Journal of Industrial and Management Optimization

JIMO, 22(12): 6339−6373.
DOI: 10.3934/jimo.2026213
Received: 19 August 2026
Revised: 02 October 2026
Accepted: 29 October 2026
Published: 10 November 2026

https://www.aimspress.com/journal/jimo

Research article

Energy-aware dynamic scheduling in hybrid flow shops via integrated multi-objective evolutionary optimization and Actor–Critic reinforcement learning

Kuo-Feng Huang*

Department of Information Technology, Ling Tung University, Taichung, Taiwan

* **Correspondence:** Email: sailerhuang@teamail.ltu.edu.tw.

Abstract: Dynamic hybrid flow shops must simultaneously balance production efficiency, volatile energy prices, carbon emissions, and stochastic shop-floor disruptions. Optimization-based schedulers cannot react to real-time events, whereas purely learning-based dispatchers converge slowly and struggle to satisfy operational constraints. In this study, we proposed a Hybrid Push–Pull Reinforcement Learning (HP²-RL) framework for energy-aware dynamic scheduling in hybrid flow shops with parallel heterogeneous machines, finite buffers, and dynamic lot splitting. An upper-layer MOEA/D-DE optimizer with Tchebycheff decomposition handled seven objectives (makespan, tardiness, energy cost, carbon emissions, WIP dwell time, machine idle time, and material-delay penalty) and generated Pareto-oriented scheduling templates; a lower-layer Actor–Critic agent performed event-driven dispatching and lot splitting guided by the selected template; and a closed-loop feedback mechanism triggered rolling-horizon re-optimization every 4 hours or when KPI deviations exceeded 15%. Simulation results over 30 independent runs showed that HP²-RL reduced makespan by 14.5% relative to rule-based dispatching (1240.5 min) and total energy cost by 11.5% relative to static MOEA/D-DE ($8,450), achieved the lowest carbon footprint (1,241 kg), and converged within approximately 300 training episodes. Under machine failures, recovery took 58.6 minutes versus 115 minutes for rule-based dispatching. Ablation experiments and Wilcoxon signed-rank tests ($p < 0.05$) confirmed that each component contributes significantly, and an inference time below 3.8 ms per decision supports real-time deployment.

Keywords: dynamic scheduling; multi-objective optimization; reinforcement learning; energy-aware manufacturing; low-carbon production

Mathematics Subject Classification: 90B35, 90C29

1. Introduction

The proliferation of Industry 4.0 technologies has transformed production systems into highly dynamic, data-driven environments that must simultaneously address volatile demand, frequent disruptions, and increasingly stringent sustainability regulations [1−4]. Manufacturers are required to jointly optimize productivity, energy efficiency, and carbon emissions without compromising delivery reliability and system stability, elevating production scheduling from an operational task to a strategic decision-making problem with direct economic and environmental implications [5−8].

Hybrid flow shop (HFS) systems, characterized by multi-stage processing routes, parallel heterogeneous machines, and finite inter-stage buffers, represent a particularly challenging scheduling environment. In such systems, routing flexibility, machine heterogeneity, and stochastic events, including rush orders, machine breakdowns, and energy price spikes, render traditional priority rules and deterministic optimization approaches inadequate, as they typically assume static conditions and lack real-time adaptability [9−14]. The widespread adoption of time-of-use (TOU) electricity tariffs, demand charges, and carbon pricing further complicates scheduling decisions by introducing time-varying energy and emission costs that must be explicitly considered [15−17].

Evolutionary multi-objective optimization (EMO) algorithms, such as NSGA-II and MOEA/D, have been extensively applied to job shop and flow shop scheduling problems, demonstrating strong capabilities in exploring Pareto-optimal trade-offs among conflicting objectives such as makespan, tardiness, energy cost, and carbon emissions [18−25]. However, most EMO-based approaches operate in offline or rolling-horizon modes and cannot react promptly to unforeseen shop-floor events, limiting their effectiveness in highly dynamic manufacturing environments [26−28].

