Hybrid Flow Shop Scheduling through Reinforcement Learning: A systematic literature review

Victor Ulisses Pugliese
Universidade Federal de São Paulo
São José dos Campos, São Paulo
Brazil
pugliese@unifesp.br

Oséias Faria de Arruda Ferreira
EMBRAER S.A.
São José dos Campos, São Paulo
Brazil
oseias.ferreira@embraer.com.br

Fabio A. Faria[*]
Instituto Superior Tecnico,
Universidade de Lisboa
Lisboa, Portugal
fabio.faria@tecnico.ulisboa.pt

Abstract

This paper reviews the application of Reinforcement Learning (RL) in solving Hybrid Flow Shop Scheduling (HFS) problems, a complex manufacturing scheduling challenge. HFS involves processing jobs through multiple stages, each stage has multiple machines that can work in parallel, aiming to optimize objectives like makespan, tardiness, and energy consumption. While traditional methods are well-studied, RL's in HFS problem is relatively new. The review analyzes 26 studies identified through IEEE Xplore, Scopus, and Web of Science databases (as of April 2024), categorizing them based on RL algorithms, problem types, and objectives. Our analysis reveals the increasing adoption of advanced RL methods like Deep Q-Network (DQN) and Proximal Policy Optimization (PPO) to handle the complexities of HFS, often achieving superior performance compared to metaheuristics and scheduling heuristics. Furthermore, we explore the trend of integrating RL with other optimization techniques and discuss the potential for real-world applications, model interpretability, and the consideration of additional constraints and uncertainties. This review provides valuable insights into the current state and future directions in HFS using RL.

CCS Concepts

• **Computing methodologies** → **Artificial intelligence**; • **Planning and scheduling** → *Planning for deterministic actions.*

Keywords

Reinforcement Learning, Hybrid Flow Shop, Scheduling

ACM Reference Format:
Victor Ulisses Pugliese, Oséias Faria de Arruda Ferreira, and Fabio A. Faria. 2025. Hybrid Flow Shop Scheduling through Reinforcement Learning: A systematic literature review. In *The 40th ACM/SIGAPP Symposium on Applied Computing (SAC '25), March 31-April 4, 2025, Catania, Italy*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3672608.3707903

[*]Currently, Dr. Fabio A. Faria is an assistant professor at the Instituto Superior Tecnico and Collaborator Researcher at the Universidade Federal de São Paulo (UNIFESP).

1 Introduction

Smart Manufacturing has revolutionized the productive sector scenario, aligning itself with the principles of the Fourth Industrial Revolution, also known as Industry 4.0. Market dynamics played a key role, urging companies to shift from traditional production lines to more dynamic and flexible systems. This revolution is leveraging advances in cyber-physical systems, the Internet of Things (IoT), artificial intelligence (AI), and machine learning (ML) areas [29, 34].

Despite the wide range of opportunities in these areas, implementing robust and complex programming solutions often requires attention in real-world production environments.

In a production environment, the items being produced, called jobs, can typically be broken down into multiple tasks. Each task represents a small unit of work with constraints (e.g., raw materials, subassemblies, machinery, equipment, conveyors, and operators), most notably its required processing time. Scheduling determines when each task and on which resources will be processed at that time [6]

Scheduling tasks are often performed manually, documented in data sheets, or managed through limited software systems [12]. To address this challenge, scheduling heuristic and numerical optimization emerge as potential solutions, each with unique advantages and constraints [5].

Scheduling heuristics such as 'Early Due Date' (EDD), which prioritizes items based on the earliest due date, provides an efficient decision-making process for situations with high initial inventory and no tardiness. However, they may be inefficient when dealing with problems involving longer time horizons and lower initial stock levels [43]. On the other hand, numerical optimization methods find demonstrably optimal solutions, but their application is more complex as the production environment, and objectives need to be described mathematically in detail and then solved numerically [1]

The scheduling environment is a sociotechnical system comprising numerous interconnected components (e.g., weather, worker behavior, tools, machines, supply chain, spatial arrangement, buffers, and jobs) with stochastic behavior, such as fluctuations in worker productivity. Therefore, the interconnectivity and mutual influence among those components introduce additional complexity layers, manifesting in explicit relationships (machine-tool) and less apparent connections (weather-process parameters). As a result, manually constructing an accurate mathematical model proves challenging due to these complexities [5].

Flow shop scheduling problem (FSP) is a scheduling problem class wherein all tasks follow the same processing sequence across a set of machines. It is a typical combinatorial optimization problem, categorized as NP-hard, and widely applied to production and service systems. Also, there is an extent concept called the hybrid flow shop scheduling problem (HFS), which it involves multiple parallel machines per stage, making this problem even more challenging [40, 53]

Reinforcement Learning (RL) is a paradigm that continuously learns and adjusts to its environment through exploitation and exploration. It is well-suited for sequential decision-making across gaming, robotics, and control systems [10] such as scheduling problems.

The introduction of RL methods for solving scheduling problems in manufacturing has shown promising results in terms of minimization of makespan [33], total tardiness [22], processing time of workpieces [60], and energy consumption [51] including better solutions than heuristic and metaheuristic [47]. In addition, RL does not require an explicit environment model, so the agent learns to make decisions by observing the rewards of its actions from a state [67]. Finally, it is well-suited when dealing with complex, unstructured, and uncertain tasks, whereas metaheuristics may encounter big challenges to overcome those scenarios [59].

To the best of our knowledge, we have not found works in the literature that address a review of RL methods for the HFS. In this regard, our efforts have identified only sixteen articles in the literature, with most studies focusing on sequential, discrete, and deterministic environments.

This work is organized as follows: The next section 2 briefly describes Hybrid Production Flow Shop Scheduling, including its types and goals, six reinforcement learning methods, thirteen metaheuristics, and a set of scheduling heuristics. Section 3 describes our methodology. Section 4 presents the research results and explores related work. Section 5 discusses related work and answers the research questions. The main findings and conclusions are presented in Section 6.

2 Background

This section briefly introduces the main definitions and concepts related to the areas of HFS (Section 2.1), RL (Section 2.2), and metaheuristics (Section 2.3), and Scheduling Heuristic (Section 2.4) addressed in this review.

2.1 Hybrid Flow Shop Scheduling

Flow Shop Scheduling (FSP) is a classic problem in manufacturing systems, involving the sequencing jobs through multiple production stages with the same machine order. This combinatorial problem is NP-hard, indicating the lack of known algorithms that provide polynomial-time solutions for all instances. Consequently, various authors have proposed approximation solutions to tackle this complexity [21, 65].

Companies have adopted strategies to increase production capacity and optimize specific objective functions to meet the growing demand for customized products in various industrial sectors. One of these strategies involves the integration of multiple parallel machines within a production stage, a concept known as Hybrid Flow Shop Scheduling (HFS) [21].

Based on our research in selected papers, we found these types of HFS as listed below.

