Learning to Schedule Machines and Operators: A Human-Aware Deep Q-Learning Framework for Dynamic Flexible Job Shops

TAJI HAJAR[1], AYAD GHASSANE[2], ZAKI ABDELHAMID[3], KHAMMAL ADIL[4]
Mechanical, Material and Thermal Laboratory, National School of Mines in Rabat (ENIM), Rabat, Morocco[1, 2]
Laboratory of Intelligent Systems Industrial and Mechanical Engineering (LISIME), National School of Arts and Crafts (ENSAM) Casablanca, Morocco[3]
Mathematics and Computer Science Department-Ben M'Sik Faculty of Sciences, University Hassan II, Casablanca, Morocco[4]

***Abstract*—Most learning-based schedulers for job shop problems assume static workforce performance. However, dynamic dual-resource job shops must navigate stochastic disturbances and time-varying task durations due to human factors. This study proposes a human-centered Deep Reinforcement Learning framework, DD4LQN, for dynamic flexible job shop scheduling under operator learning and forgetting dynamics. The environment integrates a disturbance-aware scenario generator, bounded logistic learning-forgetting dynamics, and deterministic Dual Pair Ranking to ensure auditable decisions. The training used a plan that included ENTRY and EXIT greedy evaluations and a Quote-then-Commit process. Tests over ten weeks showed that DD4LQN-EXIT did better than other scheduling methods. Empirical evaluations over a ten-week horizon demonstrate that DD4LQN-EXIT got the average schedule reward of 0.5843. It was better than Earliest Due Date (EDD) and Shortest Processing Time (SPT) by 17% and 14%, respectively. It also completed jobs and had fewer risky orders. Even though Shortest Processing Time had delays on average, it completed fewer jobs because it focused on short tasks. Additionally, an exploratory ablation across the same scenarios demonstrated that Dual Pair Rank and learning–forgetting dynamics come up with better results. Furthermore, representation diagnostics on the 227,717-parameter network confirm structural stability, with layer matrices retaining up to 98.2% of maximum effective rank without capacity collapse.**

Keywords*—Deep reinforcement learning; dynamic flexible job shop; dual-resource scheduling; human-centered manufacturing; learning and forgetting

I. INTRODUCTION

Scheduling production, or Job Shop Scheduling (JSS), is one of the core issues in production planning, where Job Shop Scheduling with flexible resources (FJSS) or Flexible Job Shop Scheduling (FJSP) assigns precedence-constrained tasks to available resources to minimize makespan, lateness, or even resource utilization [1]. FJSP has been a challenging problem in the field of scheduling, since it is NP-hard [2], even harder than regular JSS [3], because every task can be assigned to more than one machine [4], thus increasing the search space even more [5]. Even for JSS, most methods are not able to find a satisfactory solution for problems of medium to large sizes [6]. The classical problems related to Job Shop Scheduling (JSS) and its flexible variation (FJSS) have been tackled in the past decades by numerous mathematical models, as well as by a wide variety of heuristic and meta-heuristic techniques that aim to find a solution to production planning problems, by assigning the tasks to the resources to minimize some performance measures such as makespan, lateness, or even the amount of resources which are used for executing the tasks, in environments where the number of resources is finite and where each task has a set of precedents that must be satisfied before the task can be executed [7]. For a long time, a large set of models [8], heuristics [2], and also of complete and incomplete search techniques for solving Flexible Job Shop Scheduling Problems (FJSP) have been studied, since allowing different machines for one task is considered to be one of the most challenging issues for FJSS problems, since FJSS is NP-hard in general [2] and even many techniques that find solutions for such problems in reasonable running times, have limitations when they are applied to instances of FJSS Problem of large dimensions [9]. Another critical challenge that has not been fully addressed in the FJSS domain is related to dual-resource (M&R and H&R) assignment, especially when it comes to modeling and handling the human factor variability while jobs are being executed. Although the variability of the human factor has been recently highlighted in production management literature, the majority of studies have been modeling human resources as static resources, while in reality, human execution times follow a dynamic distribution that is influenced by various factors, including the operator's competencies, skill levels, and prior experiences [10]. In fact, there are only a few studies in the field that have modeled human performance, and those have been using simplistic models and have not been incorporating learning and forgetting phenomena that have been proven to play critical roles in operators' performance over time. While there is an increasing interest in modeling human competencies and their variability in order to enhance production performance and improve production operations in Industry 4.0 and 5.0 environments, more research is clearly needed to fully address this critical challenge and to effectively integrate human factors in production settings [11]. This work tackles the issues of dynamic production in order to deal with disturbances by proposing a DRL paradigm for the dynamic dual-resource scheduling (JFDRS) to stabilize the deterministic decisions of a job shop with two types of resources: machines and humans. To model the human resources, this work explicitly models human performance using a learning-

forgetting operator model that is fed to the DD4LQN (Dynamic Deep Four Layers Q-Network) by an environment specifying the Dual Pair-Rank that is produced by a weekly scenario generator that is given a disruption-aware shop floor.

The study is organized as follows. Section II presents a clear review of related work concerning DRL-based production scheduling and the human factors involved in shop-floor management. The third section elaborates on the disturbance-aware scenario generator, the Bounded Logistic Learning and Forgetting (BLLF) modelization, the Dual Pair-Rank (DPR) modelization, as well as the hierarchical reward structure. Section IV focuses on the DD4LQN training protocol and the network architecture; it also gives in detail the framework results while assessing its performance in comparison with SPT and EDD scheduling techniques. Finally, Section VI summarizes the main contributions of the study, discusses the limitations of the work, and outlines future research avenues.

II. Related Work and Research Gaps

A. Deep Reinforcement Learning for Job Shop and Flexible Job Shop Scheduling

Deep reinforcement learning (DRL) scheduling is commonly formulated based on a Markov decision process (MDP), in which an agent selects actions to maximize cumulative reward [12], [13]. When transition-relevant workforce memory is not included in the agent observation, however, the controller operates under partial observability. Recently, learned dispatch heuristics are being replaced by deep value-based policies and by more structured state-action space representations [14], [15]. However, for these problems of scheduling abstract resources in abstract environments, abstraction, incentives, and instances have a huge impact on the performance of operation control. However, for these abstract representations of manufacturing environments, operation control is too complex and hard to manage[16]. In order to increase the stability of training in DRL models for scheduling, a target network, a replay buffer, Double DQN, prioritized replay, and distributional RL are applied in order to deal with delayed rewards, poor generalization, and a lack of interpretability of learned controllers. However, these methods for increasing the stability of training have a lot of serious drawbacks, such as a high computational cost of training [17]. In addition, a major open research question for assessment and tie management in DRL for scheduling remains: how to guarantee environment-level determinism to guarantee repeatable and auditable dispatching in the presence of disturbances in shop states and in arrival times of events in a scheduling environment [18].

B. Dynamic Scheduling Under Disturbances

Dynamic JSS/FJSP schedules take into account machine failures, new task arrivals, and processing time variance. Scheduling in cyber-physical systems is adaptive but has to be feasible after interruptions [19], [20]. In recent DRL approaches, ranking or rule selection is implemented at event points, but feasibility is often not ensured [21]. Stochasticity, implementation issues, and hardware characteristics strongly affect DRL outputs[12], [22]. In scheduling situations, learning targets are already unstable due to non-determinism of tie-breaking, state-altering evaluation, and post-selection duration re-computation. Thus, there is a severe gap for disturbance-aware scheduling to guarantee repeatable and auditable dispatching. This can only be achieved by imposing determinism on the scheduling environment [23]. For that purpose, the respective status records have to be treated as hard constraints, so-called scheduling constraints, and have to be provided with an authoritative status record and auditable outcome labels.

C. Human-Centric Scheduling with Learning and Forgetting

However, human factors are not yet sufficiently modeled in scheduling [10]. Skills for Industry 4.0/5.0 should enable employees to deal with job changes and to work in a repetitive and, to some extent, also interrupted manner [11], [24]. The classical learning-forgetting-curves (LFCs) are still interpretable; however, in many cases they are restricted by re-assignment, by decay thresholds, and by warm-up procedures [25], [26]. In many cases, human dynamics are not, or only poorly, calibrated and are left to the RL loop or are not integrated at all in human-machine scheduling, as it has been requested several times [4], [27].

III. Modelization and Environment Definition

A. Problem Formulation and Event-Driven Time Assumptions

Building upon the previously detailed limitations, the environment is designed as a dynamic dual-resource FJSP, requiring the duration of tasks along with machine and operator availability to complete them based on their precedence constraints. The model tackles three main limitations that had not been addressed by previous research on FJSP: 1) unrealistic treatment of operators as static resources, 2) Lack of audibility of infeasibility due to disruptions, and 3) environment-level non-determinism leading to unstable DRL targets. These hard constraints are verified by means of authoritative status logs and by the next-free proxies, i.e., a human operator executes decisions under continuous event-driven time, (stochastic) weekly demand, (stipulated) due dates, and operator/machine unavailability. For cross-week continuity and interpretable skill progression, human execution is constrained to an ABS-time representation of learning-forgetting dynamics, i.e., a (stipulated) week starts at an ABS-time (absolute begin time) and ends at a forget-time. Four hypotheses are to be evaluated by the experiments: 1) ABS-time memory leads to more coherent human-state transitions over weeks than week-reset memory, (H1); 2) Dual Pair Rank is controllable and efficient for meeting hard deadlines in a dual-resource environment, (H2); 3) Result labels and logs of authoritative status are audible for a human operator for infeasibility caused by disruptions, (H3); and 4) a Quote-then-Commit framework with deterministic Quote and a specified then-commit rule and tie-breaking rule stabilizes the DD4LQN environment, (H4).

B. Weekly Scenario Generator and Continuous-Time Simulation

To train the DRL agent in an environment that is as realistic as possible but at the same time under control, it is important to isolate the weekly stochastic realizations from the master data. In this way, in order to avoid an artificial drift but at the same time ensure to keep the cross-week continuity. The master-data

layer of the environment describes a fixed shop with 20 machines and 12 operators, as well as a product catalog that consists of an occasional pool (P16–P20) and a known pool (P1–P15); see Fig. 1. The operators are described by their initial experience, their warm-up units, as well as by their forgetting thresholds. In this way, the skill of the operators can be transferred from one week to another. Each product goes through a fixed number of 3 to 5 tasks. Each task has a set of eligible machines as well as a set of standard processing times for these machines. A product also has a deadline for each weekly scenario that includes the external constraints of the client.