Reinforcement learning (RL) has emerged as a promising paradigm for dynamic scheduling and dispatching due to its ability to learn adaptive policies through continuous interaction with complex, uncertain environments [29−31]. Deep RL and Actor–Critic architectures have shown superior adaptability compared with static dispatching rules under stochastic arrivals and machine failures [32−34]. Nevertheless, purely RL-based methods often suffer from slow convergence, large state–action spaces, and feasibility enforcement issues in large-scale manufacturing systems with strict operational constraints [35−37].

To address these limitations, hybrid frameworks that integrate global optimization with local adaptive control have attracted increasing attention [38−40]. By combining the global search capability of EMO algorithms with the real-time responsiveness of RL, such hybrid push–pull architectures can achieve long-term optimality and short-term adaptability. Motivated by these observations, we propose a Hybrid Push–Pull Reinforcement Learning (HP²-RL) framework for HFS systems with finite buffers and dynamic lot splitting. The framework tightly integrates an upper-layer MOEA/D–DE optimizer with a lower-layer RL-based adaptive controller under energy and carbon constraints. A closed-loop feedback mechanism further enables continuous performance monitoring and cross-layer parameter tuning, leading to self-improving scheduling performance in dynamic manufacturing environments.

Compared with other studies, the novelty of this work lies in three aspects. First, energy-aware EMO schedulers are typically executed offline or in a rolling-horizon fashion [18–25], and hybrid EMO–RL algorithms mainly use RL to improve the search of the evolutionary algorithm [6,36]; in contrast, the Pareto-oriented templates generated here become reference trajectories for an event-driven RL controller,

so that global optimization and real-time control operate in one loop. Second, RL-only dispatchers for dynamic (flexible) job shops [32–34,38,39] must learn feasibility and energy awareness from scratch, whereas HP²-RL embeds buffer, limited-wait, kitting, TOU, and carbon considerations in the template and in a feasibility-repair operator, and extends the action space with dynamic lot splitting and merging. Third, a KPI-driven bidirectional feedback mechanism with explicit trigger rules (a 4-hour rolling horizon and a 15% KPI-deviation threshold) lets the observed shop-floor performance re-tune both layers, which is absent in most one-directional hybrid schemes. These differences are verified by the ablation study, statistical tests, and computational-cost analysis reported in Section 4.

The main contributions of this work are as follows:

(1) A hierarchical push–pull scheduling architecture is proposed for HFS systems, in which MOEA/D–DE generates Pareto-optimal scheduling templates and an Actor–Critic RL agent performs event-driven real-time control.

(2) Seven objectives, including makespan, total tardiness, total energy cost, carbon emissions, WIP dwell time, machine idle time, and material delay penalty, are simultaneously optimized under TOU pricing and carbon emission constraints.

(3) A dynamic lot splitting and compatible job merging mechanism is incorporated to enhance routing flexibility and load balancing while respecting buffer, limited-wait, and logistics constraints.

(4) A closed-loop feedback learning mechanism is designed to trigger rolling-horizon re-optimization and policy adaptation based on KPI deviations, ensuring robustness against stochastic disturbances.

(5) Comprehensive simulation experiments demonstrate that HP²-RL outperforms rule-based dispatching, static MOEA/D–DE, and pure RL baselines in terms of production efficiency, energy performance, environmental impact, and system resilience.

(6) Rigorous validation is provided through a component-wise ablation study, 30 independent runs with Wilcoxon signed-rank tests, an analysis of computational overhead (re-optimization time, inference time, and memory footprint), and a mechanism-level explanation of why the framework is effective.

2. Literature review

2.1. Multi-objective scheduling in job shop and hybrid flow shop systems

Multi-objective scheduling in job shop and HFS systems has been extensively studied, with typical objectives encompassing makespan, total tardiness, work-in-process (WIP) levels, and machine utilization. Evolutionary multi-objective (EMO) algorithms, such as NSGA-II and MOEA/D, have demonstrated strong performance in exploring Pareto-optimal trade-offs among these conflicting objectives due to their population-based search mechanisms and explicit handling of dominance relations. Extensions incorporating flexible routing, finite buffer capacities, and sequence-dependent setup times have further enhanced the applicability of EMO methods to realistic manufacturing environments [23−26].

