Journal of Intelligent Manufacturing (2023) 34:905–929
https://doi.org/10.1007/s10845-021-01847-3

Reinforcement learning applications to machine scheduling problems: a comprehensive literature review

Behice Meltem Kayhan[1] · **Gokalp Yildiz**[1]

Received: 2 December 2020 / Accepted: 17 September 2021 / Published online: 19 October 2021

Abstract

Reinforcement learning (RL) is one of the most remarkable branches of machine learning and attracts the attention of researchers from numerous fields. Especially in recent years, the RL methods have been applied to machine scheduling problems and are among the top five most encouraging methods for scheduling literature. Therefore, in this study, a comprehensive literature review about RL methods applications to machine scheduling problems was conducted. In this regard, Scopus and Web of Science databases were searched very inclusively using the proper keywords. As a result of the comprehensive research, 80 papers were found, published between 1995 and 2020. These papers were analyzed considering different aspects of the problem such as applied algorithms, machine environments, job and machine characteristics, objectives, benchmark methods, and a detailed classification scheme was constructed. Job shop scheduling, unrelated parallel machine scheduling, and single machine scheduling problems were found as the most studied problem type. The main contributions of the study are to examine essential aspects of reinforcement learning in machine scheduling problems, identify the most frequently investigated problem types, objectives, and constraints, and reveal the deficiencies and promising areas in the related literature. This study can help researchers who wish to study in this field through the comprehensive analysis of the related literature.

Keywords Reinforcement learning · Q-learning · Machine scheduling · Job shop scheduling problem · Parallel machine scheduling problems

Introduction

The history of RL goes back to the early days of neuroscience, computer science, psychology, and cybernetics. In the most general sense, RL mimics the simple principles of animal or human cognition. Animal or human cognition tends to perform the actions which have positive consequences and learns these consequences by trial and error (Sigaud & Buffet, 2013). RL is generally formulated as a decision problem to find optimal policies in which actions are mapped to states. The learning system does not tell which action is the best; instead, it only evaluates the consequences of the action for the existing state (Barto & Mahadevan, 2003).

The RL approach is widely used in production and operations management problems. Particularly in decision problems with dynamic environments, it is shown that the RL approach can model a wide range of problems, and it can get better results than conventional methods. Machine scheduling problems are one of these decision problems where the RL algorithms are frequently studied, and good results are obtained (Lihu & Holban, 2009).

Machine scheduling is defined as the sequencing of n jobs that have to be processed on m machines to minimize or maximize predefined objective function under certain constraints (Graham et al., 1979). It is directly related to the operational activities of a company and has a very crucial impact on competitiveness, efficiency, diminution of production cost, on-time delivery, and customer satisfaction (Fuchigami & Rangel, 2018). Therefore, machine scheduling problems have been studied in the literature broadly, and different algorithms developed to solve these problems efficiently (Lihu & Holban, 2009; Neto & Godinho Filho, 2013; Fuchigami & Rangel, 2018).

The first application of the RL method to a scheduling problem was conducted by Zhang and Dietterich (1995) for the NASA space shuttle payload process to obtain a feasible schedule without any violation of resource constraint while

✉ Behice Meltem Kayhan
meltem.kayhan@deu.edu.tr

[1] Department of Industrial Engineering, Dokuz Eylul University, 35397 Buca, Izmir, Turkey

minimizing the duration of the process. Afterward, many researchers have applied RL algorithms to different machine scheduling problems (Kim & Lee, 1996; Riedmiller & Riedmiller, 1999; Aydin & Öztemel, 2000; Paternina-Arboleda & Das, 2001). Especially in machine scheduling problems that have a large set of data with dynamic environments, it is shown that the RL algorithms work well, and they are among the top five most encouraging methods for scheduling literature (Lihu & Holban, 2009).

The RL methods are applied to machine scheduling problems actively in recent years. Parente et al. (2020) showed that machine learning methods and Industry 4.0 concepts are getting to attract the attention of researchers and frequently applied to production planning, control, and scheduling problems. Cadavid et al. (2020) indicated that RL methods are actively applied to machine scheduling problems in recent years. Therefore, this paper aims to conduct a comprehensive literature review about the RL method applications to machine scheduling problems and shed light on the related literature gap.

For this purpose, related databases are searched from 1995 up today exhaustively, and 80 papers were found in the scope of the study. These papers were reviewed considering different aspects of the problem such as applied algorithm, machine environment, job and machine characteristics, objectives, benchmark method, and a detailed classification scheme was constructed. Afterward, the papers were analyzed and interpreted in the context of the machine environment, constraints, and objectives. Through the detailed analysis, this study provides insight to researchers who would like to work in this field and reveals the trends and deficiencies in the literature, the most studied dimensions of the problem, the potential research areas, etc.

This paper is separated into seven sections: "Methodology and framework" section introduces the methodology and framework of the research. In "Reinforcement learning secion, the RL method is defined. "Machine scheduling" section describes the essential aspects of machine scheduling. "Classification and quantitative analysis of the literature" section shows the classification and quantitative analysis of the relevant literature. "Overview of the literature" section gives an overview of the literature according to machine scheduling problem types, and "Conclusion" section presents the concluding remarks.

Methodology and framework

In this study, the papers that handle the application of RL algorithms to machine scheduling problems have been reviewed comprehensively. Primarily, Scopus (https://www.scopus.com) and Web of Science (webofknowledge.com/WOS) databases were searched using the following keywords in the whole articles:

Reinforcement Learning" and "Scheduling
Q-Learning" and "Scheduling
Neuro-dynamic Programming" and "Scheduling

In the first part of the study, the literature search was made very inclusively to prevent any paper in this field from being overlooked. All search results were scanned by reading the abstracts of the papers, and the papers that include scheduling problems in production systems were identified. Both conference papers and articles were included. As a result of the comprehensive search, 110 papers were found. Firstly, these papers were examined in terms of problem type, and 16 papers dealing with supply chain management, economic lot scheduling, load carrier scheduling, maintenance scheduling, gantry scheduling, job remaining time prediction, transfer unit scheduling, etc. eliminated. Afterward, solution methods in these papers were analyzed, and 14 papers in which RL algorithms were applied as an auxiliary method were eliminated. After these eliminations, 80 papers have been found in the scope of this study.

The papers are classified into eight following categories for further analysis and are given in Table 1.

1. *Learning algorithm* defines the base algorithm of RL used in the paper.
2. *Problem type* gives information about the machine environment, job characteristics, and objective function of the problem is explained in detail in "Machine scheduling" section.
3. *Objective* indicates whether the problem studied is a multi-objective (MO) problem or a single objective (SO) problem.
4. *Agent* indicates whether systems are modeled as Single-Agent (S) or modeled as Multi-Agent (M).
5. *Benchmark method* indicates the methods used to compare the performance of RL.
6. *Action selection* indicates how the exploration and exploitation processes are balanced.
7. *State definition* indicates the state definition strategy to handle the curse of dimensionality.
8. *System type* indicates whether the analyzed problem is modeled as stochastic or deterministic.

Reinforcement learning

RL is one of the branches of machine learning. The primary purpose of RL is to maximize the rewards gained by an agent in consequence of the action about the state of the environment. Unlike other machine learning algorithms, the agent is

Table 1 Classification of the literature review

	Author	Year	Learning algorithm	Problem type	Obj.	Agent	Benchmark method	Action selection	State definition	System type
1	Zhou et al.	2020	Q-learning	$R_m//C_{max}$	SO	S	No Benchmark	Not Mentioned	Neural Network	Stochastic
2	Park et al.	2020	Deep Q Network	$J/S_{jk}/C_{max}$	SO	M	GA and Dispatching Rules	ε-greedy	Neural Network	Stochastic
3	Guo et al.	2020	Q-learning	$Q_m//T_{max} + \sum U_i + \frac{1}{n}\sum W_i$	MO	S	Dispatching Rules	ε-greedy	Aggregation	Stochastic
4	Wang et al.	2020	Q-learning	$J//\sum w_i E_i + TC$	MO	S	Dispatching Rules	ε-greedy	Clustering	Deterministic
5	Fang Guo et al.	2020	Q-learning	$HF//C_{max}$	SO	S	GA	ε-greedy	All States	Deterministic
6	Lee et al.	2020	Deep Q Network	$J//\sum wT_i$	SO	M	Dispatching Rules	ε-greedy	Neural Network	Deterministic
7	Luo	2020	Deep Q Network	$J//\sum T_i$	SO	S	Dispatching Rules	Boltzmann Probability	Neural Network	Stochastic
8	Liu et al.	2020	Actor Critic	$J//C_{max}$	SO	M	Dispatching Rules	ε-greedy	Neural Network	Stochastic
9	Ren et al.	2020	Actor Critic	$J//C_{max}$	SO	S	Benchmark Problems	Boltzmann Probability	Neural Network	Deterministic
10	Jiménez et al.	2020	Q-learning	$J//C_{max}$	SO	M	Benchmark Problems	ε-greedy	All States	Deterministic
11	Ábrahám et al.	2019	Q-learning	$R_m/prec/C_{max}$	SO	S	Benchmark Problems	Boltzmann Probability	All States	Deterministic
12	Reyna et al.	2019a	Q-learning	$F//C_{max}$	SO	S	Benchmark Problems	ε-greedy	All States	Deterministic
13	Reyna et al.	2019b	Q-learning	$HF/S_{jk}prec/C_{max}$	SO	M	Benchmark Problems	ε-greedy	All States	Deterministic
14	Han et al.	2019	Q-learning	$HF//C_{max}$	SO	S	AIS and GA	Boltzmann Probability	All States	Deterministic
15	Zhao et al.	2019	Q-learning	$J/bkdwn/C_{max}$	SO	S	Dispatching Rules	ε-greedy	Aggregation	Stochastic
16	Lin et al.	2019	Deep Q Network	$J//C_{max}$	SO	S	Dispatching Rules	ε-greedy	Neural Network	Deterministic
17	Waschneck et al.	2018a	Deep Q Network	$J/S_{jk}/U$	SO	M	Dispatching Rules	ε-greedy	Neural Network	Stochastic
18	Waschneck et al.	2018b	Deep Q Network	$J/S_{jk}/CT$	SO	M	Dispatching Rules	ε-greedy	Neural Network	Stochastic
19	Reyna et al.	2018	Q-learning	$F//C_{max}$	SO	S	Benchmark Problems	ε-greedy	Not Mentioned	Deterministic
20	Stricker et al.	2018	Q-learning	$J/bkdwnblock/U$	SO	S	Dispatching Rules	ε-greedy	Neural Network	Stochastic
21	Shiue et al.	2018	Q-learning	$J//CT + \sum U_i + \text{TP}$	MO	S	Dispatching Rules	ε-greedy	Neural Network	Stochastic
22	Wang	2018	Q-learning	$J//\sum E_i + \sum T_i$	MO	M	RL Algorithms	ε-greedy	Clustering	Stochastic
23	Thomas et al.	2018	Q-learning	$J//TP$	SO	S	Benchmark Problems	ε-greedy	Neural Network	Stochastic
24	Zhang et al.	2018	Q-learning	$J//CT$	SO	S	Dispatching Rules	ε-greedy	Neural Network	Stochastic
25	Bouazza et al.	2017	Q-learning	$J/S_{jk}prec/\sum w_i W_i$	SO	M	Dispatching Rules	Boltzmann Probability	Not Mentioned	Stochastic
26	Zhang et al.	2017	Q-learning	$J//CT$	SO	S	Dispatching Rules	ε-greedy	Neural Network	Stochastic
27	Xiao et al.	2017	Q-learning	$1//CT$	SO	S	Relative Value Iteration	SA Temperature Procedure	Aggregation	Stochastic
28	Kim and Shin	2017	Actor Critic	$J/S_{jk}bkdwn/CT, WIP$	MO	S	Dispatching Rules	Not Mentioned	Not Mentioned	Stochastic
29	Arviv et al.	2016	Q-learning	$F//C_{max}$	SO	S	Dispatching Rules	ε-greedy	Aggregation	Stochastic
30	Qu et al.	2016a	Q-learning	$HF/S_{jk}block/TC + \sum U_i$	MO	M	No Benchmark	ε-greedy	Gradient Descent Method	Stochastic
31	Qu et al.	2016b	Approximate RL	$J/block/WIP + \sum U_i$	MO	S	Dispatching Rules	ε-greedy	Gradient Descent Method	Stochastic
32	Wang and Yan	2016	Q-learning	$J//\frac{1}{n}\sum T_i$	SO	M	RL Algorithms	ε-greedy	Clustering	Stochastic
33	Yuan et al.	2016	Q-learning	$P_m/bkdwn/L_{max}, \sum U_i$	SO	S	Dispatching Rules	ε-greedy	Aggregation	Stochastic