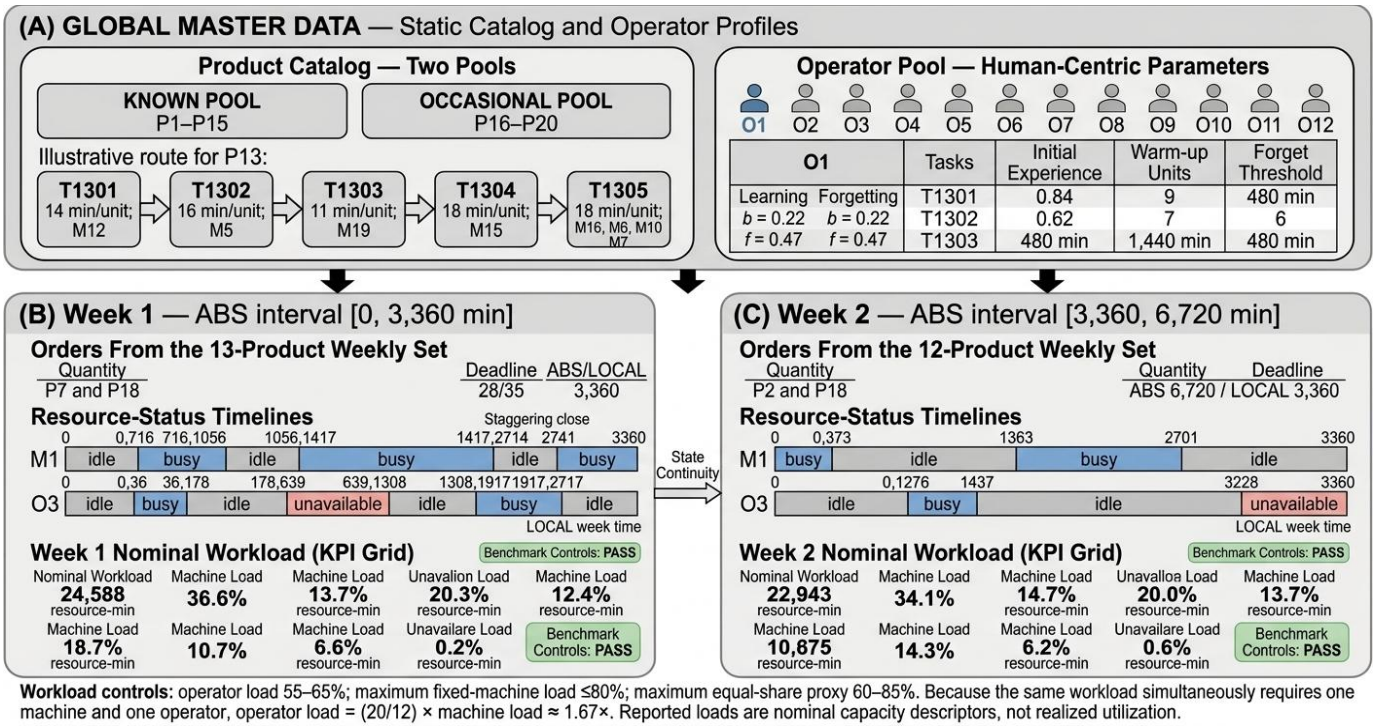

Fig. 1. Weekly scenario generator and continuous-time simulation logic.

As presented in Fig. 1, the generator randomly generates an order mix of 8-10 regular products and 1-5 irregular products on a weekly basis over 3360 minutes. The batch sizes are between 25 and 40 units. The local windows are defined for idle, busy, maintenance, absent, and unavailable times of machines and operators. The due dates are generated on an ABS timeline spanning weeks or more, and then mapped to the local week timeline in order to calculate feasibility and rewards. The test instances use 22,228–26,067 resource-minutes every week to analyze 11–14 products and 44–53 tasks. The 20 machines (67,200 min) and 12 operators (40,320 min) result in an operator-to-machine load ratio of 1.67 since each operation requires both a machine and an operator at the same time. Therefore, system restrictions are driven by operator load (55.1–64.7%) rather than machine load (33.1–38.8%). Limited flexibility (1.39–1.73 machines/operation) results in 84.8% of localized bottlenecks. This 20-machine setup incorporates practical dual-resource, due-date, and human-performance limitations while offering a scale context similar to traditional FJSP benchmarks of Brandimarte [28] and Hurink [29] work. The current generator does not include sequence-dependent setup, repair rules, transport, and multi-operator activities.

C. Logistic Learning–Forgetting Performance Model

More accurate models of learning and forgetting are required. Although traditional learning-forgetting models, such as those proposed by Wright [30] and Ebbinghaus [42], are simple to understand and follow, they treat learning and forgetting as separate processes, failing to capture constrained, non-linear learning trajectories affected by the warmup and interruption effects. Others, such as ODE-based models [31] unify the two learning processes, yet are less transparent and are difficult to calibrate for the type of disturbance analysis in question. Alternatively, learning processes have been modeled using data-driven techniques such as LSTM networks [32] and Model-Agnostic Meta-Learning (MAML) [33]. However, as black-box models, they are very adaptive but require a lot of data to learn from. There are also more complete cognitive models of human learning, such as the Adaptive Control of Thought–Rational (ACT-R) cognitive architecture [32]. Yet, such models are also too complex and thus not suitable for industrial, widespread application. Finally, Distributed Reinforcement Learning (DRL) has been applied for worker-aware scheduling and task allocation; however, as in the case of learning, the performance evolution of such systems is also difficult to understand [34].

To represent the evolution of operator performance across repeated task assignments, this study adopts bounded learning and forgetting mechanisms [4], [35]. For every unit processed by operator O on task τ, performance is computed according to Eq. (1):

$$P(o,\tau)^{n+1} = P(o,\tau)^{n} + \frac{Pmax - P(o,\tau)^{n}}{1+exp[-So(n-Mo,\tau)}} \quad (1)$$

where, $P(o,\tau)^n$ is the operator's performance after n processed units, Mo,τ is the learning midpoint related to the operator–task pair, Pmax is the upper performance constraint, and So is the operator learning rate. The updated performance falls within [P0, Pmax]. As a result, performance continuously improves toward Pmax with repeated execution, and each unit's effective processing time is calculated by dividing its standard time by the associated performance level.

The model attempts to maintain operator-task memory throughout subsequent weeks, allowing the same operator to be reassigned to the same work. The absolute idle duration is computed as the difference between the start time of the new assignment and the prior absolute completion time.

Only when this duration is superior to the task-operator specific threshold So,τ does forgetting become active. Eq. (2) determines the performance at the initial stage of the new assignment:

$$P(o,\tau,\Delta t)^{Start} = Pmin + \frac{P(o,\tau,)^{Last} - Pmin}{1+exp[-Fo(\Delta t - So,\tau)} \quad (2)$$

where, Fo is the forgetting curve, Pmin is the lower performance bound, Δt is the absolute idle duration, and $P(o,\tau)^{Last}$ the performance attained following the previous task execution.

As a result, performance is maintained prior to the forgetting threshold and then monotonically declines toward Pmin as inactivity increases. Instead of starting from the operator's initial experience level, learning resumes from this retained performance after reassignment. Fig. 2 illustrates the resulting bounded learning and threshold-activated forgetting dynamics.

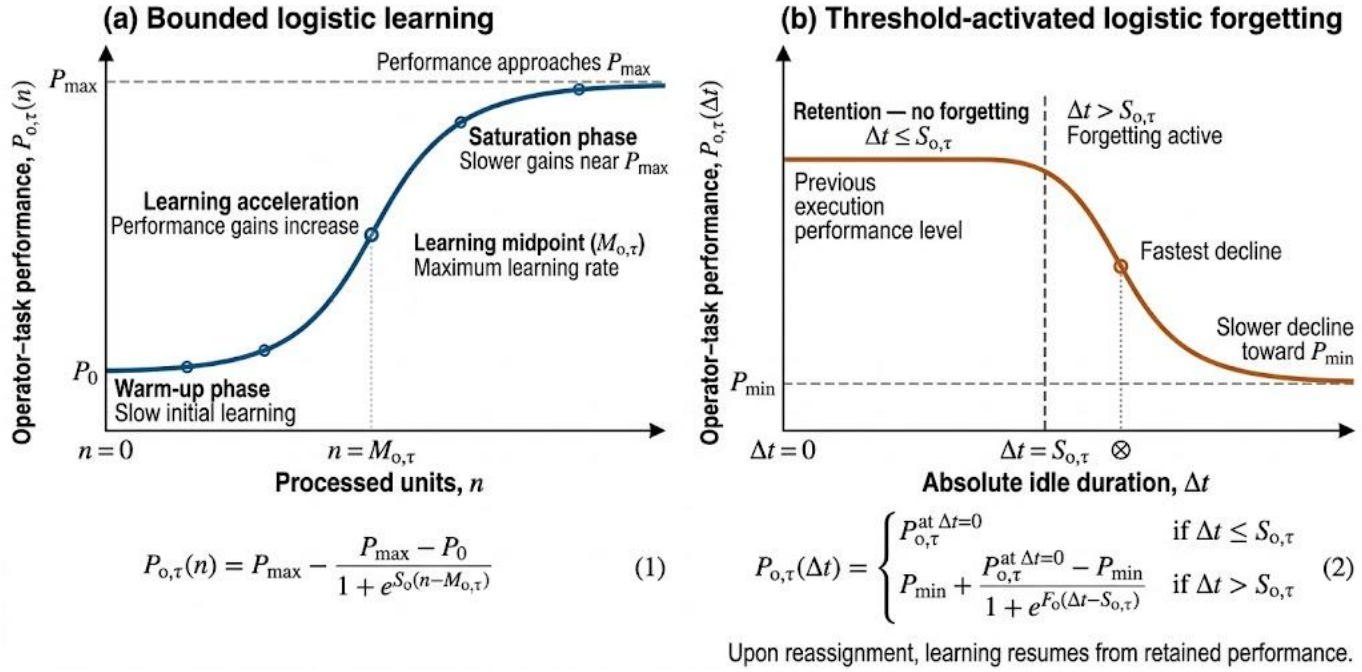

Fig. 2. Operator–task performance dynamics under repeated execution and interruption: (a) logistic learning toward the upper performance bound and (b) retention followed by logistic forgetting toward the lower performance bound.

As illustrated in Fig. 2 the bounded learning and forgetting curves reflect task-specific operator performance under repeated execution and protracted inactivity. By incorporating these human-performance dynamics into the RL scheduling loop, the proposed model compensates for workforce skill evolution when making production-planning decisions.

The BLLF model provides an explicit synthetic framework to evaluate the robustness of DRL policies against variability in human performance. The qualitative mechanisms (e.g., threshold-activated decay and bounded retention) are consistent with empirical ergonomics literature of Baily [36] Nembhard [37] and Hoedt [38] contributions but the particular numerical parameter ranges (P(o ,τ) ∈[0.60, 0.85], So ∈ [0.20, 0.35], Fo ∈ [0.35, 0.60]) are artificial scenarios. For starters, the study's findings are based on artificial, simplified conditions rather than real-world testing. Future validation is still needed for empirical calibration utilizing shop-floor cycle-time records (e.g., Lago Alvarez, 2025 [39]).

D. Machine-Operator Dual Pair Ranking

The study considers a dynamic dual-resource job shop in which every operation requires one eligible machine and one eligible operator. Machine and operator unavailability, different release times, and time-dependent human processing times make pair assignment a multi-criteria decision. For a dispatchable task Tij at decision time t, a candidate c = (Tij, t, m, o) is retained only when machine m and operator o are simultaneously feasible and its quoted completion time lies within the planning horizon H. Dual Pair Rank then converts the feasible pair attributes into one deterministic cost so that the selected pair is stable and reproducible. Although DD4LQN selects the product identifier, Dual Pair Rank performs the subsequent machine-operator assignment for the current

operation. This separation implies that the learned policy and resource-pairing must be evaluated as distinct components of the system.

