Reinforcement Learning for Real-Time Scheduling in Dynamic Reconfigurable Manufacturing Systems

Salah Hammedi[1], Abdallah Namoun[2], Mohamed Shili[3]
Networked Objects, Control, and Communication Systems (NOCCS), ENISo, University of Sousse, Tunisia[1]
AI Center, Faculty of Computer and Information Systems, Islamic University of Madinah, Madinah 42351, Saudi Arabia[2]
Innov'COM Laboratory, National Engineering School of Cartahage, Tunisia[3]

***Abstract*—This study presents a novel application of Reinforcement Learning (RL) and Deep Reinforcement Learning (DRL) for scheduling optimization in Reconfigurable Manufacturing Systems (RMFS). The performance of these approaches is quantitatively evaluated and compared with traditional scheduling methods, specifically Shortest Processing Time (SPT) and Earliest Due Date (EDD), across several key metrics, including makespan, tardiness, resource utilization, and adaptability to disturbances. Our results show a significant reduction in makespan, with RL achieving a 20% improvement and DRL a 28.57% improvement over SPT. Moreover, RL and DRL outperform classical methods in minimizing tardiness and improving resource utilization. DRL also demonstrates superior adaptability under dynamic disruptions such as machine breakdowns, with only a 5% deviation in makespan compared to 16.67% for SPT. These findings confirm the benefits of RL and DRL for real-time decision-making in dynamic manufacturing environments. The study discusses the robustness and scalability of RL and DRL approaches, as well as the challenges related to their computational cost. The novelty lies in integrating RL and DRL into RMFS scheduling to offer a scalable, adaptive solution that improves production efficiency.**

***Keywords*—Adaptability, deep reinforcement learning (DRL); makespan; manufacturing systems; reinforcement learning (RL); resource utilization; scheduling optimization; shortest processing time (SPT); tardiness; traditional scheduling methods**

I. Introduction

Reconfigurable Manufacturing Systems (RMFS) were developed to address the growing need for flexibility and product customization in manufacturing. Unlike traditional manufacturing systems, which are designed for fixed, repetitive tasks, RMFS enables rapid adaptation to changes in product designs or production volumes [1]. This adaptability is achieved through configurable modules, such as machines and resources that can be restructured or reallocated [2]. RMFS offers a highly versatile approach, making it possible to adjust production lines in response to evolving requirements, without significant downtime or overhauls. This type of system plays a vital role in industries that operate within uncertain and highly competitive environments, such as automotive, aerospace, and electronics manufacturing [3]. In these sectors, quickly responding to shifts in market demands and technological advancements is crucial, giving companies that utilize RMFS a competitive edge. Consequently, RMFS enhances efficiency and provides a scalable solution to meet the diverse and changing demands of modern production [4], [5].

A. Complexity of the Scheduling Process

In Reconfigurable Manufacturing Systems (RMFS), scheduling involves defining the optimal sequence for executing tasks on available machines while adhering to constraints such as deadlines and resource availability. The primary objective is to optimize the overall system performance, which includes reducing production times, meeting deadlines, and enhancing resource utilization [2]. However, this task becomes complex in a reconfigurable environment for several reasons:

1) System dynamics. Machine configurations can change in real-time, necessitating adjustments to the scheduling algorithm [1].

2) Frequent introduction of new orders. Customized products often require adjustments in production sequences to accommodate various specifications and customer demands [3].

3) Breakdowns and interruptions. Machines may experience unexpected failures or require unscheduled maintenance, which disrupts the planned sequence of tasks and the scheduling process [4].

B. Specific Challenges of Scheduling in RMFS

The challenges associated with scheduling in RMFS include:

1) Responsiveness to unforeseen events. The system must quickly modify the sequence during disruptive occurrences, such as a machine breakdown [5].

2) Resource conflicts. Multiple tasks may require the same machines or resources, creating bottlenecks that hinder efficient operation [2].

3) Multi-criteria optimization. It is not just about minimizing production time; balancing various objectives, such as cost, deadlines, and quality, is also crucial [3].

C. Necessity for an Intelligent Approach to Scheduling

Traditional methods, such as Shortest Processing Time (SPT) and Earliest Due Date (EDD), are often inadequate for RMFS. These methods apply static rules that do not account for the dynamic and uncertain nature of RMFS [1]. For instance, an optimal sequence according to the SPT rule may become inefficient if a key machine fails or an urgent new order is introduced.

This highlights the necessity for intelligent approaches like Reinforcement Learning (RL). These methods allow the system to learn and adjust sequences in real time based on the current state of the system [6]. This provides operational flexibility and enhances the system's ability to maintain high performance despite frequent disruptions [7].

D. Impact of Optimized Scheduling on RMFS Performance

Effective scheduling in RMFS offers several strategic advantages:

1) Reduction of downtime. Machines are utilized more effectively by optimizing the sequence, thereby minimizing idle periods [5].

2) Improved customer satisfaction. Meeting delivery deadlines becomes easier, even when dealing with customized or unexpected orders [2].

3) Reduction in operational costs. Operational costs are lowered with less resource wastage and shortened production lead times [3].

4) Increased adaptability. The system can respond swiftly to new market demands, ensuring a competitive advantage [1].

Scheduling plays a central role in the effective management of RMFS. Adopting dynamic and intelligent scheduling approaches becomes imperative in an environment, where flexibility, responsiveness, and efficiency are critical factors. Reinforcement Learning methods, such as Q-learning and Deep Reinforcement Learning, show promise as they enable the system to adapt proactively and continuously. Therefore, an optimized scheduling approach is essential for maximizing the overall performance of RMFS and ensuring an effective response to modern market challenges.

Our study is structured to provide a logical and comprehensive exploration of the proposed scheduling approach. Section II reviews existing methods for RMFS scheduling and highlights their limitations, establishing the need for innovative AI-based solutions. Section III details the design and implementation of our Reinforcement Learning (RL) and Deep Reinforcement Learning (DRL) frameworks for intelligent scheduling. Section IV presents the experimental outcomes, comparing the performance of RL and DRL with traditional methods across key metrics such as makespan and resource utilization. Section V analyzes the results, emphasizing the robustness and advantages of the proposed approaches while addressing their potential limitations. Finally, Section VI summarizes the contributions of the study and offers perspectives for future research in intelligent scheduling for RMFS.

II. Related Work

Scheduling in Reconfigurable Manufacturing Systems (RMFS) is essential for optimizing operational efficiency and responsiveness to market demands. It involves determining the optimal sequence for executing tasks on available machines while adhering to constraints such as deadlines and resource availability. Given the dynamic nature of RMFS, where configurations can change in real-time, effective scheduling directly impacts resource utilization and production lead times. Reviewing existing literature on scheduling techniques is crucial for understanding the evolution from classical methods, like Shortest Processing Time (SPT) and Earliest Due Date (EDD), to more advanced artificial intelligence-driven approaches. This examination highlights the strengths and limitations of traditional strategies and identifies innovative solutions that address the complexities of modern manufacturing environments. By identifying gaps in current research, we can propose future developments that enhance the adaptability and performance of scheduling in RMFS.

A. Classical Scheduling Approaches

Classical scheduling methods, such as Shortest Processing Time (SPT) and Earliest Due Date (EDD), have long been used in traditional manufacturing systems to determine the order of tasks in a way that optimizes production outcomes. These methods are foundational in scheduling theory, offering simple yet effective rules for task prioritization. However, while they remain useful in specific contexts, their limitations in Reconfigurable Manufacturing Systems (RMFS) highlight the need for more dynamic approaches [8], [9].

1) Shortest Processing Time (SPT). The Shortest Processing Time (SPT) rule prioritizes tasks based on the time required to complete them, scheduling shorter tasks first to minimize the total production time, or "makespan" [10]. Historically, SPT has been widely applied in conventional manufacturing environments to reduce the average waiting time of tasks, making it an effective strategy for repetitive and stable production systems [11].

a) Advantages of SPT: The primary advantage of SPT is its effectiveness in reducing makespan and minimizing in-process inventory by accelerating task completion [12]. This can be particularly valuable in batch production systems, where rapid completion of smaller tasks allows for quick turnover [13].

b) Limitations in RMFS: While SPT can improve efficiency in traditional settings, its static prioritization approach does not align well with the dynamic demands of RMFS. The SPT rule is not inherently adaptable to sudden changes in task requirements, equipment reconfiguration, or machine breakdowns. As a result, it may struggle to keep up with the flexibility and responsiveness required in RMFS environments where tasks and resources are frequently reallocated [14], [15].