Table 1 continued

	Author	Year	Learning algorithm	Problem type	Obj.	Agent	Benchmark method	Action selection	State definition	System type
34	Qu et al.	2015	Q-learning	$HF/S_{jk}block/WIP+TC$	MO	S	Dispatching Rules	ε-greedy	Not Mentioned	Deterministic
35	Reyna et al.	2015	Q-learning	$J//C_{max}$	SO	S	Benchmark Problems	ε-greedy	Not Mentioned	Deterministic
36	Xanthopoulos et al.	2013	Q-learning	$1/S_{jk}/\frac{1}{n}\sum T_i+\frac{1}{n}\sum E_i$	MO	S	Fuzzy Logic MOEA	ε-greedy	Aggregation	Stochastic
37	Wang and Yan	2013a	Q-learning	$J/bkdwnblock/\frac{1}{n}\sum T_i$	SO	M	RL Algorithms	ε-greedy	Clustering	Stochastic
38	Wang and Yan	2013b	Q-learning	$J//\frac{1}{n}\sum E_i$	SO	S	Dispatching Rules	ε-greedy	Gradient Descent Method	Stochastic
39	Atighehchian and Sepehri	2013	Q-learning	$1/S_{jk}/\sum w_iT_i$	SO	S	Dispatching Rules	ε-greedy	Neural Network	Stochastic
40	Yuan et al.	2013	Q-learning	$P_m//L_{max},\sum U_i$	SO	S	Dispatching Rules	ε-greedy	Aggregation	Stochastic
41	Aissani et al.	2012	SARSA	$J/prec/C_{max}$	SO	M	Genetic Algorithm MILP	Boltzmann Probability	Not Mentioned	Not Mentioned
42	Zhang et al.	2012	R-learning	$R_m//\sum w_iT_i$	SO	S	Dispatching Rules	ε-greedy	Gradient Descent Method	Stochastic
43	Palombarini and Martínez	2012a	Relational RL	$R_m/S_{jk}bkdwn/\sum T_i$	SO	S	Benchmark Problems	ε-greedy	Relational Regression Tree	Stochastic
44	Palombarini and Martínez	2012b	Relational RL	$R_m/S_{jk}bkdwn/\sum T_i$	SO	S	No Benchmark	ε-greedy	Relational Regression Tree	Stochastic
45	Zhang et al.	2011	SARSA	$R_m/S_{jk}prec/TC$	SO	S	Benchmark Problems	ε-greedy	Gradient Descent Method	Stochastic
46	Gabel and Riedmiller	2011	Q-learning	$J//C_{max}$	SO	M	Benchmark Problems	ε-greedy	All States	Stochastic
47	Iwamura et al.	2010	Q-learning	$R_m//\sum F_i+TC$	MO	S	Average Improvement	ε-greedy	All States	Not Mentioned
48	Palombarini and Martínez	2010	Relational RL	$R_m//\sum T_i$	SO	S	Average Improvement	ε-greedy	Relational Regression Tree	Stochastic
49	Aissani et al.	2009	SARSA	$J//C_{max}$	SO	M	Genetic Algorithm	Boltzmann Probability	Not Mentioned	Stochastic
50	Yingzi et al.	2009	Q-learning	$J//C_{max}$	SO	S	Dispatching Rules	ε-greedy	Aggregation	Stochastic
51	Ramírez-Hernández and Fernandez	2009	Temporal Difference	$J/blockS_{jk}bkdwn/WIP$	SO	S	Dispatching Rules	ε-greedy	Aggregation	Stochastic
52	Yang and Yan	2009	Q-learning	$J//\frac{1}{n}\sum T_i$	SO	S	Dispatching Rules	ε-greedy	Aggregation	Stochastic
53	Aissani and Trentesaux	2008	SARSA	$J/bkdwn/C_{max}+MDT$	MO	S	Benchmark Problems	Boltzmann Probability	Not Mentioned	Stochastic
54	Csáji and Monostori	2008	Q-learning	$J/S_{jk}pmtn/\sum U_i$	SO	S	Benchmark Problems	Boltzmann Probability	SVR	Stochastic
55	Gabel and Riedmiller	2008	Q-learning	$J//C_{max}$	SO	M	Dispatching Rules	ε-greedy	Neural Network	Stochastic
56	Gabel and Riedmiller	2007a	Q-learning	$J//C_{max}$	SO	M	Dispatching Rules	ε-greedy	Neural Network	Stochastic
57	Gabel and Riedmiller	2007b	Q-learning	$J//C_{max}$	SO	M	Dispatching Rules	ε-greedy	Neural Network	Stochastic

Table 1 continued

	Author	Year	Learning algorithm	Problem type	Obj.	Agent	Benchmark method	Action selection	State definition	System type
58	Yang and Yan	2007	Q-learning	$J//\frac{1}{n}\sum T_i$	SO	S	Dispatching Rules	ε-greedy	Clustering	Stochastic
59	Zhang et al.	2007	Q-learning	$R_m/S_{jk}/\frac{1}{n}\sum w_i T_i$	SO	S	Dispatching Rules	ε-greedy	Gradient Descent Method	Stochastic
60	Wang and Usher	2007	Q-learning	$J/S_{jk}/\frac{1}{n}\sum T_i$	SO	S	Dispatching Rules	ε-greedy	Aggregation	Stochastic
61	Gabel and Riedmiller	2006a	Q-learning	$J/S_{jk}/\frac{1}{n}\sum T_i$	SO	S	Dispatching Rules	ε-greedy	Neural Network	Deterministic
62	Gabel and Riedmiller	2006b	Q-learning	$J/precbkdwn/\sum T_i$	SO	M	Dispatching Rules	ε-greedy	CBR	Stochastic
63	Csáji et al.	2006	Temporal Difference	$J/bkdwn/C_{max}$	SO	M	No Benchmark	SA Temperature Procedure	Neural Network	Stochastic
64	Monostori and Csáji	2006	Q-learning	$J/pmtnS_{jk}/\sum L_i$	SO	S	Benchmark Problems	Boltzmann Probability	SVR	Stochastic
65	Idrees et al.	2006	SMART	$1//\frac{1}{n}\sum T_i + TC$	MO	S	Dispatching Rules	ε-greedy	Not Mentioned	Stochastic
66	Wang and Usher	2005	Q-learning	$1//L_{max}, \sum U_i, \frac{1}{n}\sum L_i$	SO	S	Dispatching Rules	ε-greedy	Aggregation	Stochastic
67	Kong and Wu	2005	Q-learning	$1//\frac{1}{n}\sum T_i, T_{max}, \sum U_i$	SO	S	Dispatching Rules	SA Temperature Procedure	Aggregation	Stochastic
68	Csáji and Monostori	2005	Q-learning	$R_m/bkdwnprec/C_{max}$	SO	S	No Benchmark	SA Temperature Procedure	Gradient Descent Method	Stochastic
69	Ramírez-Hernández and Fernandez	2005	Q-learning	$J/block/TC$	SO	S	Modified Policy Iteration	ε-greedy	All States	Stochastic
70	Monostori et al.	2004	Temporal Difference	$J/prec/C_{max}$	SO	M	Branch and Bound	Boltzmann Probability	Neural Network	Stochastic
71	Hong and Prabhu	2004	Q-learning	$F/S_{jk}/DD + SC$	MO	S	Dispatching Rules	ε-greedy	All States	Deterministic
72	Wang and Usher	2004	Q-learning	$1//\frac{1}{n}\sum T_i$	SO	S	Dispatching Rules	ε-greedy	Aggregation	Stochastic
73	Wei and Zhao	2004	Q-learning	$J//\frac{1}{n}\sum T_i$	SO	S	Dispatching Rules	ε-greedy	Aggregation	Stochastic
74	Paternina-Arboleda and Das	2001	SMART	$F/bkdwn/WIP$	SO	S	Kanban CONWIP Behavior-Based Control (BBC) policy	ε-greedy	All States	Stochastic
75	Liu et al.	2001	Temporal Difference	$J/block/TP$	SO	S	Dispatching Rules	Not Mentioned	All States	Deterministic
76	Aydin and Öztemel	2000	Q-learning	$J//\frac{1}{n}\sum T_i$	SO	S	Dispatching Rules	ε-greedy	Aggregation	Stochastic
77	Miyashita	2000	Temporal Difference	$J//\sum w_i T_i + WIP$	MO	S	Random Operator	ε-greedy	CBR	Stochastic
78	Riedmiller and Riedmiller	1999	Q-learning	$J//\sum T_i$	SO	M S	Dispatching Rules	ε-greedy	Neural Network	Stochastic
79	Zhang and Dietterich	1996	Temporal Difference	$J/prec/C_{max}$	SO	S	Neural Network	SA Temperature Procedure	Neural Network	Stochastic
80	Zhang and Dietterich	1995	Temporal Difference	$J/prec/C_{max}$	SO	S	Iterative Repair	SA Temperature Procedure	Neural Network	Stochastic

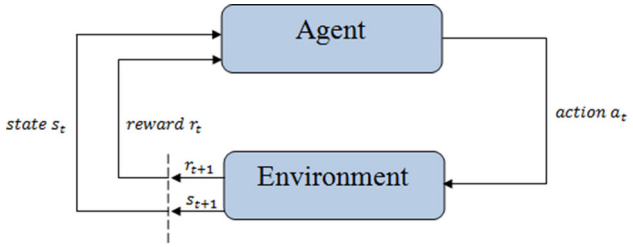


Fig. 1 The framework of the RL

not told which action should be taken, but instead, the agent should learn which actions return the most reward by trial and error (Liu & Wang, 2009).

In the standard RL model, the agent refers to the decision-making unit which observes the environment and takes action. Each action will produce a result, which could be a reward or a punishment according to the state of the environment. Rewards or punishments received from the environment are used to evaluate actions. The agent decides which actions will be taken by itself, considering which actions yield the most rewards. The result of executing an action is evaluated via a reward function regarding the current state and the current action. The framework of RL is given in Fig. 1 (Kaelbling et al., 1996).

At each decision step t, the agent observes the system state s_t, executes an action a_t and receives a scalar reward r_t. After executing action a_t the current state of the environment changes and the agent observes the next state s_{t+1} and takes the next action a_{t+1}. The algorithm keeps the values of the action and current state of the environment at each decision step and uses this information to evaluate the next step (Kaelbling et al., 1996).

After several iterations with a sufficient learning level, the agent takes the action to optimize the total reward in the long term. The primary purpose of the agent is to discover a policy that maximizes long-run objectives, which is defined as a value function. The value function evaluates the states in terms of the total amount of reward gained by an agent (Sutton & Barto, 1998).

RL primarily solves Markov Decision Problems (MDP), which consists of the following variables: a finite set of environment states S, a finite set of actions A, a reward function $R_a(s, s')$, and transition probabilities between states $T_a\left(s, s'\right) \rightarrow [0,1]$.

The solution of an MDP consists of a policy $\pi : S \rightarrow A$ that maps actions to states. An optimal policy $\left(\pi^*\right)$ defines the best action in each state which provides a maximum reward in the long term. This long-term reward is named "value" and calculated by value functions. The decision-maker selects the value function depending on the type of problem (Sutton & Barto, 1998).

In this study, we analyzed the papers by considering the categories of learning algorithm, action selection, and the state definition which are explained in "Methodology and framework" section. In the following subsections, we will briefly introduce the methods frequently encountered under these categories in the related literature. Detailed information on the methods can be found in the references given in the relevant sections.

Learning algorithm

The learning algorithm defines the base algorithms of the RL used in the papers. According to issues such as reward system, state representation, value update procedure, and policy adaptation, these algorithms differ from each other. The learning algorithm category shows the base RL algorithms that are used in the papers. As a result of the analysis, the primary learning algorithms used in the paper are *Q-learning, SARSA, R-Learning, SMART, Deep Q Network, Approximate RL, Relational RL, Temporal Difference (λ)* and *Actor-Critic*.

In the Q-learning algorithm, a frequently used RL algorithm, action-value functions are stored in Q-values, $Q(s, a)$, expressing the utility of selecting action a in state s. The principle of the algorithm depends on the value iteration update. At each decision step t, the agent observes the state s_t, chooses the action a_t with the maximum Q-value, receives the reward r_{t+1} and observes the new state s_{t+1}. Then Q-value $Q(s, a)$ is updated as in the following formula:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[r_{t+1} + \gamma max_a Q(s_{t+1}, a) - Q(s_t, a_t)\right] \quad (1)$$

In this equation, γ denotes the discount factor, $0 \leq \gamma \leq 1$, and adjusts the balance between immediate rewards and long term rewards, α is learning rate, $0 < \alpha \leq 1$, and denotes how much newly acquired information influences the old information. Determination of these parameters is a crucial factor in terms of the performance of algorithms (Sutton & Barto, 1998).

Unlike the Q-learning algorithm, the Q-values update procedure is based on the taken actions instead of the best possible action in the SARSA algorithm. In the Q-learning algorithm, Q-value is updated considering the action with maximum Q-value in the next state (Sutton & Barto, 1998). In contrast, in the SARSA algorithm, the agent chose an action with the same policy in the next state, and Q-value is updated considering the Q-value of this action as represented in the following formula:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[r_{t+1} + \gamma\, Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right] \quad (2)$$

R-learning is an RL algorithm proposed by Schwartz (1993) which takes into consideration the average reward. *R-learning* considers the best possible action to update the Q-values same as in the Q-learning, but the average reward

is substituted from the immediate reward as depicted in the following formula:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[r_{t+1} - \rho + \gamma max_a Q(s_{t+1}, a) - Q(s_t, a_t)\right] \tag{3}$$

$$\rho = \sum_{t=1}^{u} \frac{r_t}{u} \tag{4}$$

In this equation, ρ is the approximation of the expected reward per decision time step under policy π, and u is the number of the decision time step.

Another average reward RL algorithm *SMART* (Semi-Markov Average Reward Technique) is developed by Das et al. (1999). Differently from R-learning, it takes into account the T, which represents the total time till the decision step, instead of the number of the decision step and calculated as follows:

$$\rho = \frac{\sum_{t=0}^{T} r}{T} \tag{5}$$

Deep Q Network combines the RL with a class of neural networks named deep neural networks. In a deep neural network, multiple hidden layers are used to gradually construct more precise representations of the data (Mnih et al., 2015). This architecture is utilized to approximate the value functions in the RL problems.

Approximate RL is also based on the value function approximation instead of the precise Q-values. A weight factor is assigned to each feature of the state, and the algorithm approximates a Q-value. After each decision step, the weight is updated considering the difference between the prediction value and the target value (Qu et al., 2016b).

Relational RL utilizes the RL with the aid of logical and relational representations. In this algorithm, the states and the action are depicted as logical facts, and the Q-values are stored in a regression tree instead of an explicit Q-table (De Raedt, 2008).

Temporal Difference (λ) algorithm pays regard to action history, which contributes to reaching the goal state, unlike the Q-learning and SARSA algorithm, which considers the last action before the goal state. When a reward is received for the current state, all states are updated according to their eligibility, which means the degree of the states has been visited recently. The algorithm aims to estimate the value function of the states instead of the state-action values (Sutton & Barto, 1998).

As mentioned before, the agent takes action in the state of the environment to reach a goal. In the *Actor-Critic* algorithm, this agent consists of two main interacting processes: actor and critic. The critic architecture evaluates the current policy to help the actor. The actor architecture tries to improve current policy considering the feedback that comes from the critic (Szepesvári, 2010).

Action selection

Another critical issue that directly affects the performance of algorithms is the *action-selection* mechanism corresponding to the trade-off between exploration and exploitation. In this context, exploration defines finding more information about the environment, and exploitation defines taking actions with available information to maximize reward (Sutton & Barto, 1998). In the exploitation process, the agent may take advantage of known actions that yield better rewards, but the exploration process may allow the agent to find the maximum total reward in the long term.

This trade-off issue is generally handled with the ε-*greedy Method*. This method involves selecting an action randomly with a small probabilityϵ and selecting the action with the maximum Q-value with probability $(1-\epsilon)$ (Gosavi, 2015). In the absence of information about the environment, the agent should discover the environment by exploration, which is crucial for learning. In this case, random selection probability ε initialized with a high value to perform exploration, and it is decreased over time. *Boltzmann Probability* and *Simulated Annealing (SA) Temperature Procedure* approaches are also widely used to balance exploration and exploitation in the relevant literature.