(1) HFS: The most basic form of HFS.
(2) Hybrid Flow Shop on Batch Processing Machines (HFS-BPM) is a scheduling problem where jobs are processed in batches and must go through multiple stages with both batch processing and single-item processing machines [54].
(3) Hybrid Flow Shop with Learning and Forgetting effects (HFS-LF) is a scheduling problem where worker experience affects task durations. Learning decreases processing times, while forgetting or interrupting increases them [55].
(4) Multi-stage Hybrid Flow Shop (MHFS) is a scheduling problem where jobs are scheduled across multiple stages, and machines within each stage can undergo condition-based maintenance [36].
(5) Two-Stage Hybrid Flow Shop (THFS) is about finding the best way to schedule jobs on machines in a two-stage production system with special rules and limitations [17].
(6) Distributed Two-Stage Hybrid Flow Shop (DTHFS) is a scheduling problem of organizing tasks across multiple factories, each factory with two-stage production [69]
(7) Re-entrant Hybrid Flow Shop (RHFS) is scheduling multiple jobs on parallel machines, with some jobs needing to revisit machines [11]
(8) Dynamic Re-entrant Hybrid Flow Shop (DRHFS) is a scheduling problem in a hybrid flow shop where jobs revisit stages and unexpected events disrupt production [49]
(9) Distributed Hybrid Flow Shop (DHFS) is a scheduling problem with multiple factories, each having several stages with multiple machines [2].
(10) Distributed Assembly Hybrid Flow Shop (DAHFS) is a scheduling problem involving multiple factories and assembly operations [38].
(11) Distributed Assembly Hybrid Flow Shop with Flexible Preventive Maintenance (DAHFS-FPM) is an approach for scheduling tasks in a distributed assembly line, taking into account machine wear and tear and the need for maintenance [48].
(12) Distributed Blocking Hybrid Flow Shop (DBHFS) is a scheduling problem in multiple factories' production with limited buffer space [20].
(13) Distributed Heterogeneous Hybrid Flow Shop Scheduling (DHHFS) is scheduling jobs with different priorities across multiple, unique factories, each having several production stages with various machines that can work in parallel.
(14) Distributed Hybrid Flow Shop with Multiprocessor Tasks (DHFS-MT) is when multiple factories work together to complete tasks with multiple parts [56].
(15) Energy-Aware Distributed Hybrid Flow Shop (EADHFS) is scheduling jobs in a distributed manufacturing system with hybrid flow shops to minimize both production time and energy use [68].
(16) Fuzzy Distributed Hybrid Flow Shop Considering On-Time-Delivery (FDHFS-OTD) is a scheduling problem in multiple factories, balancing speed, energy consumption, and on-time delivery [31].

The list below provides an overview of the objective functions that we found in our research.

- Minimization of Makespan: This refers to the total time it takes to complete all jobs in a given schedule. It is essentially the duration from the start of the first job to the finish of the last job in the schedule [17].
- Minimization of Total Tardiness: It measures the accumulated delay of jobs that are completed after their due date [17].
- Minimization of Processing Time of Workpieces: It aims to schedule and assign workpieces to machines in a way that reduces the total time required to complete all the workpieces [60].
- Minimization of Energy Consumption: Reduces energy consumption without compromising production efficiency [20].
- Minimization of Total Production Cost: It means finding ways to reduce both the expenses associated with production delays and the costs of maintaining the machines used in the process [36].
- Minimization of Fuzzy Makespan: It is similar to makespan, but it considers the reality where processing times can be uncertain due to factors like machine wear, operator skill, and variations in materials [31].
- Delivery Accuracy: It is calculated using the similarity between the measures of makespan and due dates. A higher value indicates a greater degree of overlap and thus better delivery accuracy [31].
- Penalties for exceeding due dates: It penalizes late deliveries, calculating the difference between completion time and due date, and applying penalty cost per unit of time [3].

2.2 Reinforcement Learning Methods

Reinforcement Learning (RL) addresses the problem of automatically learning optimal decisions over time. Despite that uses well-established supervised learning methods such as deep neural networks for function approximation, stochastic gradient descent (SGD), and backpropagation, RL applies them differently with no supervisor through a reward signal, and the delayed feedback [63]. Therefore, an RL agent receives dynamic states from an environment and performs actions that aim to obtain maximum rewards in trial-and-error interactions. Thus, the agent learns behavior from data [32]. [67] model the RL cycle similar to a Markov Decision Process (MDP), as shown in Figure 1. In Figure 1, the agent and

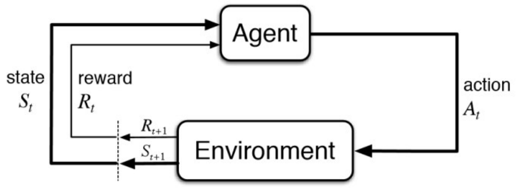

Figure 1: RL cycle

environment interact at each discrete time step of a sequence, $t = 0, 1, 2, 3, \cdots$. At each time step t, the agent receives some representation of the environment's state $s_t \in S$, where S is the set of possible states and the action is $a_t \in A(s_t)$, where $A(s_t)$ is the set of actions available in state st. At time $t + 1$, as a consequence of its actions, the agent receives a numerical reward $r_{t+1} \in R$ and finds itself in a new state s_{t+1} [67].

At each time step, the agent implements a mapping from state to probabilities of each possible action. This mapping is called the agent's policy and is denoted as π_t, where $\pi_t(s, a)$ is the probability that $a_t = a$ if $s_t = s$. Reinforcement learning methods specify how the agent changes policy due to experience. The agent aims to maximize the total reward received over the long run [67].

2.2.1 Q-learning. Q-learning is an RL method that implements a mapping in a Q-table from state to probabilities of each possible action, aiming to maximize future rewards. Notably, it does not require prior knowledge about the dynamics of the environment. Q-learning is an off-policy algorithm that allows one to learn from experiences, even when the current policy is suboptimal [45].

The equation mentioned in [67] essentially updates the Q-value for the current state-action pair (S_t, A_t) based on the immediate reward R_{t+1} and the estimated maximum future cumulative reward. The update is a weighted sum of the current estimate and the new information, where the learning rate α determines the weights and γ is the discount factor, a constant between 0 and 1 that represents the importance of future rewards. The goal is to iteratively improve the Q-values over time through experience and learning from the consequences of different actions in different states. This process helps the agent learn a policy that maximizes cumulative rewards over time.

$$\begin{aligned} Q(S_t, A_t) &\leftarrow Q(S_t, A_t) \\ &\quad + \alpha \left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\right] \end{aligned} \tag{1}$$

It solves the issue of iteration over the full set of states, but still can struggle with situations when the count of the observable set of states is very large [63].

2.2.2 State-Action-Reward-State-Action (SARSA). SARSA is an on-policy reinforcement learning algorithm used to train a MDP model on a new policy. It operates by taking actions based on rewards received from previous actions, storing a table of state-action estimate pairs for each Q-value, known as a Q-table [67].

What sets SARSA apart is that it's an "on-policy" algorithm. This means it learns based on the actual actions it takes, following a specific policy or strategy. It's like a student learning from their own experiences rather than just watching others. This makes SARSA a cautious learner, preferring safe and reliable actions over potentially risky ones with higher rewards [71].

2.2.3 Deep Q-Network (DQN). DQN is a variation of *Q-learning* that employs Deep Neural Networks. The method has been successfully applied to Atari 2600 games. It estimates Q functions as the expected value of future rewards when performing a certain action in a specific state. For example, the network inputs several game frames in Atari and calculates state values for each action [44]. To stabilize training, DQN employs replay memory and a target network.

Replay memory randomly samples past transitions to smooth the training distribution, reducing the correlation between consecutive experiments. A target network, a copy of the main network, stabilizes training by calculating Q values for the next state, preventing fluctuations in Q values due to constant weight updates [44].

Furthermore, it uses a ϵ-greedy method, randomly exploring some actions while mostly sticking to the best-known action, gradually learning the best way to win through exploration and exploitation [44].