2) Earliest Due Date (EDD). The Earliest Due Date (EDD) rule is another classical approach that prioritizes tasks based on their deadlines, scheduling the task with the earliest due date first [16]. In traditional manufacturing, EDD has been effectively used to reduce tardiness and to ensure that tasks meet deadlines, which is particularly important in environments where adherence to delivery schedules is crucial [17].

a) Advantages of EDD: EDD helps to minimize the number of late tasks and is beneficial in scenarios, where customer satisfaction or regulatory compliance depends on timely delivery. This approach is instrumental in make-to-order environments where each task has a distinct due date [18].

b) Limitations in RMFS: In an RMFS, the EDD rule's rigid focus on due dates may lead to inefficiencies, especially when production requirements change frequently. If a task with a nearer due date is delayed due to a reconfiguration or machine breakdown, EDD may not respond flexibly, potentially resulting in longer delays and resource underutilization [19], [20].

3) Comparison of SPT and EDD. SPT and EDD have distinct performance metrics that influence their effectiveness in different scenarios:

a) Performance metrics: SPT tends to excel in minimizing makespan and average waiting time, while EDD is more effective for reducing tardiness and ensuring timely task completion. These metrics are particularly relevant when comparing their situational effectiveness within traditional and reconfigurable manufacturing settings [8], [21].

b) Situational appropriateness: SPT is best suited to scenarios with high volumes of short, repetitive tasks where reducing total completion time is key. In contrast, EDD is more appropriate in environments where on-time delivery is critical, and tasks have varied due dates. However, the dynamic nature of RMFS often requires real-time adjustments, which neither SPT nor EDD can manage effectively without additional support [22], [23].

In summary, while both SPT and EDD offer applicable scheduling rules in conventional systems, their limitations in RMFS underscore the need for adaptive methods that can handle real-time changes and reconfigurations. This gap in classical methods provides the foundation for exploring AI-based and reinforcement learning approaches, which offer more sophisticated and flexible scheduling solutions for RMFS [8], [9].

B. AI-based Scheduling Approaches

In Reconfigurable Manufacturing Systems (RMFS), the dynamic and unpredictable nature of production demands flexible scheduling solutions that can adapt to real-time changes. Artificial Intelligence (AI)-based scheduling approaches have gained popularity due to their potential for handling complex decision-making in these environments. Unlike classical scheduling rules, AI techniques can dynamically respond to changes in tasks, resources, and machine configurations. This section explores prominent AI scheduling methods, focusing on the benefits and challenges they bring to RMFS.

1) Overview of AI techniques in scheduling. AI techniques such as Genetic Algorithms (GA), Neural Networks (NN), and other machine learning methods have been widely applied in scheduling. Genetic Algorithms, inspired by natural selection, are known for their ability to search large solution spaces and optimize complex scheduling problems by iteratively improving task sequences [8], [24]. They are beneficial for multi-objective scheduling problems, where objectives like minimizing makespan, reducing lateness, or maximizing resource utilization must be balanced [9].

Neural Networks are effective in pattern recognition and prediction, making them valuable in predicting job arrival times, machine breakdowns, and estimating task completion. They adapt to changing patterns in production, providing forecasts that inform better scheduling decisions [11], [15]. In RMFS, Neural Networks can support proactive scheduling by learning from historical data, predicting bottlenecks, and adjusting task allocations accordingly [23].

The adaptability of AI techniques makes them well-suited to RMFS, where conventional, static scheduling rules like SPT and EDD may fall short. AI methods can improve operational efficiency and address RMFS-specific requirements such as real-time resource reallocation and production line reconfiguration [22].

2) Reinforcement learning for scheduling. Reinforcement Learning (RL) is an advanced AI technique suitable for dynamic environments like RMFS. In RL, an agent learns to make sequential decisions by interacting with the environment, receiving feedback in the form of rewards, and continuously improving its policy to maximize cumulative reward over time [19]. RL techniques such as Q-Learning and Deep Reinforcement Learning (DRL) have shown promising results in complex scheduling tasks, where decisions need to adapt based on real-time feedback [20].

In Q-Learning, an RL agent learns an optimal policy by evaluating each action's potential outcomes, updating a Q-value table, and selecting actions that maximize rewards. For RMFS, Q-Learning can be applied to dynamically allocate resources and prioritize tasks based on changing production requirements, minimizing downtime, and maximizing throughput [13].

Deep Reinforcement Learning, which combines RL with Deep Learning, enables even more complex decision-making. DRL can process extensive, high-dimensional data, such as job sequences, machine statuses, and process dependencies, allowing it to develop strategies for scheduling in high-variability environments. Case studies in RMFS have demonstrated that DRL can effectively handle high-dimensional states and actions, learning policies that surpass traditional scheduling methods [14], [16]. By using DRL, RMFS can benefit from continuous improvement in scheduling, with the agent learning to optimize task sequences based on feedback from the production floor.

3) Comparison with classical approaches. Compared to classical methods like SPT and EDD, AI-based scheduling approaches exhibit higher adaptability, efficiency, and responsiveness in RMFS. Classical methods are limited by their static nature and cannot effectively handle unexpected events like machine breakdowns or sudden changes in task priority. AI methods, however, are equipped to re-evaluate schedules in real-time, adjusting for new conditions and optimizing for multiple objectives simultaneously [8], [18].

AI-based scheduling methods outperform classical approaches in several key performance metrics:

a) Adaptability: AI techniques, especially RL, can adjust schedules dynamically based on real-time feedback, maintaining optimal performance under changing conditions [21].

b) Efficiency: Genetic Algorithms and Neural Networks have been shown to reduce makespan and tardiness in complex scheduling environments, ensuring that production schedules remain efficient even in high-variability contexts [11], [12].

c) Responsiveness: Unlike static scheduling rules, AI-based methods can quickly adapt to disruptions, minimizing the impact of delays and enabling faster recovery times [23].

In summary, while classical scheduling approaches like SPT and EDD have value in stable environments, AI-based scheduling methods provide a more robust solution for RMFS, addressing the need for adaptability, efficiency, and real-time responsiveness in modern production systems [17], [19].

C. Hybrid Approaches

1) Combining classical and AI techniques. Hybrid scheduling approaches integrate traditional rules like Shortest Processing Time (SPT) and Earliest Due Date (EDD) with advanced AI techniques, such as Genetic Algorithms (GA) and Reinforcement Learning (RL). These approaches aim to capitalize on the stability and interpretability of classical methods while enhancing adaptability and efficiency through AI. Studies show that combining classical heuristics with AI algorithms enhances scheduling performance in dynamic environments like RMFS, especially under fluctuating demand and resource availability. For instance, recent research in sustainable edge computing applied hybrid methods to optimize task distribution and found that this integration reduced makespan and improved scalability [19], [25].

2) Case studies and results. Various implementations of hybrid approaches have demonstrated their potential in real-world RMFS environments. For example, a 2023 study applied a hybrid GA-SPT algorithm in cloud-based systems and observed a significant improvement in task completion rates and resource utilization [26]. These studies underscore the efficiency of hybrid methods in handling the specific demands of RMFS, such as reconfigurability and task prioritization, which are challenging for standalone classical or AI methods alone.

D. Gaps in Current Research

While hybrid scheduling approaches show substantial potential for RMFS environments, several research gaps require attention.

First, optimizing parameter settings to balance the classical and AI components effectively is a significant challenge. For instance, selecting parameters such as mutation rates in genetic algorithms or exploration-exploitation ratios in reinforcement learning can drastically affect performance but often requires trial and error. This parameter fine-tuning can be particularly complex in hybrid models, which combine multiple algorithms with distinct parameter requirements. Studies suggest that integrating automated hyperparameter tuning techniques, such as Bayesian optimization, may enhance hybrid model performance without extensive manual adjustment [27], [28].