In the *Boltzmann Probability*, the selection probability of each action is proportional to its Q-value. Unlike the ε-greedy method that takes random action or takes action with maximum Q-value, this approach takes action with weighted probabilities (Sutton & Barto, 1998). The selection probability of action in a given state is calculated as in Eq. 6, where τ_t is a temperature parameter that decreased over time.

$$p_t(a|s_t) = \frac{e^{\frac{Q(s_t,a)}{\tau_t}}}{\sum_a e^{\frac{Q(s_t,a)}{\tau_t}}} \tag{6}$$

Simulated Annealing (SA) Temperature Procedure is a hybrid approach used to improve the *Boltzmann Probability* or the ε-*greedy Method.* The temperature parameter of Boltzmann probability or value of the ϵ is decreased over time as the agent gathered enough information. In this way, the exploration will be decreased gradually and will converge to an optimum value.

State definition

As mentioned before, the agent observes the state and then takes action in RL. *State definition* is very crucial since many problems have continuous states or state-space is very large. It is too hard to calculate and store the value function for

each state-action pair in such circumstances. This problem is known as the curse of dimensionality (Gosavi, 2015). To deal with this problem, state aggregation methods or function approximators are used to approximate the value functions.

As a result of the analysis of the related literature, the mainly handled state definition strategies were found and these strategies categorized under the sections as *All States, Aggregation, Clustering, Neural Network (NN), Case-Based Reasoning (CBR)*, S*upport Vector Regression (SVR), Regression Tree*, and *Gradient Descent Method.*

The category of *All States* indicates that each state of the environment is situated in the Q-table. This method is applied only to small-scale problems, and generally, states are defined by a single feature. In cases where the state space gets a bit bigger, the *Aggregation* strategy can be applied. In this strategy, states are lumped together to reduce the state-space. States are generally defined by one or two features, and the ranges based on these features are determined in advance to categorize states. In *Clustering*, the states are defined by a set of features. The aim is to cluster these states on the basis of these features. Clusters are constructed considering the similarity or dissimilarity of the states.

NN is a learning technique that is utilized to match input data to output data and inspired by the neural structure of the human brain. It consists of many neurons that help link given input data to output data by calculating weights of the linkages. It is trained on labeled data (i.e., inputs with output) to calculate the input weights and tunes the input weights until minimizing the network's error. Once the training process is completed, *NN* gains the ability to generalize new, unseen inputs into accurate outputs (Russel & Norvig, 2010). In the related literature, it is used to predict the Q-value of action in a given state.

CBR relies on the idea that similar problems have similar solutions. This idea can be explained in the RL concept as similar states have the same Q-value for the same action. A case consists of a problem part which includes the state characteristics, and a solution part includes the Q-values of actions. Visited state-action pairs are stored in a case base. When an agent observes a new state, a similarity value between the new state and each state in the case base is calculated. The state with the highest similarity value is selected, and the action that yields the highest Q-value is executed (Gabel & Riedmiller, 2006b).

SVR is a statistical learning technique that aims to find output values based on input values by finding a function with an acceptable error value (Vapnik, 2000). The features of state and actions are stored in a vector form. The value of action depending on these features is found by modeling nonlinear relationships, and never experienced states can be estimated (Csáji & Monostori, 2008).

A *Regression Tree* is a particular type of decision tree that handles real-valued variables (Van Otterlo, 2009). The regression tree creates different input space subsets by splitting input space based on specific limit values and aims to predict a target value. In the related literature, this approach was used to split state-action space into subsets, and the Q-values of the state-action pair were stored in the regression tree (Palombarini & Martínez, 2012a) .

In the *Gradient Descent Method*, states are represented by feature vectors, and the value functions are constructed by a linear combination of these vectors. The main purpose is to find weights of the linear functions minimizing the difference between real Q-values and predicted Q-values. In this regard, the value function is reorganized by updating the weights using the gradient of Q-values with respect to weights of functions (Zhang et al., 2012).

Machine scheduling

Machine scheduling deals with the sequencing of jobs that have to be processed on a collection of machines under some constraints to optimize the predefined objective function (Nahmias & Olsen, 2015). The general restriction about the problem is that each machine can process at most one job, and each job can be processed on at most one machine simultaneously (Graham et al., 1979). Proper scheduling ensures meeting demands timely, minimizing work-in-process inventory, reducing set up times, increasing the utilization of machines, and minimizing worker and production costs (Nahmias & Olsen, 2015).

In the literature, scheduling problems are symbolized with three attributes using the notation $\alpha/\beta/\gamma$, where α represents the machine environment, β represents the job characteristics, and γ represents the optimality criteria (Graham et al., 1979). This classification scheme is known as Graham's notation. Further details of these attributes are given in the following sections.

Machine environment

In machine scheduling problems, the machine environment also represents the type of problem. Mainly studied machine environments are (1) Single Machine, (2) Parallel Machine, (3) Flow Shop, (4) Job Shop, and (5) Open Shop.

1. Single Machine: There is only one machine on which all jobs have to be processed.
2. Parallel Machine: There are two or more machines, but all jobs must be processed on only one of these machines. Machines in this environment can be *Identical, Uniform, and Unrelated. Identical Parallel Machine* indicates that all machines can process all jobs at the same speed. *Uniform Parallel Machine* indicates that all machines can process all jobs, but the speed of the machines varies.

Unrelated Parallel Machine indicates that processing time depends on both job and machine.

3. Flow Shop: There is a set of jobs that have to be processed by a set of machines in series, and each job follows the same order through the machines. A *Hybrid Flow Shop* integrates flow shop and parallel machine in which there are s stages in series, and at least one of these stages contains multiple machines. This type of machine environment is also named *Flexible Flow Shop*.
4. Job Shop: Differently from the Flow Shop, each job has a specific route through the machines.
5. Open Shop: The route of the jobs can be changed; in other words, there is no constraint on the processing order of each job.

Job characteristics

The second attribute, which is symbolized as β gives information about the constraints of the problem and processing characteristics. Common characteristics and constraints are *release date, preemption, precedence constraints, sequence-dependent setup times, limited buffer size*, and *machine breakdown*. The list and abbreviations of the job characteristics and constraints are given in Table 2.

Release date implies that the release date of the job may vary.
Preemption implies that the processing of a job may be abandoned and resumed later.
Precedence constraints imply that a job can be started only after its predecessor jobs have been completed.
Sequence-dependent setup times imply that there is a need for setup time between different types of jobs. Each machine's setup time of a particular job is determined by both that job and the previous job that has been processed on that machine.
Limited buffer size implies that a blockage of the machine may occur due to the limited buffer area of the next machine.
Machine breakdown indicates that machines cannot work in some periods due to damage, repair, etc.

Optimality criteria

Let us assume that there are n jobs J_i, $(i = 1,2,\ldots,n)$ with the attributes of t_i, r_i, p_i, d_i, w_i which refers to the start time, release date, processing time, due date, and weight of J_i, respectively and C_i refers to the completion time of J_i. The basic properties of job i are as follows:

$$\text{Lateness: } L_i = C_i - d_i \tag{7}$$

Table 2 Graham notations

Machine environment (α)		Optimality criteria (γ)	
Acronym	Meaning	Acronym	Meaning
1	Single machine	C_{max}	Makespan
P_m	Identical parallel machine	L_{max}	Maximum lateness
Q_m	Uniform parallel machine	T_{max}	Maximum tardiness
R_m	Unrelated parallel machine	$\sum F_i$	Total flow time
J	Job shop	$\sum T_i$	Total tardiness
F	Flow shop	$\sum L_i$	Total lateness
HF	Hybrid flow shop	$\sum E_i$	Total earliness
O	Open shop	$\sum W_i$	Total waiting time
Job Characteristics (β)		$\sum U_i$	Total number of tardy jobs
Acronym	Meaning	$\sum w_i T_i$	Total weighted tardiness
r_j	Release date	$\sum w_i W_i$	Total weighted waiting time
$pmtn$	Preemption	$\frac{1}{n}\sum w_i T_i$	Mean weighted tardiness
$prec$	Precedence constraints	$\frac{1}{n}\sum L_i$	Mean lateness
S_{jk}	Sequence dependent setup times	$\frac{1}{n}\sum T_i$	Mean tardiness
$bkdwn$	Machine breakdown	WIP	Work-in-process
$block$	Limited buffer size	CT	Cycle time
		MDT	Machine down time
		TC	Total cost
		DD	Due-date deviation
		TP	Throughput
		U	Utilization
		SC	Setup cost

$$\text{Tardiness: } T_i = \max(C_i - d_i, 0) \tag{8}$$

$$\text{Earliness: } E_i = \max(d_i - C_i, 0) \tag{9}$$

$$\text{Flow Time: } F_i = C_i - r_i \tag{10}$$

$$\text{Waiting Time: } W_i = C_i - r_i - p_i \tag{11}$$

$$\text{Makespan: } C_{max} = \max\{C_1, C_2, C_3, \ldots, C_n\} \tag{12}$$

$$\text{Tardy Job: } U_i = \begin{cases} 1, \; if \; C_i > d_i \\ 0, \; otherwise \end{cases} \tag{13}$$

An optimality criterion states the objective of the scheduling problem and is defined with one or more performance

measures such as makespan, mean tardiness, total flow time, etc. Besides these mainly used performance measures, there are other performance measures in the analyzed literature such as *Work-in-Process, Cycle Time, Machine Downtime, Cost, Utilization, Due Date Deviation*, and *Throughput*. The list of all optimality criteria is given above in Table 2.

Classification and quantitative analysis of the literature

As a result of an exhaustive literature review, a total of 80 papers, including 51 journal articles and 49 conference papers, that handle the scheduling problems with the RL algorithms were found, and the papers are classified according to eight attributes. Table 1 presents the papers found in the related literature review.

The first analysis conducted on the papers is related to the publication date. The literature review has been conducted until December 2020. The first paper in this field was published in 1995. Therefore, the analysis includes the years between 1995 and 2020, as seen in Fig. 2. Considering this analysis, it is clear that studies in this field are in an increasing trend.

The publication years of the papers were analyzed based on the manufacturing environment, given in Fig. 3. Until 2000, the researchers addressed only the job shop scheduling problem. After 2001, they began to apply the RL algorithm to different manufacturing environments.

Figure 4 demonstrates the distribution of manufacturing environments studied in the papers. The majority of the papers handle the job shop scheduling problem (50 out of 80). The unrelated parallel machine scheduling problem (10 out of 80) is the second most studied problem, and it is followed by single machine scheduling problems (7 out of 80). It can be seen that identical parallel machines and uniform parallel machines scheduling problems were not discussed broadly.

Figure 5 summarizes the base RL algorithms used in the papers. It is clear that the Q-learning algorithm is the most used, and it is followed by Temporal Difference and Deep Q Network algorithms.

Another analysis is related to the objective type of problem. Most of the problems were studied as single objective (SO) problems, as illustrated in Fig. 6. Especially in the studies that handle the single machine and identical parallel machines, there is a lack of studies that consider multi-objective (MO) problems.

In addition to these, the type of agent used in the papers was analyzed, as illustrated in Fig. 7. In most of the papers, the RL method was applied with the single agent (58 out of 80).

The action selection mechanism is a critical factor that directly affects the trade-off between exploration and exploitation processes. In the related literature, due to the simplicity of the procedure, the $\epsilon - greedy$ method was selected frequently (60 out of 80), followed by the Boltzmann probability method (11 out of 80) and SA temperature procedure (6 out of 80) as seen in Fig. 8.

Especially in the problems with continuous state space or very large state-space, another vital issue in RL is the state definition strategy. As it is seen in Fig. 9, to deal with this problem, state aggregation and NNs methods are applied chiefly.

Finally, the system types studied in these papers were analyzed as depicted in Fig. 10. The RL methods frequently applied to problems in stochastic systems. In some papers, the RL algorithms were applied to deterministic problems to compare the effectiveness of the algorithm with benchmark problems.