2.2.4 *Proximal Policy Optimization (PPO).*

PPO uses the actor-critic and trains a policy in an on-policy way, meaning it samples actions based on the latest policy iteration [26]. In this way, two neural networks play the "actor" and "critic" roles. The "actor" learns the policy, while the "critic" estimates the value of the action value function (or advantage) that is used to train the "actor"

The training process also involves computing future rewards and advantage estimates to refine the policy and adjust the value function. It is updated using a stochastic gradient ascent optimizer, while the value function is optimized through a gradient descent algorithm as outlined in [61].

The degree of randomness in action selection is contingent upon the initial conditions and the training procedure. Typically, as training progresses, the policy becomes less random due to an updated rule that encourages exploration of previously discovered rewards [64].

2.2.5 *Advantage Actor-Critic (A2C).*

A2C, often perceived as a distinct algorithm, is revealed in “A2C is a special case of PPO" as a specific configuration of Proximal Policy Optimization (PPO) operating within the actor-critic approach. A2C shares similarities with PPO in employing separate neural networks for policy selection (actor) and value estimation (critic). Its core objective aligns with PPO when the latter's update epochs are set to 1, effectively removing the clipping mechanism and streamlining the learning process [42].

A2C is a synchronous adaptation of the Asynchronous Actor-Critic (A3C) policy gradient approach. It operates deterministically, waiting for every actor to complete its experience segment before initiating updates, averaging across all actors. This strategy enhances GPU utilization by accommodating larger batch sizes [46].

2.2.6 *REward Increment = Nonnegative Factor × Offset Reinforcement × Characteristic Eligibility (REINFORCE).*

REINFORCE is a fundamental algorithm in reinforcement learning based on Monte Carlo that directly optimizes an agent's policy. It operates by estimating the gradient of the expected return concerning the policy parameters using samples collected from the environment. These samples typically consist of trajectories, which are sequences of states, actions, and rewards experienced by the agent. The key idea is to increase the probability of actions that lead to higher rewards and decrease the probability of actions that lead to lower rewards. REINFORCE achieves this by adjusting the policy parameters toward the estimated gradient. The algorithm is known for its simplicity and effectiveness, making it a popular choice for various reinforcement learning tasks. [27, 67].

2.3 Metaheuristic

Our literature review found benchmark papers using metaheuristics and RL to handle HFS. Some authors combine both methods, as described in this paper [69]

2.3.1 *Artificial Bee Colony (ABC).*

is inspired by honey bees' foraging behavior. Employed bees search for solutions, while onlooker bees choose solutions based on their fitness. Scouts randomly explore new solutions [70].

2.3.2 *Artificial Immune Systems (AIS).*

is inspired by the human immune system. It uses techniques like clonal selection and negative selection to mimic learning and adaptation, enabling it to solve complex problems effectively [15].

2.3.3 *Cooperative Memetic Algorithm (CMA).*

combines evolutionary algorithms and local search. It uses a population of solutions that compete and undergo refinement to find near-optimal solutions for complex problems [24].

2.3.4 *Cuckoo Optimization Algorithm (COA).*

is inspired by cuckoos laying eggs in other birds' nests. It represents solutions as habitats and eggs, with eggs surviving based on their similarity to host eggs. It quickly finds good solutions but is limited to continuous optimization [18].

2.3.5 *Genetic Algorithms (GAs).*

are inspired by natural selection. They evolve a population of solutions through processes like mutation and alteration to find improved outcomes [28].

2.3.6 *Iterated Greedy (IG).*

is a search method that repeatedly destroys and reconstructs solutions. Unlike other algorithms, it only yields one solution per iteration [20, 66].

2.3.7 *Migrating Birds Optimization (MBO).*

is inspired by migrating birds' V-formation. It uses a leader-follower structure and information sharing to efficiently explore and find high-quality solutions for complex problems [14].

2.3.8 *Multiobjective Evolutionary Algorithms (MOEAs).*

solve problems with multiple conflicting objectives. They evolve a population of solutions using selection, reproduction, and population updating to find a set of Pareto optimal solutions [8, 8].

2.3.9 *Multi-population memetic algorithms (MPMA).*

combine global and local search using multiple subpopulations. They excel in solving complex problems with multiple peaks and variables by sharing knowledge between subpopulations [4, 7].

2.3.10 *Particle Swarm Optimization (PSO).*

is inspired by bird flocks' social behavior. Particles in PSO adjust their positions based on their own and the swarm's best-found locations to find optimal solutions [19].

2.3.11 *Shuffled Frog Leaping Algorithm (SFLA).*

combines memetic and particle swarm optimization. It uses a population of virtual frogs that search for solutions through population division, memeplex search, and shuffling [9, 23].

2.3.12 *Teaching-Learning-Based Optimization (TLBO).*

is inspired by classroom learning. It uses teaching and learning phases to optimize an objective function [13, 41].

2.3.13 *Variable Neighborhood Search (VNS).*

explores the solution space by systematically changing neighborhoods. It starts with an initial solution and moves to different neighborhoods to avoid local optima [19].

2.4 Scheduling Heuristics

It is also possible to apply scheduling heuristics such as Shortest Processing Time (SPT) which selects the task with the shortest processing time to run first; Longest Processing Time (LPT) which chooses the job with the longest processing time to run first; First Come, First Served (FCFS) which selects tasks in order of arrival, Nawaz-Enscore-Ham (NEH) which works by iteratively inserting jobs into a partial schedule to minimize the makespan; Shortest Remaining Time (SRT) which prioritizes the task with the shortest remaining processing time [16, 47]; Family Production Tatic which groups jobs by family to reduce setup times [17]; Proximity-first assigns tasks to the closest available resource; First-In-First-Out (FIFO) processes tasks in the order they are received; Process-first prioritizes completing a single task from start to finish before moving on to the next [57]; Most Work Remaining (MWKR) is to tackle the longest jobs first; and Least Work Remaining (LWKR) is to complete as many jobs as possible quickly [52].

3 Methodology

Although we have identified previous reviews such as [37] and [40] that conducted extensive studies on the HFS problem, none of them included the use of RL for this type of scheduling problem. [37] studied works published between 2010 and 2019, while [40] describe works published until 2010. According to our study, the first work about RL for scheduling is from 2019 with [60]. We extensively analyzed papers that employ RL methods to address challenges in HFS problems. Therefore, this section provides an overview of the literature review from January/2019 to April/2024, focusing on recent and significant contributions to the application of RL to the HFS problem. The following subsections, namely 3.3, 3.1, and 3.2 delve into the research questions, search strings, and inclusion/exclusion criteria for papers, respectively, adopted to carry out the review.

3.1 Research Questions

The research protocol was defined to return works that employ RL in HFS problem to have a state-of-the-art view of the topic and answer the following research questions (RQ): RQ1: Trends and statistics: when and where were studies published?; RQ2: What RL methods have already been used in the literature? RQ3: How were RL methods evaluated in the literature?; and RQ4: Is any improvement gained from using RL over HFS?

3.2 Search strings

Once the objective was established, the research protocol was followed by selecting paper searchers across relevant scientific databases and determining the keywords for the search string. We chose these three digital libraries: IEEE[1], Scopus[2], and Web of Science[3].

We employed the search keywords “flow", “shop", “reinforcement", and “learning" to retrieve relevant papers from digital libraries. This targeted search produced 371 results, meticulously gathered in BibTeX format across diverse areas of study to guarantee comprehensive coverage and avoid overlooking pertinent papers.