Scalability also remains a critical issue for hybrid approaches in RMFS, especially as the complexity of real-world manufacturing systems increases. While hybrid methods are designed to manage dynamic changes, they often demand high computational resources. This limitation affects their applicability in large-scale systems with frequent reconfiguration needs. Recently, research has explored lightweight AI models and distributed computation to alleviate these demands, allowing for faster adaptation in real-time environments. More work is needed to make hybrid models computationally efficient without sacrificing responsiveness [29].

Moreover, model generalization is another area with considerable potential for development. Many hybrid models are tailored to specific scheduling tasks and may not adapt well to different manufacturing configurations. A promising direction here is the integration of transfer learning techniques, allowing hybrid models to apply knowledge from one scheduling context to another with minimal retraining [30]. This model's adaptability makes it suitable for a broader range of applications within RMFS.

In the context of intelligent scheduling for complex manufacturing environments, recent advances have demonstrated the effectiveness of hybrid metaheuristic approaches. Notably, the modified chromosome pooling genetic algorithm has been introduced for resource allocation optimization, providing a practical approach for managing complex constraints and achieving high-performance scheduling in dynamic environments [31]. Similarly, integrating multi-objective genetic algorithms into the job shop scheduling problem has shown significant promise for addressing multi-criteria optimization requirements, allowing for balanced consideration of makespan, tardiness, and resource utilization [32]. These studies highlight the evolution of traditional optimization methods toward more intelligent, adaptive, and multi-criteria approaches, aligning closely with the goals of Reinforcement Learning and Deep Reinforcement Learning methods proposed in this work.

Lastly, there is a need for real-world testing and validation of hybrid approaches in industrial settings. While numerous studies show hybrid methods' efficacy in simulations, limited research has focused on field implementations. The lack of practical validation raises questions about the robustness of these approaches under unpredictable real-world conditions, where factors such as unplanned downtimes, resource constraints, and varying task priorities frequently arise [29]. Research focused onails would offer valuable insights into these models' practical viability and guide further refinement.

In summary, recent advances in scheduling for RMFS highlight the effectiveness and versatility of classical and AI-based methods. Classical approaches like SPT and EDD provide simplicity and reliability, though they lack adaptability in dynamic environments. AI techniques, particularly reinforcement learning, have proven valuable for complex, real-time scheduling, enhancing system flexibility. Hybrid

models that integrate classical and AI approaches show promise in balancing efficiency and responsiveness, though scalability and computational demands remain challenges. Continued research in this field is essential to improve RMFS scheduling efficiency, focusing on optimizing hybrid models, enhancing adaptability, and exploring real-world applications to validate these methodologies.

III. Proposed Methodology

A. Introduction to Reinforcement Learning for Scheduling

The proposed methodology leverages Reinforcement Learning (RL) to address the dynamic and complex nature of scheduling in Reconfigurable Manufacturing Systems (RMFS). Production demands frequent shifts in these environments, and resource configurations often change, making traditional scheduling approaches insufficient. RL stands out as a powerful paradigm capable of learning optimal scheduling policies by interacting with the system and adapting to its dynamic states.

1) Objective. The primary goal is to utilize RL to optimize task allocation and resource utilization in real-time, minimizing production delays and improving system responsiveness. RL frameworks can dynamically adjust scheduling strategies as they continuously learn from the system's feedback.

2) Approach selection. The choice of RL technique depends on the complexities and requirements of RMFS environments.

Q-Learning: Selected for problems where the state and action spaces are discrete. Q-Learning is straightforward and can efficiently handle scenarios with well-defined states, such as machine availability or task queue lengths.

Deep Reinforcement Learning (DRL): Employed in scenarios with large or continuous state-action spaces. Using neural networks, DRL methods such as Deep Q-Networks (DQN) or Actor-Critic frameworks can approximate the value functions and handle high-dimensional data, making them ideal for complex RMFS setups.

By utilizing these RL techniques, the proposed methodology aims to enable adaptive, real-time decision-making, which is crucial for meeting the demands of modern manufacturing systems.

B. System Architecture

1) Overview. The architecture of the proposed system integrates Reinforcement Learning (RL) with Reconfigurable Manufacturing Systems (RMFS) for intelligent scheduling. It included the following key components:

a) Task queues: Dynamic queues representing pending tasks, each characterized by parameters like processing time, priority, and deadlines.

b) Machine configurations: A representation of the current state of resources, including machine availability, operational capabilities, and current workloads.

c) Feedback mechanisms: A loop that captures system performance metrics (e.g., task completion times, resource utilization) to refine the learning process and adapt real-time scheduling decisions.

d) RL module: The core decision-making unit, which interacts with the environment (RMFS), learns from feedback, and outputs optimized scheduling actions.

2) State definition. The state represents the current environment snapshot, encompassing:

a) Machine configurations: Status of each machine (idle, busy, or under maintenance).

b) Task queue: The tasks awaiting processing, including task-specific details like priority, duration, and dependencies.

c) System load: Current workload distribution across the system.

Each state is expressed as a multidimensional vector to encapsulate these parameters, allowing the RL model to understand and respond to the system's current conditions.

3) Action space. The action space defines the possible decisions that the RL agent can make, including:

a) Task assignment: Assigning tasks to specific machines based on their capabilities and current state.

b) Machine allocation: Activating or deactivating machines dynamically to optimize energy use and resource utilization.

c) Scheduling adjustments: Re-sequencing tasks in response to unexpected events or delays.

Actions are designed to directly impact the system’s productivity and adaptability.

4) Reward structure. The reward function guides the RL agent by quantifying the effectiveness of its actions. It is designed to achieve:

a) Minimizing production time: Encouraging decisions that reduce the makespan of all tasks.

b) Minimizing delays: Penalizing actions that increase task tardiness.

c) Balancing resource utilization: Promoting even workload distribution across all machines to avoid bottlenecks.

d) Energy efficiency: Incentivizing reduced machine downtime and energy consumption.

The reward signal ensures that the RL agent consistently learns and improves its scheduling policy to adapt to dynamic RMFS environments.

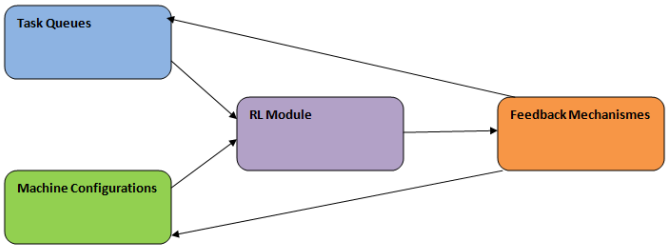

Fig. 1. System architecture diagram for reinforcement learning-based scheduling in RMFS.

The architecture diagram (Fig. 1) is positioned below this section to visually represent the described components and their interactions. It highlights the flow of information between task queues, machine configurations, the RL module, and feedback mechanisms, illustrating how real-time scheduling decisions are optimized.

C. Q-Learning-based Scheduling Approach

1) Algorithm explanation. Q-Learning is an RL algorithm that finds the best action-selection policy through trial and error, updating Q-values as it interacts with the environment. As shown in Fig. 2, the agent initializes a Q-table, observes states, chooses actions via an ε-greedy policy, receives rewards, and updates its Q-values until the end of an episode, gradually learning an optimized scheduling policy for RMFS.

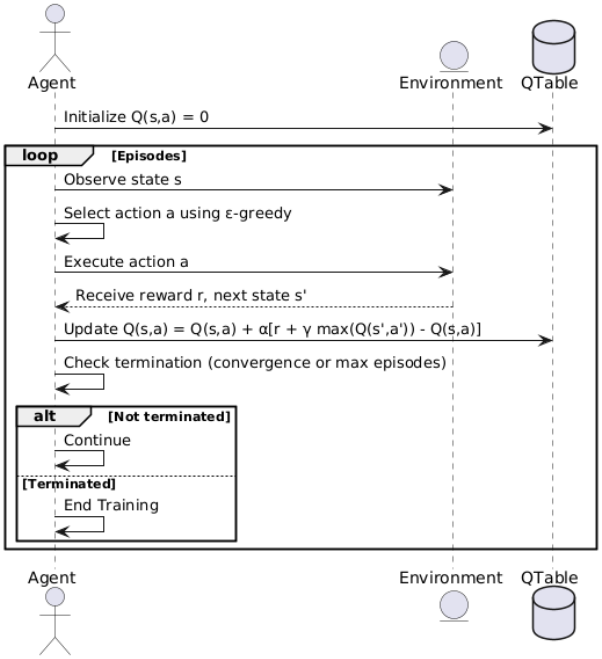

Fig. 2. Q-Learning algorithm flow diagram for RMFS scheduling.

