Deep Reinforcement Learning for multiobjective Scheduling in Industry 5.0 Reconfigurable Manufacturing Systems ⋆

Madani Bezoui, Abdelfatah Kermali, Ahcène Bounceur, Saeed Mian Qaisar

Deep Reinforcement Learning for multiobjective Scheduling in Industry 5.0 Reconfigurable Manufacturing Systems*

Madani Bezoui[1][0000−0001−6930−1088], Abdelfatah Kermali[1][0009−0007−8937−4875], Ahcene Bounceur[2][0000−0002−0043−7742], and Saeed Mian Qaisar[3,4][0000−0002−4268−3482] Abdulaziz Turki Almaktoom[4][0000−0003−4958−5946]

[1] CESI LINEACT, UR 7527, Nice, France
{mbezoui,akermali}@cesi.fr
[2] KFUPM, ICS Department, Dhahran, Saudi Arabia
Ahcene.Bounceur@kfupm.edu.sa
[3] CESI LINEACT, UR 7527, Lyon, France
smianqaisar@cesi.fr
[4] Electrical and Computer Engineering Department, Effat University, 22332, Jeddah, KSA
abalmaktoom@effatuniversity.edu.sa

Abstract. In modern-day manufacturing, it is imperative to react promptly to altering market requirements. Reconfigurable Manufacturing Systems (RMS) are a significant leap forward in achieving this criteria as they offer a flexible and affordable structure to comply with evolving production necessities. The ever-changing nature of RMS demands a sturdy induction of learning algorithms to persistently improve system configurations and scheduling. This study suggests that using Reinforcement Learning (RL), specifically, the Double Deep Q-Network (DDQN) algorithm, is a feasible way to navigate the intricate, multi-objective optimization landscape of RMS. Key points to consider regarding this study include cutting down tardiness costs, ensuring sustainability by reducing wasted liquid and gas emissions during production, optimizing makespan, and improving ergonomics by reducing operator intervention during system reconfiguration. Our proposal consists of two layers. Initially, we suggest a hierarchical and modular architecture for RMS which includes a multi-agent environment at the reconfigurable machine tool level, which improves agent interaction for optimal global results. Secondly, we incorporate DDQN to navigate the multi-objective space in a clever manner, resulting in more efficient and ergonomic reconfiguration and scheduling. The findings indicate that employing RL can help solve intricate optimization issues that come with contemporary manufacturing paradigms, clearing the path for Industry 5.0.

* The authors are thankful to the CESI LINEACT, Effat University,and King Fahd University of Petroleum and Minerals for the technical support. They are also thankful to the Effat University for financially supporting this project under the grant number (UC No. 9/12June2023/7.1-21(4)11).

Keywords: Reconfigurable Manufacturing Systems · Sustainability · Deep Reinforcement Learning · Multiobjective Scheduling · Industry 5.0

1 Introduction

In an era of rapid technological evolution and fluctuating market dynamics, industries, particularly the manufacturing sector, are in a constant search for innovative solutions to meet the escalating demand for product customisation and to skilfully navigate the unpredictability of market trends [10]. In the face of these challenges, Reconfigurable Manufacturing Systems (RMS), a concept introduced by Koren et al. [6], has emerged as a seminal solution. RMS represents a pivotal evolution in manufacturing paradigms, seeking to merge harmoniously the high-throughput features of Dedicated Manufacturing Lines (DML) - characterised by the use of Dedicated Machines (DM) - with the adaptability and responsiveness of Flexible Manufacturing Systems (FMS), characterised by the use of Computer Numerical Control (CNC) machines. This transformation from DML to FMS and finally to RMS represents a profound shift in manufacturing philosophy, where adaptability, modularity and rapid reconfiguration are essential to meet the dynamic demands of today's industrial landscapes [6].

The core challenges associated with RMS are orchestrated around three key stages: design, implementation and optimisation. The design stage provides the fundamental blueprint that defines the architecture of the system, which in turn significantly influences the subsequent implementation and optimisation stages. Crucial decisions regarding system components and their potential reconfigurations are made during this phase [2]. The design phase is followed by the implementation phase, which translates theoretical designs into tangible, operational systems. This transition is fraught with intricacies, encapsulating the integration of diverse system components, ensuring operational coherence, and accommodating practical considerations [7]. Ultimately, the optimisation stage continually refines system operations to increase efficiency and productivity while reducing costs and waste. This stage employs a range of optimisation tools and strategies such as scheduling, machine allocation and production planning, and also requires the consideration of real-time data and prospective reconfigurations [3].

In recent years, the infusion of Artificial Intelligence (AI) and learning methods, particularly Reinforcement Learning (RL), has gained considerable traction in solving complex puzzles in various domains. These technological advancements promise to improve decision making, system adaptation and optimisation, thus addressing some of the critical challenges associated with RMS implementation.

2 Literature Review

The advent of Reconfigurable Manufacturing Systems (RMS) marked a significant milestone towards addressing the dynamic demands of modern manufacturing landscapes. The application of various optimization techniques within

the RMS domain has garnered substantial attention, owing to the potential to address complex scheduling and reconfiguration problems.