[1]IEEE https://ieeexplore.ieee.org
[2]Scopus https://www.scopus.com
[3]Web of Science http://webofknowledge.com/WOS

3.3 Search criteria and inclusion/exclusion of articles

In the process of selecting articles, we employed a Python script that followed specific criteria to filter the works from the initial search results in BibTeX format. First, duplicate entries were eliminated. Then, for an article to be considered, its title, abstract, or keywords had to include all the tokens from the search string, which encompassed terms like "Hybrid," "DQN," "PPO," and "A2C." Additionally, only papers published from 2019 onward, marking the first year reinforcement learning was applied in Hybrid Flow Shop Scheduling (HFS), were included in the final selection.

To select the relevant papers, we manually reviewed the content of each returned result and applied specific inclusion criteria. Only papers that were available for download, addressed Hybrid Flow Shop (HFS) and its variations, and were written in English were included in the final selection.

Finally, from a total of 371 papers that were initially identified, after applying the inclusion and exclusion criteria, 26 papers emerged for in-depth analysis.

4 Research Findings

This section thoroughly reviews the literature found using our search method adopted for this work. Section 4.1 provides useful statistics about the papers we selected, while Section 4.2 summarizes the papers chosen.

4.1 Overview and Statistics

This subsection discusses statistics on the selected papers during the systematic review.

RL for the HFS problem emerged in 2019 with the publication of [60]. Since then, interest in this approach has steadily grown, with 1 publication in 2020 [58], 3 in 2021 [16, 20, 68], 5 in 2022, and a significant increase to 11 in 2023 [23, 25, 30, 31, 36, 39, 48–50, 54, 56]. In 2024, 5 publications have already appeared until May [2, 3, 11, 52, 55]. No individual journal has dedicated more than two papers to this topic.

HFS papers have predominantly focused on common objective functions. Among these, 17 papers prioritize the minimization of makespan [2, 11, 16, 17, 23, 30, 31, 48–50, 52, 54–57, 68, 69], 6 aim at minimizing total tardiness [17, 25, 35, 39, 47, 49], another 7 target energy consumption reduction [2, 3, 20, 30, 31, 39, 68], and 2 focus on minimizing the processing time of workpieces [58, 60]. Additionally, 6 papers explore other objectives [3, 31, 36, 49, 50, 60]. Notably, 8 studies adopt a multi-objective approach, addressing two or more of these criteria simultaneously [2, 3, 17, 30, 31, 39, 49, 50].

Reinforcement Learning (RL) methods demonstrate considerable diversity. Among the reviewed articles, 13 employed Q-learning [2, 20, 23, 30, 31, 36, 47, 48, 55, 56, 58, 60, 69], 7 DQN [11, 25, 35, 36, 39, 47, 57], 5 PPO [16, 49, 50, 52, 57], and 1 each A2C [17], REINFORCE [68], and SARSA [3]. Furthermore, 2 papers used other RL approaches [36, 54]. Regarding the agent structure, 7 studies used multi-agent [11, 17, 35, 36, 49, 50, 52], while 19 focused on single-agent.

4.2 Summary of selected papers

In 2019, Han et al. [60] pioneered RL in HFS. The researchers treated HFS as MDP, where the state represents the current stage and workpiece, and actions involve selecting machines. Han et al. compared the performance of Q-learning to a GA and an AIS in two configurations: automotive engines and aircraft carriers, where the Q-learning outperformed its counterparts and minimized the processing time of workpieces. It also achieves satisfactory real-time performance in deterministic initial sequence HFS through the application in this paper

Guo et al. [58] also showed that Q-learning works better than GA. Their goal was to minimize the processing time of workpieces in steel production. Thus, they modeled the state as a tuple (piece, workpiece), an action as the machines in the next state, and a production process with four sequential stages (steelmaking, refining, casting, and rolling) as HFS. Both results obtained by these papers are not necessarily optimal, but they provided some reference for HFS compared with manual scheduling and some intelligent algorithms scheduling.

In other studies, Reinforcement Learning methods were combined with Neural Networks. For instance, Wang et al. [47] implemented DQN based on classical scheduling rules to handle dynamic order arrivals in an HFS simulated environment. To minimize total tardiness, a set of seven standardized state features, ranging between 0 and 1, is employed to represent the status of each rescheduling point. They also defined the action space for selecting the priority processing job according to the scheduling rules. Comprehensive numerical experiments are conducted across various production environments. Wang et al. found advantages in using DQN and Q-learning over simple scheduling rules such as FCFS, SRT, EDD, SPT, and LPT.

Gil and Lee [57] redesigned the material scheduling in HFS with many machines, considering real-world constraints, such as the dynamic location of the Robotic Transfer Unit (RTU) responsible for material transportation, which impacts scheduling decisions. Moreover, grouping machines in this RL application has mitigated the challenge of dealing with high-dimensional spaces by using mass machines in unmanned lines. The study introduced a reward prediction technique to minimize the time gap between actions and rewards to enhance learning. They used the PPO method to minimize makespan, and its performance is better than DQN and Proximity-First, FIFO, and Process-First scheduling rules.

Ni et al. [16] present a Reinforcement Learning algorithm designed for large-scale HFS from a multi-graph perspective. This approach showcases efficiency in policy search and impressive generalization across diverse problem distributions and scales. The algorithm's efficacy was demonstrated through training the PPO method using Huawei's supply chain data, focusing on minimizing makespan. The results were superior to NEH, IG, and a Bilevel Deep Reinforcement Learning Scheduler (BDS). The BDS operates with a two-tier structure, where the upper level explores an initial global sequence, and the lower level refines partial sequences through exploitation, employing a sliding window sampling mechanism to connect both levels.

Gerpott et al. [17] implemented A2C to train multiple agents to schedule jobs in THFS with family-dependent setup times, searching to minimize total tardiness and makespan as multiobjective optimization. It performed better than conventional scheduling heuristics like EDD and Family Production in an OpenAI Gym discrete event simulator.

Wang et al. [35] also deals with THFS, where jobs arrive dynamically, and machines can process jobs in batches, proposing a novel approach called MA-IDDQN, which leverages multi-agent reinforcement learning with independent double DQN. The system is designed with two agents, one for batch forming and another for scheduling, aiming to minimize total tardiness in a simulated environment. A key innovation was including a "waiting" action in the batch forming agent's policy, allowing for improved batch utilization and reduced processing times. Thus, the cooperative environment and multi-agent analysis for joint optimization improved performance, and MA-IDDQN achieved better than heuristics (e.g., EDD, SPT, FIFO), DDQN, and MA-IDDQN variations.

Xi and Lei [69] addressed the DTHFS with uncertain processing times and sequence-dependent setup time. They proposed a novel approach called Q-learning-based teaching-learning optimization (QTLBO), combining the strengths of TLBO and Q-learning. QTLBO employs four distinct phases, teacher, learner, teacher's self-learning, and learner's self-learning, to efficiently explore the solution space. The integration of Q-learning dynamically adjusts the algorithm's structure, enhancing its ability to find optimal solutions. Through experiments, QTLBO performs better than metaheuristics such as TLBO, ABC, IG in minimizing makespan. This research offers valuable insights into integrating reinforcement learning with metaheuristics for tackling scheduling problems with uncertainty in distributed manufacturing environments.

Liu et al. produced two scientific papers [49, 50] on HFS, considering the presence of multi-skilled workers and their fatigue states.