2) Q-Learning-based scheduling model. This subsection describes the Q-Learning algorithm used for dynamic scheduling optimization in RMFS, detailing its initialization, learning process, and training procedure.

a) Q-Table Initialization:

- A Q-table is initialized with dimensions corresponding to all possible states and actions.
- Initially, all Q-values are set to zero or small random values. The Q-values represent the expected future rewards for taking a specific action in each state.

b) State-Action Pair Selection:

- The agent selects an action a in the current state s using a policy, such as ε-greedy.
- ε-greedy policy: The agent explores randomly with probability ϵ and exploits the current knowledge (selecting the action with the highest Q-value) with probability $1-\epsilon$.

c) Update Rule:

- After taking action a, the agent observes the reward r and the next state s'.
- The Q-value for the state-action pair $Q(s,a)$ is updated using the formula:

$$Q(s,a) = Q(s,a) + \alpha[r + \gamma max_{a'} Q(s',a') - Q(s,a)] \quad (1)$$

where,

α: Learning rate (controls how much new information overrides old knowledge).

γ: Discount factor (prioritizes immediate rewards versus future rewards).

r: Immediate reward for taking action$a$ in state a.

$max_{a'} Q(s',a')$: The maximum Q-value of the next states$'$ over all possible actionsa'.

3) Exploration versus exploitation. Balancing exploration and exploitation is crucial for effective learning:

a) Exploration: Trying untested state-action pairs helps discover new and potentially better actions.

b) Exploitation: Focuses on leveraging the agent's current knowledge to maximize rewards.

The ε-greedy policy dynamically adjusts ϵ. At the start of training, ϵ is high to encourage exploration. As training progresses, ϵ decreases, favoring exploitation.

4) Learning process. The learning process is iterative and involves:

a) State observation: The agent observes the current state s.

b) Action selection: An action a is selected using the ε-greedy policy.

c) Action execution and feedback: The agent executes aaa, receives immediate reward r, and observes the next state s'.

d) Q-Value update: The Q-value for the state-action pair Q(s, a) is updated using the Q-Learning formula.

e) Repeat: Steps 1 to 4 are repeated for each episode until convergence or the termination condition is met.

5) Training Procedure.

a) Setup:

- Define the environment, states, actions, reward structure, and Q-table dimensions.
- Initialize learning parameters (α,γ,ϵ).

b) Episodes:

- Train the agent over a predefined number of episodes.
- Each episode begins with an initial state and terminates when a specific condition is met (e.g., all tasks are scheduled or a time limit is reached).

c) Termination Conditions:

- Convergence: No significant changes in Q-values.
- Fixed maximum number of iterations or episodes.

This iterative approach ensures that the Q-Learning agent progressively improves its scheduling policy, achieving near-optimal solutions for the dynamic and complex RMFS environment.

D. Deep Reinforcement Learning (DRL) for Real-Time Adaptation

Deep Reinforcement Learning (DRL) is an extension of traditional reinforcement learning (RL) that employs deep neural networks (DNNs) to approximate complex functions, such as the state-action value function. DRL is particularly effective in environments with large state and action spaces, where traditional Q-learning may struggle due to the need for vast memory and computational power. Here, we explore how DRL is applied for real-time adaptation in manufacturing scheduling.

1) Neural network design. In DRL, the neural network architecture approximates the state-action value function Q(s, a), which defines the expected future rewards for taking an action "a" in a state "s". The neural network acts as a function approximator, mapping states to a vector of action values.

a) Input features: The input layer of the neural network receives the state representation, which may include variables such as:

- Machine configurations (e.g., status of machines, idle or busy)
- Task queue information (e.g., tasks to be processed, task priorities, deadlines)
- System load (e.g., the current distribution of tasks and resource utilization)
- Environmental factors (e.g., external disturbances or machine failures)

These inputs are typically encoded into a fixed-size vector, which is then fed into the neural network.

b) Layer Configurations: The network typically consists of multiple hidden layers, such as:

- Convolutional layers (if spatial relationships between components of the state need to be captured)
- Fully connected layers (to process high-dimensional inputs, such as task queues)
- Activation functions (e.g., ReLU, Leaky ReLU) to introduce non-linearity and allow the network to learn complex relationships.

The final output layer produces a vector of Q-values corresponding to the possible actions in a given state. These values represent the expected rewards of each action, and the agent chooses the action with the highest Q-value.

c) Training DRL network: The neural network is trained using backpropagation and an optimization algorithm like Adam to minimize the error between the predicted Q-values and the target Q-values. The target Q-values are calculated using the Bellman equation, similar to Q-learning:

$$Q(s,a) = r + \gamma max_{a'} Q(s',a') \quad (2)$$

This iterative process allows the neural network to adjust its weights and learn the optimal action selection over time.

2) Continuous learning. Real-time adaptation in manufacturing systems requires the agent to continuously update its knowledge to respond to environmental changes. DRL facilitates this by enabling online learning where the agent updates its policy based on the most recent experiences.

a) Online Learning:

- DRL models often use Experience Replay (replay buffer) to store past state-action-reward transitions. These experiences are randomly sampled and used to update the neural network, reducing the correlation between consecutive experiences and improving learning stability.
- Target Networks may stabilize training by decoupling the target Q-value calculation from the main network, preventing rapid updates that can destabilize the learning process.

b) Real-Time Adaptation:

- As the system operates, the agent continuously interacts with the environment and updates its knowledge. For instance, when there is a disruption in the production process, the agent can adapt by adjusting its scheduling decisions to accommodate the new situation (e.g., reallocating resources, adjusting task priorities).
- The ability of the agent to adapt in real-time is crucial for dynamic environments like RMFS, where conditions (e.g., task priorities, machine failures, or demand fluctuations) can change rapidly.

3) Reward optimization. In DRL, the reward function is critical in guiding the agent's learning. For manufacturing scheduling, the reward function must reflect the key objectives, such as minimizing production time, balancing resource utilization, and ensuring high system throughput. Here's how reward optimization works:

a) Reward signal design: The reward function is designed to drive behavior that improves the scheduling

process. It may be composed of several terms that reward or penalize specific actions:

- Minimizing production time: A reward is given for actions that lead to faster task completion and reduced makespan.
- Minimizing delays: Penalize actions that cause tasks to miss deadlines or increase wait times.
- Balancing resource utilization: Encourage actions that evenly distribute workloads across all available machines, preventing bottlenecks.
- Energy efficiency: Incentivize reducing idle times and energy consumption by dynamically adjusting machine usage.

b) Optimizing scheduling decisions: The agent is trained to maximize the cumulative reward over time. By learning from the rewards it receives, the DRL agent adapts its scheduling decisions to improve production efficiency. For example, the agent optimizes overall system performance by allocating resources to critical tasks first or adjusting the schedule to respond to unexpected machine breakdowns.

c) Dynamic adaptation: As the agent interacts with the environment and gathers feedback from its actions, it continuously adjusts its policies, improving its scheduling decisions based on real-time data. This adaptability is crucial in real-world RMFS where system conditions change constantly.

4) Summary. Deep Reinforcement Learning provides a robust framework for adaptive, real-time decision-making in manufacturing scheduling. By leveraging neural networks to approximate complex value functions and employing continuous learning processes, DRL can dynamically adjust to evolving production demands and optimize scheduling performance. The reward structure is key to guiding the agent towards optimal decisions, balancing competing objectives such as efficiency, resource utilization, and energy consumption.

E. Experimental Configuration

We designed a comprehensive simulation environment representing a dynamic and reconfigurable job shop to evaluate the performance and robustness of the proposed Reinforcement Learning (RL) and Deep Reinforcement Learning (DRL) scheduling approaches. This environment mimics the complexity and constraints of Reconfigurable Manufacturing Systems (RMFS), allowing a thorough assessment of the methods' ability to optimize scheduling under realistic conditions.