Several studies have underscored the application of mathematical programming approaches like mixed-integer linear programming (MILP) in optimizing scheduling and reconfiguration in RMS settings. For instance, Aljuneidi and Bulgak [1] proposed a MILP model for designing reconfigurable cellular manufacturing systems with hybrid manufacturing-remanufacturing capabilities. The model incorporated constraints related to machine capacities, inventory levels, and processing requirements. Khezri et al. [4] developed a multi-objective MILP model to generate sustainable process plans in a reconfigurable context. The model incorporated objectives related to minimizing production cost, time as well as environmental impacts measured through liquid waste generation and greenhouse gas emissions. The model was solved using an augmented epsilon-constraint method. In another study, Khezri et al. [5] proposed a bi-level decomposition approach involving two MINLP models to integrate diagnosability and sustainability considerations in RMS design. The upper-level model focused on preventive maintenance planning to minimize hazardous energy consumption, while the lower-level model concerned sustainable process plan generation by minimizing energy losses. Several studies have also adapted metaheuristic techniques like genetic algorithms, particle swarm optimization, ant colony optimization to effectively solve optimization problems related to scheduling and reconfiguration in RMS settings under complexity constraints.Musharavati and Hamouda [9] developed a simulated annealing algorithm for optimizing process plans in RMS environments. The proposed approach outperformed a standard SA implementation in the comparative analysis. Mohapatra et al. [8] employed an adapted NSGA-II algorithm for integrating process planning and scheduling decisions in an RMS through adaptive setup planning. The objectives of minimizing completion time and cost were simultaneously optimized through the NSGA-II implementation.

The integration of artificial intelligence techniques like reinforcement learning, neural networks and fuzzy logic has also garnered attention for automating and enhancing RMS scheduling and reconfiguration capabilities. For instance, Yang and Xu [12] proposed a multi-agent deep reinforcement learning architecture for optimized scheduling and reconfiguration in smart RMS environments. The model design demonstrated the versatility of deep reinforcement learning for automated decision-making in complex RMS settings. Tang and Salonitis [11] explored a deep reinforcement learning-based scheduling policy tailored for reconfigurable manufacturing systems, elucidating the role of deep reinforcement learning in devising adaptive scheduling policies. Zhou et al. [14] presented a dynamic scheduling approach based on deep reinforcement learning to address scheduling challenges in smart manufacturing contexts enabled by RMS.

In summary, mathematical programming, metaheuristics and artificial intelligence have been widely leveraged to tackle scheduling and reconfiguration challenges in RMS environments, with the techniques offering complementary strengths. While mathematical models support optimization under precisely de-

fined constraints, metaheuristics and AI facilitate solving highly complex problems through exploration of large search spaces. Further hybridization of these techniques can pave the way for more holistic and powerful RMS optimization capabilities.

3 Problem Description

The main focus of this work is to explore the complexity of scheduling and reconfiguration in the field of reconfigurable manufacturing systems, especially in the emerging framework of Industry 5.0. The problem is multi-objective in nature, involving the simultaneous optimisation of objectives such as minimising tardiness costs, reducing reconfiguration operator interventions, optimising makespan, and improving system ergonomics. This section unfolds the core aspects of the proposed modelling of the RMS architecture, the underlying multi-objective problem and the contextual relevance of Industry 5.0, providing a solid foundation for the proposed methodology and its subsequent implementation and experimentation.

3.1 Reconfigurable Manufacturing System (RMS)

The key components of RMS include reconfigurable machine tools (RMTs), modular equipment and intelligent control systems. These elements work in a complementary way to facilitate rapid reconfiguration, rescheduling and adaptation to new manufacturing tasks. The modular design of RMT enables a wide range of configurations to meet different product specifications and production volumes. Meanwhile, the intelligent control systems use advanced algorithms and real-time data to orchestrate the seamless transition between different configurations and schedules.The proposed hierarchical environment for RMS architecture consists of a top-level RMS environment that includes several lower-level RMT sub-environments. Each of these RMT sub-environments is as a multi-agent environment.

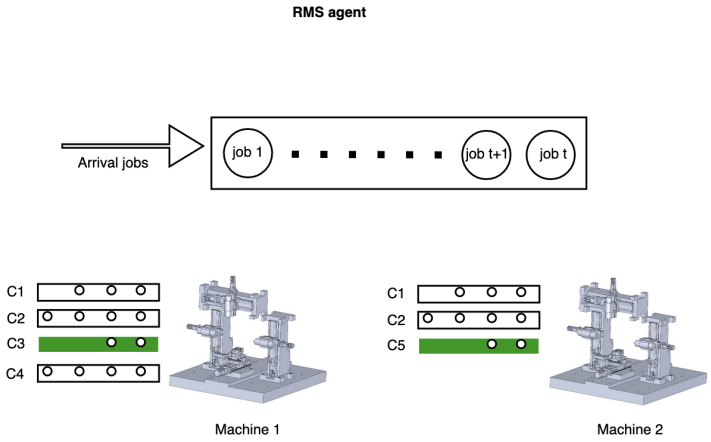


Fig. 1: RMS Modelling with RMT machines with differents configurations.