Despite these advances, most studies assume relatively static problem settings and focus on offline optimization under deterministic or mildly stochastic conditions. In such formulations, the scheduling solution is typically computed once and executed without systematic mechanisms for real-time adaptation to unforeseen shop-floor events, such as machine breakdowns, rush orders, or abrupt changes in energy prices [26−28]. Consequently, while EMO-based approaches can generate high-quality Pareto fronts in terms of long-term performance metrics, their direct deployment in highly

dynamic HFS environments remains limited. This limitation motivates the integration of EMO methods with adaptive, event-driven control mechanisms capable of responding to real-time disturbances while preserving the strategic guidance provided by global optimization.

Researchers continue to extend EMO to energy-oriented distributed scheduling. For example, Pan et al. [37] integrated factory assignment and machine assignment into an extended machine assignment for distributed energy-efficient parallel machine scheduling and proposed a knowledge-based two-population algorithm in which NSGA-II and differential evolution cooperate to minimize total energy consumption and total tardiness, supported by problem-specific properties and knowledge-based local search. Such DE-assisted, knowledge-based designs support the use of differential evolution operators in the present upper layer; however, they remain offline optimizers without a mechanism for reacting to disturbances during execution.

2.2. Energy-aware and low-carbon scheduling

Energy-aware and low-carbon scheduling has gained significant attention in response to rising energy costs, time-varying electricity tariffs, and increasingly stringent environmental regulations. Researchers have incorporated time-of-use (TOU) pricing, demand charges, and carbon emission factors into scheduling objectives and constraints, demonstrating the potential for jointly reducing energy consumption, energy cost, and carbon emissions [24,27−30]. Representative works have addressed single-machine, parallel-machine, job shop, and flow shop configurations, often formulating energy and carbon considerations as additional objectives or constraints within multi-objective optimization frameworks [23,24,27−30].

However, most energy-aware and low-carbon scheduling studies remain predominantly optimization-driven and assume deterministic or pre-specified disturbance scenarios. Dynamic events such as stochastic machine failures, unexpected order arrivals, and real-time fluctuations in electricity prices are either neglected or modeled in a simplified manner, which limits the practical robustness of the resulting schedules in volatile production environments. Moreover, few researchers explicitly consider the interaction between long-term energy-aware optimization and short-term adaptive control, thereby overlooking the potential benefits of hierarchical or hybrid architectures that can simultaneously address strategic energy planning and operational responsiveness. For the proposed HP²-RL framework, we aim to fill this gap by embedding energy and carbon constraints within a push–pull architecture that couples global MOEA/D–DE optimization with RL-based real-time control.

2.3. Reinforcement learning for dynamic scheduling

Reinforcement learning (RL) has emerged as a promising paradigm for dynamic scheduling and dispatching due to its ability to learn adaptive policies through continuous interaction with complex and uncertain environments [29−33]. Deep RL and Actor–Critic architectures, in particular, have shown superior adaptability compared with static dispatching rules under stochastic job arrivals, machine failures, and time-varying system states [34−36]. Applications have covered job shop, flexible job shop, and flow shop environments, where RL agents learn dispatching policies that balance productivity, responsiveness, and resource utilization.

Nevertheless, purely RL-based scheduling approaches face several critical challenges in large-scale manufacturing systems. First, the state–action space grows rapidly with the number of jobs, machines, and

operational constraints, leading to slow convergence and high sample complexity [35−37]. Second, ensuring feasibility with respect to intricate shop-floor constraints, such as finite buffers, limited-wait requirements, and material readiness conditions, remains non-trivial within standard RL formulations, often requiring additional repair mechanisms or constraint-handling techniques. Third, the lack of explicit long-term optimization guidance can result in locally adaptive but globally suboptimal behaviors, particularly when multiple conflicting objectives (e.g., makespan, energy cost, carbon emissions) must be simultaneously considered. These limitations highlight the need for hybrid frameworks that leverage the global search capability of optimization algorithms to guide RL exploration and constraint satisfaction, while retaining the real-time adaptability of RL for event-driven control.