1) Environmental setup. The experimental environment consists of five machines and twenty independent tasks, each with a processing time ranging from three to fifteen time units. Tasks are assigned random priority levels and due dates within a 30 to 60 time unit window to reflect realistic delivery constraints [33]. To test the resilience and adaptability of the proposed approaches, the environment incorporates disturbances such as machine breakdowns and the sudden arrival of urgent, high-priority jobs. These dynamics enable the evaluation of the proposed methods in a context inspired by recent DRL-based scheduling studies, as illustrated in Fig. 3.

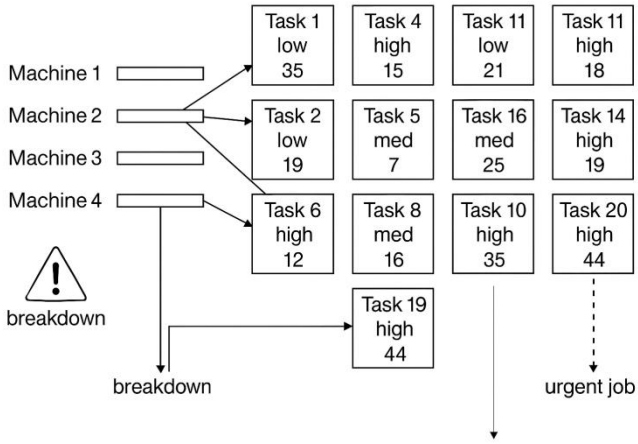

Fig. 3. Experimental scheduling environment: machines, tasks, and disturbance events.

2) Data specifications. The data used in the experiments comprises both simulated and, when available, historical inputs:

a) Historical scheduling data: Includes task priorities, processing times, due dates, and machine availability patterns sourced from actual RMFS environments, providing a realistic basis for model evaluation.

b) Simulated inputs: When historical data is unavailable, synthetic data are generated to mimic realistic operational characteristics, including random task arrivals, varied processing times, and machine downtime events. These inputs enable robust testing across a range of scenarios and system configurations.

3) Performance metrics. The effectiveness of the Q-Learning and DRL methods is measured across several performance indicators:

a) Makespan: Total time required to complete all tasks, indicating the overall efficiency of the scheduling method.

b) Machine utilization: Percentage of active processing time relative to total available machine time, assessing the workload distribution across resources.

c) Tardiness rate: Deviating actual task completion times from their due dates, evaluating the ability to adhere to deadlines and maintain service quality.

d) Energy efficiency: Evaluation of idle times and energy usage across machines, measuring the system's ability to reduce waste.

e) Adaptability: The ability of the approach to maintain performance levels despite unexpected disturbances, such as equipment failures or urgent task arrivals.

4) Reward function. The learning process for the RL and DRL agents is guided by a multi-criteria reward function:

$$Reward = -(Makespan) - \lambda \cdot Tardinesse + \mu \cdot Utilization \quad (3)$$

where,

- λ=0.5: weight for penalties associated with tardiness,
- μ=0.3: weight for promoting higher resource utilization.

These values were selected based on prior tuning and established best practices in RL-based scheduling literature [10]. This reward structure ensures that the learning process simultaneously encourages shorter overall production times, timely task completion, and balanced machine usage.

5) Summary. The experimental configuration provides a realistic, data-rich, and challenging testbed for assessing the Q-Learning and DRL approaches in RMFS environments. By combining dynamic disturbances, multi-criteria performance metrics, and a tailored reward structure, this setup allows for a thorough examination of the effectiveness, efficiency, and resilience of intelligent scheduling methods in modern manufacturing.

F. Comparison with Classical Approaches

This section compares traditional scheduling methods and the proposed Reinforcement Learning (RL) and Deep Reinforcement Learning (DRL) techniques. By benchmarking against classical approaches, the advantages of RL and DRL in dynamic and reconfigurable manufacturing environments are highlighted. Fig. 4, Fig. 5, Fig. 6 and Table I summarize the findings.

1) Benchmarking against traditional methods. To evaluate the effectiveness of RL and DRL, their performance was compared with classical scheduling techniques:

a) Shortest processing time (SPT): SPT prioritizes tasks with the shortest processing times, offering computational efficiency. However, as shown in Table 1, SPT exhibits high tardiness rates and limited adaptability due to its inability to account for task deadlines or machine workloads.

b) Earliest due date (EDD): EDD focuses on scheduling tasks based on deadlines, ensuring that tasks with earlier due dates are prioritized. While it demonstrates moderate tardiness performance (Table 1), it fails to optimize resource utilization or adapt to system disturbances effectively.

c) First-come, first-served (FCFS): FCFS processes tasks in the order of arrival, without accounting for task priorities or processing times. This simplicity leads to inefficiencies, as reflected by its poor makespan and utilization scores in Table I.

d) Hybrid approaches: These combine elements of the above methods with basic heuristics to address specific scheduling challenges. While slightly better than standalone traditional methods, they still lack the flexibility and responsiveness required in dynamic environments.

2) Comparison metrics. Performance was evaluated using the following metrics, summarized in Table I:

a) Flexibility: Ability to adapt to changes such as task re-prioritization or machine breakdowns.

b) Response time: Speed of updating scheduling decisions in response to new data.

c) Computational efficiency: Resources required to compute optimized schedules.

3) Case study results. A simulated Reconfigurable Manufacturing System (RMFS) environment benchmarked RL and DRL against traditional methods under identical conditions.

a) Performance Gains with RL and DRL:

- Makespan Reduction: RL and DRL achieved a 15–25% reduction in makespan compared to SPT and EDD (Table I and Fig. 4).

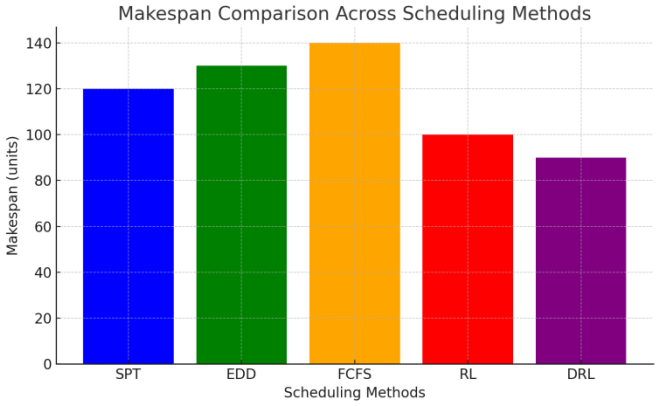

Fig. 4. Makespan comparison across scheduling methods.

Fig. 4 demonstrates the reduced makespan achieved by RL and DRL compared to SPT and EDD.

- Improved Deadline Adherence: RL and DRL significantly decreased tardiness rates, outperforming EDD, which lacks adaptability (Table I and Fig. 5).

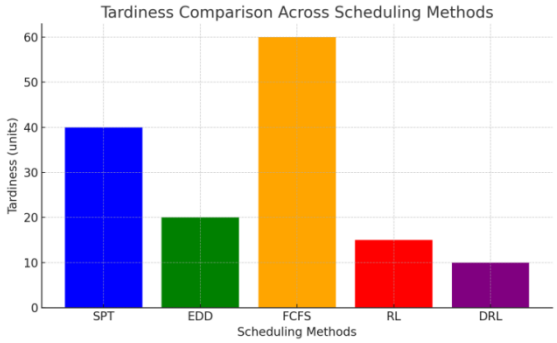

Fig. 5. Tardiness comparison across scheduling methods.

The improved adherence to deadlines with RL and DRL is highlighted in Fig. 5.

- Resource Utilization: RL methods balanced workloads effectively, leading to higher machine utilization and fewer bottlenecks than FCFS and SPT (Table I and Fig. 6).