The first work [50] focused on HFS, using multiple processing stages with parallel machines, require assignment of multi-skilled workers who experience fatigue. The authors developed an agent-based simulation system combining GA and PPO to handle the uncertainty introduced by worker fatigue. The GA optimizes job sequencing and machine selection, while the PPO focuses on assigning workers to tasks efficiently, considering their fatigue levels and skill sets.

The second work [49] focused on DRHFS with unforeseen events like new job arrivals, machine breakdowns, and worker unavailability to disrupt production plans, aims to minimize tardiness. Thus, they employed PPO agents, representing machines, workers, jobs, and stages, which collaborate and negotiate to respond to disruptions and maintain production flow. Two RL methods were developed to address specific sub-problems: one for job sequencing and machine selection, utilizing a reward function that balances long-term and short-term goals, and another for worker assignment, employing an attention-based neural network to efficiently extract relevant features and make informed decisions.

In both papers [49, 50], PPO outperforms other algorithms like GA and SRPT, EDD, SPT, Minimum Fatigue, and Earliest Available Worker to minimize makespan.

Gholami and Sun [56] investigate the DHFS-MT aiming to minimize makespan among the factories. The paper introduces a novel framework called Conditional Markov Chain Search (CMCS) to automate the design of heuristic algorithms, eliminating the need for manual parameter tuning. The study proposes two new concepts, "weight" and "impact," to assess job resource usage and influence at different stages, enabling intelligent load balancing between factories. By formulating HFS-MT as a Markov Decision Process (MDP) and employing a hybrid Q-learning-local search approach, the CMCS effectively schedules jobs within each factory. Experiments on benchmark instances demonstrate the superiority of CMCS over IG and SFLA variations, achieving significant improvements in makespan reduction and solution quality

Qin et al. [20] propose a Q-Learning-based IG metaheuristic (IGQs) in a DBHFS. The IGQ aims to minimize energy consumption by efficiently allocating jobs to factories and determining their processing order while considering blocking constraints (limitations due to limited buffer space between stages). They detail the model of the DBHFS and the design of the IGQ algorithm, including initialization with NEH Heuristic, global search, local search, and the Q-learning selection mechanism. The results demonstrate the effectiveness of IGQ in reducing energy consumption compared to metaheuristics such as ABC, IG, PSO, MBO, and GA variations.

Wang and Wang [68] are also concerned about energy awareness, addressing the challenge of EADHFS, aiming to minimize makespan and energy consumption. They propose a novel approach using a CMA with a RL-based policy agent. The CMA incorporates problem-specific heuristics for initialization, a policy network trained by REINFORCE to guide operator selection, and a solution selection method based on decomposition. Additionally, local intensification and energy-saving strategies are employed to further enhance the algorithm's performance, and it showed the best results compared to CMA using Random Selection, SFLA or GA variations.

Cai et al. [23] introduced QSFLA, a novel algorithm that merges the strengths of the SFLA metaheuristic with the adaptive learning capabilities of Q-learning to solve DAHFS with fabrication, transportation, and assembly, minimizing makespan. By dynamically selecting the most suitable search strategy based on the current population's state, QSFLA effectively explores the solution space while maintaining a balance between exploration and exploitation. This integration of reinforcement learning empowers QSFLA to achieve superior performance compared to SFLA CMA, and variants of PSO, VNS, and COA, as demonstrated through extensive computational experiments and a real-life case study in a furniture company.

Cai and Wong [2] published a new paper, employing the QSFLA to minimize the makespan and energy consumption in DHFS this time. The Q-learning algorithm dynamically chooses the most effective search strategy based on the current state of the population, leading to improved convergence and uniformity of solutions. QSFLA outperforms GA and QSFLA variants, making it a competitive method for solving this complex scheduling problem

Jai et al [48] addressed the DAHFS-FPM, focusing on a three-stage assembly process with the added complexities of machine deterioration and the need for preventive maintenance. This research establishes a mathematical model for it and introduces flexible preventive maintenance strategies to minimize makespan. They employed a MPMA combined with Q-learning (MPMA-QL) which divides the population into subpopulations, each employing distinct crossover strategies to maintain diversity and avoid local optima. Furthermore, Q-learning is used to dynamically adjust the number of individuals within each subpopulation, leading to more efficient information exchange and improved problem-solving capabilities. Experimentation demonstrated that MPMA-QL outperforms MPMA, SFLA, and ABC, including their reinforcement learning variations.

Luo et al [54] proposes a RL approach utilizing actor-critic techniques to solve a production scheduling problem within the Industrial Internet of Things context. Specifically, it addresses the HFS-BPM to minimize makespan and the process involves determining machine allocation, processing start times, and job completion times, enabling flexible manufacturing. The proposed approach is evaluated on the publicly available dataset and real steel plant production dataset from Nanjing Iron and Steel Co., Ltd. (NISCO). It performs better than existing baselines such as FIFO, LPT, SPT, ABC, GA on both datasets, resulting in a minimized makespan. The paper does not mention the name of RL actor-critic method.

Luo et al [25] employed an HFS with unrelated parallel machines with dynamically arrived jobs, using online scheduling based on DQN and scheduling rules to minimize total tardiness. The DQN learns to select the best scheduling rule based on the current state of the production environment, which is represented by seven features such as machine utilization and job completion rates. The paper also presents the experimental results showing the effectiveness of the DQN algorithm compared to classic scheduling rules such as SPT, FCFS, and EDD under different production scenarios.

Li et al. [39] addressed the DHHFS with multiple priorities of jobs, aiming to minimize tardiness and energy consumption simultaneously. They proposed a solution (D2QCE) combining a co-evolutionary algorithm with a double DQN to balance computational resources between global and local searches, incorporating problem-specific knowledge to improve convergence and using this DQN variation to select the best scheduling operators intelligently. The D2QCE outperforms the mainstream MOEAs algorithms, making it a valuable tool for real-world manufacturing scenarios.

Lin et al. [11] solved the RHFS in an automated material handling system using two DQN agents. The problem involves scheduling jobs in a manufacturing environment where jobs may need to revisit specific workstations multiple times, including temporary storage areas called stockers, used to manage work-in-process inventory. DQN1 selects the machine for the next job, and DQN2 selects the next job for processing. Simulation results showed that both DQN approaches outperform metaheuristics like GA and PSO, especially for large-scale problems. The study emphasizes the potential of RL for solving complex scheduling problems in smart factories and highlights the impact of stocker placement on makespan.

Cui and Yuan [3] focused on energy-aware production scheduling in the photovoltaic glass industry, whose layout is an HFS with batch and non-batch machines. Their goals were minimized energy consumption and penalties for exceeding due dates. They proposed a hybrid GA enhanced by reinforcement learning design, in which expected Sarsa is used to extract critical knowledge about algorithmic parameters during the population evolution to guide the exploration of the GA. The chromosome is encoded by a priority list representing the priority relationship among all jobs, which is then decoded by a constructive heuristic based on the problem feature analysis. The results were greater than other metaheuristics like GA and PSO.

Zhang el al. [55] tackle the complex challenge of scheduling tasks in an HFS-LF, where both employee learning and forgetting impact production efficiency. They introduce two models for learning and forgetting, incorporating sequential setup times. The proposed solution, called Meta-Reinforcement Learning-based Metaheuristic (MRLM), aims to minimize makespan by initially creating a schedule using a NEH heuristic and then refining it with various search operators. MRLM's distinctive feature is its adaptive search framework, which involves a meta-training phase to learn about operator selection and a Q-learning-driven search phase to dynamically choose operators based on feedback from previous searches. It outperforms metaheuristics like IG, CSA, PSO, ABC, VNIS variations.