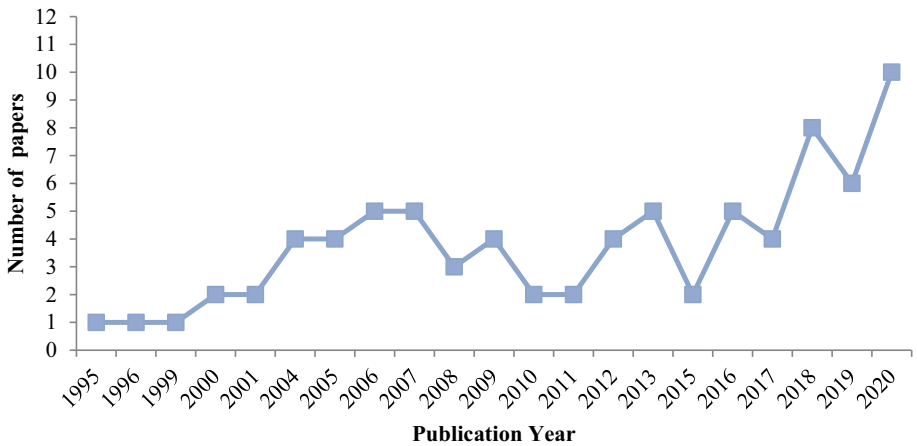


Fig. 2 The yearly number of papers

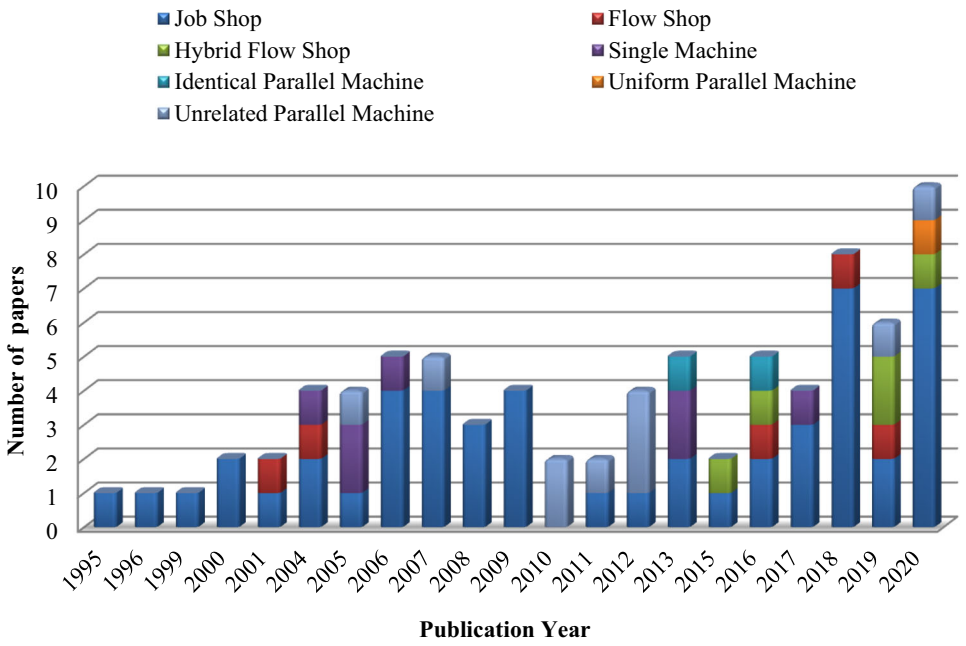


Fig. 3 The yearly number of papers according to the manufacturing environment

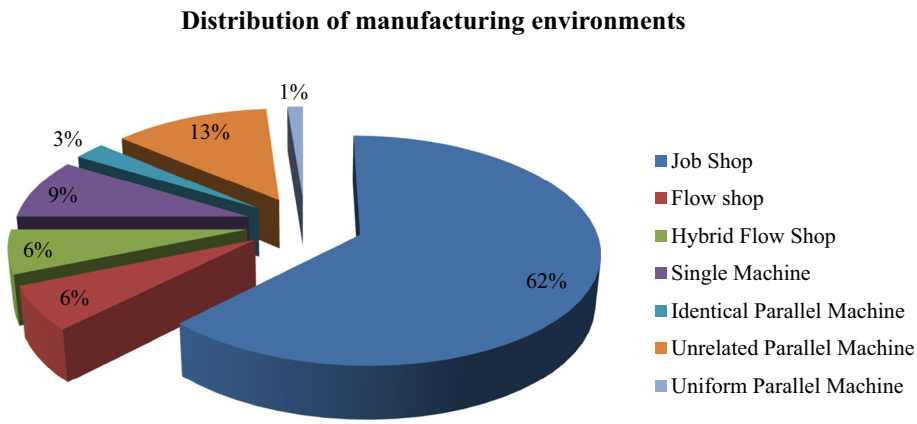


Fig. 4 Distribution of manufacturing environments in the papers

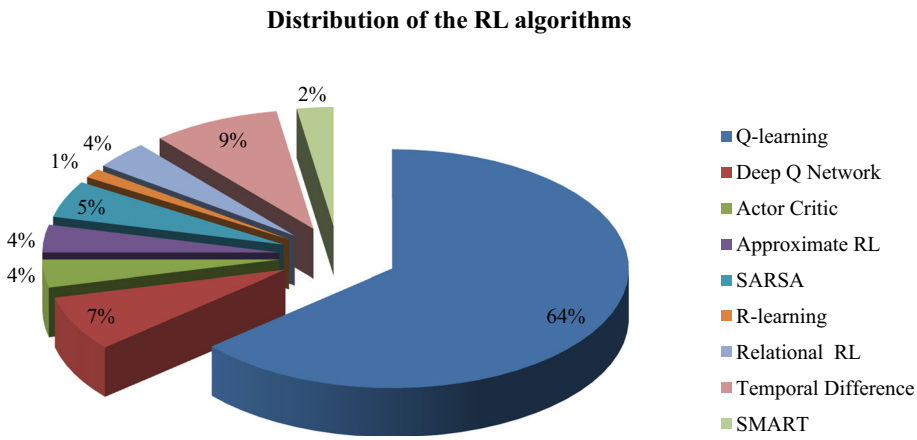


Fig. 5 Distribution of the RL algorithms

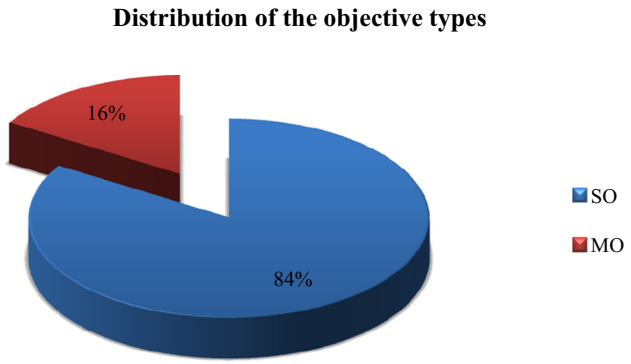


Fig. 6 Distribution of objective types in the papers

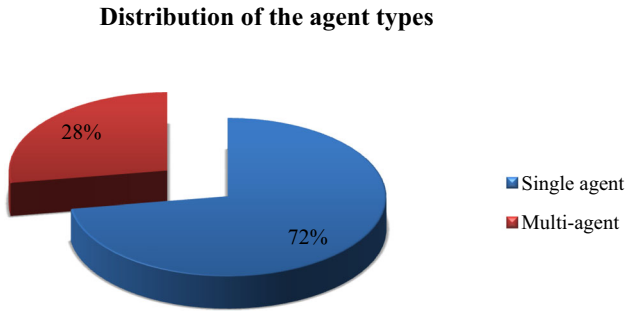
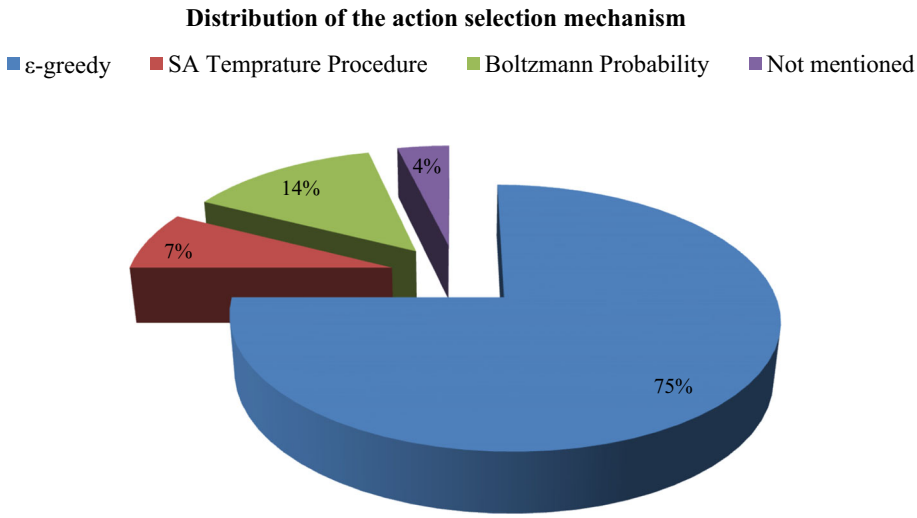


Fig. 7 Distribution of the agent types

Fig. 8 Distribution of the action selection mechanism



Overview of the literature

In this section, machine scheduling problems are analyzed according to manufacturing environments. The abbreviations used in the following sections are given in Table 3.

Single machine scheduling problems

In the related literature, we have found seven papers that handle the single machine environment. Wang and Usher (2004), Kong and Wu (2005) and Wang and Usher (2005) applied the Q-learning algorithm to the problem of $1//\frac{1}{n}\sum T_i$, $1//\frac{1}{n}\sum T_i, T_{max}, \sum U_i$, $1//L_{max}, \sum U_i, \frac{1}{n}\sum L_i$, respectively. The performances of the algorithm were compared with dispatching rules and the rule selection percentages of the agent were analyzed. The results of the studies show that the Q-learning algorithm learns to select the best dispatching rule which is previously known for each objective function. Wang and Usher (2004) also conducted an experiment aiming to find essential factors that affect the performance of the Q-learning algorithm. They simulated the system under four different system conditions that consider the due date

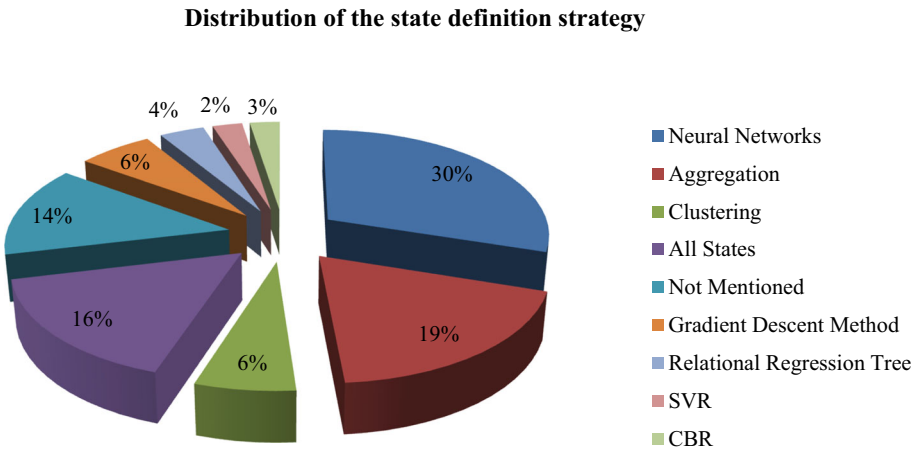


Fig. 9 State definition strategy

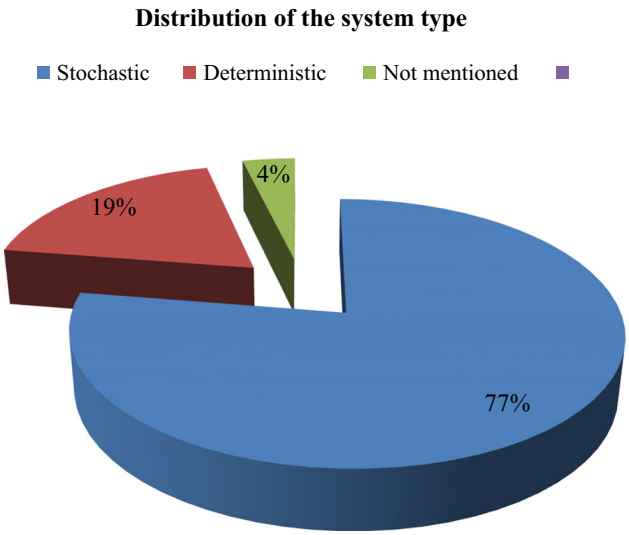


Fig. 10 System type of the problems

Table 3 The explanations of abbreviations used in this section

Abbreviation	Explanation	Abbreviation	Explanation
MOEO	Multi-objective Evolutionary Optimization	WCOVERT	Weighted Cost Over Time
GA	Genetic Algorithm	AIS	Artificial Immune System
WSPT	Weighted Shortest Processing Time	WMDD	Weighted Modified Due Date
BBC	Behavior Based Control	CONWIP	Constant Work in Process
MST	Minimum Slack Time	LPT	Longest Processing Time
EDD	Earliest Due Date	FIFO	First In First Out
ATC	Apparent Tardiness Cost	SPT	Shortest Processing Time

tightness and loading frequency of jobs to the system and tried to determine the best factor level combinations for these conditions. The Q-learning obtained the best result in heavy loading and loose due date condition. Another important finding is that the agent's learning performance improves when the policy table contains narrower ranges for the states and more ranges for the reward function. Idrees et al. (2006) applied the SMART algorithm to the problem of $1//\frac{1}{n}\sum T_i + TC$. They aimed to minimize the mean tardiness and the cost of hiring extra workers. The agent decides on the number of workers and dispatching rules together in every state. This study is the first study that addresses the single machine scheduling problem as a multi-objective problem, and the study showed that the RL algorithm obtained better results than dispatching rules in the multi-objective case. Atighehchian and Sepehri (2013) developed a new function-based approach with Simulated Annealing and Neural Network (SANN) and applied the new approach to the problem of $1/S_{jk}/\sum w_i T_i$. In this approach, contrary to previous studies, the job that will be processed next is selected rather than the dispatching rule at each decision point. The results of the algorithm were compared with dispatching rules and the Q-learning algorithm. It is concluded that the new approach outperforms both the Q-learning algorithm and the dispatching rules. Xanthopoulos et al. (2013) studied the problem of $1/S_{jk}/\frac{1}{n}\sum T_i + \frac{1}{n}\sum E_i$ as a multi-objective optimization problem. They aimed to develop two adaptive schedulers based on RL, and the other is based on Fuzzy Logic. They compared the results of the schedulers with 15 dispatching rules. The results revealed that RL schedulers and Fuzzy schedulers outperform all dispatching rules when mean tardiness and mean earliness objectives are handled simultaneously. Xiao et al.

(2017) studied the problem of $1//CT$ in a batch processing system using the Q-learning algorithm and compared the method with the Relative Value Iteration (RVI) method. This study showed that the Q-learning algorithm obtained only 0.141–1.118 % above the optimal value obtained by RVI; besides, it got results 100 times faster than RVI.

Generally, in the related literature, single machine scheduling problems have been handled without constraints, or only sequence-dependent setup time has been considered in a few of the papers. The effectiveness of the algorithm has been compared with dispatching rules. These studies showed that the Q-learning algorithm tends to select the best dispatching rule, which is previously known for each objective function. However, it was concluded that it did not outperform the dispatching rules. Also, the application of the dispatching rule is effortless in comparison with the RL algorithms. Therefore, RL does not provide an advantage for this type of scheduling problem.

On the other hand, although there are a limited number of studies that examine the problem as a multi-objective problem, these studies showed that the RL algorithm gets better results than dispatching rules. For multi-objective problems, there is no single dispatching rule that yields optimal results. Besides, dispatching rules are myopic. However, in these cases, the RL algorithm combines dispatching rules properly and benefits from the combination of dispatching rules in balancing the objectives. Since the real-life scheduling problems usually include multiple objectives, the application of the RL algorithms to multi-objective problems should be analyzed deeply.

Parallel machine scheduling problems

In the related literature, two studies handle identical parallel machine problems. Yuan et al. (2013) and Yuan et al. (2016) applied the Q-learning algorithm to the problem of $P_m//L_{max}, \sum U_i$ and $P_m/bkdwn/L_{max}, \sum U_i$, respectively, for the selection of dispatching rules SPT, EDD, and FIFO. The studies show that the learning agent tends to choose the proper dispatching rule among SPT, EDD, and FIFO, which gives better results for each objective function.

In the context of uniform parallel machine environment, Guo et al. (2020) handled the problem of $Q_m//T_{max}+\sum U_i+\frac{1}{n}\sum W_i$ and proposed a novel multi-stage Q-learning method to balance the different objectives. They compared the results of the proposed method with dispatching rules and the single-objective Q-learning method. The study results showed that the proposed method chooses the proper dispatching rules consistent with the objective more frequently and that the proposed method is more successful in balancing the different objectives.

