Human-Aware Task Scheduling in Human–Robot Collaboration via Deep Reinforcement Learning

Francesco Boscolo Meneguolo*, Alberto Dalla Libera, Davide Frizzo and Gian Antonio Susto

***Abstract*— Human–Robot Collaboration (HRC) has become a key enabler of productivity, adaptability, and improved ergonomics in modern manufacturing environments. Despite this, many existing task allocation strategies for HRC are based on restrictive assumptions and rely on static or heuristic methods that limit their ability to cope with human variability and human factors, such as stress and work-load. In this work, we explore the use of value-based Reinforcement Learning (RL) for online task scheduling in a collaborative assembly context. We developed a simulated two different simulated setups. The first setup assessed the ability of the RL scheduler to optimize system performance under conditions of high variability in human execution. In the second setup, we modeled the operator's stress level and incorporated it directly into the objective function optimized by the RL agent. Results from the first setup indicate that the RL agent can effectively adapt task allocation in response to fluctuating human performance. Meanwhile, the second setup demonstrates the agent's capacity to balance productivity with operator well-being by explicitly trading off performance and stress.**

I. Introduction

Modern manufacturing systems are characterized by increasing product variability, shorter production cycles, and stringent efficiency requirements, which place significant demands on assembly operations. These processes require precise coordination of multiple tasks involving heterogeneous components, often under dynamic and uncertain conditions. Despite advances in automation, many assembly activities still require human intervention due to the need for adaptability, dexterity, and decision-making capabilities that remain difficult to fully replicate with autonomous systems [1], [2]. Consequently, Human–Robot Collaboration (HRC) has emerged as a key paradigm in advanced manufacturing, leveraging the complementary strengths of humans and collaborative robots (*cobots*) to enhance productivity, flexibility, and operational robustness [3], [4].

A critical challenge in HRC environments lies in the efficient allocation and scheduling of tasks between human operators and robots. Conventional approaches typically rely on static scheduling methods defined prior to execution, assuming predictable system behavior and stable human performance. However, such approaches fail to account for real-time variability arising from human fatigue, performance fluctuations, or unforeseen operational disturbances, which can negatively impact both productivity and worker well-being [5], [6], [7], [8]. To address these limitations, recent research has increasingly focused on dynamic task allocation strategies capable of adapting to changing system conditions.

Machine Learning (ML) techniques, and Reinforcement Learning (RL) in particular, have demonstrated strong potential for enabling adaptive and autonomous decision-making in complex HRC systems, and in general in dynamic task scheduling [9]. By learning optimal policies through interaction with the environment, RL can effectively manage large state and action spaces while accounting for uncertainty and variability inherent to human-in-the-loop processes [10], [11].

Building upon this paradigm, this paper proposes a Deep Reinforcement Learning–based task scheduling framework for HRC assembly environments. We considered as a case study the assembly of color sample folders. We formulated the scheduling problem as a Markov Decision Process (MDP) and implemented a simulator of the environment. Our experiments considered two setups. In the first, we evaluated the ability of the RL policy to adapt to variable human performance, exploring different levels of uncertainty—from mild variability to highly fluctuating behaviors modeled through bimodal distributions. In the second scenario, we additionally simulated the operator's stress level by introducing into the MDP a simple model, linear with respect to work and rest time. This signal was provided both to the policy and to the reward function, enabling the agent to learn a strategy that balances performance with operator well-being. In the first scenario, we compared the RL scheduler with an optimal static scheduler, while in the second we adopted a heuristic derived from the optimal solution. Results from both scenarios confirm the effectiveness of our approach.

II. Problem formulation & Related works

A. Problem formulation

This work builds on the same framework described in [12] and serves as its direct continuation, focusing on human involvement in HRC systems. Human participation introduces inherent variability, as individual performance fluctuates due to factors such as fatigue, skill level, and cognitive workload. This variability complicates system modeling, prediction, and optimization, thereby highlighting the need for adaptive policies capable of responding to dynamic human behavior. The objective of this study is to derive an optimal task scheduling policy for collaborative assembly applications by means of model-free RL. Two optimization setups are considered. In the first the policy aims to minimize the total assembly completion time while explicitly accounting for the stochastic nature of human task execution. In the second

* Main author. Email: francesco.boscolomeneguolo@studenti.unipd.it
All authors are with the Department of Information Engineering, University of Padua, Padua PD 35131, Italy.