Deng et al. [31] introduce a novel approach called 3D-EDA/RL to solve an FDHFS-ODT, where multiple factories collaborate to complete orders while minimizing energy consumption fuzzy makespan, and delivery accuracy. 3D-EDA is a three-dimensional matrix where X and Y axes represent the jobs, Z is the factories, and Value is the probability of a job being assigned to a specific factory and the likelihood of it being processed before another job at the same stage within that factory. This method addresses the uncertainties inherent in real-world manufacturing by representing processing times as fuzzy numbers and measuring delivery accuracy using fuzzy relative entropy. The algorithm incorporates several innovative techniques, including hybrid initialization, a 3-D probability matrix to guide offspring generation, and an adaptive biased decoding mechanism based on Q-learning. The 3D-EDA/RL achieved the best results compared to GA, CMA, and MOEAs variations.

Xu et al. [30] focus on optimizing production processes, and minimize makespan and energy consumption in a steel manufacturing setting, where machine congestion and transportation time are significant factors. They use adaptive Q-learning to make real-time decisions based on job, machine, and queue information, using scheduling rules like SPT, FCFS, and LPT in a THFS context. A key innovation is the adaptive selection of objectives using a t-test at each decision point, resulting in improved performance compared to SPT, FCFS, LPT, Q-learning with linear weighting, and a variation of GA.

Zhao et al. [52] propose an end-to-end architecture based on Heterogeneous Graph Neural Network (HGNN) structure and PPO to solve HFS with scalability, aiming to minimize makespan. It addresses this problem by modeling the scheduling state as a Heterogeneous Graph and employing a specially designed HGNN to extract scheduling information. This information is then used by a PPO algorithm to learn optimal scheduling policies. This RL method outperforms LPT, SPT, FIFO, MWKR, and LWKR.

Zhang et al. [36] worked to scheduling maintenance and production in a MHFS, where machines deteriorate over time and jobs have uncertain processing times, aiming to minimize the total production cost. They employed a decentralized partially observable Markov decision process (Dec-POMDP) model to handle this challenge, but the vast state and action spaces posed a hurdle for conventional reinforcement learning methods. In response, They introduced the Counterfactual Attention Multi-Agent Reinforcement Learning (CAMARL) framework, which has three main parts: a way to focus on important details, a method to simplify the many possible actions, and a unit to help find the best schedules quickly. CAMARL outperforms methods such as variations of GA, VNS, Q-learning, and DQN in different production situations, proving its effectiveness in optimizing schedules while considering machine wear and job uncertainties.

5 Discussion

This section delves into the essential findings and implications of selected papers in our systematic literature review. We discuss their effectiveness in HFS scenarios and highlight promising directions for future research. We also answer the research questions about RL in HFS, which we defined in the Methodology section.

Although research on RL for HFS began in 2019, there are already 26 works with different scheduling problems (e.g., two-stage distributed assembly, batch processing machines, and energy-aware distribution). However, there are few works by type, and only nine articles focus on the primary form of HFS.

The pioneering studies by [58, 60] have profoundly impacted the field of RL for HFS. These studies, which demonstrated the effectiveness of RL, particularly Q-learning, in HFS scheduling by treating the problem as an MDP, set the stage for subsequent research. Their focus on minimizing processing time not only highlighted the superiority of Q-learning over GA but also paved the way for the adoption of more advanced approaches in the field.

The evolution of RL algorithms in HFS research is a testament to the field's progress. The initial studies, which focused on basic RL algorithms like Q-learning, have paved the way for more sophisticated approaches such as DQN and PPO. This shift in focus reflects the increasing complexity of HFS problems and the need for algorithms that can handle high-dimensional state spaces and dynamic environments. For example, [47] implemented DQN to handle dynamic order arrivals, [57] used PPO to manage the dynamic location of the RTU, and [16] designed a PPO for large-scale HFS.

RL can address the complexities of the HFS by incorporating real-world constraints, multi-objective optimization, and uncertainty management. Techniques like tailored reward functions, terminal state definitions, and restricted action spaces streamline learning. However, most studies are limited to simulated environments, highlighting the need for real-world validation that accounts for factors like worker fatigue, machine breakdowns, and processing time uncertainties. Future research should focus on practical implementation and data collection in real-world HFS scenarios.

The articles also applied the RL in different areas, such as steel production, aircraft carriers, automotive engines, furniture, material, and glass. Thus, they did not compare their experiments with each other nor share a dataset, and we cannot conclude which approach is the recommended one. However, we observed scheduling cases in which PPO outperforms DQN, Q-learning, and heuristics approaches [16, 52, 57].

PPO due to its ability to handle different action spaces such as discrete, multi-discrete, and even continuous, making it perfect for precise control. Unlike DQN, which focuses on values, PPO directly improves policies, making learning more steady and efficient. It uses reduced objective functions and advantage estimation to make policy updates smoother, improving learning's stability and reliability. Additionally, PPO is better at using data efficiently, learning from recent experiences and maximizing data use through importance sampling and on-policy learning. Its trust region constraint and adaptive learning rate also enhance stability and robustness. Overall, these qualities help PPO converge faster to optimal solutions, especially in complex environments with high-dimensional state and action spaces [16, 52, 57].

Some studies explore the benefits of combining RL with other optimization techniques, such as [49] applied RL with a GA, [69] Q-learning with TLBO, and [23] uses Q-learning with SFLA.

The interpretability of RL models for HFS deserves more attention because understanding the decision-making process behind RL agents would enhance trust and provide valuable insights for human schedulers.

Finally, We can synthesize the selected papers to answer the research questions established as presented below:

- RQ1: Trends and statistics: when and where were studies published?
  Section 4.1 presents an annual breakdown of published papers detailing the HFS types, objectives, and executed RL methods. Additionally, it explores whether a multiobjective or multi-agent approach characterizes the examined works.
- RQ2: What RL methods have already been used? And RQ3: How were they evaluated?
  Section 4.2 summarizes the selected papers, reporting which RL methods were employed, such as Q-learning, DQN, PPO, A2C, REINFORCE, and SARSA, applied in areas like metal, aircraft sortie, and simulators, using makespan, tardiness, and energy consumption as function objectives. Some works compare the RL with metaheuristics and scheduling rules. There are also scientific works that combine RL with other methods.
- RQ4: Exists an improvement obtained by using RL in HFS?
  RL has the potential to surpass conventional metaheuristic approaches due to its ability to understand interdependencies related to problems [62]. This capability allows you to generate comparable or even superior HFS solutions.
  RL's adaptability and learning capabilities are advantageous in complex and dynamic environments because they can treat uncertain events [62].
  RL is also designed to strike a balance between exploration and exploitation. It can explore new strategies to discover solutions while leveraging the knowledge gained to improve performance, avoiding local optimum, a recurring challenge metaheuristics face.