The application of RL to the unrelated parallel machine scheduling problem was examined from different perspectives by researchers. Csáji and Monostori (2005) applied variants of the Q-learning algorithm to the problem of $R_m/bkdwnprec/C_{max}$ in various scheduling environments (deterministic-stochastic, static-dynamic). There is no benchmark method; instead, the performance was evaluated by the adaptivity of the algorithm to unexpected events such as new job arrivals, machine breakdowns, etc. As a result, it is shown that this system easily adapts to unexpected events using previously obtained information. Zhang et al. (2007, 2012) handled the problem of $R_m/S_{jk}/\frac{1}{n}\sum w_i T_i$ and $R_m//\sum w_i T_i$, respectively, and compared the results of the algorithm with the results of selected heuristic rules. The results of the study indicated that the Q-learning algorithm outperforms heuristic rules in all test problems remarkably. They indicated that a heuristic is myopic; however, Q-learning can combine the advantages of different heuristics. Iwamura et al. (2010) studied the problem of $R_m//\sum F_i + TC$ as a multi-objective problem and applied the Q-learning algorithm. They only analyzed the average improvement ratio of the objective function values, and they showed proposed methods improve the objective function values. Palombarini and Martínez (2010, 2012a, b) applied the Relational RL algorithm to the problem of $R_m//\sum T_i$, $R_m/S_{jk}bkdwn/\sum T_i$, and $R_m/S_{jk}bkdwn/\sum T_i$ for repairing schedules, respectively. Unlike other studies, the RL algorithm was applied to choose predetermined repair operators for repairing schedules instead of scheduling all jobs. They analyzed the average improvement as a performance measure, and it was indicated that the RL algorithm could learn how to repair schedules against unexpected events. Palombarini and Martínez (2012b), differently from these studies, only analyzed the learning ability of the agent under different objectives and implied that the agent achieves the goal after 450 episodes, and only between 5 and 8 repair steps were sufficient to repair the schedule. Zhang et al. (2011) studied the problem of $R_m/S_{jk}prec/TC$ in a semiconductor test scheduling problem which includes re-entrance of jobs. They applied the SARSA algorithm to industrial datasets and compared the results with individual dispatching rules and the Industrial Method (IM) that the company had used for scheduling up to that time. The study results demonstrated that the SARSA algorithm outperforms all the investigated methods in each test problem. Ábrahám et al. (2019) studied the problem of $R_m/prec/C_{max}$ and they defined the states as the order of the tasks and calculated Q-values for task sequencing. They compared their method with benchmark problems and showed that the method could find the optimal solutions. The benchmark problems were solved by CPLEX. The results showed that RL obtained the same or better results much faster than CPLEX. Zhou et al. (2020) handled the problem of $R_m//C_{max}$ in smart manufacturing and applied a deep RL-based method to two case studies to analyze the usefulness of the proposed method.

The results of this section reveal that in comparison to identical and uniform parallel machine scheduling problems, RL mostly applied to the unrelated parallel machine scheduling problem. The study in the uniform parallel machine environments demonstrates that Q-learning is more successful than dispatching rules in balancing objective functions simultaneously. In the unrelated parallel machine environments, the performance of the RL was examined in various aspects. In three of these studies, different from others, RL algorithms were applied to repair the production schedules instead of constructing the schedule. As a result, it is demonstrated that the RL easily adapts to unexpected events. Studies in which the RL algorithm was compared with dispatching rules in unrelated parallel machine environments present that the RL compounds the advantages of dispatching rules and learns how to combine them appropriately considering the objective of the problem; in this way, RL outperforms the dispatching rules.

There is only one study in unrelated parallel machine environments that handles the problem as a multi-objective problem. However, in this study, the algorithm results have not been compared with other dispatching rules or other methods. The studies in the single machine and uniform parallel machine environments demonstrate the effectiveness of the RL algorithm in multi-objective cases. This case also can be investigated for unrelated parallel machine scheduling problems. The use of metaheuristics as a solution approach occupies a significant place in machine scheduling problems. Hence, the performance of the RL algorithms can be compared with metaheuristics, and the superior and weak properties of the algorithm compared to metaheuristics can be revealed.

Flow shop scheduling problems

In the related literature, five studies were found in which the RL algorithms were applied to flow shop scheduling problems. Paternina-Arboleda and Das (2001) applied the SMART algorithm to the problem of $F/bkdwn/WIP$ to develop intelligent dynamic control policies. They analyzed a serial line production system with four machines and a single product type. The performance of the algorithm was compared with Kanban, CONWIP, and BBC policy based on average WIP. The results of the study showed that RL and BBC policy yielded better results than other policies. Hong and Prabhu (2004) studied the problem of $F/S_{jk}/DD + SC$ as a multi-objective scheduling problem and applied the Q-learning algorithm for batch sequencing and sizing in a deterministic manner. They aimed to develop an approach suitable for Just-in-Time (JIT) production systems balancing the due-date deviation cost and setup cost. The performance of the approach was compared with four dispatching rules (EDD, MST, SPT, and LPT). The study showed that RL outperformed dispatching rules, especially for jobs with a tight due date in complex manufacturing systems. Arviv et al. (2016), Reyna et al. (2018, 2019a) applied the Q-learning algorithm to the problem of $F//C_{max}$. Unlike classical flow shop problems, Arviv et al. (2016) considered scheduling the robots to minimize makespan for different collaboration levels. Reyna et al. (2018) analyzed the effects of parameters on the learning process. The algorithm solved benchmark problems with the reported optimal results to validate the quality, and it was shown that the average relative error of the Q-learning algorithm is less than 1.00 %. Reyna et al. (2019a) incorporated the Nawaz-Enscore-Ham (NEH) heuristic into the Q-learning algorithm and the results are compared with various metaheuristics from the literature. Experimental results show that Q-learning outperforms most of the metaheuristics.

Applications of RL to hybrid flow workshop scheduling problems have been studied in recent years. Qu et al. (2015) studied the problem of $HF/S_{jk}block/WIP + TC$ for proactive scheduling. They applied the approximate Q-learning algorithm and compared the performance of the algorithm with the FIFO method and heuristic greedy method. The study results verified that approximate Q-learning outperforms other methods and converges to optimal schedule faster. Qu et al. (2016a) studied the problem of $HF/S_{jk}block/TC + \sum U_i$ with a multi-skill workforce and multiple machine types. Different from the previous study, they combined machine scheduling with workforce scheduling. The Q-learning algorithm was applied to update the production schedules dynamically and enable real-time collaboration between the machines and the workforce. However, they did not compare the results of the algorithm with any other method. They only analyzed the average reward for different workforce skill sets. Han et al. (2019) and Fang Guo et al. (2020) applied the Q-learning algorithm to the problem of $HF//C_{max}$ and handled a small size case study to verify the algorithm. They compared the results of the Q-learning algorithm with the results of the GA. The results of the studies demonstrated that the Q-learning algorithm got better results than GA in these examples. Reyna et al. (2019b) studied the problem of $HF/S_{jk}prec/C_{max}$ in a deterministic environment. This study is essential due to considering the hybrid flow shop scheduling problem with additional constraints. They showed that the Q-learning algorithm obtains good results in complex scheduling problems.

To sum up, RL algorithms were mainly applied to minimize makespan in flow shop scheduling problems. To minimize the makespan, one of the studies tried to schedule robot transfers. Two others tried to find a suitable permutation of jobs. However, in these studies, state definitions and actions were not defined clearly. Besides, the state space of the problem was too large, and it was too hard for the agent to visit each state. One study handles the problem as

a multi-objective problem, and it concludes that RL outperforms dispatching rules.

In the related literature, the minimization of makespan is the most studied optimality criteria in hybrid flow shop problems. However, these studies handled a single case or small-scale problems. Further analysis is needed to investigate the effectiveness of the RL algorithm in large-scale problems. Another critical deficiency is that although real-life scheduling problems include dynamic properties, the studies in the related literature generally handled the problem in a static environment. The studies in the parallel machine scheduling problems demonstrated that the RL algorithm adapts to sudden changes. Therefore, this case can be a promising area for hybrid flow shop problems. For the multi-objective case, only one of the two studies compared the performance of the algorithm with other methods and remarks that RL outperforms other methods. The other study only analyzed the average reward for different conditions. This issue also should be handled to compare the effectiveness of the RL algorithms with other methods.

Job shop scheduling problems

Job shop scheduling problems have been frequently studied in the RL field. We have found 50 papers dealing with job shop scheduling problems and analyzed them based on the optimality criterion in the following sections.

Job shop scheduling problems with makespan criterion

Job shop scheduling problem with makespan optimality criteria $J//C_{max}$ is addressed by Gabel and Riedmiller (2007a, b, 2008), Yingzi et al. (2009), Aissani et al. (2009), Gabel and Riedmiller (2007a, 2011), Reyna et al. (2015), Lin et al. (2019, 2020), Ren et al. (2020), Jiménez et al. (2020) applied the Q-learning algorithm to job shop benchmark problems. Gabel and Riedmiller (2007b) analyzed the performance of the algorithm on large-scale benchmark problems. Gabel and Riedmiller (2008) developed an approach in which each machine was assigned to an agent, and each agent decides which job to process next. These three problems were modeled as multi-agent cases and utilized a NN for value function approximation. Aissani et al. (2009) applied the SARSA algorithm utilizing Boltzmann probability in the action selection process. The study results were compared with GA, and it was shown that RL outperformed the GA. Gabel and Riedmiller (2011) applied the policy gradient RL algorithm and handled the problems as sequential decision problems utilized by independent learning agents. Reyna et al. (2015) analyzed the effectiveness of the Q-learning algorithm by solving benchmark instances from the literature. They solved 15 instances that are the combination of 5 machines and 10, 15, and 20 jobs. The obtained result showed that the algorithm finds optimum value for 11 out of 15 problems and approximates very close the rest of the instances. Lin et al. (2019) applied the deep Q-Network (DQN) in a smart manufacturing factory. Liu et al. (2020) applied the actor-critic algorithm to problems in static and dynamic job shop environments. Yingzi et al. (2009) applied the Q-learning algorithm for the dynamic scheduler. All of them compared the algorithm results with dispatching rules and demonstrated that the RL algorithms get better results in each case compared to a single dispatching rule strategy. Ren et al. (2020) applied the actor-critic algorithm and compared the effectiveness of the method with benchmark problems. Jiménez et al. (2020) developed a job shop scheduling tool using the MA-RL approach. The tool allows users to set the RL method parameters and make adjustments in the schedule according to the conditions of the shop floor.

Park et al. (2020) addressed the problem with sequence-dependent setup times ($J/S_{jk}/C_{max}$). They applied deep Q-network and analyzed the efficiency of the proposed algorithm comparing with GA and dispatching rules. Numerical experiments showed that the proposed algorithm is superior to GA and dispatching rules, and besides, this proposed method ensures shorter computing time, approximately ten times from GA.

Zhang and Dietterich (1995), Zhang and Dietterich (1996), Monostori et al. (2004), and Aissani et al. (2012) handled the problem considering precedence constraints ($J/prec/C_{max}$). Zhang and Dietterich (1995) applied a temporal difference algorithm to the problem to obtain a feasible schedule without any violation of constraint and compared the results with the iterative repair method. Zhang and Dietterich (1996) also studied the same problem, but they compared the proposed algorithm with the NN method. Monostori et al. (2004) applied a temporal difference algorithm utilizing Boltzmann probability in the action selection process. They investigated how much computation is required to construct a schedule in case of new job arrivals. The flexibility of the algorithm was compared with the Branch and Bound algorithm. Aissani et al. (2012) applied the SARSA algorithm and compared the algorithm with a mixed-integer linear program (MILP) and GA. The study results showed that the RL algorithm falls behind the MILP by an average of 3% deviations, but it reaches these results in less computational time. Besides, it is demonstrated that the RL algorithm is faster than GA in reacting to disturbances.

Csáji et al. (2006) and Zhao et al. (2019) addressed the problem with machine breakdowns ($J/bkdwn/C_{max}$). Csáji et al. (2006) aimed to investigate the reactivity of the algorithm to machine and job disturbances. The study results showed that the proposed algorithm used previous information in unexpected events, thus accelerating the scheduling process. On the other hand, Zhao et al. (2019) compared

the results of the method with dispatching rules. The study results showed that the Q-learning algorithm learns which dispatching rule is the right choice in specific states and thus gets better results than single dispatching rule selections in the case of machine failures. Aissani and Trentesaux (2008) handled the job shop scheduling problem as a multi-objective problem, and they aimed to minimize both makespan and machine downtime ($J/bkdwn/C_{max} + MDT$) and applied the SARSA algorithm to the benchmark problems in the literature. They determined the makespan as effectiveness criteria and machine idle time as efficiency criteria. The study results indicated that the proposed algorithm is effective and efficient, especially in scheduling problems that include machine disturbances.

In the relevant literature, most of the papers (11 out of 19) handle the problem without constraints or job characteristics. These studies generally compared the RL with dispatching rules in dynamic environments, and studies showed that RL outperformed dispatching rules. In static environments, the problem instances with only up to 15 jobs were solved. Therefore, large-scale problem instances also should be solved to analyze the performance of the algorithm in static environments. A significant part of the studies in this section analyzed the reactiveness of the algorithm to dynamic events such as new job arrival, machine breakdowns, and job cancellation. Studies demonstrated that the RL algorithm easily compensates for the disturbances, uses previous information to reschedule, accelerates the scheduling process, and reacts more quickly than other methods. These findings are significant in that they demonstrated the RL algorithms are practical for real-life scheduling problems. Also, the sequence-dependent setup times are frequently encountered in many real-life scheduling problems; this constraint is handled only in one paper. It can be noted that this problem can be studied with different job characteristics and constraints such as blockage, sequence-dependent setup times, which are frequently encountered in real-life problems.

Job shop scheduling problems with tardiness related criterion

Job shop scheduling problems with mean tardiness criterion ($J//\frac{1}{n}\sum T_i$) are studied by Aydin and Öztemel (2000), Wei and Zhao (2004), Yang and Yan (2007), Yang and Yan (2009), Wang and Yan (2016). Aydin and Öztemel (2000) developed a new version of the Q-learning algorithm called Q-III that uses past experiences of the agent with a strong predictor. The agent decides which dispatching rule will be used. The behaviour of the agent was compared with individual dispatching rules in different due date tightness conditions. Study results showed that the agent obtained better results than dispatching rules except in one case. Wei and Zhao (2004) applied the Q-learning algorithm for composite rule selection, including job selection and machine selection. They indicated that the agent behaves than other rules by giving flexibility of switching rules in real-time. In the studies of Yang and Yan (2007) and Yang and Yan (2009), dispatching rules selection was made by a version of Q-learning named B-Q learning. Experiments showed that the agent gets better results than the single dispatching rule in each different due date urgency conditions. Wang and Yan (2016) applied the Q-learning algorithm and incorporated a fuzzy reward into the Q-value updating process. They aimed to handle the large state space problem of the Q-learning and compared the results with different RL approaches.

Gabel and Riedmiller (2006a), Wang and Usher (2007) addressed the problem of $J/S_{jk}/\frac{1}{n}\sum T_i$. Gabel and Riedmiller (2006a) developed a method that tracks the learning process and decides whether it is better to stop the learning process and utilized a NN for value function approximation. The proposed method prevents overfitting and gets better results than classical Q-learning. Wang and Usher (2007) aimed to investigate the impacts of the Q-learning algorithm factors such as state determination criteria, number of reward ranges, reward magnitude, and range interval size. They tested these factors under different system conditions considering the time between job arrivals and due date tightness and recommend factor levels for each system condition.