3.2 Reconfigurable Machine Tool (RMT)

Reconfigurable Machine Tool (RMT) accepts multiple configurations, each configuration is represented by a virtual buffer in this model. A configuration refers to a specific arrangement of the machine's modules to obtain certain capabilities. RMT is controlled by two cooperative but independent agents inspired from the work of [12]:

- ✓ The Reconfiguration Agent (RCA) handles dynamic Reconfiguration of the machine's modules among a predefined set of configurations to adapt its capabilities.
- ✓ The Scheduling Agent (SA) is responsible for optimizing the scheduling and sequencing of jobs on the machine in its current configuration.

By jointly learning coordinated policies, the RCA and SA agents can intelligently RCAigure the machine's modules and adjust the production SAule to optimize objectives like minimizing job tardiness as conditions change.

3.3 Reconfiguration agent RCA

The objective is to select reconfiguration actions that minimize any rise in total tardiness costs of jobs during the Reconfiguration interval. Let t_S and t'_S be the start and end times of a reconfiguration.

The reward R_t for a reconfiguration is:

$$R^{RCA} = -\frac{1}{t_{S'} - t_S}(TC_{WJ} + TC_{CJ}) \tag{1}$$

$$TC_{WJ} = \sum_{k=1}^{X} \sum_{j=1}^{n_k} \alpha_j z_{j_{S'}} \left[t_{S'} - \max\left(t_S, d_j\right) \right]$$

$$z_{j_{S'}} = \begin{cases} 1, & \text{if } d_j < t_{S'} \\ 0, & \text{else} \end{cases}$$

$$TC_{CJ} = \sum_{j=1}^{n_{FNS}} \alpha_j z_{jC} \left[C_j - \max\left(t_S, d_j\right) \right]$$

$$\mathrm{z}_{jC} = \begin{cases} 1, & \text{if } d_j < C_j \\ 0, & \text{else.} \end{cases}$$

Where TC_{WJ} and TC_{CJ} are the newly added tardiness costs of waiting and completed jobs. This rewards actions that keep the increase in total tardiness low.

Reconfiguration Trigger Reconfiguration is triggered when:

- ✓ The current buffer is empty
- ✓ Overdue jobs exist and the current buffer has relatively low average tardiness
- ✓ The number of finished jobs exceeds a threshold and average tardiness is low

This balances reconfiguration and scheduling optimization.

Action Space: Four actions are defined for selecting a new configuration k from $1, 2, ..., K$:

- ✓ **Action 1 ($act1$):** Choose k with maximum total tardiness cost $\sum_j \beta_j$ of its jobs
- ✓ **Action 2 ($act2$):** Choose k with maximum average tardiness cost $\frac{1}{n_k} \sum_j \beta_j$
- ✓ **Action 3 ($act3$):** Choose k with maximum number of jobs n_k
- ✓ **Action 4 ($act4$):** Choose k with minimum average safe time $\frac{1}{n_k} \sum_j ST_j$

If selected k matches current, no reconfiguration is done.

State Features: Four state features are used:

- ✓ F_1: Number of jobs in each configuration.
- ✓ F_2: Total tardiness cost in each configuration.
- ✓ F_3: Average tardiness cost in each configuration.
- ✓ F_4: Average safe time in each configuration.

The state space includes statistical measures on these features. States are normalized using $\min - \max$ scaling.

3.4 Scheduling agent SA

The goal of SA is selecting the next job from the current buffer (selected RMT configuration) to execute.

Reward Function It is inversely related to the change in total tardiness per second:

$$R^{SA} = -\frac{1}{t_{s'} - t_s}(TC_{WF}) \tag{2}$$

Where TC_{WF} is the newly added tardiness cost of waiting jobs in $[t_s, t_{s'}]$. This is calculated as:

$$TC_{WF} = \sum_{j=1}^{n} y_{js'}\beta_j[t_{s'} - \max(t_s, q_j)] \tag{3}$$

Where $y_{js'}$ indicates if job j is overdue at $t_{s'}$. Rewarding lower incremental tardiness aims to have SA learn optimized job sequencing.

Action Space: The SA has five actions for selecting the next job j from the current buffer:

- ✓ **Action 1** ($act1$)**:** Choose job with maximum tardiness cost ψ_j.
- ✓ **Action 2** ($act2$)**:** Choose job with maximum unit tardiness cost β_j.
- ✓ **Action 3** ($act3$)**:** Choose job with minimum safe time ST_j.
- ✓ **Action 4** ($act4$)**:** Choose job with nearest due date q_j.
- ✓ **Action 5** ($act5$)**:** Choose job with minimum processing time $\sum_i t_{ij}$.

The diverse actions enable SA to learn nuanced scheduling policies.