Moreover, work has strengthened the learning side of dynamic scheduling. Wang et al. [38] proposed a dual Q-learning method for assembly job shops with uncertain assembly times, in which a two-level learning process improves adaptability to environmental changes. Zhao et al. [39] developed a heterogeneous graph reinforcement learning framework with question-aware neighborhood aggregation and interoption prompt attention for the dynamic flexible job shop scheduling problem. These studies show that richer value estimation and graph-based state representations improve adaptability under uncertainty. To the best of our knowledge, however, they focus on time-related objectives and do not jointly address time-of-use energy cost, carbon emissions, finite buffers, and lot splitting, nor do they couple the learner with a multi-objective optimizer. The proposed framework is complementary: Its lower layer could adopt such representations, while its upper layer supplies the multi-objective, energy-aware structure that they lack.

2.4. Hybrid optimization–learning frameworks

Hybrid approaches that combine evolutionary multi-objective optimization and reinforcement learning aim to leverage the complementary strengths of both paradigms [38−40]. Researchers typically adopt one of the following strategies: (i) using EMO algorithms to generate high-quality initial solutions or reference trajectories that guide RL exploration; (ii) employing RL to tune heuristic parameters or dispatching rules within an optimization loop; or (iii) decomposing the scheduling problem into a long-term planning layer and a short-term control layer, each handled by optimization and learning modules, respectively [38−40]. Such hybridization has shown potential for improving convergence speed, solution quality, and adaptability in dynamic scheduling contexts.

However, several important gaps remain in the literature. First, most hybrid frameworks lack systematic closed-loop feedback mechanisms that continuously monitor operational performance and adjust optimization and learning parameters based on real-time key performance indicators (KPIs). As a result, the interaction between the optimization and learning components is often one-directional or static, limiting the system's ability to self-improve over time in response to evolving shop-floor conditions [38−40]. Second, energy and carbon considerations are rarely integrated into hybrid optimization–learning architectures in a unified manner; instead, they are typically treated as secondary objectives or post-processing criteria, rather than as integral components of the decision-making loop. Third, few researchers explicitly address hybrid flow shop configurations with finite buffers, dynamic lot splitting, and TOU-based energy pricing within a single hierarchical framework.

The proposed HP²-RL framework addresses these gaps by introducing a hierarchical push–pull architecture that tightly integrates MOEA/D–DE-based global optimization with Actor–Critic RL-based local adaptive control under explicit energy and carbon constraints. A closed-loop feedback

mechanism is further designed to trigger rolling-horizon re-optimization and policy adaptation based on KPI deviations, thereby enabling sustained performance improvement and robustness against stochastic disturbances in HFS environments. This design positions the present work at the intersection of energy-aware scheduling, multi-objective optimization, and adaptive reinforcement learning, contributing a unified and implementable framework for smart, low-carbon manufacturing systems.

To make the positioning explicit, the proposed framework differs from the representative studies above in four characteristics: (i) a many-objective (seven-objective) upper layer solved by decomposition rather than dominance ranking; (ii) a lower-layer controller whose action space includes dispatching, lot splitting and merging, synchronization, and TOU-aware job release; (iii) explicit finite-buffer, limited-wait, and kitting constraints enforced by gating rules and feasibility repair; and (iv) a KPI-triggered closed loop that adapts both layers. Sections 4.8–4.10 quantify the contribution of these characteristics.

3. Proposed methodology

In this section, we present the methodological framework of the proposed HP²-RL system for energy-aware and low-carbon dynamic scheduling in HFS environments. The framework integrates long-term multi-objective optimization (Push control) and short-term adaptive scheduling (Pull control), coordinated through a closed-loop feedback mechanism. The system aims to minimize makespan, total tardiness, energy cost, and carbon emissions while maintaining work-in-process (WIP) balance and responsiveness to real-time shop-floor dynamics under finite buffer, limited-wait, and TOU-based energy constraints.

3.1. Overview of the proposed framework

The proposed HP²-RL framework is designed for multi-stage hybrid flow shops with parallel heterogeneous machines, finite inter-stage buffers, and dynamic lot splitting. The architecture adopts a hierarchical push–pull structure in which an upper-layer MOEA/D–DE optimizer performs global multi-objective optimization (Push control) and a lower-layer Actor–Critic reinforcement learning (RL) agent executes event-driven, real-time adaptive dispatching (Pull control). A closed-loop feedback mechanism continuously monitors key performance indicators (KPIs) and triggers rolling-horizon re-optimization and policy adaptation when KPI deviations exceed predefined thresholds, thereby enabling robust operation under stochastic disturbances such as machine failures, rush orders, and energy price spikes.