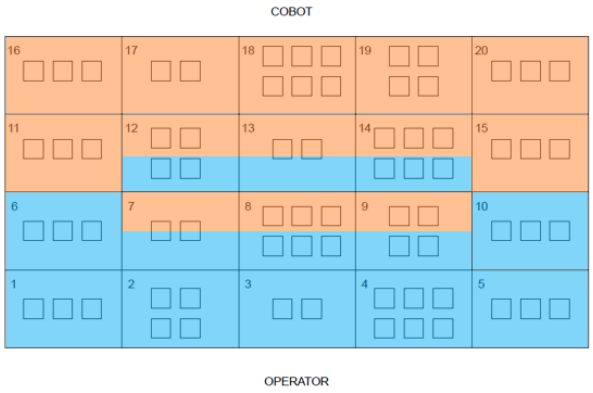

Fig. 1: Allocation table with task IDs. Blue: tasks the operator can perform; Orange: tasks the cobot can perform.

the objective is extended to minimize a composite objective function that incorporates both the total completion time and the stress level experienced by the human operator.

The proposed approach is evaluated using a case study involving the assembly of color sample folders, but it can be easily extended to other HRC tasks. These folders consist of one or more sheets onto which material samples are attached at predefined locations. In the considered HRC setup, a collaborative robot assists a human operator at a shared workstation in assembling mosaic tile sample sheets. Each experimental trial involves the completion of 20 assembly tasks characterized by different levels of complexity and execution times. Task allocation is defined according to feasibility and operational constraints, resulting in a predefined assignment in which both the human operator and the cobot are capable of performing 13 tasks each, as specified in the allocation table (Figure 1). Certain tasks are restricted to a single agent due to ergonomic or technological limitations.

The considered assembly process is repetitive and cognitively demanding, requiring accurate sample identification and precise placement. Consequently, variability arises both across different operators and within the same operator over time, due to differences in skill, experience, and fatigue. Furthermore, task allocation constraints limit the set of operations that can be performed by each agent. These characteristics make the considered application representative of real-world collaborative assembly systems and suitable for evaluating adaptive scheduling policies in HRC environments.

B. Related works

Early approaches to task allocation in HRC largely relied on deterministic optimization frameworks derived from classical scheduling theory. In these formulations, the allocation problem was modeled as a constrained optimization task aimed at minimizing performance metrics such as makespan, operational cost, or workload imbalance, while respecting feasibility constraints on agent capabilities [13], [14], [15]. Although these methods could generate theoretically optimal allocations under predefined assumptions, they inherently depended on accurate prior knowledge of task durations and stable human performance. Such assumptions rarely hold in real manufacturing environments, where human efficiency evolves over time due to fatigue, learning, or variability, and where unexpected disruptions such as equipment failures or quality deviations may arise [16]. Consequently, static schedules often lose optimality during execution and may even become impractical, highlighting the need for more flexible allocation strategies [17]. To address these limitations, researchers have explored dynamic scheduling approaches that adjust decisions based on real-time system feedback. These methods improve adaptability but are frequently based on heuristic rules, whose design requires expert knowledge and whose performance may degrade when system complexity increases or operating conditions change significantly [18], [19]. More recently, artificial intelligence techniques have emerged as promising alternatives for handling the uncertainty and stochastic nature of HRC systems. Unlike traditional rule-based approaches, these methods enable the automatic derivation of allocation policies directly from data or interaction with the environment.

One line of research has applied supervised learning techniques, including neural networks, to learn mappings between system states and scheduling decisions. Early studies demonstrated that neural networks could approximate scheduling policies by learning from historical production data, using input features such as processing times, due dates, and system load to predict task priorities or allocation decisions[20] and [21]. However, the effectiveness of supervised learning strongly depends on the availability of representative and high-quality labeled datasets. In highly dynamic HRC environments, where conditions continuously evolve, historical data may fail to capture all relevant scenarios, limiting the generalization capability of these models.