State Features Key state features for SA include:

- ✓ F_1: Number of waiting jobs.
- ✓ F_2: Current tardiness cost of each job.
- ✓ F_3: Unit tardiness cost of each job.
- ✓ F_4: Safe time of each job.
- ✓ F_5: Due date of each job
- ✓ F_6: Processing time of each job

Statistical measures on array features produce a 23-dim state space. States are normalized via $\min - \max$ scaling.

3.5 RMSA Agent

The RMSA supervises scheduling and reconfiguration decisions across multiple Reconfigurable Machine Tool (RMT) environments. This decentralised structure enables independent learning while reducing dimensionality. At each timestep, the RMSA assigns the next pending job to an RMT based on aggregated state data. Once initial jobs are assigned, the RMSA steps the RMTs to simulate system operation. This allows system oversight with machine autonomy.

Multiobjective scheduling and reconfiguration Optimization: The heart of the scheduling and reconfiguration problem in RMS is to determine the optimal sequence of operations on various machines, along with the optimal configuration path of machines and production lines to meet specified manufacturing objectives. The multiobjective nature of the problem requires a careful optimisation approach to find a Pareto-optimal solution that represents a balanced trade-off between the conflicting objectives. Each objective represents a critical dimension of operational performance and cost efficiency. For example, minimising the tardiness cost is critical to maintaining delivery scheduling and customer satisfaction, while reducing number of reconfiguration operator interventions is critical to improving system responsiveness to market changes. Optimising makespan has a direct impact on throughput and inventory levels, and improving system ergonomics is critical to ensuring operator safety and job satisfaction. Exploring these objectives, both individually and collectively, reveals a complex optimisation landscape with countless local optima and intricate constraints.

Industry 5.0 Context: The advent of Industry 5.0 heralds a human-centred approach to manufacturing, with an emphasis on fostering collaboration between people and intelligent systems. Unlike previous industrial revolutions, Industry 5.0 seeks to harmonise the strengths of human creativity and machine precision. The problem described in this paper is directly aligned with the ethos of Industry 5.0, as it seeks to optimise the operational efficiency of RMS while promoting ergonomic considerations for human operators. The envisaged solution aims to harness the capabilities of intelligent algorithms to not only enhance manufacturing performance, but also to create a conducive and ergonomic working environment. The synergy between man and machine is seen as a linchpin for achieving sustainable and inclusive growth in the manufacturing sector.

Multiobjective Reward: The RMSA agent employs a weighted multiobjective reward function to optimize multiple performance goals:

$$R^{\text{RMSA}} = w_1 F_t - w_2 F_s - w_3 F_o \tag{4}$$

Where:

- ✓ F_t = Average of RMTs agents' Tardiness cost reward

- ✓ F_s = Sustainability objective
- ✓ F_o = Operator intervention objective
- ✓ w_1, w_2, w_3 = Weight factors, which indicated the importance of each criterion. Usualy, these weights are given by the Decision Maker.

The tardiness cost objective F_t aims to minimize job delays. The sustainability objective F_s encourages energy-efficient operation with lower emissions. The operator intervention objective F_o penalizes excessive reconfigurations. By combining these objectives with tunable weights, the multiobjective reward allows the RMSA agent to learn policies that balance productivity, efficiency, human effort and other critical goals. The weights enable prioritizing the different objectives as needed for the manufacturing application.

Optimizing this composite reward function leads to Pareto optimal solutions that make trade-offs between the individual objective costs. This provides coordinated reconfiguration and scheduling control to maximize overall system performance across key dimensions.

Tardiness cost: The RMSA tardiness cost reward is the average of the RMT tardiness cost rewards at each timestep. This aligns the global objective of maximizing productivity with the RMT rewards to minimize tardiness.

$$F_t = \frac{1}{N} \sum_{i=1}^{N} R_i^{RMT} \tag{5}$$

where:

$$R_i^{RMT} = \frac{R_i^{RCA} + R_i^{SA}}{2} \tag{6}$$

Where N is the number of RMTs.

Sustainability Objective The sustainability objective F_s in the RMSA reward function aims to minimize the environmental impact of the manufacturing system operations. It accounts for both liquid hazardous waste (LHW) and greenhouse gas (GHG) emissions associated with each job and machine configuration inspired from the work of [4].

Where the sustainability cost F_S is:

$$F_S = w_{\text{LHW}} \sum_{i=1}^{N} \sum_{j=1}^{n_i} \frac{(LHW_{ij} - LHW_{\min})}{(LHW_{\max} - LHW_{\min})} + w_{\text{GHG}} \sum_{i=1}^{N} \sum_{j=1}^{n_i} \frac{(GHG_{ij} - GHG_{\min})}{(GHG_{\max} - GHG_{\min})} \tag{7}$$

Where:

- ✓ N is the number of RMT environments
- ✓ n_i is the number of finished jobs in RMT i
- ✓ LHW_{ij} and GHG_{ij} are the LHW and GHG emissions for job j in RMT i

- ✓ $LHW_{\min}$, $LHW_{\max}$, $GHG_{\min}$, $GHG_{\max}$ are normalization constants
- ✓ w_{LHW}, w_{GHG} are objective weights

Minimizing this sustainability cost encourages configurations and job assignments that reduce hazardous wastes and emissions. The weights allow tuning the relative importance of LHW versus GHG reduction. Normalization accounts for variability in emission levels across jobs and machines.