1) Deterministic scalarization for pair ranking: For each feasible candidate, Dual Pair Rank combines five bounded costs: Machine Rank MR(c), Operator Rank OR(c), deadline-risk cost URG(c), normalized earliest-start cost EST_N(c), and normalized predicted-underperformance cost UP_N(c). Lower values are preferred.

$$\Pi P(\chi) = \omega_M MP(\chi) + \omega_O OP(\chi) + \omega_Y Y P\Gamma(\chi) + \omega_E E\Sigma T_N(\chi) + \omega_\Pi Y\Pi_N(\chi) \quad (9)$$

where, $\sum wi = 1$ and $wi \geq 0$ are satisfied by the weight vector $w = (w_M, w_O, w_U, w_E, w_P) = (0.25, 0.30, 0.20, 0.15, 0.10)$. A lower score PR(c) implies a more advantageous candidate assignment because each component corresponds to a cost metric. Table I presents operational definitions and significance of each criterion.

TABLE I. PARAMETERIZATION, OPERATIONAL DEFINITIONS, AND SIGNIFICANCE OF DUAL PAIR RANK COMPONENTS

Criteria	Wi	Operational Definition	Domain Significance
Machine Rank (MR)	**0.25**	Prevents machine overload and bottlenecks by evaluating a machine's past workload and future queue to determine how busy it is.	Keeps machine workloads evenly distributed around the shop floor by preventing the system from overloading popular machines and accumulating work **[40]**.
Operator Rank (OR)	**0.30**	Takes into account operator slowdowns, memory loss during idle periods, and warm-up delays when estimating the time and effort required by an operator.	By accounting for skill loss and warm-up time, it avoids depending solely on the best employees and promotes equitable operator rotation while maintaining a rapid pace of work **[41], [42]**.
Emergency (URG)	**0.20**	Determines how close a task is to its due date, which results in advancing critical tasks and preventing past-due tasks from totally breaking the system's logic.	Accelerates critical tasks as their due dates approach, but limits past-due tasks to prevent them from disrupting or taking over the decision-making process **[43], [44]**.
Earliest Start Time (EST_N)	**0.15**	Selects earlier possibilities to save needless waiting time by comparing acceptable start timings to start a task.	Eliminates needless waiting and prevents the system from selecting a late start time just because the worker or machine score appears to be high **[45], [46]**.
Underperformance Penalty (UP_N)	**0.10**	Penalizes worker-machine pairings that are abnormally slow in order to prevent selecting incredibly inefficient combinations.	Eliminates incredibly sluggish worker-machine combinations without applying excessive penalties or double counting operator slowdowns.

Table I shows that by integrating workforce priorities, deadline buffers, and efficiency protections into an elementary [0, 1] cost score, PR(c)'s design achieves a balance between shop-floor stability, human performance, and timetable timeliness. The approach keeps machine usage balanced while avoiding overworking the best operators and preventing burnout. It also aims to prevent late orders from breaking the decision logic and introduces small safeguards to prevent pointless waiting time and avoid poor resource pairings.

2) Weight derivation and limits of the justification: Although the literature currently supports these five criteria, it does not impose particular numerical weights. The weight vector $\{w\} = (0.25, 0.30, 0.20, 0.15, 0.10)$ was formalized using the Revised Simos rank-order process [47] to guarantee strict scientific repeatability. The process produces non-normalized scores of 1.0, 1.5, 2.0, 2.5, and 3.0 that normalize straight to their final values using an importance ratio $z = 3$ and become 0.10, 0.15, 0.20, 0.25, and 0.30, respectively dividing by their sum of 10 yields across the ordered criteria (UPN < ESTN < URG < MR < OR): Because operator performance varies depending on task experience, idle time, forgetfulness, and warm-up, OR is given the highest weight [41], [42]. This decision is also in line with the workload benchmark, where nominal machine load (33.1%-38.8%) is less than nominal operator load (55.1%-64.7%). Because machine eligibility might concentrate workload and cause local delays, MR comes in second [40]. Remaining deadline slack is safeguarded by URG [43], [44]. Although EST_N avoids unnecessary delays, it somewhat overlaps with URG [45], [46]. UP_N is a low-weight precaution as OR and QED already exhibit the anticipated slowness.

3) Determinism and auditability: At each decision step, the scheduler reads task descriptors (QSD, quantity, and week-local deadline), dual-resource availability, and ABS-time operator memory. It quotes QED without mutating operator state, rejects candidates that violate the horizon or availability logs, computes PR(c), and commits the lowest-cost feasible pair. The technique ensures that the same decision is taken each and every time when scores are tied by zeroing out identical numbers and using a predetermined backup checklist.

4) Case identification and interpretability in stochastic manufacturing settings: The following code block outlines the rule set to dispatch an eligible product-task, producing either a committed assignment or an informative outcome label to enable auditability:

- Rule 1: If job, routing, or standard time for the task is missing, then the instance is ill-posed and should be flagged as such.
- Rule 2: Quotations for the job (deterministic quotations for stateless tasks) are to be assessed by the dispatching process.
- Rule 3: Declare the step unfeasible when no candidate meets the planning horizon of the step.
- Rule 4: Record blocking disturbance. Remove tasks that are blocked by tasks within non-working log-intervals of the same task of different jobs of the same product.

- Rule 5: Record the Dual Pair Rank for the viable candidates.
- Rule 6: The score, start time, end time, machine ID, and operator ID of the chosen Dual Pair are determined in a deterministic way to resolve any remaining open issues for the pair selection process and thereby also for the step execution process.
- Rule 7: There is an ABS-time idle gap for forgetting to be applied.
- Rule 8: Commit without recalculating using the saved quote.
- Rule 9: Log any exceptions that have occurred during the commit step and cancel them.

The decision time, job ID, considered candidates, deadline and slack, size of the considered set, selected machine-operator pair, quoted duration for the selected pair, outcome labels for the considered pair(s), tie-breaking information, and information about any forgetting that occurs at commit time are recorded for reproducibility

E. Reward Configuration

The suggested Dynamic Deep 4-layer Q-Network for Dynamic Job Shop Scheduling (DD4LQN-DJSS) relies on a robust reward design to guide the RL agent in identifying the best schedules for a given setting. A hierarchical incentive scheme will enable the agent to evaluate tasks' execution quality, product compliance with deadlines, and overall schedule performance. The industrial context of a flexible job shop is modeled as a dual-resource flexible job shop where each activity is assumed to require an eligible machine as well as an available operator. For this model, a human-centric gap is addressed via a dynamic performance formulation in order to capture its variability. The skill improvement and degradation of the operators are modeled by a Bounded Logistic Learning and Forgetting (BLLF) model, presented in the previous section. This formulation presents the advantage of being compatible with the use of RL-based training in a dynamic DJSS setting.

1) Problem definition and constraints

a) Sets and indices: For a given job shop, let m ∈ M be the set of machines, and k∈ $\mathcal{K}$ be the set of operators. For a given product/jobs j ∈ $\mathcal{J}$, let Tij⊆ $\mathcal{O}$ij be the set of ordered tasks (or operations). The weekly horizon is denoted by H. For a given task (or operation) Tij ∈ $\mathcal{O}$ij, the set of eligible machines is denoted by M{i,j} ⊆ M, and the set of eligible operators is denoted by $\mathcal{K}${i,j} ⊆ $\mathcal{K}$.

b) Parameters: The model contains several parameters that can be used. The due date of a job Dj can be used as a reward shaping parameter, and the batch size qj of a job can also be used. The nominal unit processing time of a task Tij is denoted by stdij, and a stabilizer ε>0 can also be added to keep the reward function stable. Lastly, a dimensionless operator k performance level for an operation Tij on a machine M at time t (minutes) for a unit l ∈ {1…,qj} is denoted by Perf (k,i,j,t,l,m).

c) Decision variables: Assigning tasks to machine-operator pairs is denoted by xijmk; execution of tasks within a given time horizon is denoted by uij; starting and completion times are denoted by Sij and Cij, respectively. Incomplete jobs are denoted by Ij, and completion time and tardiness are denoted by Cj and Ťj, respectively. Machine and operator ordering constraints are modeled by disjunctive variables.

2) Flexible Job Shop constraints and resource capacity

Flexible Job Shop and resource capacity constraints impose:

- Each executed operation is assigned to exactly one eligible machine-operator pair;
- Route precedence compliance within each job;
- Completion Time = Start Time + Effective Processing Time of task at hand;
- No overlap on both machines and operators through disjunctive sequencing;
- Horizon feasibility.

For the purpose of penalizing partial products, Ij=1 and Cj are set to H for any job that is not completed in full by its corresponding decision variables.

3) Reward design: task shaping, product shaping, and schedule-level objective

a) Task-level reward: Eq. (4) presents the task-level reward for a task (i,j) that has been executed. It shows the Quantity of Effective Duration (QED) that has been achieved in comparison to the nominal workload (QSD) for the task (i,j), which is defined as QSD = (stdij * qj), and Effective Duration for a unit as stdij divided by operator unit performanceij. Add up the Effective Duration of all units to get the QED for a task.

$$\text{Rij} = \frac{QSD-QED}{QSD+\varepsilon} \times \text{uij} \quad (4)$$

Late and early completion of tasks are penalized and rewarded, respectively. There is no task reward for unscheduled tasks. The task reward is managed at the product level.

b) Unified product reward: Let's establish λj, μj, and γj as the fractions of early-completed, late-completed, and unscheduled workload using nominal workloads as weights. λj+ μj+ γj = 1 Lateness and earliness for a given task are defined in Eq. (5) below:

$$\Delta\text{j} = \frac{Dj-Cj}{\max(\varepsilon,\text{Dj})} \quad (5)$$

where, Δj^- and Δj^+ represent the normalized lateness and normalized earliness, as follows: $\Delta j^- = \min(0, \Delta j)$ and $\Delta j^+ = \max(0, \Delta j)$. Given that R̄ij is the mean task reward for task (i,j), the unified product reward is defined as follows in Eq. (6):

$$P_{\pi\rho\delta\varphi} = P\square_{\iota\varphi} + (1 + K.\lfloor_{\varphi}).\otimes_{\varphi^+} + (1 + K.\lceil_{\varphi}).\otimes_{\varphi} - K_{\iota\nu\chi} \times \copyright_{\varphi} \quad (6)$$

Notes: 1) The last term vanishes and γj=0 if the product is finished. 2) If incomplete, γj>0 induces the explicit unscheduled-workload penalty, but Cj=H implies Δj becomes the conservative end-of-week due-date term.

c) Schedule reward with tardiness: To make sure the total tardiness is taken into account by the agent, Eq. (7) is

modified to the schedule reward, where $\check{T}$ is the total tardiness (for all the products) and $\bar{R}$prod is the mean product reward.

$$B\check{T} = \frac{H}{H+\check{T}+\varepsilon} \quad (7)$$

Eq. (8) below shows the ultimate schedule reward:

$$P_{\sigma\chi\eta\varepsilon\delta} = P\square_{\pi\rho\sigma\delta} + B\square \quad (8)$$

In summary, our model uses a hierarchical reward structure to force the DD4LQN scheduler to optimize global due-date satisfaction, product completion quality, and local execution quality in order to achieve a single hierarchical goal.