Reinforcement Learning offers a more flexible paradigm by enabling agents to learn allocation strategies through interaction and feedback rather than relying solely on predefined datasets. In the context of human–machine manufacturing systems, [5] proposed a deep RL framework capable of learning allocation policies that implicitly account for human-related variability, including differences in skill and fatigue, through training in simulated environments. Similarly, [3] introduced an RL-based task allocation method for collaborative assembly, incorporating models of human fatigue and stochasticity to better capture real-world dynamics. Their results demonstrated that adaptive RL policies can improve both system efficiency and responsiveness compared to static or heuristic approaches. [22] proposes a task scheduling algorithm based on a deep RL architecture to dynamically schedule tasks with precedence relationship to cloud servers to minimize the task execution time.

Building on these developments, the present work demonstrates that a relatively simple RL-based approach can effectively address task allocation challenges in HRC systems characterized by dynamic and uncertain operating conditions, while maintaining robustness and adaptability.

III. Proposed approach

A. MDP formulation

The considered HRC scheduling problem is formulated as a Markov Decision Process (MDP), where the environment is characterized by a state space and an action space, summarized in Table I. The state representation includes information about task completion, task assignment, execution characteristics, and the human operator's stress level (if considered) for both the robot and the human operator.

Specifically, the vectors ρ_{done} and ω_{done} are encoded as multi-hot representations that capture which tasks remain available, which tasks have already been completed, and by whom. The variables $\rho_{scheduled}$ and $\omega_{scheduled}$ identify the task currently assigned to the robot and the human operator, respectively. These variables take values in the corresponding feasible task sets, denoted by P_{tasks} for the robot and Ω_{tasks} for the human operator. A null value indicates that the corresponding agent is idle and ready to receive a new assignment. Furthermore, ρ_{times} and ω_{times} represent the task execution durations for the robot and the operator. Robot execution times are deterministic and fixed, whereas operator execution times are stochastic and sampled from probability distributions estimated from experimental observations. These distributions were obtained through an experimental campaign and reflect the intrinsic variability of human performance. Although execution times are sampled at the beginning of each episode to ensure realistic simulation dynamics, they are not included in the agent's observation space and therefore do not directly influence the learned policy.

The decision process evolves in discrete timesteps, where each step corresponds to the execution of a single task by either the robot or the human operator. Therefore, at each timestep t, the agent observes the current system state and selects an action a_t that assigns a new task to the available agent, with the objective of optimizing the scheduling policy over time.

To model operator stress, a scalar stress variable Λ_t is introduced, whose evolution follows a linear accumulation model:

$$\Lambda_{t+1} = \Lambda_t + \lambda \cdot \Delta_{a_t}, \tag{1}$$

where $\lambda > 0$ is a scaling parameter and Δ_{a_t} denotes the execution duration of the task associated with action a_t.

In the stress-aware scenario, the action space is extended to include optional rest periods for the human operator. After completing all tasks in a cycle, the scheduler may assign a fixed one-minute recovery period, during which the operator stress level is reduced by a factor $\eta > 0$, thereby enabling the policy to balance productivity and operator well-being over repeated task sequences.

An iteration corresponds to the completion of the full set of 20 tasks. The definition of an episode depends on whether operator stress is considered. In the absence of stress modeling (first setup), an episode corresponds to an iteration. When stress is incorporated (second setup), the episode consists of 15 consecutive iterations, enabling the modeling of cumulative stress effects.

The reward function is defined to encourage efficient task completion. In the baseline scenario, the reward at each time step is proportional to the negative task duration:

$$r_{t+1} = -\Delta_{a_t}. \tag{2}$$

When stress is explicitly considered, an additional penalty term is introduced to discourage excessive stress accumulation:

$$r_{t+1} = -\Delta_{a_t} - \beta \cdot (\Lambda_{t+1} - \Lambda^*) \cdot \mathbb{1}\{\Lambda_{t+1} \geq \Lambda^*\}, \tag{3}$$

where $\beta > 0$ is a weighting coefficient, Λ^* is a predefined stress threshold, and $\mathbb{1}\{\cdot\}$ denotes the indicator function.

Under this reward formulation and assuming a discount factor $\gamma = 1$, maximizing the expected return in the first setup corresponds to minimizing the total episode completion time. In the second setup, the optimization objective balances assembly efficiency and operator well-being by minimizing a weighted combination of completion time and stress-related penalties.

TABLE I: The state space and the action space.