Operator Intervention Objective The operator intervention objective F_o aims to minimize the total number of reconfigurations across all RMT environments. It penalizes frequent machine reconfiguration that requires human effort.

The operator intervention cost is:

$$F_o = \sum_{i=1}^{M} N_{r,i} \tag{8}$$

Where M is the number of RMTs and $N_{r,i}$ is the number of reconfigurations performed in RMT i.

The total reconfigurations N_r is calculated by summing the reconfiguration counts $N_{r,i}$ from each RMT environment i.

By penalizing the total reconfiguration count through this objective term, the RMSA agent is incentivized to learn policies that minimize operator effort required for machine reconfiguration across the manufacturing system. The weight w_r controls the relative importance of this goal.

Action Space The RMSA actions assign each pending job to an RMT selected based on aggregated states to optimize system performance.
There are 5 actions corresponding to RMT with:

- ✓ minimum configuration buffer length
- ✓ maximum configuration buffer safe time
- ✓ minimum configuration buffer total tardiness
- ✓ minimum configuration buffer average tardiness

RMT already running the configuration, or minimum length

By considering factors like queue occupancy, safe time, and tardiness, the RMSA can improve system scheduling.

State Features The RMSA state summarizes high-level features across RMTs, including normalized:

- ✓ Queue lengths per RMT
- ✓ Total tardiness per RMT
- ✓ Average tardiness per RMT
- ✓ Job safe times per RMT.

It tracks RMT metrics like queue occupancy and scheduling to inform system job assignments. The compact representation allows assessing global status for joint reconfiguration and scheduling.

4 Implementation and Experimentation

4.1 Implementation of DDQN

The DDQN algorithm, an extension of the conventional Deep Q-Networks (DQN) algorithm, is employed to tackle the multiobjective optimization problem at hand. DDQN mitigates the overestimation bias of Q-values, a known issue in standard Q-learning and DQN, by decoupling the selection and evaluation of actions.

The core components of the DDQN algorithm include:

- ✓ **State Representation**: The state of the system is represented using a set of features that encapsulate the current configuration and status of the RMS.
- ✓ **Action Space**: The action space comprises all possible reconfiguration and scheduling actions that can be executed at any given time.
- ✓ **Reward Function**: The reward function is designed to reflect the objectives of the optimization problem, such as minimizing tardiness costs, optimizing makespan, and enhancing system ergonomics.
- ✓ **Q-Network and Target Q-Network**: Two separate neural networks are employed to approximate the Q-values of state-action pairs, aiding in reducing overestimation bias.

The DDQN algorithm in this work is implemented in Python 3.10 version using GYMNASIUM API and Ray library tailored for deep reinforcement learning applications, in a machine with 1 GPU and 24 CPU and 32 GO of RAM. The implementation follows a systematic procedure:

1. **Initialization**: Initializing the Q-network and Target Q-network with random weights, and setting initial values for the learning rate = $4e^{-4}$ and exploration rate = $8e^{-4}$, which are the defaults used values for *Ray Library*.
2. **Training**: Training the DDQN algorithm using a replay buffer (with length of 5000) to store and sample experience tuples, and updating the Q-network weights using mini-batch gradient descent.
3. **Evaluation**: Evaluating the trained DDQN model on a set of test instances, generated like shown in Algorithm 2 to assess its performance in solving the scheduling and reconfiguration problem.

The algorithm begins by initialising the RMS agent with random weights and proceeds to execute a series of RMS episodes. Within each episode, the RMS environment is reset, and for each RMT environment in the set, machine configurations and job distributions are sampled. The algorithm maintains states for both the RMS and RMT environments, updating them as actions are taken.

Algorithm 1: Hierarchical RL for RMS and RMT

Input: RMS environment E_{RMS}, Set of RMT environments $\{E_{\text{RMT}}\}$, Initial jobs Learning rates α_{RMSA}, ,