F. Internal Simulator State and Partial Observation

Operator forgetting and learning are influenced by previous task assignments. As a result, the simulator keeps the data from the past that may influence the subsequent change. At the moment of choice t, the entire internal state is presented in Eq. 9:

$$x_t = (\tau_t, J_t, M_t, O_t, H_t, Q_t) \quad (9)$$

The components are as follows: O_t, the corresponding operator information; H_t, operator-task proficiency, accumulated learning exposure, last absolute completion time, and elapsed idle time; Q_t, pending events and remaining disruption information; J_t, product progress, current operations, quantities, deadlines, completion times, and feasibility flags; and M_t, machine status, next-free time, current assignment, and remaining availability calendar.

The entire simulator transition -*presented in Eq. 10*- meets the following since x_t contains all dynamic variables that are used to calculate the next event, effective duration, resource assignment, reward, and operator-memory update:

$$P(x_{t+1} \mid x_{0:t}, a_{0:t}) = P(x_{t+1} \mid x_t, a_t) \quad (10)$$

The Dynamic Deep Four-Layer Q-Network (DD4LQN) does not observe x_t directly. Its compact observation is presented in Eq. (11):

$$o_t = \varphi(x_t) = [b_t^O, b_t^M, g_t] \quad (11)$$

In this case, operator-availability indications are in b_t^O, machine-availability indicators are in b_t^M, and product-operation progress is in g_t. The same output can have varying effective lengths depending on the operator's history. DD4LQN product selection is a partially observable Markov decision process (POMDP), although the simulator is still Markovian in x_t. Only the product identifier is chosen by DD4LQN. The machine-operator pair is assigned using the rule-based Dual Pair Rank using all of the simulator data.

IV. DRL Agent and Training Protocol

A. Action Space and Environment-Step Semantics

The scheduling system is not end-to-end learnt, but rather hybrid. DD4LQN chooses a product from the compact observation o_t described in Section III-F at each decision epoch. The environment maintains the entire internal state x_t for machine-operator assignment, operator-memory updates, duration computation, and feasibility checking. The suggested DD4LQN scheduler is a decision-making process that is event-driven. The AI selects a product ID from a predetermined universe (P1–P20) at each decision point. It identifies the dispatchable job of the product (the subsequent unscheduled activity in compliance with priority restrictions) and chooses the dispatch focus without assigning any machines or operators. The environment keeps moving to the remaining products if the chosen one is already finished or flagged. Additionally, the environment processes completion events to maintain an ongoing weekly timetable. For every dispatchable task assignment attempt, the agent first determines the earliest start time by assessing the viability of each (machine, operator) candidate; DD4LQN does not choose the machine-operator pairing. The candidate interval must respect the weekly horizon as well as the status-log window (machine and/or operator unavailability). Dual Pair Ranking is used as tie break to select the most accurate pairing for a given task, leading to operator task and machine memory calendars being updated after completion. Learning-forgetting updates are only updated at the commit stage to maintain consistent action labels.

The quoted duration is subsequently booked by Quote-then-Commit, which also modifies the operator-task memory, machine calendar, and operator calendar. As a result, the rule-based approach is used for the resource allocation and feasibility check, while the learnt contribution is restricted to the product-level dispatch priority. Thus, the incremental effect of the learned component is evaluated against EDD and SPT product-selection rules while keeping the same scenario, Dual Pair Rank, learning-forgetting model, Quote-then-Commit rule, and tie-breaking convention.

B. DD4LQN Network Architecture

A fixed-size observation vector comprising 1) machine-availability indicators, 2) operator-availability indicators, and 3) fixed product-operation progress windows is sent to DD4LQN. The Q-function is approximated using a grouped feature encoder in a Dynamic Deep 4-Layer Q-Network (DD4LQN). These blocks are processed by distinct operator, machine, and product observation encoders before shared trunk layers combine their outputs. In order to prevent conflict at the first level of representation, this grouping makes sure that job progression and resource availability are caused by various physical factors. An advantage stream and an observation-value stream are created by the dueling head from the Q-values. This enables the network to determine the baseline attractiveness of the compact observation as well as the proportionate benefit of each product activity. This is particularly useful for dispatching, where the learning signal is often driven by fine ranking of actions rather than absolute value, and several actions may be similarly bad under congestion. Fig. 3 outlines the entire architecture to link learning and execution explicitly – at both the policy level and the workforce level.

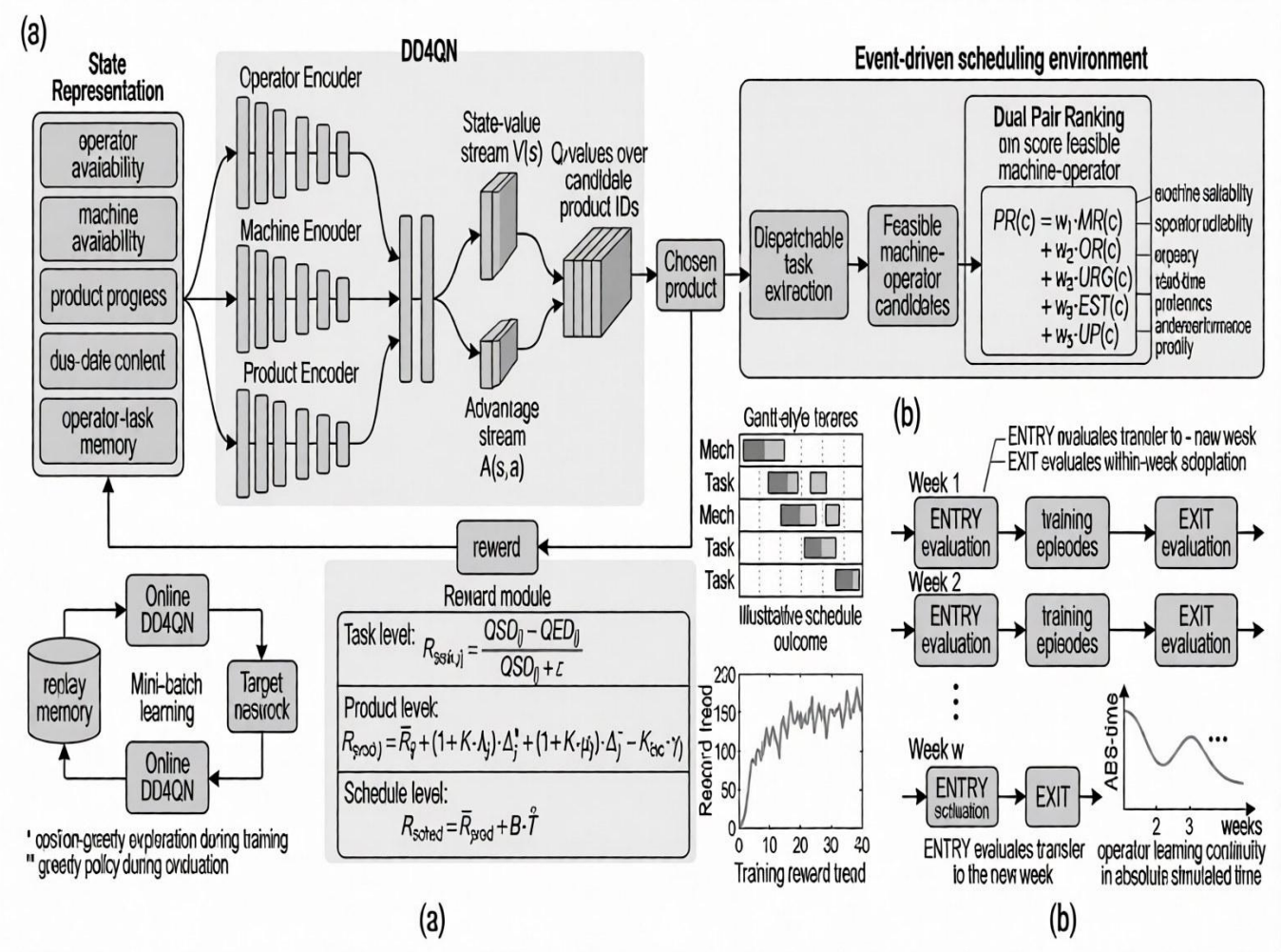

Fig. 3. DD4LQN Architecture and weekly protocol: (a) architecture, environment, ranking, reward, (b) weekly ENTRY/EXIT curriculum and ABS learning continuity.

The compact DD4LQN observation o_t provided to the neural network from the entire internal simulator state x_t utilized by the environment is shown in Fig. 3.

Moreover, Fig. 3 illustrates two interconnected learning loops: a micro loop that lies beneath the ABS-time learning-forgetting continuity during the weekly ENTRY/EXIT evaluations, and a macro-RL loop based on replay memory, minibatch learning, and a target network. The training employs a replay buffer containing 10,000 transitions, prioritized replay, minibatches of 32 samples, γ=0.99, lr=10−4, SmoothL1 loss, and has the target network refreshed every 50 episodes [17], [48]. The curriculum includes ten weekly scenarios, each consisting of 1000 episodes, and the greedy ENTRY/EXIT assessments (with ε = 0) are used to distinguish between within-week adaptation and cross-week transfer.

V. Experimental Setup and Results

A. Reporting Protocol and Logged Indicators

Week 1 is considered the cold-start reference, while week 10 is the late-curriculum reference; the comparison between the two aims to make the learning path comprehensible. Reporting includes numerical representation of schedule viability and resource allocation. This link is established through product- and schedule levels rewards trajectories, ENTRY/EXIT KPI tables, and machine and operator Gantt charts. Reporting includes product tardiness, flagged products, completion rate, sample efficiency (E90), and convergence stability (CV (last20)). These metrics are meant to evaluate training stability, feasibility robustness, adaptation quality and due-date management.

B. Statistical Evaluation Unit

To evaluate algorithmic variability independently of the stochastic environment, every configuration was assessed five times each week (K=5) to determine the degree of variation in each policy's outcomes. During these five rounds, the weekly scenario did not vary, guaranteeing a fair comparison of the policies.

Using the fixed seed 123, ten consecutive weekly scenarios were produced, and the same weekly demand, resource availability, and operating conditions were used to evaluate every policy. The mean (μ) and standard deviation (σ) were computed from the five evaluation runs for each week. The outcomes across the ten weekly means are then summarized in

the cross-week tables. The calculations of μ and σ are defined by Eq. (10) and (11), respectively:

$$\mu = \frac{1}{K}\sum_{1}^{K} Xw,i \qquad (10); \sigma = \sqrt{\frac{\sum_{1}^{K}(Xw,i-\mu)^2}{K}} \qquad (11)$$

Here, Xw,i is the KPI obtained during rollout i of week w (e.g., total tardiness, mean completion rate).

1) Cold-start learning behavior: Week 1: Week 1 is used as the baseline for the agent's behavior during the early stages of exploration when the value targets are unstable. The reward trajectory varies greatly from one episode to the next, exhibiting a weak ENTRY baseline and frequent oscillations, as shown in Fig. 4.