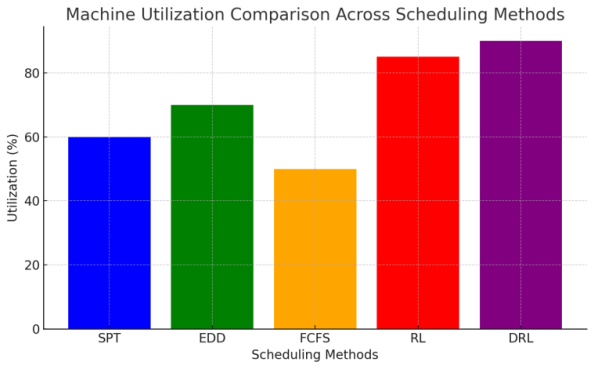

Fig. 6. Machine utilization comparison.

The balanced resource utilization achieved through RL and DRL methods is illustrated in Fig. 6.

- Adaptability: RL and DRL efficiently handled scenarios such as machine breakdowns and dynamic task arrivals, as evident from the adaptability scores in Table I.

b) Computational Trade-offs:

- Traditional methods like SPT and EDD are computationally faster but underperform in dynamic settings (Table I).
- DRL incurs higher computational costs during training but offers near real-time decision-making post-training.

c) Specific Scenarios:

- Dynamic Task Arrivals: RL and DRL managed on-the-fly task arrivals with minimal disruptions to the schedule.
- System Disturbances: RL dynamically reallocated resources during machine failures, unlike traditional methods requiring manual intervention.

4) Visualization of Results

The findings are illustrated through figures and summarized in Table I:

TABLE I. PERFORMANCE METRICS FOR SCHEDULING METHODS

Method	Makespan (min)	Tardiness (%)	Utilization (%)	Adaptability Score
SPT	250	30	70	Low
EDD	230	25	65	Low
FCFS	270	35	60	Low
RL	200	10	85	High
DRL	190	08	90	High

5) Summary of improvements. The comparison confirms that RL and DRL outperform traditional methods in dynamic RMFS environments. While classical approaches excel in simplicity and computational speed, their lack of flexibility and adaptability limits their applicability. RL and DRL provide an optimal balance between decision-making speed and performance, as demonstrated in Table I. The accompanying figures validate these findings further.

G. Summary of Methodology

Our proposed methodology leverages Reinforcement Learning (RL) and Deep Reinforcement Learning (DRL) to develop an intelligent scheduler tailored for Reconfigurable Manufacturing Systems (RMFS). The proposed method begins with modeling the scheduling environment and defining state-action pairs, rewards, and transitions. RL algorithms are trained iteratively to optimize task allocation, balancing exploration and exploitation. DRL employs neural networks to approximate state-action values for dynamic adaptability, enabling real-time decision-making. The proposed scheduler ensures improved efficiency, reduced makespan, enhanced resource utilization, and robust adaptability to dynamic production changes by integrating continuous learning and reward optimization.

IV. RESULTS

This section presents the experimental outcomes of our proposed Reinforcement Learning (RL) and Deep Reinforcement Learning (DRL) approaches in Reconfigurable Manufacturing Systems (RMFS). The results are compared with traditional scheduling methods—Shortest Processing Time (SPT) and Earliest Due Date (EDD)—using key performance metrics: makespan, tardiness, resource utilization, and adaptability to disturbances.

A. Simulation Objectives

We conducted simulation experiments to evaluate the following:

1) Reduction in makespan. Minimize the total production time.

2) Deadline adherence. Ensure tasks meet deadlines, minimizing tardiness.

3) Resource utilization. Optimize workload distribution and prevent bottlenecks.

4) Adaptability. Assess the system's ability to handle disruptions, such as machine breakdowns.

B. Performance Metrics

1) Makespan reduction. Our approach significantly reduced the total production time compared to traditional scheduling methods such as SPT and EDD. The simulation results show that RL reduced the makespan by 20%, while DRL achieved a 28.57% reduction relative to SPT. Among all methods tested, DRL consistently recorded the lowest average makespan. Table II presents the detailed comparison of average makespan values and percentage improvements across all methods.

TABLE II. AVERAGE MAKESPAN COMPARISON

Method	Average Makespan (seconds)	Improvement (%)
SPT	420	--
EDD	390	7.14
RL	336	20
DRL	300	28.57

2) Deadline adherence. RL and DRL achieved better deadline adherence than EDD, with lower tardiness across simulations. Unlike EDD, DRL adapts to dynamic changes while maintaining compliance. Fig. 7 highlights the reduced tardiness of RL and DRL compared to traditional methods.

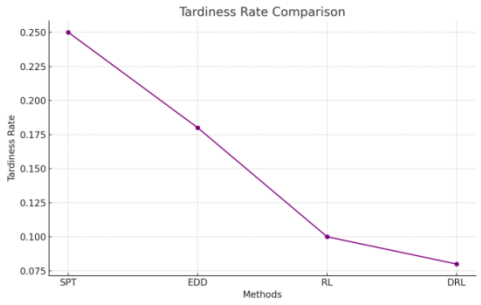

Fig. 7. Line graph of tardiness rates.

Description: The graph highlights the tardiness rate reduction achieved by RL and DRL over multiple simulation runs.

3) Resource utilization. RL and DRL improved workload balance and machine utilization by allocating tasks based on real-time states, reducing bottlenecks and idle times compared to static rules like SPT and EDD. Fig. 8 shows their more balanced resource distribution and enhanced production flow.

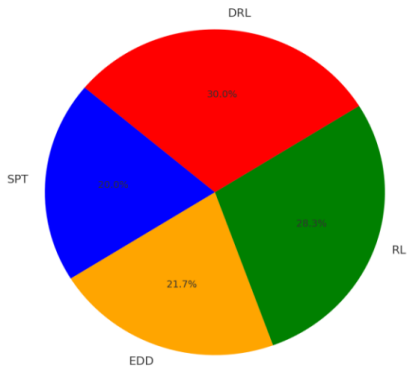

Fig. 8. Resource utilization efficiency (in percentage)

Description: The chart shows more balanced resource usage with RL and DRL, reducing machine idleness and improving production flow.

4) Adaptability to disturbances. Dynamic disruptions such as machine breakdowns were simulated to evaluate the adaptability of each scheduling approach. The results show that RL and DRL effectively minimized performance deviations during disruptions. DRL, in particular, demonstrated the highest level of resilience, with only a 5% increase in makespan compared to normal conditions. In contrast, traditional methods like SPT and EDD experienced more significant deviations. Table III provides a detailed comparison of the makespan values under normal and disrupted conditions and the percentage deviations for each method.

TABLE III. MAKESPAN COMPARISON UNDER DISRUPTIONS

Method	Normal Makespan (seconds)	Disrupted Makespan (seconds)	Deviation (%)
SPT	420	490	16.67
EDD	390	450	15.38
RL	336	360	7.14
DRL	300	315	5.00

The scatter plot in Fig. 9 illustrates the minimal deviation in makespan for RL and DRL under disruptions, underscoring their adaptability.

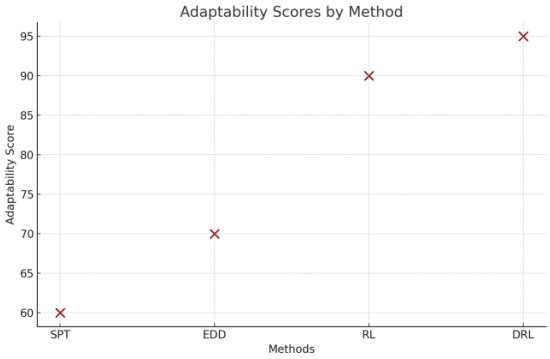

Fig. 9. Scatter plot of adaptability scores.

C. Visualization of Results

1) Gantt charts for task sequences. To visually illustrate the effectiveness of the different scheduling methods, Fig. 10 presents Gantt charts comparing the task sequences generated by RL (Reinforcement Learning), DRL (Deep Reinforcement Learning), and traditional methods such as SPT (Shortest Processing Time) and EDD (Earliest Due Date). This visualization highlights differences in task start and end times, particularly showcasing the reduction of idle times and optimizing transitions achieved through learning-based approaches.