Initialize RMSA with random weights θ_{RMS};
for *each RMS episode* **do**
  Reset E_{RMS};
  for *each* E_{RMT} *in* $\{E_{RMT}\}$ **do**
    Sample machine configurations;
    Sample new job distribution;
    Initialize state s_0;
    Reset buffers B, completed jobs J;
  Aggregate info from $\{E_{\text{RMT}}\}$ into $s_{0,\text{RMS}}$;
  while E_{RMS} *not done* **do**
    Observe $s_{t,\text{RMS}}$;
    Sample $a_{t,\text{RMS}} \sim A_{\text{RMS}}(s_{t,\text{RMS}}; \theta_{\text{RMS}})$;
    for *each* E_{RMT} *in* $\{E_{RMT}\}$ **do**
      if RCA *triggered reconfiguration* **then**
        Calculate reconfiguration time Δt;
        Update state $s_{t+\Delta t,\text{RMT}}$;
        Reset E_{RMT};
        Sample new machine configuration;
        Sample new job distribution;
        Reset B, J;
        Initialize state $s_{0,\text{RMT}}$;
        Update aggregated RMS state $s_{t+\Delta t,\text{RMS}}$;
      while E_{RMT} *not done* **do**
        Observe state $s_{t,\text{RMT}}$;
        Sample $a_{t,\text{RCA}} \sim RCA(s_{t,\text{RMT}};)$;
        Sample $a_{t,\text{SA}} \sim SA(s_{t,\text{RMT}};)$;
        Execute $a_{t,\text{RCA}}$, $a_{t,\text{SA}}$ in E_{RMT};
        Update $s_{t+1,\text{RMT}}$, B, J;
        Calculate rewards $r_{t,\text{RCA}}$, $r_{t,\text{SA}}$;
        Aggregate RMT info into $s_{t+1,\text{RMS}}$;
    Store $(s_{t,\text{RMS}}, a_{t,\text{RMS}}, r_t, s_{t+1,\text{RMS}})$;
    Train RMSA on batch from replay buffer with learning rate α_{RMS};
    Update θ_{RMS};
Train all RMT agents $\{RCA, SA\}$ on batches with learning rates $\alpha_{RCA}, \alpha_{SA}$;

The hierarchical structure becomes apparent as the algorithm handles reconfigurations initiated by the RCA agents and scheduling decisions made by the SA agents within the RMT environments. Information is aggregated between the RMS and RMT environments to enable holistic decision making. Throughout the process, the RL agents learn from their experience and adapt their policies. The algorithm demonstrates a multi-agent, hierarchical approach to resource management, which can be particularly useful in complex systems where coordination and decision making span multiple levels.

4.2 Experimental instances generation

In this algorithm, we first generate a set of jobs with random parameters including unit tardiness cost in this interval, due date, completion time, configuration choices, load handling weights, and greenhouse gas emissions. All these values are mentioned in Algorithm 2. We then create a Job object for each job instance and store it in the list of jobs. Finally, we provide a comment on how to represent the job's attributes, including its name, due date, and completion time.

Algorithm 2: Input Jobs Generation

Data: Number of Jobs num_of_jobs, Available Configurations $available\_configs$
Result: List of generated jobs $jobs$
for *each* j *in range* num_of_jobs **do**
  Generate a unique job name $name$ based on j;
  Generate unit tardiness cost $unit\_tard\_cost$ randomly between 2 and 10;
  Generate due date due_date randomly between 10 and 50 after the arrival of the job;
  Generate completion time $completion\_time$ randomly between 3 and 20;
  Generate the number of configuration choices $num\_choices$ randomly between 1 and 2;
  Generate $num\_choices$ random configuration choices $configs$ from $available\_configs$;
  Generate quantity of liquid emitted by job of each configuration $configs\_lhw$;
  Generate $num\_choices$ random configurations' greenhouse gas emissions of each configuration $configs\_ghg$;
  Create a job object job with attributes $name$, $unit\_tard\_cost$, due_date, $completion\_time$, $configs$, $configs\_lhw$, $configs\_ghg$;
  Append job to the list of jobs $jobs$;

4.3 Training methodology

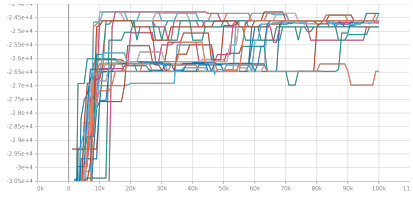
In our exploration of training efficiency, we adopted a segmented approach, particularly focusing on the Reconfigurable Manufacturing Systems (RMS) and the Reconfigurable Manufacturing Tools (RMTs) within a hierarchical environment. Initially, we directed our attention towards individually training the multi-agent system associated with RMTs. A meticulous process of hyperparameter tuning was undertaken to find the optimal settings for the Learning Rate (LR) and the exploration factor ε. This tuning process employed a grid search technique, which is a systematic way of traversing through a manually specified subset of the hyperparameter space. The grid search mechanism allowed us to scrutinize various combinations of LR and ε to pinpoint the configuration that yielded superior training outcomes.

The training curves for the Reconfiguration Agent (RCA) in Figure 2a demonstrate its ability to learn effective policies for initiating reconfiguration actions based on the state of the manufacturing system. The smoothing trend of the reconfiguration time metric indicates the agent's proficiency in generalizing its experience to make timely and impactful reconfiguration decisions.

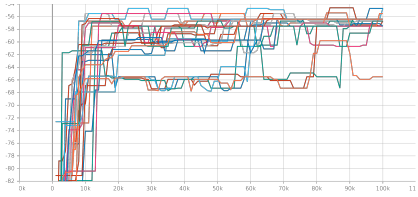
For the scheduling Agent (SA), Figure 2b exhibits convergence in the tardiness cost. This highlights the agent's competence in developing robust job scheduling sequences that minimize delays and workflow duration even as manufacturing conditions evolve. The stability of the learning process enables the SA agent to consistently generate high-quality solutions, as noted by [13] in their work on deep reinforcement learning for job shop scheduling.