3.1.1. System architecture and functional layers

The proposed scheduling framework consists of two interconnected functional layers: an upper-layer push control for multi-objective optimization and a lower-layer pull control for real-time intelligent decision-making, as illustrated in Figure 1.

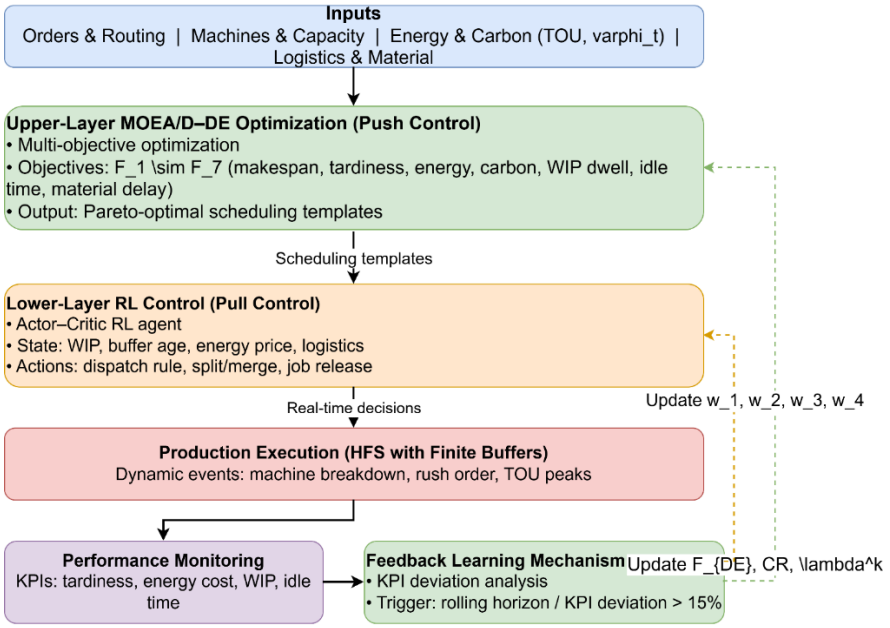


Figure 1. Overall research framework of the proposed hybrid push–pull scheduling system.

(1) Upper-Layer Push Control (Global Optimization)

The upper layer executes offline and mid-term multi-objective optimization to generate Pareto-optimal scheduling templates. It employs the MOEA/D–DE algorithm, which decomposes the optimization problem into multiple scalar sub-problems and evolves them simultaneously using differential evolution operators. Inputs include: (i) Order data and routing structure; (ii) Machine availability and production capacity; (iii) Time-of-Use (TOU) energy pricing and carbon emission factors; and (iv) Material readiness and logistics availability.

The upper layer outputs scheduling templates that represent near-optimal decisions for routing, sequencing, resource allocation, and lot-splitting configurations. These templates are transmitted to the lower-layer RL controller for execution in real-time production environments.

(2) Lower-Layer Pull Control (Real-Time Execution)

The lower layer performs real-time adaptive control using a Reinforcement Learning (RL) mechanism. This agent-based system responds to dynamic shop-floor changes by adjusting: (i) Dispatching rules and job sequencing; (ii) Lot splitting and merging decisions; (iii) Synchronization of production and logistics; and (iv) Energy-aware job release under TOU tariffs.

The RL agent observes the current state of production (WIP, buffer age, energy price, logistics congestion) and selects an action to optimize short-term objectives. Through feedback-based learning, the policy continuously improves over time, achieving adaptive stability under uncertainty.

(3) Feedback Learning Mechanism

A continuous feedback loop connects both layers. Performance indicators, such as tardiness, energy consumption, and WIP dwell time, are analyzed and used to tune algorithmic parameters in the MOEA/D–DE and RL modules. This mechanism enables the system to evolve dynamically, achieving self-optimization through iterative learning.

The figure shows the conceptual system architecture integrating the upper-layer MOEA/D–DE optimizer and the lower-layer RL-based controller. Inputs (orders, machines, energy, logistics) feed

into the optimizer, which generates Pareto scheduling templates. The RL controller adapts these templates in real time, while feedback metrics continuously refine both layers.

