-Reiter. 2013. Evolutionary generation of dispatching rule sets for complex dynamic scheduling problems. *International Journal of Production Economics* 145, 1 (2013), 67–77.
[161] A. Teymourifar, G. Ozturk, Z. Ozturk, and O. Bahadir. 2020. Extracting new dispatching rules for multi-objective dynamic flexible job shop scheduling with limited buffer spaces. *Cognitive Computation* 12 (2020), 195–205.
[162] A. Asghari, M. Sohrabi, and F. Yaghmaee. 2020. Online scheduling of dependent tasks of cloud's workflows to enhance resource utilization and reduce the makespan using multiple reinforcement learning-based agents.*Soft Computing* 24 (2020), 16177–16199.
[163] Y. Wang and J. Usher. 2007. A reinforcement learning approach for developing routing policies in multi-agent production scheduling. *International Journal of Advanced Manufacturing Technology* 33 (2007), 323–333.
[164] Y. Lee and R. Sikora. 2019. Application of adaptive strategy for supply chain agent. *Information Systems and e-Business Management* 17, 1 (2019), 117–157.
[165] S. Baer, J. Bakakeu, R. Meyes, and T. Meisen. 2019. Multi-agent reinforcement learning for job shop scheduling in flexible manufacturing systems. In *2019 Second International Conference on Artificial Intelligence for Industries (AI4I'19),* 22–25.
[166] I. Park, J. Huh, J. Kim, and J. Park. 2020. A reinforcement learning approach to robust scheduling of semiconductor manufacturing facilities. *IEEE Transactions on Automation Science and Engineering* 17, 3 (2020), 1420–1431.
[167] R. Lu, Y. Li, Y. Li, J. Jiang, and Y. Ding. 2020. Multi-agent deep reinforcement learning based demand response for discrete manufacturing systems energy management. *Applied Energy* 276 (2020), 115473.
[168] S. Qu, J. Wang, and J. Jasperneite. 2018. Dynamic scheduling in large-scale stochastic processing networks for demand-driven manufacturing using distributed reinforcement learning. In *IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA'18),* 433–440.
[169] T. Gabel and M. Riedmiller. 2012. Distributed policy search reinforcement learning for job-shop scheduling tasks. *International Journal of Production Research* 50, 1 (2012), 41–61.

[170] A. Kuhnle, J. Jakubik, and G. Lanza. 2019. Reinforcement learning for opportunistic maintenance optimization. *Production Engineering* 13 (2019), 33–41.
[171] A. Orhean, F. Pop, and I. Raicu. 2018. New scheduling approach using reinforcement learning for heterogeneous distributed systems. *J. Parallel and Distrib. Comput.* 117 (2018), 292–302.
[172] M. Silva, S. Souza, M. Souza, and A. Bazzan. 2019. A reinforcement learning-based multi-agent framework applied for solving routing and scheduling problems. *Expert Systems with Applications* 131, 1 (2019), 148–171.
[173] Q. Tan, Y. Tong, S. Wu, and D. Li. 2019. Modeling, planning, and scheduling of shop-floor assembly process with dynamic cyber-physical interactions: A case study for CPS-based smart industrial robot production. *International Journal of Advanced Manufacturing Technology* 105 (2019), 3979–3989.
[174] A. Asghari, M. Sohrabi, and F. Yaghmaee. 2019. A cloud resource management framework for multiple online scientific workflows using cooperative reinforcement learning agents. *Computer Networks* 179, 9 (2019), 107340.
[175] Y. Wang, H. Liu, W. Zheng, Y. Xia, Y. Li, P. Chen, K. Guo, and H. Xie. 2019. Multi-objective workflow scheduling with deep-Q-network-based multi-agent reinforcement learning. *IEEE Access* 7 (2019), 39974–39982.
[176] S. Chen, S. Fang, and R. Tang. 2019. A reinforcement learning based approach for multi-projects scheduling in cloud manufacturing. *International Journal of Production Research* 57, 10 (2019), 3080–3098.
[177] Y. Kim, S. Lee, J. Son, H. Bae, and B. Chung. 2020. Multi-agent system and reinforcement learning approach for distributed intelligence in a flexible smart manufacturing system. *Journal of Manufacturing Systems* 57 (2020), 440–450.

Received 16 August 2021; revised 16 October 2022; accepted 14 February 2023