The tuning config was defined as:

```
config["search_space"] = {
"lr": tune.grid_search([1e-4, 2e-4, 3e-4, 4e-5, 5e-4, 6e-4]),
"epsilon": tune.grid_search([6e-4, 7e-4, 8e-4, 9e-4])
}
```



(a) Reconfiguration agent (RCA) training curves



(b) Scheduling agent (SA) training curves

Fig. 2: Training curves for RCA and SA

Upon obtaining the desired tuning, the training proceeded, during which the Q-Network and Target Network were evolved and refined. These networks embody the core learning mechanisms that guide the agents in making intelligent decisions. Once the training of the RMT multi-agent system reached a satisfactory level, the trained networks were preserved.

Transitioning to the broader hierarchical environment of RMS, we integrated the pre-trained networks from the RMT multi-agent system. This integration serves as a bedrock, providing a substantial head start for the agents operating within the RMS environment. By leveraging the insights and learned behaviors encapsulated in the pre-trained networks, the RMS hierarchical agent could navigate the environment with a higher degree of competency right from the outset. This strategy significantly expedited the training phase, fostering a more efficient learning trajectory and accelerating the attainment of desirable performance levels within the RMS hierarchical environment.

5 Results and Discussion

The results section encapsulates the insights obtained from the implementation and experimentation phase. Various performance metrics were analyzed to ascertain the effectiveness and efficiency of the proposed DDQN algorithm in managing multi-objective scheduling and reconfiguration challenges in reconfigurable manufacturing systems.

At the higher level, the manufacturing system agent (RMSA) in Figure 3 demonstrates its capability to learn meta-policies that coordinate the lower-level RCA and SA agents to enhance overall system performance. The simultaneous optimization across all key metrics shows the agent's proficiency in balancing trade-offs through hierarchical reinforcement learning.

In summary, the empirical results provide quantitative evidence for the efficacy of the proposed hierarchical DDQN technique in enabling intelligent reconfiguration planning, scheduling, and coordination for next-generation reconfigurable manufacturing systems.

Fig. 3: Manufacturing system agent (RMSA) training curves

Additionally, the Pareto frontier in Figure 4 obtained through the multi-objective DDQN approach exhibits a rich set of non-dominated solutions that make strategic compromises between the conflicting metrics. This highlights the adaptability of the intelligent agents in tailoring decisions to dynamic manufacturing conditions.

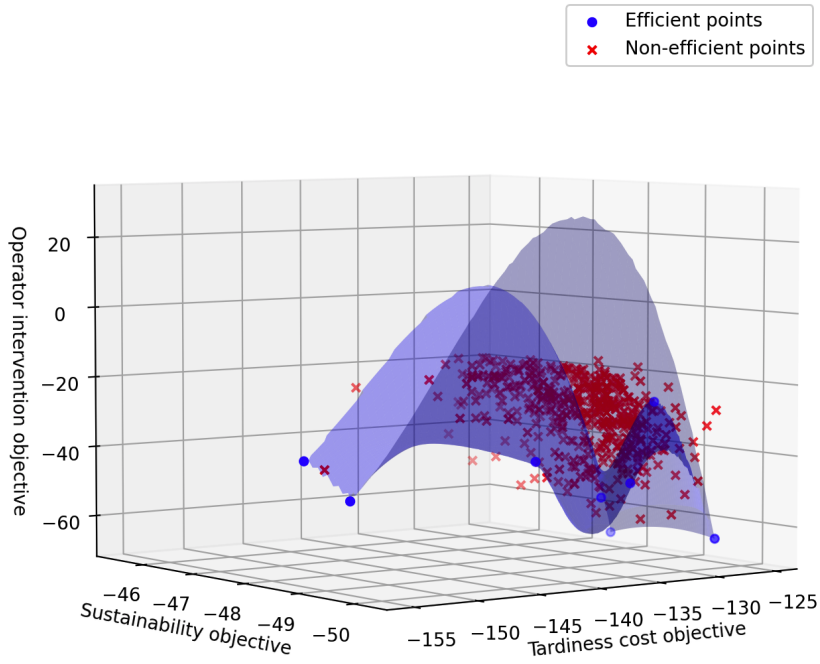


Fig. 4: Pareto frontier of non-dominated solutions

5.1 Discussion on Findings

The findings from the experimentation phase underscore the potential of deep reinforcement learning, particularly the DDQN algorithm, in revolutionizing resource management in reconfigurable manufacturing systems. The ability of the DDQN algorithm to dynamically adapt to changing manufacturing conditions, optimize resource allocation, and significantly improve operational efficiency elucidates its promise as a formidable tool for modern manufacturing environments.