6 Conclusion and future perspectives

In this paper, we performed a Systematic Literature Review focusing on applying Reinforcement Learning to solve the Hybrid Flow-Shop Scheduling. The literature suggests that HFS is a widely studied problem, but applying RL is an incipient approach. These algorithms have improved scheduling in various scenarios, such as metal processing workshops and aircraft sortie scheduling. The application of RL, particularly methods like DQN and PPO, has shown significant potential in outperforming traditional scheduling rules and metaheuristic. These methods excel at learning optimal policies from real-world data and adapting to dynamic environments, effectively minimizing total tardiness, makespan, and others. However, the HFS problem is NP-hard, characterized by a complex search space involving multiple stages, parallel machines, and variable processing times. This complexity, coupled with the stochastic nature of processing times influenced by factors such as worker fatigue and machine breakdowns, (for example), can limit the performance of RL algorithms, potentially leading to suboptimal solutions, as mentioned by [60]. It can also be a problem for scheduling heuristic and metaheuristic. Some papers propose multi-agent RL approaches for HFS, especially in complex production environments. Furthermore, there is a trend toward integrating RL with metaheuristic, such as TLBO, GA, and SFLA. This combination can lead to promising results, including a new way to dynamically adjust algorithm hyperparameters for solving complex scheduling problems. The findings suggest several potential avenues for future research. These include investigating the impact of additional constraints, multiple objectives, and uncertainties in scheduling problems. There is also a focus on exploring the potential of different RL algorithms and their integration with metaheuristics in production scheduling.

Acknowledgements

The authors would like to thank the National Council for Scientific and Technological Development (CNPq) for granting a scholarship to Victor Pugliese through the Academic Master's and Doctorate Program in Innovation (MAI/DAI) in collaboration with the EMBRAER S.A. company.

References

[1] 2016. Scheduling: Theory, algorithms, and systems. (2016).
[2] Jingcao Cai and Lei Wang. 2024. A Shuffled Frog Leaping Algorithm with Q-Learning for Distributed Hybrid Flow Shop Scheduling Problem with Energy-Saving. *Journal of Artificial Intelligence and Soft Computing Research* 14, 2 (2024), 101–120.
[3] Weiwei Cui and Biao Yuan. 2024. A hybrid genetic algorithm based on reinforcement learning for the energy-aware production scheduling in the photovoltaic glass industry. *Computers & Operations Research* 163 (2024), 106521.
[4] Leonardo de Lima Corrêa and Márcio Dorn. 2020. A multi-population memetic algorithm for the 3-D protein structure prediction problem. *Swarm and Evolutionary Computation* 55 (2020), 100677.
[5] Constantin Waubert de Puiseau et al. 2022. On reliability of reinforcement learning based production scheduling systems: a comparative survey. *Journal of Intelligent Manufacturing* 33, 4 (2022), 911–927.
[6] Hamilton Emmons and George Vairaktarakis. 2012. *Flow shop scheduling: theoretical results, algorithms, and applications*. Vol. 182. Springer Science & Business Media.
[7] Andrès Gutierrez et al. 2016. A multi population memetic algorithm for the vehicle routing problem with time windows and stochastic travel and service times. *IFAC-PapersOnLine* 49, 12 (2016), 1204–1209.
[8] Aimin Zhou et al. 2011. Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and evolutionary computation* 1, 1 (2011), 32–49.
[9] Bestan B. Maaroof et al. 2022. Current studies and applications of shuffled frog leaping algorithm: a review. *Archives of Computational Methods in Engineering* 2, 1 (2022), 1–16.
[10] Ching-An Cheng et al. 2021. Heuristic-guided reinforcement learning. *Advances in Neural Information Processing Systems* 34 (2021), 13550–13563.
[11] Chun-Cheng Lin et al. 2024. Reentrant hybrid flow shop scheduling with stockers in automated material handling systems using deep reinforcement learning. *Computers & Industrial Engineering* (2024), 109995.
[12] Duarte Alemão et al. 2021. Smart manufacturing scheduling approaches—Systematic review and future directions. *Applied Sciences* 11,