State Space			
ρ_{done}	‖	completed tasks by robot	$\{0,1\}^{13}$
ω_{done}	‖	completed tasks by operator	$\{0,1\}^{13}$
$\rho_{scheduled}$	‖	still in progress task by robot	P_{tasks}
$\omega_{scheduled}$	‖	still in progress task by operator	Ω_{tasks}
ρ_{times}	‖	robot's task execution times	$\mathbb{R}^{13}$
ω_{times}	‖	operator's task execution times	$\mathbb{R}^{13}$
Λ	‖	operator's stress level	$[0,1]$
Action Space			
a	‖	next task to be scheduled	$\{1,2,...,20\}$
$\mathcal{R}$	‖	rest task (only in stress scenario)	0

B. Deep-RL Algorithm

To learn an optimal scheduling policy we leverage Deep Q-Network (DQN) [23] (Figure 2), in particular we adopt the Double DQN (DDQN) algorithm [24], a value-based Reinforcement Learning method designed to improve the stability and accuracy of action-value estimation. The neural network approximates the action-value function $Q(s,a)$ by taking as input the observation vector $o = [\rho_{done}, \omega_{done}, \rho_{scheduled}, \omega_{scheduled}, \rho_{times}, \Lambda]$, which encodes task completion status, current assignments, execution characteristics, and, when applicable, the operator stress level. The network produces as output the estimated Q-values corresponding to all feasible task assignments available at the current decision step.

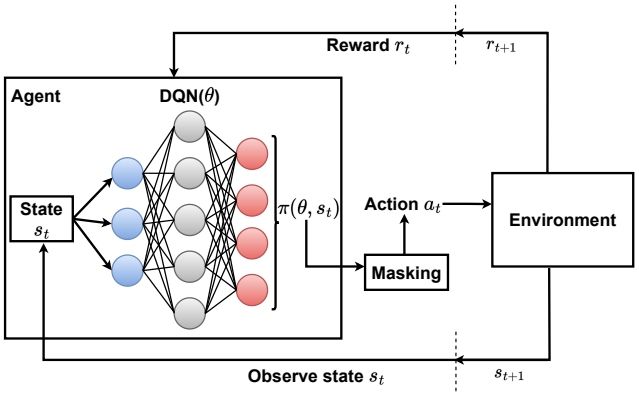

Fig. 2: A diagram of the Deep Q-Network algorithm.

When nonlinear function approximators such as deep neural networks are used, standard Q-learning may suffer from instability and systematic overestimation of action values. This issue arises because the same network is used both to select and evaluate the maximizing action when computing the temporal-difference target. The DDQN algorithm addresses this limitation by decoupling action selection from action evaluation through the use of two separate networks: an online network with parameters θ and a target network with parameters θ'. The online network selects the action that maximizes the estimated value, while the target network evaluates that action to compute the learning target. The resulting loss function is defined as

$$L(\theta) = [r + \gamma \max_{a_{t+1}} Q_{\theta'}(s_{t+1}, a_{t+1}) - Q_\theta(s_t, a_t)]^2, \quad (4)$$

where $a_{t+1} = \arg\max_{a'} Q_\theta(s_{t+1}, a')$ and γ denotes the discount factor. To further improve learning stability, the target network parameters are periodically adjusted toward the online network parameters using a soft update mechanism: $\theta' \leftarrow \tau\theta + (1-\tau)\theta'$, where $\tau \ll 1$ controls the update rate. It is also used a linear ε decay scheduler for action selection to control the exploration-exploitation trade-off during training. A key advantage of the value-based formulation is the ability to directly enforce feasibility constraints through action masking. Specifically, at each decision step, only valid task assignments are considered. Tasks that have already been completed, tasks currently in execution, or tasks that are infeasible for a given agent due to operational constraints are excluded by masking their corresponding Q-values prior to action selection. This ensures that the learned policy operates within the admissible scheduling space throughout the episode. The hyperparameters used to train the DDQN model are summarized in Table II.

TABLE II: Hyperparameters.