Moreover, the hierarchical structure of the algorithm, as delineated in the implementation section, exhibits a nuanced approach to managing multi-level decision-making processes in complex manufacturing settings. This hierarchical approach enables a harmonized coordination between scheduling and reconfiguration decisions, thereby contributing to enhanced system performance. The success of the DDQN algorithm in both simulated and real-world environments substantiates the viability of leveraging advanced machine learning techniques to tackle intricate scheduling and reconfiguration challenges inherent in reconfigurable manufacturing systems. The positive outcomes from this project pave the way for further exploration and development of intelligent solutions for modern manufacturing challenges, aligning with the broader objectives of Industry 5.0 and smart manufacturing.

6 Conclusion and Future Work

The findings from this study underscore the potential of leveraging Hierarchical Deep Q-Networks (HDQNs) in addressing the intricate challenges inherent in reconfigurable manufacturing systems (RMS). The proposed framework, through a two-tiered approach, adeptly handles the reconfiguration and scheduling tasks, demonstrating noteworthy improvements in key performance metrics such as reconfiguration time, tardiness cost, and makespan. The agents' capability to learn and adapt to dynamic manufacturing environments, as seen in the training curves and the Pareto frontier, presents a promising pathway towards the realization of more agile and efficient RMS in line with the paradigms of Industry 4.0 and 5.0.

Looking ahead, there are several avenues that can be explored to further enhance the proposed framework:

- ✓ Investigate multi-objective RL techniques like MO-PPO to explicitly optimize for conflicting objectives like cost, sustainability, and ergonomics in a pareto-optimal manner.
- ✓ Evaluate the framework on real and larger problem instances with more machines, jobs, and configurations to assess scalability.
- ✓ Study the integration of the DRL framework with other Industry 4.0 technologies like digital twins, IoT, edge computing to enable real-time intelligent optimization.
- ✓ Analyze the reliability, robustness and worst-case performance of the DRL agents using formal verification methods.
- ✓ Evaluate alternate DRL algorithm architectures like attention-based transformers for RMS optimization. Assess interpretability.
- ✓ Benchmark against traditional optimization techniques like MILP and metaheuristics on extensive problem sets to gain deeper insights.

References

1. Tariq Aljuneidi and Akif Bulgak. Designing a cellular manufacturing system featuring remanufacturing, recycling, and disposal options: A mathematical modeling approach. *CIRP Journal of Manufacturing Science and Technology*, 19, 05 2017.
2. A Bilberg, R Malik, and K Bøgh. New model for development and manufacturing of tailored solutions in the industrial market. *Journal of Manufacturing Systems*, 31(3):367–374, 2012.
3. AI Dashchenko. *Reconfigurable manufacturing systems and transformable factories*. Springer Science & Business Media, 2006.
4. Amirhossein Khezri, Hichem Haddou Benderbal, and Lyes Benyoucef. Towards a sustainable reconfigurable manufacturing system (srms): Multi-objective based approaches for process plan generation problem. *International Journal of Production Research*, 04 2020.
5. Amirhossein Khezri, Hichem Haddou Benderbal, Lyes Benyoucef, and Alexandre Dolgui. Diagnosis on energy and sustainability of reconfigurable manufacturing system (rms) design: A bi-level decomposition approach. 09 2020.

6. Y Koren, U Heisel, F Jovane, et al. Reconfigurable manufacturing systems. *CIRP Annals*, 48(2):527–540, 1999.
7. Z Li, L Li, and A Bilberg. Design and implementation of a reconfigurable manufacturing system. *International Journal of Advanced Manufacturing Technology*, 39:1181–1191, 2008.
8. Priyabrata Mohapatra, Lyes Benyoucef, and Manoj Tiwari. Realising process planning and scheduling integration through adaptive setup planning. *International Journal of Production Research*, 51, 04 2013.
9. Farayi Musharavati and A.M.S Hamouda. Simulated annealing with auxiliary knowledge for process planning optimization in reconfigurable manufacturing. *Robotics and Computer-Integrated Manufacturing*, 28:113–131, 04 2012.
10. R Rajkumar, G Ravi, and A Zalzala. Recent advances in evolutionary and adaptable manufacturing systems. *International Journal of Production Research*, 48(22):6675–6696, 2010.
11. Jiecheng Tang, Yousef Haddad, and Konstantinos Salonitis. Reconfigurable manufacturing system scheduling: a deep reinforcement learning approach. *Procedia CIRP*, 107:1198–1203, 2022.
12. Shengluo Yang and Zhigang Xu. Intelligent scheduling and reconfiguration via deep reinforcement learning in smart manufacturing. *International Journal of Production Research*, 60:4936–4953, 07 2021.
13. Cong Zhang, Wen Song, Zhiguang Cao, Jie Zhang, Puay Siew Tan, and Xu Chi. Learning to dispatch for job shop scheduling via deep reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1621–1632, 2020.
14. Longfei Zhou, Lin Zhang, and Berthold KP Horn. Deep reinforcement learning-based dynamic scheduling in smart manufacturing. *Procedia Cirp*, 93:383–388, 2020.