Lee et al. (2020) handled the problem of $J//\sum wT_i$ using deep RL and evaluated the performance of the proposed method with dispatching rules. The results of the study showed that RL outperforms the dispatching rules and also ensures better results in dynamic environments. They also indicated that RL has better performance in the situation that exact process times not known.

Wang and Yan (2013a) studied the problem of $J/bkdwnblock/\frac{1}{n}\sum T_i$. They applied the Q-learning algorithm to select a scheduling strategy based on the multi-agent system in a dynamic environment. They constituted agent groups with different missions that interact with each other to find the optimal policy. Study results showed that the proposed method learns to select proper scheduling policy in changing situations through interactions and outperforms the B-Q learning proposed by Yang and Yan (2009).

Riedmiller and Riedmiller (1999) and Luo (2020) studied the problem of $J//\sum T_i$ applying the Q-learning and DQN methods, respectively. They compared the performance of the agent with dispatching rules and analyzed the generalization ability of the agent. Study results indicated that the RL algorithm outperforms dispatching rules, and the learning agent has the capability to generalize obtained information to an unknown case. Luo (2020) also measured the performance of the DQN method against stand Q-learning. The experiments demonstrated that the DQN agent outperforms the stand Q-learning agent in most of the test instances. Gabel and Riedmiller (2006b) handled the prob-

lem of $J/bkdwnprec/\sum T_i$ in multi-agent domains and embedded the algorithm to the Case Based Reasoning framework. CBR framework utilized to tackle the high dimension state-space problem and the complexity of the large-scale problems. The study results demonstrated that the proposed method gets better results than dispatching rules.

Miyashita (2000) and Wang (2018) studied the job shop scheduling problem as a multi-objective problem. Miyashita (2000) applied a temporal difference algorithm to the problem of $J//\sum w_i T_i + WIP$. The case-based function approximation was utilized to handle high dimensional state space problem. The study results reported that the proposed method improves the schedule quality quickly. Wang (2018) handled the problem of $J//\sum E_i + \sum T_i$ and developed a new method that consists of multiple agents and a new state clustering method. The dynamic greedy strategy was utilized to improve the solution speed. The results of the proposed method were compared with other RL algorithms. The experimental results showed that the proposed method improved all objective values obtained by other algorithms.

In the job shop scheduling problems with the tardiness-related criterion, most of the papers aimed to minimize mean tardiness and did not consider different constraints and job characteristics. The studies handled the problem in a dynamic environment and compared the performance of the algorithms with dispatching rules. The results show that the RL algorithms outperform dispatching rules since the learning agent capability to switch dispatching rules based on the changing state of the systems. Besides, the studies indicated that the RL algorithm quickly reacts to unexpected events and generalizes its previous information to unknown situations. These features demonstrate that the RL algorithm a promising solution method for real-life scheduling problems that include dynamic events and require fast decision making.

On the other hand, it is challenging for the RL algorithms to have high dimensional state space for these problems. Many state features should be considered to handle job shop scheduling problems appropriately. Most of the time, it is hard to reduce state space aggregating state features. Therefore, the researchers attempted to utilize a neural network or a CBR framework. These studies showed that proposed methods improved the speed of convergence of the algorithm.

Considering real-life scheduling problems, they include more factors such as sequence-dependent setup times, machine breakdowns, blockages that affect the tardiness of the jobs, and multiple objectives conflicting with each other. These characteristics and constraints have been rarely handled in the related literature. It can be beneficial to analyze these factors to make problems more realistic. In addition, sometimes job tardiness importance varies due to the arrangement with customers. In this situation, the minimization of the weighted tardiness criterion becomes an important issue. Therefore, this problem can be researched to implement the RL algorithm in real-life scheduling problems.

Job shop scheduling problems with other optimality criteria

Liu et al. (2001) and Ramírez-Hernández and Fernandez (2005) addressed the job shop problem with reentrant lines considering blockage constraint to maximize throughput and to minimize total cost, respectively. Ramírez-Hernández and Fernandez (2009) applied a temporal difference algorithm to the problem of $J/blockS_{jk}bkdwn/WIP$ and compared the results with dispatching rules. They suggest using dispatching rules since they obtain better or similar performance than RL and more straightforward to implement. Kim and Shin (2017) also handled the reentrant lines and applied the actor-critic algorithm to the problem of $J/S_{jk}bkdwn/WIP, CT$. They analyzed the problem under two objectives separately. The RL algorithm achieved a better performance than dispatching rules only in this study among the studies mentioned above.

Monostori and Csáji (2006) studied the problem of $J/pmtnS_{jk}/\sum L_i$, and utilized the support vector machine method for value function approximation. Csáji and Monostori (2008) also studied the same problem. Different from the previous study, they aimed to minimize the number of tardy jobs.

Wang and Yan (2013b) addressed the problem of $J//\frac{1}{n}\sum E_i$ including reprocess of the jobs Qu et al. (2016b) applied an approximate RL algorithm to the problem of $J/block/WIP+\sum U_i$. Both of them compared the results of the algorithm with dispatching rules. Computational results indicated that the RL algorithm outperforms dispatching rules.

Shiue et al. (2018) studied the problem of $J//CT+\sum U_i+$ TPas a multi-objective problem and applied the Q-learning algorithm for real-time scheduling in a smart factory. They emphasized that the RL algorithm is a suitable approach for the smart factory due to autonomous dynamic decision-making properties to changing shop-floor conditions. Wang et al. (2020) studied the problem of $J//\sum w_i E_i + TC$ and proposed a dual Q-learning algorithm to balance the localized targets and global targets and compared the results of the proposed method with a single Q-learning. Findings demonstrated that the proposed method more adaptive than single Q-learning in the situation with a high job arrival rate and an increasing number of products.

Zhang et al. (2017, 2018) studied the problem of $J//CT$ problem applying the Q-learning algorithm. In the second study, they also included the job batching process in the problem. Study results showed that the RL algorithm obtained a smaller cycle time value than dispatching rules. Bouazza et al. (2017) addressed the problem of $/S_{jk}Prec/\sum w_i W_i$, and analyzed the distribution of machine and job selection

rules of the agent. The RL algorithm improved the objective value between 19 and 46% obtained by dispatching rules. Thomas et al. (2018) developed a method based on the Q-learning algorithm, named MINERVA, and solved the problem of $J//TP$ to detect bottleneck resources. They indicated that the proposed method is able to identify bottlenecks with high accuracy. Stricker et al. (2018) applied the Q-Learning algorithm to the problem of $J/bkdwnblock/U$ for adaptive order dispatching and compared the results with the FIFO rule. Experimental results demonstrated that the RL algorithm improves the utilization more than the dispatching rule in a shorter time. They also indicated that the RL algorithm is more adaptive to changing conditions. Waschneck et al. (2018a, b) addressed the job shop problem with sequence-dependent setup times and applied the DQN algorithm aiming to maximize machine utilization ($J/S_{jk}/U$) and minimize cycle time ($J/S_{jk}/CT$) respectively. They indicated that the proposed method can be a promising solution method since the algorithm develops a good schedule without human intervention. Besides, study results yielded that the RL algorithm obtained the same results which are reported in the literature without prior expert knowledge.

In this section, the studies that handled different optimality criteria out of the makespan and tardiness-related criteria are analyzed. It is seen that the minimization of WIP and Cycle Time were the frequently studied criterion. Besides, due date-related criteria such as earliness and lateness were also studied. The performance of the algorithm in dynamic environments is generally compared with dispatching rules. Most of the papers demonstrated that the RL algorithm outperforms dispatching rules. Unlike adopting single dispatching rules, the RL algorithm dynamically decides which dispatching will be used considering the changing conditions of the production system. The performance of the algorithm in the static environments was analyzed on small-scale problems. Study results showed that the RL algorithms obtained promising results on benchmark problems; however, it needs to be extended to large-scale problems. Some of the studies applied the RL algorithm to job shops with reentrant lines. Only one of the papers reported that the RL algorithm outperforms heuristic dispatching rules. Other studies said that the algorithm needs to be improved to achieve better results than dispatching rules. Another issue that can be noted is that no study handled the flow time-related criteria. This criterion may be studied for future research.

Conclusions

This study aims to conduct a comprehensive literature review concerning the applications of RL to machine scheduling problems. The scope of the study includes only machine scheduling problems. Other scheduling problems such as maintenance scheduling, carrier scheduling, gantry scheduling, economic- lot scheduling were excluded.

Initially, Scopus (https://www.scopus.com) and Web of Science (webofknowledge.com/WOS) databases were searched exhaustively, and then, 80 papers were found and included in this paper. The papers are classified into eight categories for further analysis. Afterwards, the papers were analyzed based on the machine environment in detail. From the detailed analysis of the related papers, important findings can be summarized as:

- The efficiency of the algorithm in the single machine, parallel machine and job shop environments was mostly compared with dispatching rules.
- The studies that handle the single machine and identical parallel machine problems have indicated that dispatching rules have better performance than the RL algorithm in the single-objective problems. In addition, dispatching rules provide superiority to the RL algorithms in terms of ease of implementation.
- The findings of the multi-objective problems in each machine environment have documented that the RL algorithms outperform dispatching rules since the RL algorithms learn to combine proper dispatching rules to achieve objectives simultaneously. Also, the RL algorithms learn to switch dispatching rules based on the changing state of the system.
- The minimization of the makespan is the most studied optimality criterion in the flow shop and job shop scheduling problems.
- The studies that handle the flow shop and job shop scheduling problems generally have analyzed the efficiency of the algorithm on small-scale problems and compared the results with benchmark problems from the literature. In most of the papers, it has been reported that the RL algorithms achieved the reported best result.
- A significant part of the studies that handle the parallel machine and job shop scheduling problems have analyzed the adaptivity of the RL algorithms. The results have demonstrated that the RL algorithms quickly adapt to unexpected events using previously obtained information, so it accelerates the scheduling process. This feature makes the RL algorithm more applicable to real-life problems.
- Zhang et al. (2011) and Jiménez et al. (2020) have applied the RL algorithm to real industrial problems. The experimental results of the studies have indicated that the RL algorithm obtains better results in each case compared to a single dispatching rule strategy. Also, Shiue et al. (2018), Lin et al. (2019), Zhou et al. (2020) investigated the RL application to dynamic scheduling in smart manufacturing. The studies have emphasized that the RL algorithm is a suitable application for smart factories because of the autonomous dynamic decision-making feature.

The advantages and strengths of the RL which has been reported in the related literature can be identified as follows:

- The real manufacturing systems confront dynamic events such as machine breakdown, stochastic arrival of jobs, order cancellation. The RL learns this dynamic system by getting feedback from the environment and can generalize the acquired information to new unseen situations. These features make the approach adaptive, flexible, and extensible. Therefore, it can be a suitable method for real manufacturing systems (Riedmiller & Riedmiller, 1999; Monostori et al., 2004; Csáji & Monostori, 2005; Gabel & Riedmiller, 2007a; Wang & Usher, 2007; Zhang et al., 2007, 2011; Lee et al., 2020; Liu et al., 2020).
- Dispatching rules are useful and effortless in job sequencing problems. However, it needs expert knowledge to decide which dispatching rule will be applied in which state of the system. When the RL algorithms applied to select dispatching rules it provides autonomous decision making depending on the state of the system, reduce decision-making delay and make it more applicable in fully automated manufacturing systems. Besides, the RL algorithms can consider much more characteristic of the system, switch between different dispatching rules and combine them appropriately. These characteristics of the algorithm give superiority to the algorithm against the single dispatching rule selection policy especially in multi-objective problems (Riedmiller & Riedmiller, 1999; Aydin & Öztemel, 2000; Wei & Zhaom, 2004; Zhang et al., 2007; Qu et al., 2016a; Zhao et al., 2019; Liu et al., 2020).
- The RL algorithms learn well by simulations without the need for complete information, and quickly converge to near-optimal values. (Zhang et al. 2011, 2012; Lee et al. 2020)
- The metaheuristics such as Genetic Algorithm and Tabu Search obtains good results in static scheduling problems. However, they cannot adapt quickly to sudden changes in the system because they construct the whole schedule from scratch when the system changes. Unlike these algorithms, RL algorithms do not start scheduling from scratch, but instead, use previously obtained information. Therefore, this approach reacts to sudden changes easily and modifies the schedule in a shorter time (Paternina-Arboleda & Das, 2001; Csáji & Monostori, 2005; Aissani et al., 2009, 2012; Atighehchian & Sepehri, 2013).

Despite the advantages and strengths of the RL mentioned above, the application of the RL in practice has some challenging features as follows:

- The main difficulty of the RL in applying to real-world complex problems lies in how to handle large and continuous state spaces. The design of the state space and the accurate selection of state features are very crucial for the effectiveness and the scalability of the algorithm (Wang & Usher, 2004; Gabel & Riedmiller, 2006a, b; Atighehchian & Sepehri, 2013; Wang & Yan, 2016; Xiao et al., 2017; Wang, 2018).
- The scalability of the RL is directly related to the design of the problem. The representation of all states of the environment in the Q-table can be applied only to small scale problems and renders to problem unscalable. Besides, the design of the states specific to the number of jobs or machines in the problem makes it harder to scale to the larger problem. To handle this issue state aggregation or function approximation methods are utilized. The design of the problem and selection of the proper function approximator or state aggregation directly affects the independence of the size of the state-action space from the size of the problem instance (Khadilkar, 2018; Qu et al., 2020).
- There is no restriction or strict rules in the design of the reward system. The design of the reward system can be difficult due to a large number of options (Xiao et al., 2017; Liu et al., 2020).
- Especially in multi-objective problems, it is important to design the reward system considering the importance level of the objectives. The reward function should be formulated in a way that representing and balancing each objective properly (Dulac-Arnold et al., 2019).
- The acts of the agent should be trustworthy and reasonably well in the exploration procedure when the learning occurs on the real system. Since the consequences of the actions will be observed in the real system, the actions of the agent in the exploration procedure may lead to high cost and low performance for these systems (Dulac-Arnold et al., 2019; Ding & Dong, 2020).
- The decision making in scheduling sometimes can be very critical; therefore the explainability of the RL algorithm gains importance for the decision-makers. However, as with many Machine Learning algorithms, the RL algorithm has a deficiency of explainability. The reason for the actions of the agent cannot be clarified explicitly to decision-makers. Especially in the larger size of problems where the function approximator is used to estimate the value of the action, it gets complicated to explain the reason why one action is chosen over another action in a special state (Heuillet et al., 2021; Ding & Dong, 2020).
- The states should be visited in a sufficient number to obtain good learning performance (Wang and Usher, 2004; Xiao et al., 2017).
- The RL algorithms do not guarantee to find optimal results (Xiao et al., 2017).