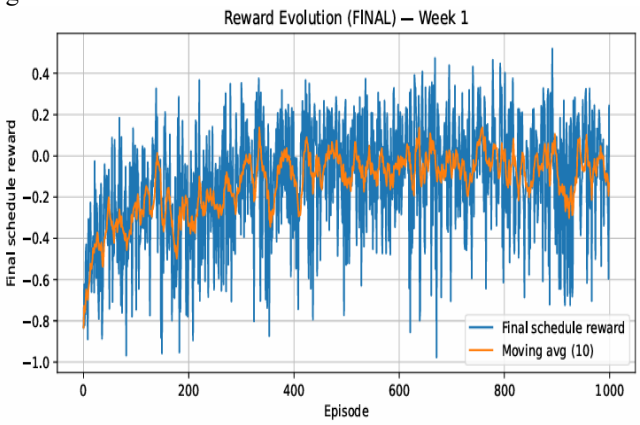

Fig. 4. Week 1 reward evolution.

Fig. 4 indicates that Week 1 emphasizes learning and exploration rather than using an existing scheduling policy. Table II shows the benchmark KPIs under greedy evaluation (ε=0) before and after training to measure this initial behavior. It provides a presentation of the schedule reward, the total tardiness, the number of flagged items, the completion, and the convergence stability CV (last20), as well as the sample efficiency E90.

TABLE II. BENCHMARK KPI SUMMARY FOR WEEK 1

KPI	ENTRY(Ɛ=0)	EXIT (Ɛ=0)
Schedule reward (mean ± std)	-0,8925 ± 0,1948	0,1194 ± 0,1699
Total tardiness (mean ± std)	384,0 ± 192,0	96,0 ± 192,0
Flagged products (mean ± std)	9,40 ± 1,02	5,40 ± 1,02
Completion rate	0,22	0,57
Stability CV (last 20 episodes)	2,415	
Sample efficiency E90 (Episodes)	668	

Overall tardiness decreases from 384±192 to 96±192, while scheduling reward rises from −0.8925±0.1948 at ENTRY to 0.1194±0.1699 at EXIT, indicating a clear within-week learning effect. While flagged products drop from 9.40±1.02 to 5.40±1.02, the completion rate rises from 0.22 to 0.57. However, the convergence indicators remain poor (CV (last20) =2.415, E90=668), suggesting that Week 1 still struggles with early-stage learning instability rather than consolidated policy maturity.

Fig. 5 shows the ideal ABS-time Week 1 product-machine schedule (R=0. 5194).

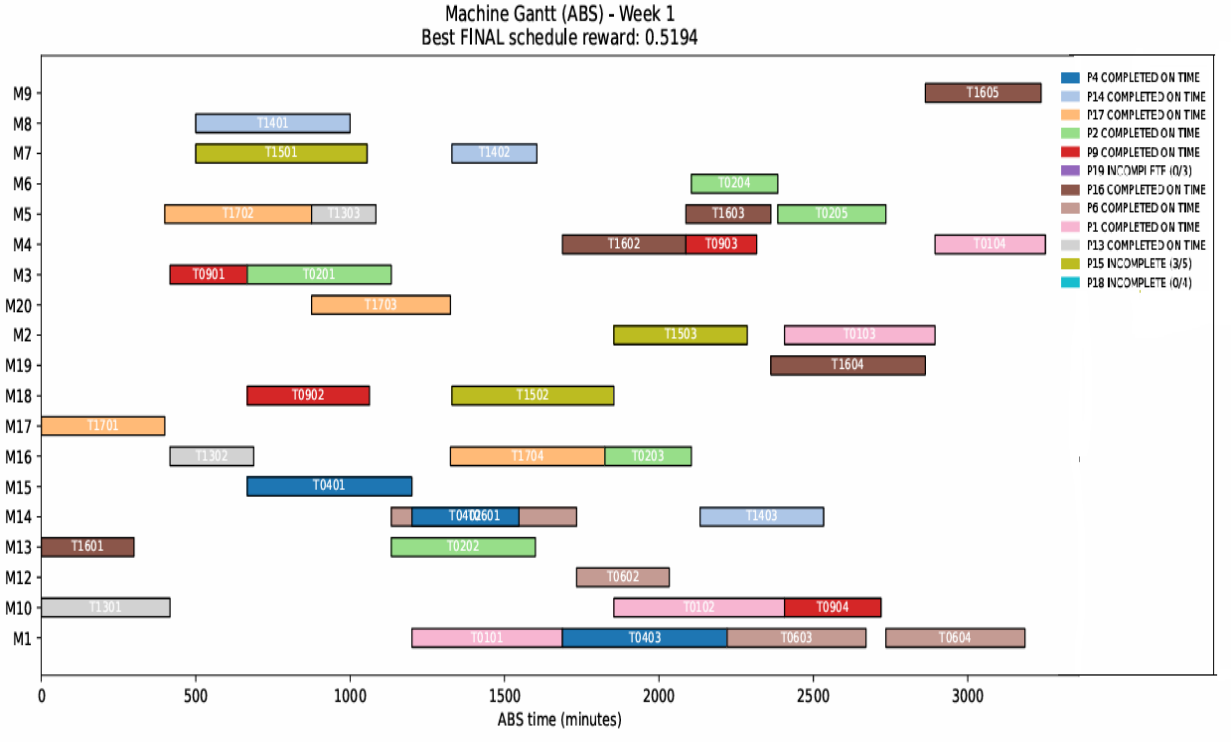

Fig. 5. Product-machine GANTT for week 1.

Since the assignment is highly dependent on both operator and machine availability, Fig. 6 displays the ideal ABS-Time product-operator Gantt for week 1, enabling structural interpretation beyond scalar reward values. This representation is vital to get the full picture of the operator and machine availability checks.

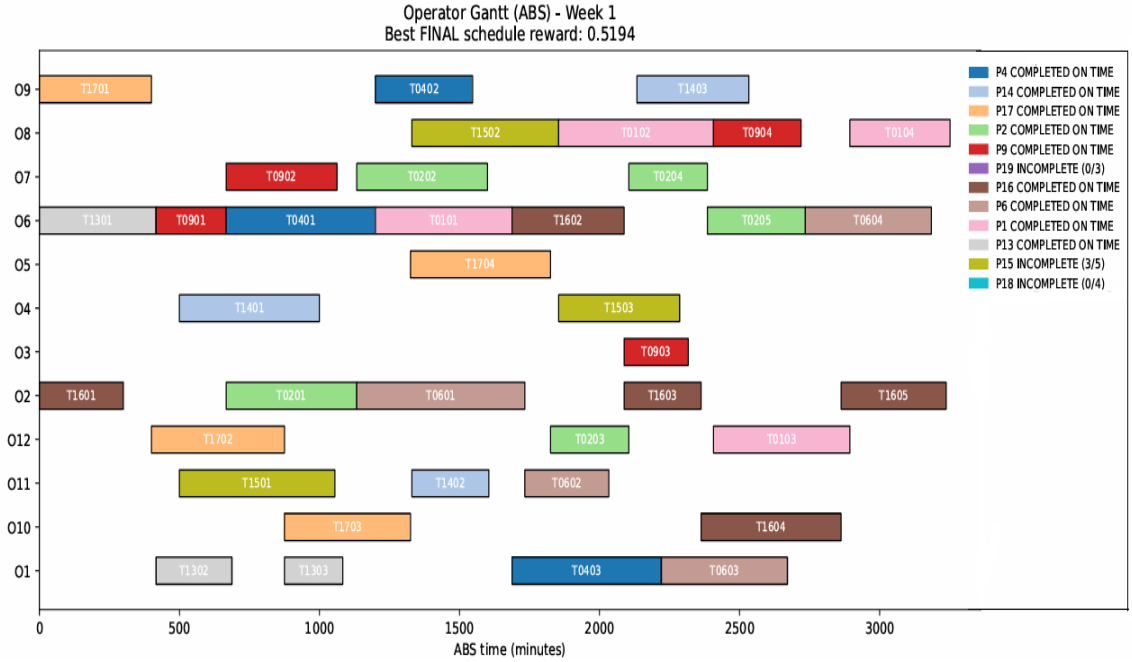

Fig. 6. Product-operator GANTT for week 1.

Fig. 5 and 6 present a structural description of the optimal Week 1 schedule. Feasibility losses are depicted in the product-machine Gantt, indicating dense processing blocks with idle spans due to operators' unavailability. The product-operator Gantt completes the schedule scope by indicating machine-operator cooperation. The best schedule indicates a small percentage of uncompleted products.

This constatation imposes a verification of product reward; Fig. 7 shows whether the product reward is distributed or concentrated on a small number of higher ones.

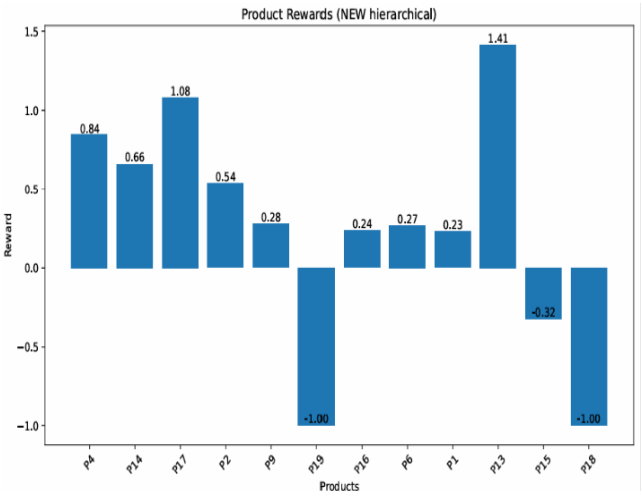

Fig. 7. Week 1 best schedule product rewards.

Fig. 7 illustrates a bimodal distribution of product rewards. High-performing products such as P13 = 1.41 and P17 = 1.08 demonstrate successful end-to-end routing, while unfinished items receive severe rewards (P15 = −0. 32, P19 = −1, P18 = −1).

When combined, Table II and Fig. 5 and 6 show that EXIT gains are a sign of broader completion improvements.

2) Late-curriculum learning behavior: Week 10: Under disrupted dual-resource conditions, the schedule reward trajectory in Fig. 8 is the first sign that the agent goes from week-specific adaptation to a certain setting to steady convergence and reliable cross-week transfer. The late-curriculum benchmark is Week 10 present a smoother pattern of the schedule reward evolution that remains concentrated at a much higher mean level. This presents an improvement from the extremely erratic low-level curve displayed in Week 1.

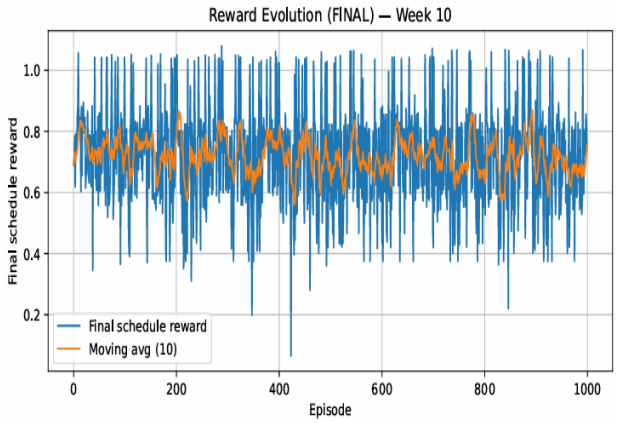

Fig. 8. Week 10 reward evolution.

This figure clearly illustrates a shift from exploration-driven instability to policy-driven patterns.