Hyperparameter	Value
Neural network	2×[32,ReLU]
Learning rate	1×10^{-3}
Replay buffer size	1×10^{5}
Training episodes in the first setup	6000
Training episodes in the second setup	500
Batch size	128
Discount factor γ	1
Soft update weight τ	1×10^{-3}
Initial exploration probability ε	1
Final exploration probability ε	0.05
Iterations for linear ε decay (1), (2), (4)	2600
Iterations for linear ε decay (3)	3000
Stress level coefficient λ	0.02
Stress reward coefficient β	100
Stress reward threshold Λ^*	0.6
Stress relief factor η	0.1

IV. Experiments

The experiments were conducted to investigate whether a Deep-RL algorithm can ensure an efficient, reliable and adaptive scheduling policy in a complicated and dynamic HRC environment in two different setups. We focused only on the cooperation between one cobot, whose execution times are considered deterministic for all its 13 tasks, and a single human operator. To simulate a realistic and diverse human behavior, we ran multiple experiments using an operator profile, characterized by a distribution of task execution times based on the data we collected during the experimental campaign. These values were used to sample, by means of a normal distribution, individual execution times for each task, introducing variability and uncertainty in the human performance. The nominal task execution times are used as means of the Gaussian distributions, while standard deviations $\sigma = 0.05$ and $\sigma = 0.2$ are used for the low variance and high variance scenarios, respectively.

The first setup comprises three different scenarios:

- (1) low variance Gaussian distributions without considering the operator's stress level;
- (2) high variance Gaussian distributions without considering the operator's stress level;
- (3) low variance Gaussian distributions with a mixture of two low variance Gaussian distributions for a common task without considering the operator's stress level. Concerning the common task whose execution time is sampled from a mixture of two Gaussian distributions, the execution time is sampled with a probability of 90% from a "faster" Gaussian distribution and with 10% probability from a "slower" Gaussian distribution, with a difference of one minute between the means of the two normal distributions.

The second setup includes a single scenario:

- (4) low variance Gaussian distributions considering the operator's stress level.

The goal of our DDQN agent is to learn an efficient

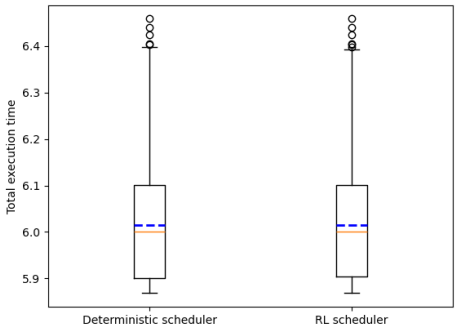

Fig. 3: Distribution of total execution time for the deterministic scheduler and the RL scheduler in (1). Orange line: median total execution time. Blue dashed line: average total execution time.

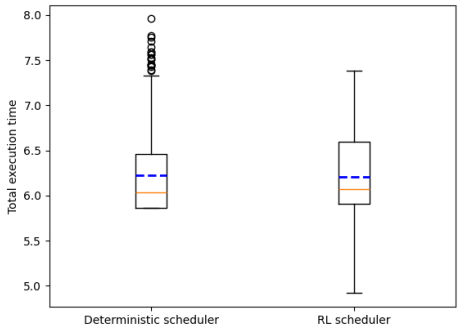

Fig. 4: Distribution of total execution time for the deterministic scheduler and the RL scheduler in (2). Orange line: median total execution time. Blue dashed line: average total execution time.

scheduling policy that appropriately allocates tasks between the robot and the human operator, taking into account the stochastic nature of human labor. The RL scheduler is evaluated against an optimal deterministic scheduler, obtained assuming deterministic execution times, equal to the nominal ones. Generally, the RL scheduler manages the randomness of human operator task times in a better way, considering both the mean times and the number of outliers of the execution times using the two schedulers. The evaluations are done over 1000 episodes.

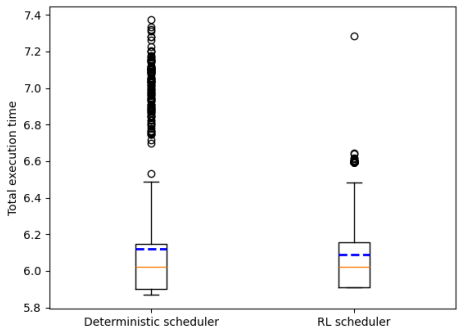

Fig. 5: Distribution of total execution time for the deterministic scheduler and the RL scheduler in (3). Orange line: median total execution time. Blue dashed line: average total execution time.