5 (2021), 2186.
[13] Di Wu et al. 2022. An improved teaching-learning-based optimization algorithm with reinforcement learning strategy for solving optimization problems. *Computational Intelligence and Neuroscience* 2022 (2022).
[14] Ekrem Duman et al. 2012. Migrating birds optimization: a new metaheuristic approach and its performance on quadratic assignment problem. *Information Sciences* 217 (2012), 65–77.
[15] Feng Liu et al. 2009. Immune clonal selection algorithm for hybrid flow-shop scheduling problem. In *2009 Chinese Control and Decision Conference*. 2605–2609.
[16] Fei Ni et al. 2021. A multi-graph attributed reinforcement learning based optimization algorithm for large-scale hybrid flow shop scheduling problem. In *27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 3441–3451.
[17] Falk T. Gerpott et al. 2022. Integration of the A2C algorithm for production scheduling in a two-stage hybrid flow shop environment. *Procedia Computer Science* 200 (2022), 585–594.
[18] G. M. Komaki et al. 2017. Improved discrete cuckoo optimization algorithm for the three-stage assembly flowshop scheduling problem. *Computers & Industrial Engineering* 105 (2017), 158–173.
[19] Guanghui Zhang et al. 2018. Scheduling distributed flowshops with flexible assembly and set-up time to minimise makespan. *Int. Journal of Production Research* 56, 9 (2018), 3226–3244.
[20] Haoxiang Qin et al. 2021. Adapting a reinforcement learning method for the distributed blocking hybrid flow shop scheduling problem. In *Asian Conference on Artificial Intelligence Technology*. 751–757.
[21] I. Maciel et al. 2022. A hybrid genetic algorithm for the hybrid flow shop scheduling problem with machine blocking and sequence-dependent setup times. *Journal of Project Management* 7, 4 (2022), 201–216.
[22] Jingru Chang et al. 2022. Deep reinforcement learning for dynamic flexible job shop scheduling with random job arrival. *Processes* 10, 4 (2022), 760.
[23] Jingcao Cai et al. 2023. A novel shuffled frog-leaping algorithm with reinforcement learning for distributed assembly hybrid flow shop scheduling. *Int. Journal of Production Research* 61, 4 (2023), 1233–1251.
[24] Jin Deng et al. 2016. A competitive memetic algorithm for the distributed two-stage assembly flow-shop scheduling problem. *Int. Journal of Production Research* 54, 12 (2016), 3561–3577.
[25] Jing Luo et al. 2023. Deep reinforcement learning for solving hybrid flow shop scheduling problem with unrelated parallel machines. In *Int. Conference on Intelligent Computing and Signal Processing*. 1642–1645.
[26] John Schulman et al. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).
[27] Junzi Zhang et al. 2021. Sample efficient reinforcement learning with REINFORCE. In *AAAI conference on artificial intelligence*, Vol. 35. 10887–10895.
[28] Kumara Sastry et al. 2005. Genetic algorithms. *Search methodologies: Introductory tutorials in optimization and decision support techniques* (2005), 97–125.
[29] Kaishu Xia et al. 2021. A digital twin to train deep reinforcement learning agent for smart manufacturing plants: Environment, interfaces and intelligence. *Journal of Manufacturing Systems* 58 (2021), 210–230.
[30] Ke Xu et al. 2023. Reinforcement Learning-Based Multi-Objective of Two-Stage Blocking Hybrid Flow Shop Scheduling Problem. *Processes* 12, 1 (2023), 51.
[31] Libao Deng et al. 2023. A Reinforcement-Learning-Based 3-D Estimation of Distribution Algorithm for Fuzzy Distributed Hybrid Flow-Shop Scheduling Considering On-Time-Delivery. *IEEE Transactions on Cybernetics* (2023).
[32] Leslie Kaelbling et al. 1996. Reinforcement learning: A survey. *Journal of artificial intelligence research* 4 (1996), 237–285.
[33] Libing Wang et al. 2021. Dynamic job-shop scheduling in smart manufacturing using deep reinforcement learning. *Computer Networks* 190 (2021), 107969.
[34] Milad Ramezankhani et al. 2021. Making costly manufacturing smart with transfer learning under limited data: A case study on composites autoclave processing. *Journal of Manufacturing Systems* 59 (2021), 345–354.
[35] Ming Wang et al. 2022. Independent double DQN-based multi-agent reinforcement learning approach for online two-stage hybrid flow shop scheduling with batch machines. *Journal of Manufacturing Systems* 65 (2022), 694–708.
[36] Nianmin Zhang et al. 2023. Counterfactual-attention multi-agent reinforcement learning for joint condition-based maintenance and production scheduling. *Journal of Manufacturing Systems* 71 (2023), 70–81.
[37] Ömür Tosun et al. 2020. A literature review on hybrid flow shop scheduling. *Int. Journal of Advanced Operations Management* 12, 2 (2020), 156–194.
[38] Qingpeng Cai et al. 2019. Reinforcement learning driven heuristic optimization. *arXiv preprint arXiv:1906.06639* (2019).
[39] Rui Li et al. 2023. Double dqn-based coevolution for green distributed heterogeneous hybrid flowshop scheduling with multiple priorities of jobs. *IEEE Transactions on Automation Science and Engineering* (2023).
[40] Rubén Ruiz et al. 2010. The hybrid flow shop scheduling problem. *European journal of operational research* 205, 1 (2010), 1–18.
[41] R. Venkata Rao et al. 2011. Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. *Computer-aided design* 43, 3 (2011), 303–315.
[42] Shengyi Huang et al. 2022. A2C is a special case of PPO. *arXiv preprint arXiv:2205.09123* (2022).
[43] Sayak Roychowdhury et al. 2017. A genetic algorithm with an earliest due date encoding for scheduling automotive stamping operations. *Computers & Industrial Engineering* 105 (2017), 201–209.
[44] Volodymyr Mnih et al. 2013. Playing Atari with Deep Reinforcement Learning. arXiv:1312.5602 [cs.LG]
[45] Volodymyr Mnih et al. 2015. Human-level control through deep reinforcement learning. *nature* 518, 7540 (2015), 529–533.
[46] Volodymyr Mnih et al. 2016. Asynchronous methods for deep reinforcement learning. In *Int. conference on machine learning*. PMLR, 1928–1937.
[47] Xinrong Wang et al. 2022. DQN-based online scheduling algorithm for hybrid flow shop to minimize the total tardiness. In *Int. Symposium on Computational Intelligence and Design*. 66–69.
[48] Yanhe Jia et al. 2023. Q-learning driven multi-population memetic algorithm for distributed three-stage assembly hybrid flow shop scheduling with flexible preventive maintenance. *Expert Systems with Applications* (2023), 120837.
[49] Youshan Liu et al. 2023. Agent-based simulation and optimization of hybrid flow shop considering multi-skilled workers and fatigue factors. *Robotics and Computer-Integrated Manufacturing* 80 (2023), 102478.
[50] Youshan Liu et al. 2023. Integration of deep reinforcement learning and multi-agent system for dynamic scheduling of re-entrant hybrid flow shop considering worker fatigue and skill levels. *Robotics and Computer-Integrated Manufacturing* 84 (2023), 102605.
[51] Yanjun Xiao et al. 2024. Study on flexible job shop scheduling problem considering energy saving. *Journal of Intelligent & Fuzzy Systems* (2024), 1–28.
[52] Yejian Zhao et al. 2024. The application of heterogeneous graph neural network and deep reinforcement learning in hybrid flow shop scheduling problem. *Computers & Industrial Engineering* 187 (2024), 109802.
[53] Zhongyuan Liang et al. 2022. A computational efficient optimization of flow shop scheduling problems. *Scientific Reports* 12, 1 (2022), 845.
[54] Zihui Luo et al. 2023. Deep Reinforcement Learning Based Production Scheduling in Industrial Internet of Things. *IEEE Internet of Things Journal* (2023).
[55] Zeyu Zhang et al. 2024. MRLM: A meta-reinforcement learning-based metaheuristic for hybrid flow-shop scheduling problem with learning and forgetting effects. *Swarm and Evolutionary Computation* 85 (2024), 101479.
[56] Hadi Gholami and Hongyang Sun. 2023. Toward automated algorithm configuration for distributed hybrid flow shop scheduling with multiprocessor tasks. *Knowledge-Based Systems* 264 (2023), 110309.
[57] Chang-Bae Gil and Jee-Hyong Lee. 2022. Deep Reinforcement Learning Approach for Material Scheduling Considering High-Dimensional Environment of Hybrid Flow-Shop Problem. *Applied Sciences* 12, 18 (2022), 9332.
[58] Fang Guo, Yongqiang Li, Ao Liu, and Zhan Liu. 2020. A reinforcement learning method to scheduling problem of steel production process. In *Journal of Physics: Conference Series*, Vol. 1486. IOP Publishing, 072035.
[59] Ricardo Luna Gutierrez and Matteo Leonetti. 2021. Meta Reinforcement Learning for Heuristic Planing. In *Int. Conference on Automated Planning and Scheduling*, Vol. 31. 551–559.
[60] Wei Han, Fang Guo, and Xichao Su. 2019. A reinforcement learning method for a hybrid flow-shop scheduling problem. *Algorithms* 12, 11 (2019), 222.
[61] FrameWork Keras. 2022. *PPO Proximal Policy Optimization.* https://keras.io/examples/rl/ppo_cartpole/
[62] Matthias Klar, Moritz Glatt, and Jan C Aurich. 2023. Performance comparison of reinforcement learning and metaheuristics for factory layout planning. *CIRP Journal of Manufacturing Science and Technology* 45 (2023), 10–25.
[63] Maxim Lapan. 2018. *Deep Reinforcement Learning Hands-On: Apply modern RL methods, with deep Q-networks, value iteration, policy gradients, TRPO, AlphaGo Zero and more.* Packt Publishing Ltd.
[64] Rigoberto Sáenz Imbacuán. 2020. Evaluating the impact of curriculum learning on the training process for an intelligent agent in a video game. (2020).
[65] Miloš Šeda. 2007. Mathematical models of flow shop and job shop scheduling problems. *Int. Journal of Physical and Mathematical Sciences* 1, 7 (2007), 307–312.
[66] Thomas Stützle and Rubén Ruiz. 2018. Iterated Greedy. *Handbook of heuristics* (2018), 547–577.
[67] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction.* MIT press.
[68] Jing-Jing Wang and Ling Wang. 2021. A cooperative memetic algorithm with learning-based agent for energy-aware distributed hybrid flow-shop scheduling. *IEEE Transactions on Evolutionary Computation* 26, 3 (2021), 461–475.
[69] Bingjie Xi and Deming Lei. 2022. Q-learning-based teaching-learning optimization for distributed two-stage hybrid flow shop scheduling with fuzzy processing time. *Complex System Modeling and Simulation* 2, 2 (2022), 113–129.
[70] Xin-She Yang. 2020. *Nature-inspired optimization algorithms.* Academic Press.
[71] Lv Zhong. 2024. Comparison of Q-learning and SARSA Reinforcement Learning Models on Cliff Walking Problem. In *Int. Conference on Data Science, Advanced Algorithm and Intelligent Computing*. Atlantis Press, 207–213.