To get a full picture, our analysis includes the Week 10 ENTRY/EXIT benchmark KPIs under greedy evaluation (ε=0) summarized in Table III.

TABLE III. BENCHMARK KPI SUMMARY FOR WEEK 10

KPI	ENTRY(Ɛ=0)	EXIT (Ɛ=0)
Schedule reward (mean ± std)	0.6520 ± 0.1925	0.9521 ± 0.1031
Total tardiness (mean ± std)	288.0 ± 235.2	96.0 ± 192.0
Flagged products (mean ± std)	3.80 ± 0.75	2.40 ± 0.49
Completion rate	0.71	0.82
Stability CV (last 20 episodes)	0,188	
Sample efficiency E90 (Episodes)	9	

Week 10 begins with a stronger foundation than Week 1, as evidenced by the ENTRY schedule reward, which is already 0.6520 ± 0.1925 (higher that week 1 schedule reward by the ENTRY: -0,8925 ± 0,1948) and rises significantly to 0.9521 ± 0.1031at EXIT (higher that week 1 schedule reward by the EXIT: 0,1194 ± 0,1699). The completion rate rises from 0.71 to 0.82 (higher than week 1 values, which went from 0.22 to 0.57). The total tardiness goes from an ENTRY value of 288.0 ± 235.2 to an EXIT value of 96.0 ± 192.0, compared to a decrease in tardiness for week 1 from 384±192 to 96±192. Furthermore, products that are flagged decrease from 3.80 ± 0.75 to 2.40 ± 0.49, compared to Week 1 flagged product evolution 9.40±1.02 to 5.40±1.02.These observations remain consistent with the much lower convergence variance reported for Week 10 (CV(last20)=0.188, E90=9) compared with the Week 1 previously displayed value (CV(last20)=2. 415, E90=668).

Similarly, Fig. 9 shows the ideal ABS-time Week 10 product-machine schedule (R=1. 0782). To complete the visual display in accordance with dual resource availability. To complete the visual display in accordance with dual resource availability, Fig. 10 displays the ideal ABS-Time product-operator Gantt for week 10.

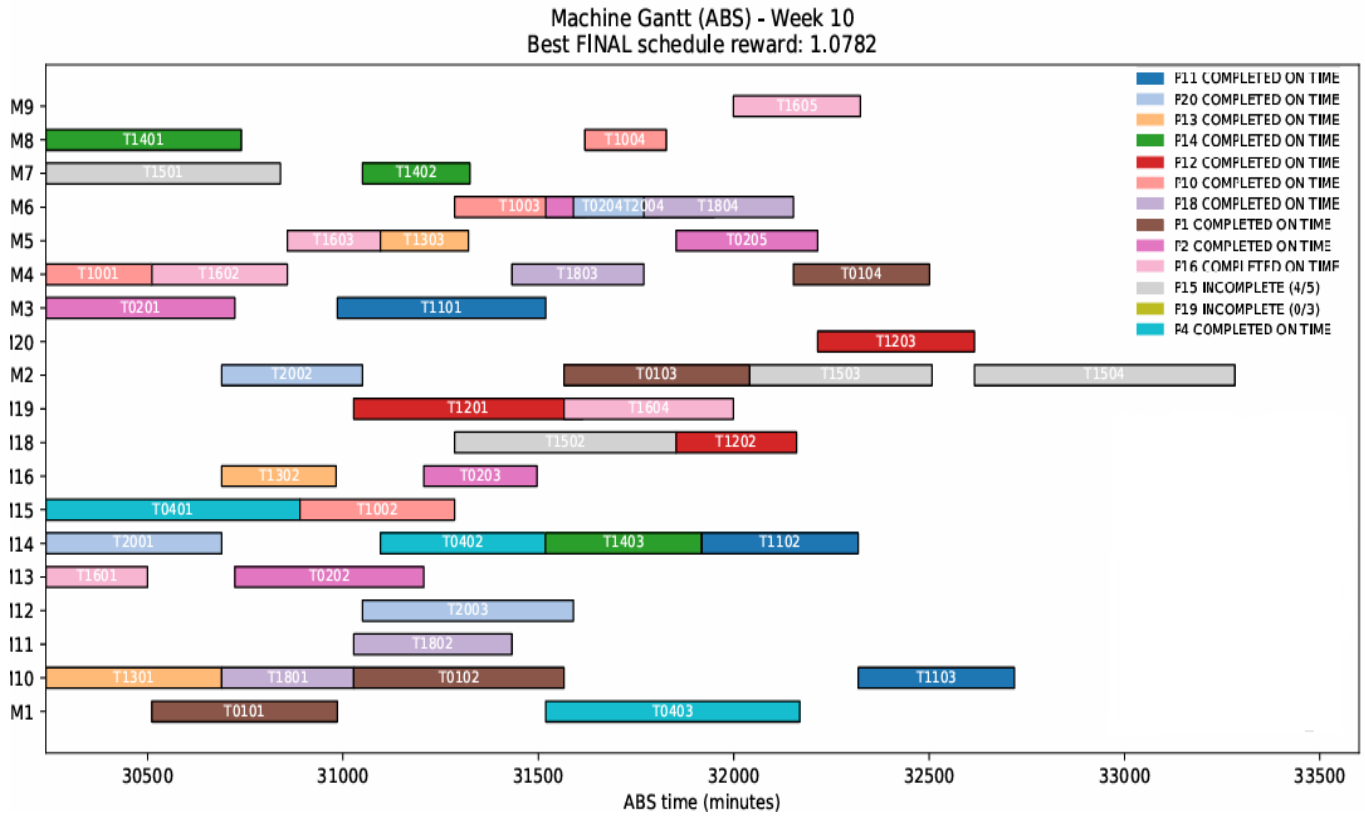

Fig. 9. Product-machine GANTT for week 10.

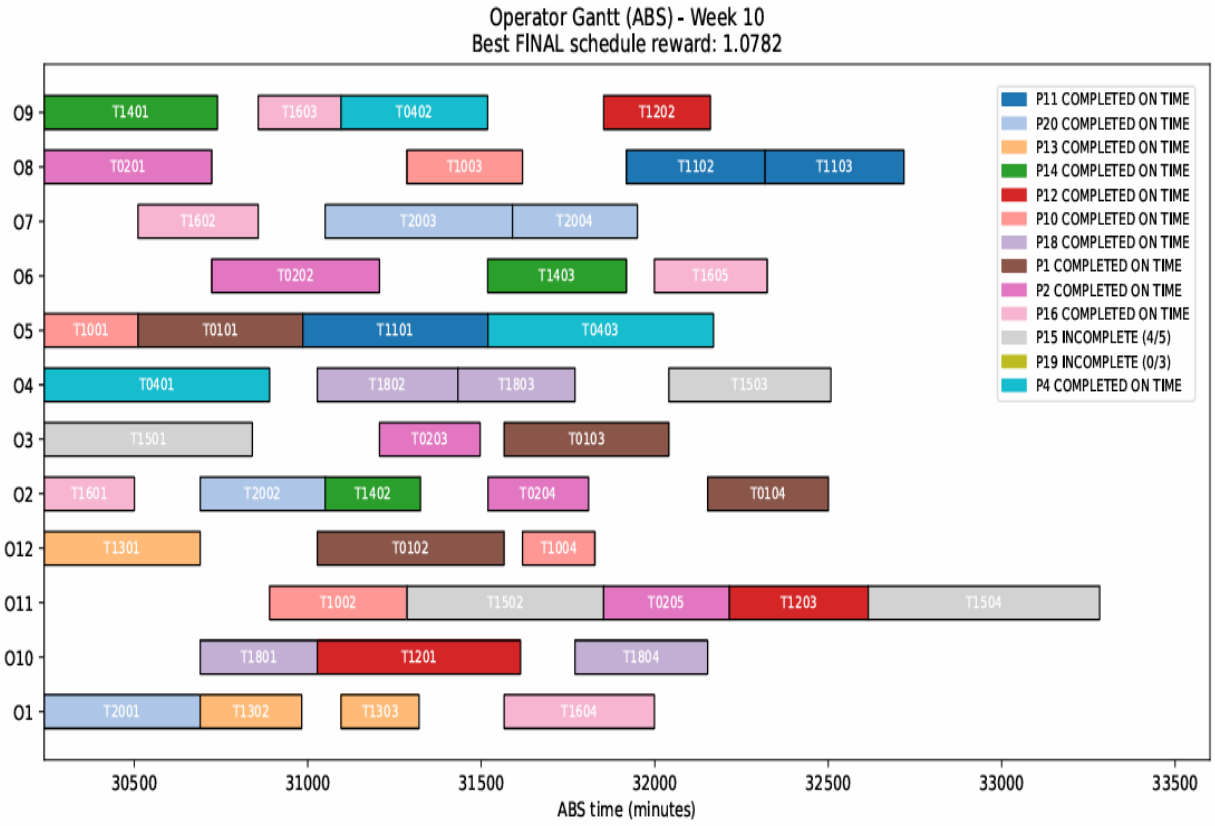

Fig. 10. Products-operator GANTT for week 10.

Fig. 9 displays a more rational machine-capacity allocation than in Week 1; in the best schedule displayed, all 11 items are completed on time, which is in line with Table III's results. The complementary operator-side perspective in Fig. 10 confirms that this result is achieved through practical machine-operator coordination rather than hidden overload or conflicts. The product reward of the 10th week distribution is depicted in Fig. 11, just like in week 1.

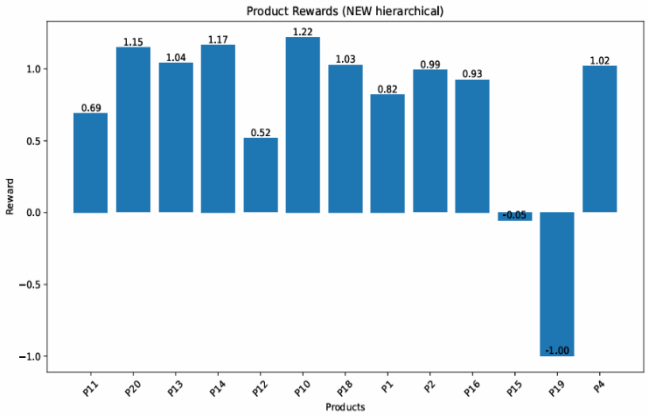

Fig. 11. Week 10 best schedule product rewards.

With several products grouped around reward values going from 0.52 to 1.22, with the exception of P15=-0,05 and P19=-1, the distribution in the figure is consistently favorable. This is a major improvement over the bimodal Week 1 pattern, where incomplete items faced severe penalties and a small number of products received strong positive rewards. As a result, Fig. 11 demonstrates that by Week 10, the policy improves feasibility and reward quality across the entire weekly mix rather than just the most basic products.

C. Cross-Week Benchmark Comparison: DD4LQN-EXIT, EDD, and SPT

We performed a post-hoc comparison between the trained DD4LQN policy and two common dispatching rules: Earliest Due Date (EDD) and Shortest Processing Time (SPT). While maintaining the same downstream scheduling processes, the experiment isolates decision-making at the product level. Each policy chose the subsequent product to be processed at each decision event:

- DD4LQN-EXIT: Applied the Deep Q-Learning policy that had been trained.
- EDD: The active product with the earliest local deadline was chosen.
- SPT: The product with the shortest nominal processing time was chosen.