- In the single machine scheduling problems with a single objective, the RL has poor performance than dispatching rules and is more complicated to implement.
- The parallel machine scheduling problems include both job sequencing and machine selecting decisions. The main challenge here is how to design the scheduling problems and how to apply the RL algorithms, taking these decisions into account. This characteristic of the problem type also cause a high-dimensional state space and this is another challenge that needs to be handled.
- The flow shop and job shop environment are constituted of multiple production stages which means a scheduling problem arises at every stage. In the flow shop scheduling problems, this issue can be handled by applying the same order of jobs or dispatching the jobs according to the FCFS rule, at the next stages. While this approach may cause the algorithm to get a worse result, the application of the algorithm to each stage also results in large state space. In job shop scheduling problems job types have different routes which mean some type of jobs may not be processed on the first stage. In this situation, a decision making problem arises at every stage making it harder to implement the algorithm causing the high dimensional state space problem.

Having regard to the findings of the literature and the advantages and challenges of the RL discussed above, some of the opportunities for future research can be summarized as:

- In the related literature, all of the studies that handle problems in a multi-objective manner have reported that the RL outperforms dispatching rules. Since the real-life scheduling problems usually include multiple objectives and the RL gets promising result in the multi-objective problem, the application of the RL algorithms to these problems should be analyzed deeply.
- The metaheuristics techniques occupy a significant place in machine scheduling problems and obtain good results, especially in large-scale static problems. Hence, the performance of the RL algorithms can be compared with metaheuristics, and the superior and weak properties of the algorithm compared to metaheuristics can be revealed.
- The studies which analyze the effectiveness of the algorithm on benchmark problems have solved small-scale static problems. Further analysis is needed to investigate the effectiveness of the RL algorithm in large-scale problems.
- The vast majority of the papers have ignored the constraints such as machine breakdown, order cancellation, sequence-dependent setup time, and precedence constraints. However, these constraints are frequently encountered in real-life scheduling problems. Hence, future researches may intensify on these issues, to make problems more realistic.
- The learning algorithm affects the performance of the RL considerably and there are very few studies that compare the performance of the different learning algorithms in scheduling problems. Therefore, it can contribute to literature to analyze the different learning algorithms by comparing each other.
- Uniform parallel machine scheduling problems can be encountered frequently in real-life situations where old and new machines exist simultaneously, but there are not enough studies that handle uniform parallel machine scheduling problems.
- Although hybrid flow shops are increasingly studied in the literature and frequently appear in the industry, there is not enough study considering it. The studies in the related literature generally have handled the problem in a static environment. The studies in the parallel machine scheduling problems have demonstrated that the RL algorithm adapts to sudden changes. It can be concluded that developing and applying RL algorithms to this problem considering dynamic events can be promising.
- Job shop scheduling problems generally have studied considering the makespan and tardiness related criterion. No study has handled the flow time-related criteria. This criterion may be studied for future research. In addition, sometimes job tardiness importance varies due to the arrangement with customers. In this situation, the minimization of the weighted tardiness criterion becomes an important issue. Therefore, this problem can be researched to implement the RL algorithm in real-life scheduling problems.

.

References

Ábrahám, G., Auer, P., Dósa, G., Dulai, T., & Werner-Stark, Ã. (2019). A reinforcement learning motivated algorithm for process optimization. *Periodica Polytechnica Civil Engineering, 63*(4), 961–970. https://doi.org/10.3311/PPci.14295

Aissani, N., Bekrar, A., Trentesaux, D., & Beldjilali, B. (2012). Dynamic scheduling for multi-site companies: A decisional approach based on reinforcement multi-agent learning. *Journal of Intelligent Manufacturing, 23*(6), 2513–2529. https://doi.org/10.1007/s10845-011-0580-y

Aissani, N., Trentesaux, D., & Beldjilali, B. (2009). Multi-agent reinforcement learning for adaptive scheduling: Application to multi-site company. In *IFAC proceedings volumes*, (Vol. 42, No. 4, pp. 1102–1107). https://doi.org/10.3182/20090603-3-RU-2001.0280.

Aissani, N., & Trentesaux, D. (2008). Efficient and effective reactive scheduling of manufacturing system using Sarsa-multi-objective agents. In *Proceedings of the 7th international conference MOSIM, Paris* (pp. 698–707).

Arviv, K., Stern, H., & Edan, Y. (2016). Collaborative reinforcement learning for a two-robot job transfer flow-shop scheduling problem. *International Journal of Production Research, 54*(4), 1196–1209. https://doi.org/10.1080/00207543.2015.1057297

Atighehchian, A., & Sepehri, M. M. (2013). An environment-driven, function-based approach to dynamic single-machine scheduling. *European Journal of Industrial Engineering, 7*(1), 100–118. https://doi.org/10.1504/EJIE.2013.051594

Aydin, M. E., & Öztemel, E. (2000). Dynamic job-shop scheduling using reinforcement learning agents. *Robotics and Autonomous Systems, 33*(2), 169–178. https://doi.org/10.1016/S0921-8890(00)00087-7

Barto, A. G., & Mahadevan, S. (2003). Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems, 13*(1), 41–77. https://doi.org/10.1023/A:1022140919877

Bouazza, W., Sallez, Y., & Beldjilali, B. (2017). A distributed approach solving partially flexible job-shop scheduling problem with a Q-learning effect. *IFAC-PapersOnLine, 50*(1), 15890–15895. https://doi.org/10.1016/j.ifacol.2017.08.2354

Cadavid, J. P. U., Lamouri, S., Grabot, B., Pellerin, R., & Fortin, A. (2020). Machine learning applied in production planning and control: a state-of-the-art in the era of industry 4.0. *Journal of Intelligent Manufacturing, 31*(6), 1531–1558. https://doi.org/10.1007/s10845-019-01531-7

Csáji, B. C., & Monostori, L. (2005). Stochastic approximate scheduling by neurodynamic learning. In *IFAC Proceedings Volumes*, (Vol. 38, No. 1, pp. 355–360). https://doi.org/10.3182/20050703-6-CZ-1902.01481

Csáji, B. C., & Monostori, L. (2008). Adaptive stochastic resource control: A machine learning approach. *Journal of Artificial Intelligence Research, 32*, 453–486. https://doi.org/10.1613/jair.2548

Csáji, B. C., Monostori, L., & Kádár, B. (2006). Reinforcement learning in a distributed market-based production control system. *Advanced Engineering Informatics*, 20(3), 279–288. https://doi.org/10.1016/j.aei.2006.01.001

Das, T. K., Gosavi, A., Mahadevan, S., & Marchalleck, N. (1999). Solving semi-Markov decision problems using average reward reinforcement learning. *Management Science, 45*(4), 560–574. https://doi.org/10.1287/mnsc.45.4.560

De Raedt, L. (2008). *Logical and relational learning*. New York: Springer. https://doi.org/10.1007/978-3-540-68856-3.

Ding, Z., & Dong, H. (2020). Challenges of reinforcement learning. *In Deep Reinforcement Learning* (pp. 249–272). Singapore: Springer. https://doi.org/10.1007/978-981-15-4095-0_7

Dulac-Arnold, G., Mankowitz, D., & Hester, T. (2019). Challenges of real-world reinforcement learning. (Online) https://arxiv.org/abs/1904.12901

Fuchigami, H. Y., & Rangel, S. (2018). A survey of case studies in production scheduling: Analysis and perspectives. *Journal of Computational Science*, 25, 425–436. https://doi.org/10.1016/j.jocs.2017.06.004

Fang, G., Li, Y., Liu, A., & Liu, Z. (2020). A reinforcement learning method to scheduling problem of steel production process.*Journal of Physics: Conference Series, 1486*(7), 072035. https://doi.org/10.1088/1742-6596/1486/7/072035

Gabel, T., & Riedmiller, M. (2006a). Reducing policy degradation in neuro-dynamic programming. In *ESANN 2006 Proceedings - European Symposium on Artificial Neural Networks* (pp. 653–658).

Gabel, T., & Riedmiller, M. (2006b). Multi-agent case-based reasoning for cooperative reinforcement learners. In Roth-Berghofer, T. R., Göker, M. H., & Güvenir, H. A. (Eds.), *Advances in case-based reasoning. ECCBR 2006* (4106 vol.). Berlin, Heidelberg: Springer. https://doi.org/10.1007/11805816_5

Gabel, T., & Riedmiller, M. (2007a). On a successful application of multi-agent reinforcement learning to operations research benchmarks. In *2007 IEEE international symposium on approximate dynamic programming and reinforcement learning* (pp. 68–75). https://doi.org/10.1109/ADPRL.2007.368171

Gabel, T., & Riedmiller, M. (2007b). Scaling adaptive agent-based reactive job-shop scheduling to large-scale problems. In *Proceedings of the 2007 IEEE symposium on computational Intelligence in scheduling, CI-Sched 2007* (pp. 259–266). https://doi.org/10.1109/SCIS.2007.367699

Gabel, T., & Riedmiller, M. (2008). Adaptive reactive job-shop scheduling with reinforcement learning agents. *International Journal of Information Technology and Intelligent Computing, 24*(4), 14–18

Gabel, T., & Riedmiller, M. (2011). Distributed policy search reinforcement learning for job-shop scheduling tasks. *International Journal of Production Research, 50*(1), 41–61. https://doi.org/10.1080/00207543.2011.571443

Gosavi, A. (2015). *Simulation-based optimization*. Berlin: Springer

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Kan, A. H. G. R. (1979). Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, 5, 287–326. https://doi.org/10.1016/S0167-5060(08)70356-X

Guo, L., Zhuang, Z., Huang, Z., & Qin, W. (2020). Optimization of dynamic multi-objective non-identical parallel machine scheduling with multi-stage reinforcement learning. In *2020 IEEE 16th international conference on automation science and engineering (CASE)* (pp. 1215–1219). https://doi.org/10.1109/CASE48305.2020.9216743

Han, W., Guo, F., & Su, X. (2019). A reinforcement learning method for a hybrid flow-shop scheduling problem. *Algorithms, 12*(11), https://doi.org/10.3390/a12110222

Heuillet, A., Couthouis, F., & Díaz-Rodríguez, N. (2021). Explainability in deep reinforcement learning. *Knowledge-Based Systems*, 214, 106685. https://doi.org/10.1016/j.knosys.2020.106685

Hong, J., & Prabhu, V. V. (2004). Distributed reinforcement learning control for batch sequencing and sizing in just-in-time manufacturing systems. *Applied Intelligence, 20*(1), 71–87. https://doi.org/10.1023/B:APIN.0000011143.95085.74

Idrees, H. D., Sinnokrot, M. O., & Al-Shihabi, S. (2006). A reinforcement learning algorithm to minimize the mean tardiness of a single machine with controlled capacity. In *Proceedings - Winter simulation conference* (pp. 1765–1769). https://doi.org/10.1109/WSC.2006.322953

Iwamura, K., Mayumi, N., Tanimizu, Y., & Sugimura, N. (2010). A study on real-time scheduling for holonic manufacturing systems - Determination of utility values based on multi-agent reinforcement learning. In *International conference on industrial applications of holonic and multi-agent systems* (pp. 135–144). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-03668-2_13

Jiménez, Y. M., Palacio, J. C., & Nowé, A. (2020). Multi-agent reinforcement learning tool for job shop scheduling problems. In *International conference on optimization and learning* (pp. 3–12). https://doi.org/10.1007/978-3-030-41913-4_1

Kaelbling, L., Littman, M. L., Moore, A. W., & Hall, S. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4, 237–285. https://doi.org/10.1613/jair.301

Khadilkar, H. (2018). A scalable reinforcement learning algorithm for scheduling railway lines. *IEEE Transactions on Intelligent Transportation Systems*, 20(2), 727–736. https://doi.org/10.1109/TITS.2018.2829165

Kim, G. H., & Lee, C. S. G. (1996). Genetic reinforcement learning for scheduling heterogeneous machines. In *Proceedings - IEEE International Conference on Robotics and Automation* (Vol. 3, pp. 2798–2803). https://doi.org/10.1109/ROBOT.1996.506586

Kim, N., & Shin, H. (2017). The application of actor-critic reinforcement learning for fab dispatching scheduling. In *2017 Winter simulation conference* (pp. 4570–4571). https://doi.org/10.1109/WSC.2017.8248209

Kong, L. F., & Wu, J. (2005). Dynamic single machine scheduling using Q-learning agent. In *2005 International conference on machine learning and cybernetics, ICMLC 2005* (pp. 3237–3241). https://doi.org/10.1109/ICMLC.2005.1527501

Lee, S., Cho, Y., & Lee, Y. H. (2020). Injection mold production sustainable scheduling using deep reinforcement learning. *Sustainability*, 12(20), 8718. https://doi.org/10.3390/su12208718

Lihu, A., & Holban, S. (2009). Top five most promising algorithms in scheduling. In *Proceedings – 2009 5th international symposium on applied computational intelligence and informatics, SACI 2009* (pp. 397–404). https://doi.org/10.1109/SACI.2009.5136281

Lin, C. C., Deng, D. J., Chih, Y. L., & Chiu, H. T. (2019). Smart manufacturing scheduling with edge computing using multiclass deep Q network. *IEEE Transactions on Industrial Informatics, 15*(7), 4276–4284. https://doi.org/10.1109/TII.2019.2908210

Liu, C. C., Jin, H. Y., Tian, Y., & Yu, H. B. (2001). Reinforcement learning approach to re-entrant manufacturing system scheduling. In *2001 International Conferences on Info-Tech and Info-Net: A Key to Better Life, ICII 2001 - Proceedings* (Vol. 3, pp. 280–285). https://doi.org/10.1109/ICII.2001.983070

Liu, C. L., Chang, C. C., & Tseng, C. J. (2020). Actor-critic deep reinforcement learning for solving job shop scheduling problems. *IEEE Access*, 8, 71752–71762. https://doi.org/10.1109/ACCESS.2020.2987820

Liu, W., & Wang, X. (2009). Dynamic decision model in evolutionary games based on reinforcement learning. *Systems Engineering - Theory & Practice, 29*(3), 28–33. https://doi.org/10.1016/S1874-8651(10)60008-7

Luo, S. (2020). Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning. *Applied Soft Computing, 91*, 106208. https://doi.org/10.1016/j.asoc.2020.106208

Miyashita, K. (2000). Learning scheduling control knowledge through reinforcements. *International Transactions in Operational Research, 7*(2), 125–138. https://doi.org/10.1016/S0969-6016(00)00014-9

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., & Hassabis, D., …. (2015). Human-level control through deep reinforcement learning. *Nature, 518*(7540), 529–533. https://doi.org/10.1038/nature14236

Monostori, L., & Csáji, B. C. (2006). Stochastic dynamic production control by neurodynamic programming. *CIRP Annals - Manufacturing Technology, 55*(1), 473–478. https://doi.org/10.1016/S0007-8506(07)60462-4