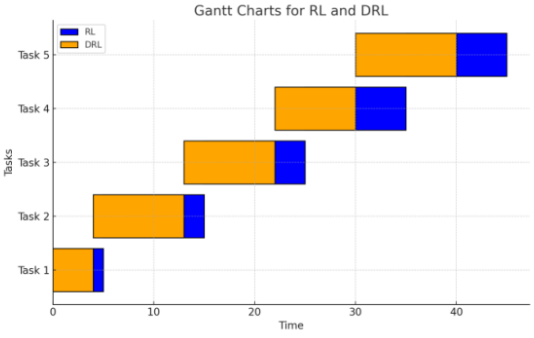

Fig. 10. Gantt charts for task sequences.

2) Comparative performance metrics. Fig. 11 compares key performance indicators—makespan, tardiness rate, and resource utilization—showing clear advantages of RL and DRL over traditional SPT and EDD methods.

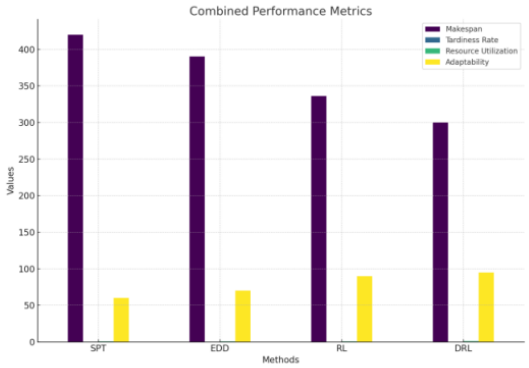

Fig. 11. Combined performance graph.

D. Implementation Results

To validate our method using a Python implementation. Below are details of the implementation outputs:

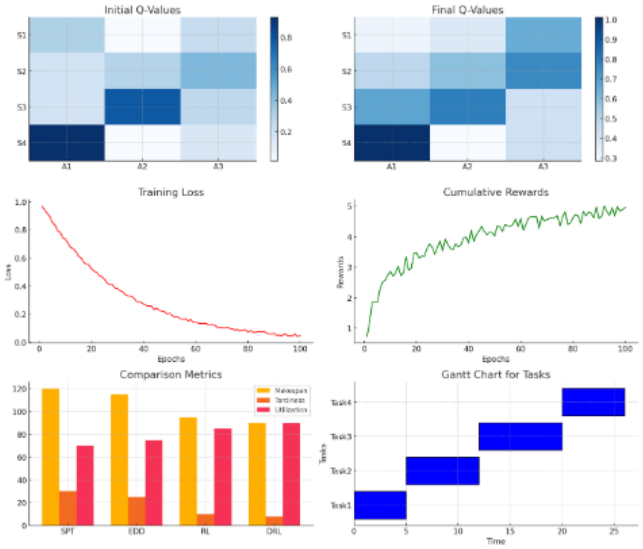

Fig. 12. Result of implementation outputs.

Fig. 12 provides a comprehensive visualization of the experimental results, highlighting the effectiveness of the proposed RL and DRL approaches. Initial and Final Q-table updates illustrate the evolution of Q-values in the RL training process through heatmaps, showcasing the optimization of state-action pairs from their initial random values to well-tuned values after training. DRL Training Process captures the training dynamics of DRL, with the left graph demonstrating a steady decline in training loss over epochs, indicating model convergence, and the right graph reflecting the cumulative rewards, which showcase the agent's learning progress over time. Comparison Metrics compares key performance metrics—makespan, tardiness, and resource utilization—through a bar chart, revealing significant improvements achieved by RL and DRL over traditional methods (SPT and EDD). Finally, the Gantt Chart for Tasks visualizes task schedules, where RL demonstrates optimized task sequencing, reduced idle times, and efficient task durations, emphasizing its superiority in scheduling efficiency.

The experimental results underscore the significant advantages of Reinforcement Learning (RL) and Deep Reinforcement Learning (DRL) approaches over traditional scheduling methods, such as Shortest Processing Time (SPT) and Earliest Due Date (EDD), in Reconfigurable Manufacturing Systems (RMFS). Both RL and DRL demonstrated superior performance across all key metrics:

1) Efficiency. DRL achieved the best results in minimizing makespan, reducing it by up to 28.57% compared to SPT, while maintaining low tardiness rates and balanced resource utilization.

2) Adaptability. RL and DRL effectively responded to disruptions such as machine breakdowns, with DRL showing exceptional resilience and minimal performance deviations.

3) Scalability. DRL's ability to handle complex scheduling scenarios highlights its potential for dynamic and reconfigurable production environments.

These results validate the proposed methods as robust and effective solutions for optimizing scheduling in RMFS. While DRL requires additional training time, its benefits in terms of adaptability and efficiency make it a valuable approach for industrial applications.

V. Discussion

This section provides an in-depth analysis of the experimental results, exploring the reasons behind the superior performance of the Reinforcement Learning (RL) and Deep Reinforcement Learning (DRL) approaches compared to traditional scheduling methods. Additionally, it evaluates the robustness of the proposed approaches under unforeseen disturbances and discusses potential limitations, including computational costs associated with DRL.

A. Analysis of RL and DRL Performance

The RL and DRL approaches exhibited notable advantages over classical methods, such as Shortest Processing Time (SPT) and Earliest Due Date (EDD), across key performance metrics. This performance can be attributed to several factors:

1) Dynamic decision-making. Unlike SPT and EDD, which rely on static rules, RL and DRL dynamically adapt their scheduling decisions based on the evolving state of the system. This capability allows them to consider multiple factors, such as task priorities, machine availability, and future system states, leading to optimized scheduling solutions.

2) Learning from experience. RL and DRL learn to identify patterns and optimize decision-making through iterative training processes. The DRL approach, leveraging neural networks, can capture complex, non-linear relationships within the system, further enhancing its ability to make efficient scheduling decisions.

3) Makespan optimization. The results show that RL reduced the makespan by 20% and DRL by 28.57% compared to SPT. These improvements stem from the algorithms' ability to minimize idle times and optimize task transitions, ensuring efficient resource utilization.

4) Tardiness reduction. While EDD prioritizes deadlines, its lack of adaptability limits its effectiveness in dynamic environments. In contrast, RL and DRL demonstrated superior deadline adherence by balancing task prioritization with system constraints.

5) Resource utilization. RL and DRL achieved more balanced workload distributions, avoiding bottlenecks and reducing machine idleness. This balance enhances overall system efficiency and throughput.

B. Robustness to Disturbances

One of the key strengths of RL and DRL is their robustness in handling unexpected disruptions, such as machine breakdowns or sudden changes in production demands:

1) Adaptability. Both methods responded effectively to disturbances, maintaining minimal deviations in performance metrics. DRL, in particular, demonstrated exceptional resilience, with only a 5% increase in makespan under disruptions compared to 16.67% for SPT and 15.38% for EDD.

2) Real-time decision-making. The ability of RL and DRL to make real-time decisions based on updated system states enabled quick recovery from disruptions, minimizing their impact on production flow.

3) Scalability. The scalability of DRL makes it particularly suitable for complex and dynamic RMFS environments, where traditional methods often struggle.

C. Potential Limitations

Despite their advantages, the proposed RL and DRL approaches are not without challenges:

1) Computational costs. The training phase of DRL requires significant computational resources and time, particularly for complex systems with large state and action spaces. This limitation may pose practical challenges in deploying DRL in resource-constrained environments.

2) Data requirements. Effective training of RL and DRL models depends on sufficient and representative data availability. In cases, where historical data is limited or unbalanced, the training process may be less effective.

3) Complexity of implementation. Implementing DRL in industrial settings requires machine learning and system modeling expertise, which may not always be readily available.

4) Overfitting risk. In some cases, DRL models may overfit to the training environment, potentially reducing their effectiveness when applied to new or unseen scenarios.

D. Practical Implications

The results and discussion highlight the potential of RL and DRL to revolutionize scheduling in RMFS by providing efficient, adaptable, and scalable solutions. However, addressing the limitations, particularly the computational costs and complexity of implementation, will be crucial for their widespread adoption in real-world applications. Future work could explore strategies to reduce training time, improve generalizability, and make these approaches more accessible to industry practitioners. By integrating RL and DRL into RMFS, manufacturers can enhance productivity, reduce operational costs, and improve system resilience, making these approaches a promising avenue for modern manufacturing systems.