Everything else in the system was maintained strictly the same across five test runs in order to maintain test fairness. The rules for selecting machine-worker pairs, managing breaks, monitoring worker skill loss, establishing deadlines, breaking ties, and computing scores were the same for each choice, and their weekly performance is displayed in Fig. 12.

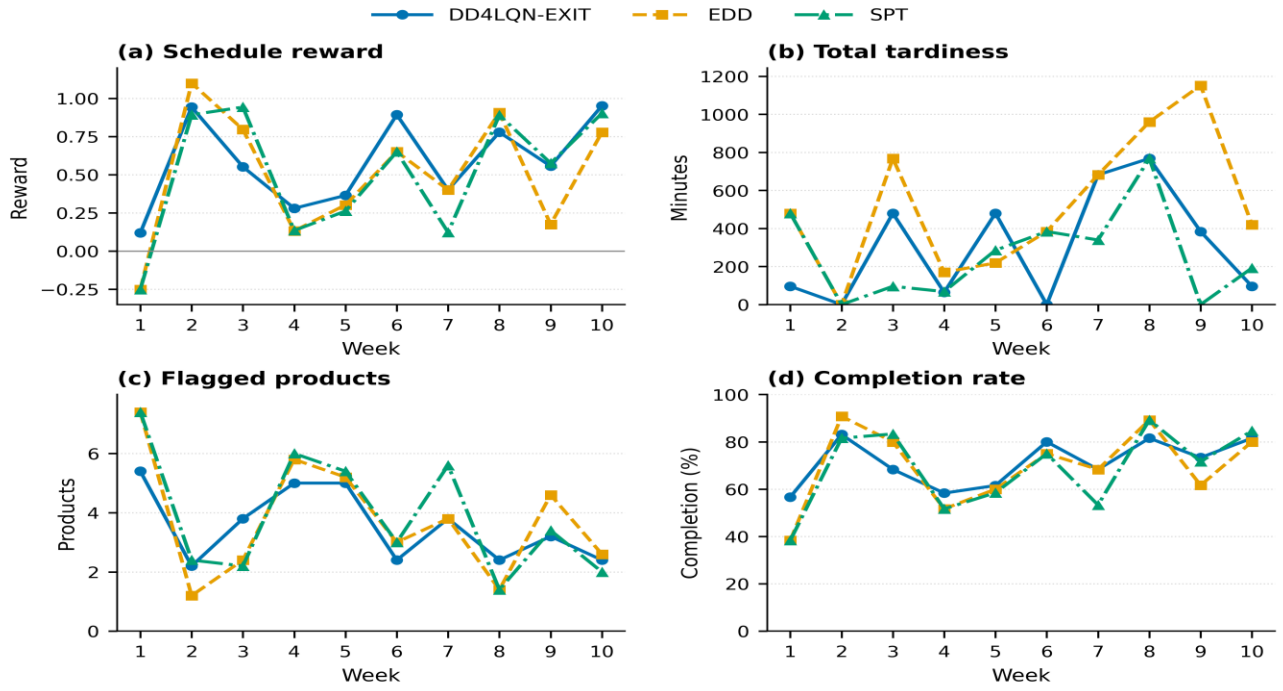

Fig. 12. Weekly performance of DD4LQN-EXIT, EDD, and SPT under identical downstream scheduling conditions. Each data point reflects the mean of five evaluation rollouts.

Fig. 12 shows that DD4LQN-EXIT outperforms baseline heuristics in terms of performance stability and balance throughout 10-week scenarios across all four evaluation metrics. SPT prioritizes short jobs to reduce tardiness during low-demand weeks, but under high workloads, it experiences important losses. On the other hand, during interruption events, EDD faces significant reward declines and significant tardiness spikes (exceeding 900 minutes in Weeks 8 and 9). By limiting overall tardiness, continuously reducing risk-flagged items (~5.5 in volatile Week 1 compared to >7 for EDD and SPT), and sustaining superior completion rates under high-workload conditions (~57% versus <40%), DD4LQN-EXIT reduces these failure modes.

Thus, robust multi-objective schedule stability is ensured by learnt product selection, which eliminates localized trade-off extremes. Table IV offers a thorough quantitative comparison of all rollouts by summarizing the whole ten-week aggregate metrics.

TABLE IV. CONTROLLED TEN-WEEK COMPARISON OF PRODUCT-SELECTION POLICIES

KPI	DD4LQN-EXIT	EDD	SPT
Schedule reward (higher)	0.5843 ± 0.2963	0.4995 ± 0.4173	0.5126 ± 0.4201
Total tardiness, min (lower)	305.0 ± 289.7	523.5 ± 364.3	261.3 ± 243.2
Flagged products (lower)	3.56 ± 1.23	3.74 ± 2.00	3.88 ± 2.05
Completion rate, % (higher)	71.27 ± 10.13	69.50 ± 16.78	68.72 ± 17.18

DD4LQN-EXIT had the best multi-objective balance over the course of the ten-week test. It increased completion by 1.77 percentage points, decreased mean tardiness by 41.7%, and outperformed EDD by 17.0% in schedule reward. DD4LQN-EXIT increased schedule reward by 14.0%, reduced risk-flagged goods by 8.2%, and increased completion by 2.55 percentage points in comparison to SPT. Although SPT produced the lowest average delay (261.3 vs. 305.0), incomplete orders are not included in this statistic. Prioritizing short, simple tasks leads to a lower total completion rate (68.72%), which is the reason for SPT's lower tardiness. In the end, learnt product selection provides the most consistent overall performance under dynamic shop conditions and avoids the failure modes of single-rule heuristics.

D. Exploratory Component Ablation Study

Table V uses the same ten frozen weekly scenarios created with seed 123 to compare the whole framework (whole) with three runtime ablations. Demand, due dates, disruptions, and resource availability are all held constant while five matching rollouts are averaged for each weekly value. DD4LQN solely chooses the product. Resource allocation is still determined by established standards. The earliest possible pair is chosen in the absence of Dual Pair Rank (NO_DPR). Duration is computed at commitment in the absence of Quote-then-Commit (NO_QTC). Operator performance is unchanged in the absence of Learning–Forgetting (NO_BLLF).

TABLE V. CONTROLLED ABLATION OF THE PROPOSED FRAMEWORK

Configuration	Controlled modification	Schedule reward, mean (Δ)	Completion rate, % (Δ)	Tardiness, min (Δ)	Flagged products (Δ)
FULL	**All components active**	0.584 (reference)	71.27 (reference)	305.02 (reference)	3.56 (reference)
NO_DPR	**Replace Dual Pair Rank with a deterministic feasible-pair rule**	0.437 (−25.2%)	66.83 (−4.44 pp)	315.40 (+3.4%)	4.08 (+0.52)
NO_QTC	**Recompute duration at commitment**	0.584 (0.0%)	71.27 (0.00 pp)	305.02 (0.0%)	3.56 (0.00)
NO_BLLF	**Hold operator performance constant**	−0.051 (−108.8%)	53.09 (−18.18 pp)	485.44 (+59.2%)	5.80 (+2.24)

Note: FULL serves as the benchmark for changes. In contrast to lesser tardiness and fewer flagged products, higher rewards and completion are desirable.

Table V shows that NO_DPR decreased completion overall and decreased reward each week. This outcome distinguishes the two scheduling levels: Dual Pair Rank enhances the succeeding machine-operator assignment, whereas DD4LQN gives the product precedence. The biggest degradation was produced by NO_LLF, demonstrating the significance of operator-performance history. Since NO_QTC did not result in a change in KPIs, its observed contribution was duration consistency and auditability.

E. Weight Interpretation and Policy Representation

A thorough examination of the final DD4LQN model checkpoint (227, 717 parameters across 28 tensors) verifies that the neural network topology is robust, well-conditioned, and free of representation collapse. Important findings include:

- Balanced Weight Distribution: All layer parameter means are close to zero (-0.0094 to 0.0006). The time encoder exhibits a greater raw weight variation (0.2385) than the machine (0.0904), operator (0.0950), and product (0.0554) encoders; nevertheless, this is solely dependent on the amount of the input.
- Normalized Variance: All feature encoders' weight variance closely aligns between 0.554 and 0.604 when scaled by input dimensions (fan-in), matching typical initialization baselines. Both the dueling output streams and the core network trunk retain consistent, small weight spreads (0.0362–0.0378).
- High Representation Capacity: Layer matrices show that the network preserves rich feature diversity instead of collapsing into redundant patterns, retaining between 77.2% and 98.2% of their maximum possible rank.

Although they address parameter geometry rather than training dynamics or feature significance, these structural tests verify appropriate network health and capacity use.

VI. CONCLUSION AND FUTURE WORK

This study introduced a human-aware Deep Reinforcement Learning framework (DD4LQN) to solve scheduling problems in dual-resource flexible job shops with unpredictable events. By using Bounded Logistic Learning and Forgetting (BLLF) patterns and Dual Pair Ranking (DPR) tasks, the approach connects performance with automated decisions. Over ten weeks, our framework showed overall better performance than traditional dispatching rules (EDD and SPT). DD4LQN-EXIT achieves the highest schedule reward, minimizes risk-flagged products, and maximizes job completion rates during high-workload periods. Even though SPT had the shortest average delay, it did so by focusing on short tasks and not completing as many jobs. EDD exhibited severe tardiness peaks and reward losses during dynamic disruptions. The proposed framework successfully avoided these extreme failure modes, proving that learned product selection delivers robust multi-objective schedule stability under dynamic shop-floor conditions. The exploratory ablation study associated learning-forgetting dynamics and Dual Pair Rank with improved scheduling outcomes in the tested scenarios, but Quote-then-Commit did not result in any change in KPIs, and hierarchical reward shaping did not demonstrate any independent advantage over the retrained simple-reward policy.

With these good results, some problems still exist. First, the learning and forgetting rates for workers were made using situations instead of real data from factories. Second, the current setup does not include real-life issues like setup times between tasks, delays in moving materials, machine repairs, and working with more than one person. Finally, even though the way tasks are ranked makes decisions clear and easy to check, the policy was tested as one part of the system and not as a solution for all parts.

Future work will focus on addressing these limitations to ensure a larger scope of this framework application. The first step is to use data from factories to make the learning and forgetting rates more accurate. Furthermore, the manufacturing limitations will be expanded in future simulations to include logistical routing, setup times, and intricate multi-worker processes. Finally, investigating the transition from the existing reinforcement learning architecture to a multi-agent architecture will be considered.

REFERENCES

[1] A. S. Jain et S. Meeran, « A STATE-OF-THE-ART REVIEW OF JOB-SHOP SCHEDULING TECHNIQUES ».

[2] P. Fattahi, M. Saidi Mehrabad, et F. Jolai, « Mathematical modeling and heuristic approaches to flexible job shop scheduling problems », J Intell Manuf, vol. 18, no 3, p. 331-342, juill. 2007, doi: 10.1007/s10845-007-0026-8.

[3] H. Taji, G. Ayad, et A. Zaki, « Optimal scheduling in a Collaborative robot environment and evaluating workforce dynamic performance », ITM Web Conf., vol. 46, p. 01004, 2022, doi: 10.1051/itmconf/20224601004.