Monostori, L., Csáji, B. C., & Kádár, B. (2004). Adaptation and learning in distributed production control. *CIRP Annals - Manufacturing Technology, 53*(1), 349–352. https://doi.org/10.1016/S0007-8506(07)60714-8

Nahmias, S., & Olsen, T. L. (2015). *Production and operations analysis*. Long Grove: Waveland Press

Neto, T. R. F., & Godinho Filho, M. (2013). Literature review regarding Ant Colony Optimization applied to scheduling problems: Guidelines for implementation and directions for future research. *Engineering Applications of Artificial Intelligence, 26*(1), 150–161. https://doi.org/10.1016/j.engappai.2012.03.011

Palombarini, J., & Martínez, E. (2010). Learning to repair plans and schedules using a relational (deictic) representation. In *Computer aided chemical engineering* (Vol. 27, pp. 1377–1382). Elsevier. https://doi.org/10.1016/s1570-7946(09)70620-0

Palombarini, J., & Martínez, E. (2012a). SmartGantt – An interactive system for generating and updating rescheduling knowledge using relational abstractions. *Computers and Chemical Engineering, 47*, 202–216. https://doi.org/10.1016/j.compchemeng.2012.06.021

Palombarini, J., & Martínez, E. (2012b). SmartGantt – An intelligent system for real time rescheduling based on relational reinforcement learning. *Expert Systems With Applications, 39*(11), 10251–10268. https://doi.org/10.1016/j.eswa.2012.02.176

Parente, M., Figueira, G., Amorim, P., & Marques, A. (2020). Production scheduling in the context of Industry 4.0: review and trends. *International Journal of Production Research, 58*(17), 5401–5431. https://doi.org/10.1080/00207543.2020.1718794

Park, I., Huh, J., Kim, J., & Park, J. (2020). A reinforcement learning approach to robust scheduling of semiconductor manufacturing facilities. *IEEE Transactions on Automation Science and Engineering, 17*(3), 1420–1431. https://doi.org/10.1109/tase.2019.2956762

Paternina-Arboleda, C. D., & Das, T. K. (2001). Intelligent dynamic control policies for serial production lines. *IIE Transactions, 33*(1), 65–77. https://doi.org/10.1023/A:1007641824604

Qu, S., Chu, T., Wang, J., Leckie, J., & Jian, W. (2015). A centralized reinforcement learning approach for proactive scheduling in manufacturing. In *IEEE international conference on emerging technologies and factory automation, ETFA* (Vol. 2015-Octob, pp. 1–8). https://doi.org/10.1109/ETFA.2015.7301417

Qu, S., Wang, J., Govil, S., & Leckie, J. O. (2016a). Optimized adaptive scheduling of a manufacturing process system with multi-skill workforce and multiple machine types: An ontology-based, multi-agent reinforcement learning approach. *Procedia CIRP, 57*, 55–60. https://doi.org/10.1016/j.procir.2016.11.011

Qu, S., Jie, W., & Shivani, G. (2016b). Learning adaptive dispatching rules for a manufacturing process system by using reinforcement learning approach. In *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA* (Vol. 2016-Novem, pp. 1–8). https://doi.org/10.1109/etfa.2016.7733712

Qu, G., Wierman, A., & Li, N. (2020). Scalable reinforcement learning of localized policies for multi-agent networked systems. In *Learning for Dynamics and Control* (pp. 256–266).

Ramírez-Hernández, J. A., & Fernandez, E. (2005). A case study in scheduling reentrant manufacturing lines: Optimal and simulation-based approaches. In *Proceedings of the 44th IEEE conference on decision and control* (Vol. 2005, pp. 2158–2163). https://doi.org/10.1109/CDC.2005.1582481

Ramírez-Hernández, J. A., & Fernandez, E. (2009). A simulation-based approximate dynamic programming approach for the control of the intel Mini-Fab benchmark model. In *Proceedings - Winter simulation conference* (pp. 1634–1645). https://doi.org/10.1109/wsc.2009.5429179

Ren, J., Ye, C., & Yang, F. (2020). A novel solution to JSPs based on long short-term memory and policy gradient algorithm. *International Journal of Simulation Modelling, 19*, 157–168. https://doi.org/10.2507/ijsimm19-1-co4

Reyna, Y. C. F., Cáceres, A. P., Jiménez, Y. M., & Reyes, Y. T. (2019a). An improvement of reinforcement learning approach for permutation of flow-shop scheduling problems. In *RISTI - Revista Iberica de Sistemas e Tecnologias de Informacao*, (E18), pp. 257–270.

Reyna, Y. C. F., Jiménez, Y. M., Cabrera, A. V., & Sánchez, E. A. (2019b). Optimization of heavily constrained hybrid-flexible flowshop problems using a multi-agent reinforcement learning approach. *Investigacion Operacional*, 40(1), 100–111

Reyna, Y. C. F., Jiménez, Y. M., & Nowé, A. (2018). Q-learning algorithm performance for m-machine n-jobs flow shop scheduling to minimize makespan. *Investigación Operacional*, 38(3), 281–290

Reyna, Y. C. F., Jiménez, Y. M., Bermúdez Cabrera, J. M., & Méndez Hernández, B. M. (2015). A reinforcement learning approach for scheduling problems. *Investigacion Operacional*, 36(3), 225–231

Riedmiller, S., & Riedmiller, M. (1999). A neural reinforcement learning approach to learn local dispatching policies in production scheduling. In *IJCAI Iiternational joint conference on artificial intelligence* (Vol. 2, pp. 764–769).

Russel, S., & Norvig, P. (2010). Artificial intelligence: A modern approach. London: Pearson.

Schwartz, A. (1993). A reinforcement learning method for maximizing undiscounted rewards. In *Proceedings of the tenth international*

conference on machine learning (Vol. 298, pp. 298–305). https://doi.org/10.1016/b978-1-55860-307-3.50045-9

Shiue, Y., Lee, K., & Su, C. (2018). Real-time scheduling for a smart factory using a reinforcement learning approach. *Computers & Industrial Engineering, 125*(101), 604–614. https://doi.org/10.1016/j.cie.2018.03.039

Sigaud, O., & Buffet, O. (2013). *Markov Decision Processes in Artificial Intelligence: MDPs, beyond MDPs and applications*. New York: Wiley

Stricker, N., Kuhnle, A., Sturm, R., & Friess, S. (2018). Manufacturing technology reinforcement learning for adaptive order dispatching in the semiconductor industry. *CIRP Annals, 67*(1), 511–514. https://doi.org/10.1016/j.cirp.2018.04.041

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. Cambridge: MIT Press

Szepesvári, C. (2010). Algorithms for reinforcement learning. *Synthesis lectures on artificial intelligence and machine learning, 4*(1), 1–103. https://doi.org/10.2200/S00268ED1V01Y201005AIM009

Thomas, T. E., Koo, J., Chaterji, S., & Bagchi, S. (2018). Minerva: A reinforcement learning-based technique for optimal scheduling and bottleneck detection in distributed factory operations. In *2018 10th international conference on communication systems & networks (COMSNETS)* (pp. 129–136). https://doi.org/10.1109/COMSNETS.2018.8328189

Van Otterlo, M. (2009). *The logic of adaptive behavior: Knowledge representation and algorithms for adaptive sequential decision making under uncertainty in first-order and relational domains*. Ios Press

Vapnik, V. N. (2000). Methods of pattern recognition. In *The nature of statistical learning theory* (pp. 123–180). New York, NY: Springer

Wang, H. X., & Yan, H. S. (2013a). An adaptive scheduling system in knowledgeable manufacturing based on multi-agent. In *10th IEEE international conference on control and automation (ICCA)* (pp. 496–501). https://doi.org/10.1109/icca.2013.6564866

Wang, H. X., & Yan, H. S. (2013b). An adaptive assembly scheduling approach in knowledgeable manufacturing. *Applied Mechanics and Materials, 433–435*, 2347–2350. https://doi.org/10.4028/www.scientific.net/AMM.433-435.2347

Wang, H. X., & Yan, H. S. (2016). An interoperable adaptive scheduling strategy for knowledgeable manufacturing based on SMGWQ-learning. *Journal of Intelligent Manufacturing, 27*(5), 1085–1095. https://doi.org/10.1007/s10845-014-0936-1

Wang, H. X., Sarker, B. R., Li, J., & Li, J. (2020). Adaptive scheduling for assembly job shop with uncertain assembly times based on dual Q- learning. *International Journal of Production Research*. https://doi.org/10.1080/00207543.2020.1794075

Wang, Y. C., & Usher, J. M. (2004). Learning policies for single machine job dispatching. *Robotics and Computer-Integrated Manufacturing, 20*(6), 553–562. https://doi.org/10.1016/j.rcim.2004.07.003

Wang, Y. C., & Usher, J. M. (2005). Application of reinforcement learning for agent-based production scheduling. *Engineering Applications of Artificial Intelligence, 18*(1), 73–82. https://doi.org/10.1016/j.engappai.2004.08.018

Wang, Y. C., & Usher, J. M. (2007). A reinforcement learning approach for developing routing policies in multi-agent production scheduling. *International Journal of Advanced Manufacturing Technology, 33*(3–4), 323–333. https://doi.org/10.1007/s00170-006-0465-y

Wang, Y. F. (2018). Adaptive job shop scheduling strategy based on weighted Q-learning algorithm. *Journal of Intelligent Manufacturing, 31*(2), 417–432. https://doi.org/10.1007/s10845-018-1454-3

Waschneck, B., Reichstaller, A., Belzner, L., Altenmüller, T., Bauernhansl, T., Knapp, A., & Kyek, A. (2018a). Optimization of global production scheduling with deep reinforcement learning. *Procedia CIRP*, 72, 1264–1269. https://doi.org/10.1016/j.procir.2018.03.212

Waschneck, B., Reichstaller, A., Belzner, L., Altenmuller, T., Bauernhansl, T., Knapp, A., & Kyek, A. (2018b). Deep reinforcement learning for semiconductor production scheduling. In *2018 29th annual SEMI advanced semiconductor manufacturing conference, ASMC 2018* (pp. 301–306). https://doi.org/10.1109/asmc.2018.8373191

Wei, Y., & Zhao, M. (2004). Composite rules selection using reinforcement learning for dynamic job-shop scheduling. In *2004 IEEE conference on robotics, automation and mechatronics* (Vol. 2, pp. 1083–1088). https://doi.org/10.1109/RAMECH.2004.1438070

Xanthopoulos, A. S., Koulouriotis, D. E., Tourassis, V. D., & Emiris, D. M. (2013). Intelligent controllers for bi-objective dynamic scheduling on a single machine with sequence-dependent setups. *Applied Soft Computing Journal, 13*(12), 4704–4717. https://doi.org/10.1016/j.asoc.2013.07.015

Xiao, Y., Tan, Q., Zhou, L., & Tang, H. (2017). Stochastic scheduling with compatible job families by an improved Q-learning algorithm. In *Chinese Control Conference, CCC* (pp. 2657–2662). https://doi.org/10.23919/ChiCC.2017.8027764

Yang, H. B., & Yan, H. S. (2009). An adaptive approach to dynamic scheduling in knowledgeable manufacturing cell. *International Journal of Advanced Manufacturing Technology, 42*(3–4), 312–320. https://doi.org/10.1007/s00170-008-1588-0

Yang, H. B., & Yan, H. S. (2007). An adaptive policy of dynamic scheduling in knowledgeable manufacturing environment. In *Proceedings of the IEEE international conference on automation and logistics, ICAL 2007* (pp. 835–840). https://doi.org/10.1109/ICAL.2007.4338680

Yingzi, W. E. I., Xinli, J., & Pingbo, H. A. O. (2009). Pattern Driven Dynamic Scheduling Approach using Reinforcement Learning. In *2009 IEEE international conference on automation and logistics* (pp. 514–519). https://doi.org/10.1109/ICAL.2009.5262867

Yuan, B., Jiang, Z., & Wang, L. (2016). Dynamic parallel machine scheduling with random breakdowns using the learning agent. *International Journal of Services Operations and Informatics*, 8(2), 94–103. https://doi.org/10.1504/IJSOI.2016.080083

Yuan, B., Wang, L., & Jiang, Z. (2013). Dynamic parallel machine scheduling using the learning agent. In *2013 IEEE international conference on industrial engineering and engineering management* (pp. 1565–1569). https://doi.org/10.1109/IEEM.2013.6962673

Zhang, T., Xie, S., & Rose, O. (2017). Real-time job shop scheduling based on simulation and Markov decision processes. In *Proceedings - Winter simulation conference* (pp. 3899–3907). https://doi.org/10.1109/WSC.2017.8248100

Zhang, T., Xie, S., & Rose, O. (2018). Real-time batching in job shops based on simulation and reinforcement learning. In *2018 Winter simulation conference (WSC)* (pp. 3331–3339). https://doi.org/10.1109/WSC.2018.8632524

Zhang, W., & Dietterich, T. G. (1995). A reinforcement learning approach to job-shop scheduling. In *1995 International joint conference on artificial intelligence* (pp. 1114–1120).

Zhang, W., & Dietterich, T. G. (1996). High-performance job-shop scheduling with a time-delay TD (λ) network. *Advances in Neural Information Processing Systems*, 91, 1024–1030

Zhang, Z., Zheng, L., Hou, F., & Li, N. (2011). Semiconductor final test scheduling with Sarsa(λ, k) algorithm. *European Journal of Operational Research*, 215(2), 446–458. https://doi.org/10.1016/j.ejor.2011.05.052

Zhang, Z., Zheng, L., Li, N., Wang, W., Zhong, S., & Hu, K. (2012). Minimizing mean weighted tardiness in unrelated parallel machine scheduling with reinforcement learning. *Computers and Operations Research*, 39(7), 1315–1324. https://doi.org/10.1016/j.cor.2011.07.019

Zhang, Z., Zheng, L., & Weng, M. X. (2007). Dynamic parallel machine scheduling with mean weighted tardiness objective by Q-learning. *International Journal of Advanced Manufacturing Technology, 34*(9–10), 968–980. https://doi.org/10.1007/s00170-006-0662-8

Zhao, M., Li, X., Gao, L., Wang, L., & Xiao, M. (2019). An improved Q-learning based rescheduling method for flexible job-shops with machine failures. In *2019 IEEE 15th international conference on automation science and engineering (CASE)* (pp. 331–337). https://doi.org/10.1109/COASE.2019.8843100

Zhou, L., Zhang, L., & Horn, B. K. P. (2020). Deep reinforcement learning-based dynamic scheduling in smart manufacturing. *Procedia CIRP, 93*, 383–388. https://doi.org/10.1016/j.procir.2020.05.163