A. *Results without stress*

In (1) the deterministic scheduler and the RL scheduler have similar performance (Figure 3). This is due to the low variability of the operator's execution times, that are similar to the nominal ones. In (2) (Figure 4) the deterministic scheduler struggles to handle the higher variance of human operator's execution times, resulting in a large number of outliers. On the other hand, the RL scheduler manages successfully the randomness of execution times. In (3) (Figure 5) the increased variability of a chosen common task makes the deterministic scheduler less effective, leading to a great number of outliers.

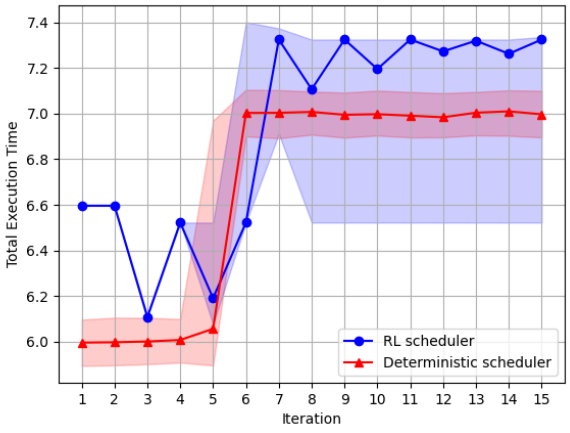

Fig. 6: Median total execution time per iteration in (4) for the deterministic scheduler and the RL scheduler. The shaded area corresponds to the interquartile range.

B. *Results with stress*

Results related to (4) are illustrated in Figures 6 and 7. We use as deterministic scheduler the optimal deterministic scheduler with the addition of the heuristic that it automatically selects the "Rest" task when it exceeds the stress threshold Λ^*. Although the RL scheduler requires longer execution times per iteration than the deterministic scheduler, it effectively reduces the stress level, being successful in keeping it below the stress threshold. This suggests improved adaptability and overall system efficiency.

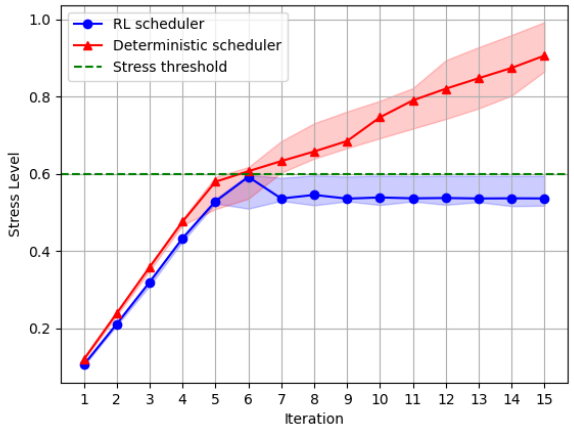

Fig. 7: Median stress level per iteration in (4) for the deterministic scheduler and the RL scheduler. The shaded area corresponds to the interquartile range.

V. Conclusion

This work presents a value-based Reinforcement Learning approach to address dynamic task scheduling in Human–Robot Collaboration. The proposed method demonstrates clear advantages over static and simple heuristic scheduling strategies, particularly in its ability to adapt to variations in human behavior. The learned scheduling policy achieves optimal performance within the considered HRC framework and shows potential for extension to more diverse application scenarios.

References

[1] W. Xu, Q. Tang, J. Liu, Z. Liu, Z. Zhou, and D. T. Pham, "Disassembly sequence planning using discrete bees algorithm for human-robot collaboration in remanufacturing," *Robotics and computer-integrated manufacturing*, vol. 62, p. 101860, 2020.

[2] J. E. C. Mateus, E.-H. Aghezzaf, D. Claeys, V. Limère, and J. Cottyn, "Method for transition from manual assembly to human-robot collaborative assembly," *IFAC-PapersOnLine*, vol. 51, no. 11, pp. 405–410, 2018.

[3] R. Zhang, Q. Lv, J. Li, J. Bao, T. Liu, and S. Liu, "A reinforcement learning method for human-robot collaboration in assembly tasks," *Robotics and Computer-Integrated Manufacturing*, vol. 73, p. 102227, 2022.

[4] L. Wang, S. Liu, H. Liu, and X. V. Wang, "Overview of human-robot collaboration in manufacturing," in *Proceedings of 5th International Conference on the Industry 4.0 Model for Advanced Manufacturing: AMP 2020*. Springer, 2020, pp. 15–58.