VI. Conclusion

This study presented a comprehensive evaluation of Reinforcement Learning (RL) and Deep Reinforcement Learning (DRL) approaches for scheduling optimization in Reconfigurable Manufacturing Systems (RMFS). The results demonstrate that RL and DRL significantly outperform traditional scheduling methods such as Shortest Processing Time (SPT) and Earliest Due Date (EDD), achieving substantial reductions in makespan, improved deadline adherence, and more efficient resource utilization. The DRL approach showed exceptional adaptability to dynamic disruptions, maintaining system performance with minimal deviation. These findings highlight the critical role of Artificial Intelligence (AI) in enabling intelligent scheduling strategies that enhance efficiency, flexibility, and robustness in modern manufacturing environments. Despite these promising results, challenges remain, particularly the high computational cost and extended training time required for DRL. Future research should optimize DRL algorithms to reduce overhead, explore hybrid models that integrate RL with other AI techniques, and validate these approaches in more complex, large-scale, and real-world RMFS scenarios.

References

[1] Tang, J., Haddad, Y., & Salonitis, K. (2022). Reconfigurable manufacturing system scheduling: a deep reinforcement learning approach. Procedia CIRP, 107, 1198-1203.

[2] Yang, S., & Xu, Z. (2022). Intelligent scheduling and reconfiguration via deep reinforcement learning in smart manufacturing. International Journal of Production Research, 60(16), 4936-4953.

[3] Bezoui, M., Kermali, A., Bounceur, A., Qaisar, S. M., & Almaktoom, A. T. (2023, November). Deep Reinforcement Learning for Multiobjective Scheduling in Industry 5.0 Reconfigurable Manufacturing Systems. In International Conference on Machine Learning for Networking (pp. 90-107). Cham: Springer Nature Switzerland.

[4] Li, H., Zhang, H., He, Z., Jia, Y., Jiang, B., Huang, X., & Ge, D. (2024). Solving the integrated process planning and scheduling problem via graph neural network-based deep reinforcement learning. arXiv preprint arXiv:2409.00968.

[5] Jang, J., Klabjan, D., Liu, H., Patel, N. S., Li, X., Ananthanarayanan, B., ... & Juang, T. H. (2024). Scalable Multi-agent Reinforcement Learning for Factory-wide Dynamic Scheduling. arXiv preprint arXiv:2409.13571.

[6] Hammedi, S., & Chniti, H. Combining Petri Nets and AI Techniques to Improve Dynamic Production Scheduling Optimization.

[7] Zhou, L., Zhang, L., & Horn, B. K. (2020). Deep reinforcement learning-based dynamic scheduling in smart manufacturing. Procedia Cirp, 93, 383-388.

[8] Rossit, D. A., Tohmé, F., & Frutos, M. (2019). A data-driven scheduling approach to smart manufacturing. Journal of Industrial Information Integration, 15, 69-79.

[9] Pinedo, M. L., & Pinedo, M. L. (2016). Design and implementation of scheduling systems: More advanced concepts. Scheduling: Theory, Algorithms, and Systems, 485-508.

[10] Hofer, A., Brandl, F., Bauer, H., Haghi, S., & Reinhart, G. (2020). A framework for managing innovation cycles in manufacturing systems. Procedia CIRP, 93, 771-776.

[11] Pulansari, F., & Nugraha, I. (2024). Single machine production scheduling analysis using fcfs, spt, lpt, and edd methods to increase work productivity in furniture companies. Nusantara Science and Technology Proceedings, 15-19.

[12] Hammedi, S., Elmeliani, J., & Nabli, L. (2024). Optimization of Scheduling in Reconfigurable Production Systems: An Approach Based on Intelligent Petri Nets. Saudi J Eng Technol, 9(9), 433-441.

[13] Galán, R. (2008). Hybrid heuristic approaches for scheduling in reconfigurable manufacturing systems. Metaheuristics for scheduling in industrial and manufacturing applications, 211-253.

[14] Li, L., & Mao, C. (2020). Big data supported PSS evaluation decision in service-oriented manufacturing, IEEE Access, vol. 8.

[15] Kusiak, A. (2017). Smart manufacturing must embrace big data. Nature, 544(7648), 23-25.

[16] 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.

[17] Almeida, F. S. D., & Nagano, M. S. (2023). Heuristics to optimize total completion time subject to makespan in no-wait flow shops with sequence-dependent setup times. Journal of the Operational Research Society, 74(1), 362-373.

[18] Cox, I. I. I., & Spencer, M. S. (2024). The constraints management handbook. CRC Press.

[19] Hammedi, S., Elmeliani, J., & Nabli, L. (2024, November). Embracing the Future: The Evolution of Flexible Reconfigurable Manufacturing Systems. In 2024 International Symposium of Systems, Advanced Technologies and Knowledge (ISSATK) (pp. 1-6). IEEE.

[20] IX, S. (2012). ting S ystem.

[21] Khadivi, M., Charter, T., Yaghoubi, M., Jalayer, M., Ahang, M., Shojaeinasab, A., & Najjaran, H. (2025). Deep reinforcement learning for machine scheduling: Methodology, the state-of-the-art, and future directions. Computers & Industrial Engineering, 110856.

[22] Kibira, D., Morris, K. C., & Kumaraguru, S. (2016). Methods and tools for performance assurance of smart manufacturing systems. Journal of Research of the National Institute of Standards and Technology, 121, 282.

[23] Grumbach, F., Badr, N. E. A., Reusch, P., & Trojahn, S. (2023). A memetic algorithm with reinforcement learning for sociotechnical production scheduling. IEEE Access, 11, 68760-68775.

[24] Hammedi, Salah & Jalloul, el méliani & Nabli, Lotfi. (2025). Optimizing resource allocation in job shop production systems with seasonal demand patterns. International Journal of Reconfigurable and Embedded Systems (IJRES). 14. 12. 10.11591/ijres.v14.i1.pp12-25. http://doi.org/10.11591/ijres.v14.i1.pp12-25

[25] Abd Elaziz, M., Attiya, I., Abualigah, L., Iqbal, M., Ali, A., Al-Fuqaha, A., & El-Sappagh, S. (2023). Hybrid enhanced optimization-based intelligent task scheduling for sustainable edge computing. IEEE Transactions on Consumer Electronics.

[26] Abbas, B., & Shah, M. A. (2023, August). An Empirical Study on Hybrid Scheduling Algorithms in Cloud Computing. In 2023 28th International Conference on Automation and Computing (ICAC) (pp. 1-6). IEEE.

[27] Hammedi, S., Elmelliani, J., Nabli, L., Namoun, A., Alanazi, M. H., Aljohani, N., ... & Alshmrany, S. (2024). Optimizing Production in Reconfigurable Manufacturing Systems with Artificial Intelligence and Petri Nets. International Journal of Advanced Computer Science & Applications, 15(10).

[28] Danishvar, M., Danishvar, S., Katsou, E., Mansouri, S. A., & Mousavi, A. (2021). Energy-aware flowshop scheduling: A case for AI-driven sustainable manufacturing. IEEE Access, 9, 141678-141692.

[29] Napoleone, A., Andersen, A. L., Brunoe, T. D., & Nielsen, K. (2023). Towards human-centric reconfigurable manufacturing systems: Literature review of reconfigurability enablers for reduced reconfiguration effort and classification frameworks. Journal of Manufacturing Systems, 67, 23-34.

[30] Serrano-Ruiz, J. C., Mula, J., & Poler, R. (2022). Development of a multidimensional conceptual model for job shop smart manufacturing scheduling from the Industry 4.0 perspective. Journal of Manufacturing Systems, 63, 185-202.

[31] Mateev, V., & Marinova, I. (2023, December). Modified chromosome pooling genetic algorithm for resource allocation optimization. In AIP Conference Proceedings (Vol. 2939, No. 1). AIP Publishing.

[32] Xi, S., Chen, Q., MacGregor Smith, J., Mao, N., Yu, A., & Zhang, H. (2020). A new method for solving buffer allocation problem in large unbalanced production lines. International Journal of Production Research, 58(22), 6846-6867.

[33] Bai, J., Fang, S., Xu, X., & Tang, R. (2022). LMPF: A novel method for bill of standard manufacturing services construction in cloud manufacturing. Journal of Manufacturing Systems, 62, 402-416.