[4] H. Taji, G. Ayad, A. Zaki, et A. Khammal, « Smart Job Shop Scheduling: Integrating Dynamic Human Performance and Deep Q-Network for Industry 4.0 », in Advanced Metaheuristics for Scheduling in Distributed

Manufacturing Systems, S. Aqil et M. Lahby, Éd., IGI Global Scientific Publishing, 2025, p. 229-276. doi: 10.4018/979-8-3373-3136-2.ch008.

[5] A. Zaki, M. Benbrahim, et B. Benchekroun, « PROPOSITION OF A MODEL FOR MULTI-PERIOD WORKFORCE ASSIGNMENT PROBLEM CONSIDERING VERSATILITY », . Vol., no 7, 2005.

[6] M. Bögl, A. Gattinger, I. Knospe, M. Schlenkrich, et R. Stainko, « Real-life scheduling with rich constraints and dynamic properties – an extendable approach », Procedia Computer Science, vol. 180, p. 534-544, 2021, doi: 10.1016/j.procs.2021.01.272.

[7] M. R. Garey, D. S. Johnson, et R. Sethi, « The Complexity of Flowshop and Jobshop Scheduling », Mathematics of OR, vol. 1, no 2, p. 117-129, mai 1976, doi: 10.1287/moor.1.2.117.

[8] J. Mohan, K. Lanka, et A. N. Rao, « A Review of Dynamic Job Shop Scheduling Techniques », Procedia Manufacturing, vol. 30, p. 34-39, 2019, doi: 10.1016/j.promfg.2019.02.006.

[9] R. A. Liaqait, S. Hamid, S. S. Warsi, et A. Khalid, « A Critical Analysis of Job Shop Scheduling in Context of Industry 4.0 », Sustainability, vol. 13, no 14, p. 7684, juill. 2021, doi: 10.3390/su13147684.

[10] J. R. Montoya-Torres, S. Sánchez, et C. Moreno-Camacho, « A literature-based assessment of human factors in shop scheduling problems », IFAC-PapersOnLine, vol. 52, no 10, p. 49-54, 2019, doi: 10.1016/j.ifacol.2019.10.025.

[11] M. Hernandez-de-Menendez, R. Morales-Menendez, C. A. Escobar, et M. McGovern, « Competencies for Industry 4.0 », Int J Interact Des Manuf, vol. 14, no 4, p. 1511-1524, déc. 2020, doi: 10.1007/s12008-020-00716-2.

[12] K. Hazeghi et M. L. Puterman, « Markov Decision Processes: Discrete Stochastic Dynamic Programming. », Journal of the American Statistical Association, vol. 90, no 429, p. 392, mars 1995, doi: 10.2307/2291177.

[13] R. S. Sutton et A. G. Barto, « Reinforcement Learning: An Introduction ».

[14] V. Modrak, R. Sudhakarapandian, A. Balamurugan, et Z. Soltysova, « A Review on Reinforcement Learning in Production Scheduling: An Inferential Perspective », Algorithms, vol. 17, no 8, p. 343, août 2024, doi: 10.3390/a17080343.

[15] M. Panzer, B. Bender, et N. Gronau, « Deep Reinforcement Learning In Production Planning And Control: A Systematic Literature Review », 2021, doi: 10.15488/11238.

[16] L. Lv, C. Zhang, J. Fan, et W. Shen, « Deep reinforcement learning for job shop scheduling problems: A comprehensive literature review », Knowledge-Based Systems, vol. 321, p. 113633, juin 2025, doi: 10.1016/j.knosys.2025.113633.

[17] T. Schaul, J. Quan, I. Antonoglou, et D. Silver, « Prioritized Experience Replay », 25 février 2016, arXiv: arXiv:1511.05952. doi: 10.48550/arXiv.1511.05952.

[18] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, et D. Meger, « Deep Reinforcement Learning that Matters », 2017, arXiv. doi: 10.48550/ARXIV.1709.06560.

[19] K. Genova, L. Kirilov, et V. Guliashki, « A Survey of Solving Approaches for Multiple Objective Flexible Job Shop Scheduling Problems », Cybernetics and Information Technologies, vol. 15, no 2, p. 3-22, juin 2015, doi: 10.1515/cait-2015-0025.

[20] S. Saeidlou, M. Saadat, et G. D. Jules, « Towards decentralised job shop scheduling as a web service », Cogent Engineering, vol. 8, no 1, p. 1938795, janv. 2021, doi: 10.1080/23311916.2021.1938795.

[21] F. Ren et H. Liu, « Dynamic scheduling for flexible job shop based on MachineRank algorithm and reinforcement learning », Sci Rep, vol. 14, no 1, p. 29741, nov. 2024, doi: 10.1038/s41598-024-79593-8.

[22] P. Nagarajan, G. Warnell, et P. Stone, « Deterministic Implementations for Reproducibility in Deep Reinforcement Learning », 2018, arXiv. doi: 10.48550/ARXIV.1809.05676.

[23] A. S. Jain et S. Meeran, « Deterministic job-shop scheduling: Past, present and future », European Journal of Operational Research, vol. 113, no 2, p. 390-434, mars 1999, doi: 10.1016/S0377-2217(98)00113-1.

[24] L. B. Liboni, L. O. Cezarino, C. J. C. Jabbour, B. G. Oliveira, et N. O. Stefanelli, « Smart industry and the pathways to HRM 4.0: implications for SCM », Supp Chain Mnagmnt, vol. 24, no 1, p. 124-146, janv. 2019, doi: 10.1108/SCM-03-2018-0150.

[25] N. Asadayoobi, M. Y. Jaber, et S. Taghipour, « A new learning curve with fatigue-dependent learning rate », Applied Mathematical Modelling, vol. 93, p. 644-656, mai 2021, doi: 10.1016/j.apm.2020.12.005.

[26] C. H. Glock, E. H. Grosse, M. Y. Jaber, et T. L. Smunt, « Applications of learning curves in production and operations management: A systematic literature review », Computers & Industrial Engineering, vol. 131, p. 422-441, mai 2019, doi: 10.1016/j.cie.2018.10.030.

[27] M. Asghari, H. Afshari, M. Y. Jaber, et C. Searcy, « Learning and forgetting interactions within a collaborative human-centric manufacturing network », European Journal of Operational Research, vol. 313, no 3, p. 977-991, mars 2024, doi: 10.1016/j.ejor.2023.09.020.

[28] P. Brandimarte, « Routing and scheduling in a flexible job shop by tabu search », Ann Oper Res, vol. 41, no 3, p. 157-183, sept. 1993, doi: 10.1007/BF02023073.

[29] J. Hurink, B. Jurisch, et M. Thole, « Tabu search for the job-shop scheduling problem with multi-purpose machines », OR Spektrum, vol. 15, no 4, p. 205-215, déc. 1994, doi: 10.1007/BF01719451.

[30] T. P. Wright, « Factors Affecting the Cost of Airplanes », Journal of the Aeronautical Sciences, vol. 3, no 4, p. 122-128, févr. 1936, doi: 10.2514/8.155.

[31] C. H. Glock, E. H. Grosse, T. Kim, W. P. Neumann, et A. Sobhani, « An integrated cost and worker fatigue evaluation model of a packaging process », International Journal of Production Economics, vol. 207, p. 107-124, janv. 2019, doi: 10.1016/j.ijpe.2018.09.022.

[32] S. Hochreiter et J. Schmidhuber, « Long Short-Term Memory », Neural Computation, vol. 9, no 8, p. 1735-1780, nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[33] C. Finn, P. Abbeel, et S. Levine, « Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks ».

[34] T. Joo, H. Jun, et D. Shin, « Task Allocation in Human–Machine Manufacturing Systems Using Deep Reinforcement Learning », Sustainability, vol. 14, no 4, p. 2245, févr. 2022, doi: 10.3390/su14042245.

[35] D. Kucharavy et R. De Guio, « Application of Logistic Growth Curve », Procedia Engineering, vol. 131, p. 280-290, 2015, doi: 10.1016/j.proeng.2015.12.390.

[36] C. D. Bailey, « Forgetting and the Learning Curve: A Laboratory Study », Management Science, vol. 35, no 3, p. 340-352, mars 1989, doi: 10.1287/mnsc.35.3.340.

[37] D. A. Nembhard, « The Effects of Task Complexity and Experience on Learning and Forgetting: A Field Study », Hum Factors, vol. 42, no 2, p. 272-286, juin 2000, doi: 10.1518/001872000779656516.

[38] S. Hoedt, A. Claeys, E.-H. Aghezzaf, et J. Cottyn, « Real Time Implementation of Learning-Forgetting Models for Cycle Time Predictions of Manual Assembly Tasks after a Break », Sustainability, vol. 12, no 14, p. 5543, juill. 2020, doi: 10.3390/su12145543.

[39] A. Lago Alvarez, « Dataset of Workers Operation Durations in Assembly Tasks ». Zenodo, 5 mai 2025. doi: 10.5281/ZENODO.15340717.

[40] I. Kacem, S. Hammadi, et P. Borne, « Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems », IEEE Trans. Syst., Man, Cybern. C, vol. 32, no 1, p. 1-13, févr. 2002, doi: 10.1109/TSMCC.2002.1009117.

[41] J. Xu, X. Xu, et S. Q. Xie, « Recent developments in Dual Resource Constrained (DRC) system research », European Journal of Operational Research, vol. 215, no 2, p. 309-318, déc. 2011, doi: 10.1016/j.ejor.2011.03.004.

[42] H. V. Kher, M. K. Malhotra, P. R. Philipoom, et T. D. Fry, « Modeling simultaneous worker learning and forgetting in dual resource constrained systems », European Journal of Operational Research, vol. 115, no 1, p. 158-172, mai 1999, doi: 10.1016/S0377-2217(98)00190-8.

[43] A. P. J. Vepsalainen et T. E. Morton, « Priority Rules for Job Shops with Weighted Tardiness Costs », Management Science, vol. 33, no 8, p. 1035-1047, août 1987, doi: 10.1287/mnsc.33.8.1035.

[44] J. L. Andrade-Pineda, D. Canca, P. L. Gonzalez-R, et M. Calle, « Scheduling a dual-resource flexible job shop with makespan and due date-related criteria », Ann Oper Res, vol. 291, no 1-2, p. 5-35, août 2020, doi: 10.1007/s10479-019-03196-0.

[45] S. S. Panwalkar et W. Iskander, « A Survey of Scheduling Rules », Operations Research, vol. 25, no 1, p. 45-61, févr. 1977, doi: 10.1287/opre.25.1.45.

[46] O. Holthaus et C. Rajendran, « Efficient dispatching rules for scheduling in a job shop », International Journal of Production Economics, vol. 48, no 1, p. 87-105, janv. 1997, doi: 10.1016/S0925-5273(96)00068-0.

[47] J. Figueira et B. Roy, « Determining the weights of criteria in the ELECTRE type methods with a revised Simos' procedure », European Journal of Operational Research, vol. 139, no 2, p. 317-326, juin 2002, doi: 10.1016/S0377-2217(01)00370-8.

[48] H. Van Hasselt, A. Guez, et D. Silver, « Deep Reinforcement Learning with Double Q-Learning », AAAI, vol. 30, no 1, mars 2016, doi: 10.1609/aaai.v30i1.10295.