[5] T. Joo, H. Jun, and D. Shin, "Task allocation in human–machine manufacturing systems using deep reinforcement learning," *Sustainability*, vol. 14, no. 4, p. 2245, 2022.

[6] L. Gualtieri, E. Rauch, R. Vidoni, and D. T. Matt, "Safety, ergonomics and efficiency in human-robot collaborative assembly: design guidelines and requirements," *Procedia CIRP*, vol. 91, pp. 367–372, 2020.

[7] K. Li, Q. Liu, W. Xu, J. Liu, Z. Zhou, and H. Feng, "Sequence planning considering human fatigue for human-robot collaboration in disassembly," *Procedia CIRP*, vol. 83, pp. 95–104, 2019.

[8] M. Calzavara, M. Faccio, I. Granata, and A. Trevisani, "Achieving productivity and operator well-being: A dynamic task allocation strategy for collaborative assembly systems in industry 5.0," *The International Journal of Advanced Manufacturing Technology*, vol. 134, no. 7, pp. 3201–3216, 2024.

[9] C. Shyalika, T. Silva, and A. Karunananda, "Reinforcement learning in dynamic task scheduling: A review," *SN Computer Science*, vol. 1, no. 6, p. 306, 2020.

[10] T. Yu, J. Huang, and Q. Chang, "Mastering the working sequence in human-robot collaborative assembly based on reinforcement learning," *IEEE Access*, vol. 8, pp. 163 868–163 877, 2020.

[11] J. Wang, Y. Yan, Y. Hu, and X. Yang, "A reinforcement learning from human feedback based method for task allocation of human robot collaboration assembly considering human preference," *Advanced Engineering Informatics*, vol. 66, p. 103497, 2025.

[12] S. Binotto, A. Dalla Libera, I. Granata, R. Minto, M. Calzavara, M. Faccio, and G. A. Susto, "Value-based reinforcement learning for task scheduling in human-robot applications," *IFAC-PapersOnLine*, vol. 59, no. 26, pp. 133–138, 2025.

[13] M. Pinedo and K. Hadavi, "Scheduling: theory, algorithms and systems development," in *Operations research proceedings 1991: Papers of the 20th annual meeting/vorträge der 20. Jahrestagung*. Springer, 1992, pp. 35–42.

[14] G. Michalos, J. Spiliotopoulos, S. Makris, and G. Chryssolouris, "A method for planning human robot shared tasks," *CIRP journal of manufacturing science and technology*, vol. 22, pp. 76–90, 2018.

[15] M. Calzavara, M. Faccio, and I. Granata, "Multi-objective task allocation for collaborative robot systems with an industry 5.0 human-centered perspective," *The International Journal of Advanced Manufacturing Technology*, vol. 128, no. 1, pp. 297–314, 2023.

[16] M. Zandieh and M. Adibi, "Dynamic job shop scheduling using variable neighbourhood search," *International Journal of Production Research*, vol. 48, no. 8, pp. 2449–2458, 2010.

[17] A. Pupa, W. Van Dijk, and C. Secchi, "A human-centered dynamic scheduling architecture for collaborative application," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4736–4743, 2021.

[18] Z. Jovanovic and S. Maric, "A heuristic algorithm for dynamic task scheduling in highly parallel computing systems," *Future Generation Computer Systems*, vol. 17, no. 6, pp. 721–732, 2001.

[19] F. NZanywayingoma and Y. Yang, "Effective task scheduling and dynamic resource optimization based on heuristic algorithms in cloud computing environment," *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 11, no. 12, pp. 5780–5802, 2017.

[20] G. R. Weckman, C. V. Ganduri, and D. A. Koonce, "A neural network job-shop scheduler," *Journal of Intelligent Manufacturing*, vol. 19, no. 2, pp. 191–201, 2008.

[21] T. Eguchi, F. Oba, and T. Hirai, "A neural network approach to dynamic job shop scheduling," in *Global Production Management: IFIP WG5. 7 International Conference on Advances in Production Management Systems September 6–10, 1999, Berlin, Germany*. Springer, 1999, pp. 152–159.

[22] T. Dong, F. Xue, C. Xiao, and J. Li, "Task scheduling based on deep reinforcement learning in a cloud manufacturing environment," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 11, p. e5654, 2020.

[23] V. Mnih, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[24] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, 2016.