3.1.2. Data flow and process integration

The hybrid system functions as a cyber-physical decision loop that integrates sensing, computation, and control. As shown in Figure 2, the data flow begins with real-time sensing of machine, material, energy, and logistics conditions. After preprocessing (filtering, normalization, and feature extraction), the refined data are provided to the upper-layer optimization. Optimized scheduling templates are transferred to the RL-based pull controller, which makes real-time adaptive decisions. Production performance is continuously monitored and fed back into the learning module to adjust system parameters.

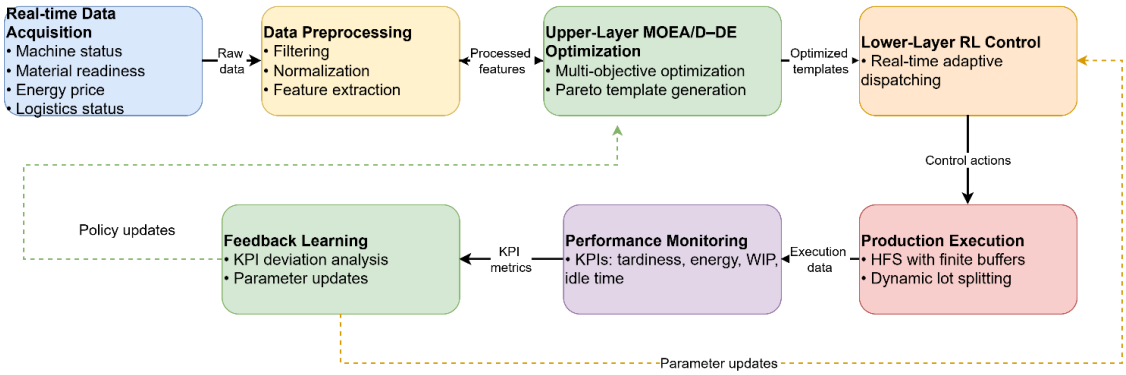


Figure 2. Data and process flow within the proposed framework.

The figure illustrates the flow: real-time data acquisition → data preprocessing → upper-layer MOEA/D–DE optimization → RL-based control → production execution → performance monitoring → feedback learning. Labeled arrows indicate data exchanges such as "optimized templates," "control actions," and "policy updates."

Design rationale of the component combination. The four components are combined because each covers a weakness of the others. MOEA/D–DE (push) provides globally coordinated, energy- and carbon-aware templates but is too slow to be invoked at every shop-floor event; the Actor–Critic agent (pull) reacts within milliseconds but, without guidance, must explore a large state–action space and has no global view of TOU tariffs; dynamic lot splitting adds routing and load-balancing flexibility that both layers exploit; and the feedback loop keeps the template consistent with the actual shop state by re-invoking the push layer only when needed. The contribution of each component is isolated in the ablation study of Section 4.8.

3.2. Problem definition and system modeling

The scheduling problem is defined in a multi-stage HFS with finite buffers, parallel heterogeneous machines, and energy and carbon constraints. Each order $i \in \{1, 2, \ldots, N\}$ requires a sequence of operations $O_i = \{1, 2, \ldots, O_i\}$ that must be processed in a predefined technological order across multiple stages. At each stage, a subset of parallel machines M_o is available, each with different processing times

$p_{i,o,m}$, power consumption profiles, and energy factors. The material flow is unidirectional from upstream to downstream stages, consistent with the HFS configuration.

The system considers finite inter-stage buffers, limited-wait constraints between consecutive operations, and TOU-based energy pricing and carbon emission factors. Decision variables include: (i) $x_{i,o,m}$: machine assignment for operation i; (ii) $S_{i,o}, C_{i,o}$: start and completion times of operation of order i; (iii) $q_{i,b}$: batch size (lot size) of sub-lot b from order i after dynamic splitting; (iv) $\delta_{spilt}, \delta_{merge}$: binary indicators for splitting or merging decisions; (v) Γ_o: limited-wait time threshold between consecutive operations; and (vi) $WIP_o(t)$: buffer occupancy at operation o and time t.

3.3. Objective formulation

The system minimizes seven objectives simultaneously:

$$\min \mathbf{F} = [F_1, F_2, F_3, F_4, F_5, F_6, F_7] \tag{1}$$

where:

- $F_1 = C_{\max}$: Makespan, defined as the completion time of the last job.
- $F_2 = \sum_{i=1}^{N} \max(0, C_i - d_i)$: Total tardiness, where C_i is the completion time of order i and d_i is its due date. This formulation ensures that only positive deviations (late completions) contribute to tardiness.
- $F_3 = \sum_{t,m} P_{m,t} \cdot p_t^{\mathrm{TOU}}$: Total energy cost, computed as the sum of power consumption $P_{m,t}$ multiplied by TOU electricity price p_t^{TOU}.
- $F_4 = \sum_{m,t} EF_m \cdot P_{m,t}$: Carbon emissions, where EF_m denotes the energy-to-carbon conversion factor for machine m.
- $F_5 = \sum_{i,b,o}\left(S_{i,b,o+1} - C_{i,b,o}\right)$: WIP dwell time, representing the waiting time between consecutive operations.
- $F_6 = \sum_{m,t} \mathrm{Idle}_m(t)$: Machine idle time, aggregated over all machines and time periods.
- $F_7 = \sum_i \sum_o max\left(0, r_{i,o}^{\mathrm{kit}} - r_{i,o}^{\mathrm{plan}}\right)$: Material delay penalty, where $r_{i,o}^{\mathrm{kit}}$ and $r_{i,o}^{\mathrm{plan}}$ denote the actual material readiness time and the planned release time for operation o of order i, respectively. The $\sum_o max\,(0,\cdot)$ operator ensures that only delayed material availability (i.e., $r_{i,o}^{\mathrm{kit}} > r_{i,o}^{\mathrm{plan}}$) is penalized.

Subject to machine, buffer, and limited-wait constraints:

$$\begin{cases} S_{i,b,o+1} \geq C_{i,b,o} + \tau_{o\to 0+1} \\ \mathrm{WIP}_o(t) \leq \mathrm{WIP}_{cap}(o) \\ 0 \leq S_{i,o+1} - \left(C_{i,o} + \tau_{o\to o+1}\right) \leq \Gamma_o \end{cases} \tag{2}$$

To avoid ambiguity, the following notational conventions are adopted throughout the paper: (1) $\mathbf{F} = (F_1, F_2, \ldots, F_7)$ denotes the objective vector; the DE mutation factor is denoted as F_{DE}; (2) Carbon

emission coefficient is denoted as φ_t (or $EF_{\mathrm{grid},t}$) to distinguish it from RL reward weights; and (3) RL reward weights are denoted as w_1, w_2, w_3, w_4.

Table 1 provides a comprehensive summary of symbols, definitions, and units consistent with these conventions.

Table 1. Symbol definition and mathematical notation summary.

Symbol	Definition	Description	Type	Unit
J	Set of production orders (jobs)	Index set $j = 1, 2, \ldots, N$	Set	—
i, j	Order (job) index	Index of production order or job	Index	—
M	Set of available machines	Index set $s = 1, 2, \ldots, \lvert M \rvert$	M	—
S	Set of processing stages	Index set $s = 1, 2, \ldots, \lvert S \rvert$	S	—
O_j	Operation sequence of order j	Set of operations $s = 1, 2, \ldots, \lvert O_j \rvert$	O_j	—
o	Operation index	Index of operation within order j	Index	—
$M_{j,o}$	Set of feasible machines for operation o of order j	Subset of M capable of processing operation o	Set	—
$p_{j,o,m}$	Processing time of operation o of order j on machine mmm	Deterministic or stochastic processing duration	Parameter	minutes
$x_{j,o,m}$	Machine assignment decision variable	Binary: 1 if operation o of order j is assigned to machine mmm, 0 otherwise	Decision variable	—
$S_{j,o}$	Start time of operation o of order j	Time at which operation o begins processing	Decision variable	minutes
$C_{j,o}$	Completion time of operation o of order j	Time at which operation o finishes processing	Decision variable	minutes
S_j	Start time of order j	Start time of first operation of order j	Decision variable	minutes
C_j	Completion time of order j	Completion time of last operation of order j	Decision variable	minutes
$q_{j,b}$	Lot size (quantity) of sub-lot b from order j	Batch size after dynamic lot splitting	Decision variable	units

Continued on next page

Symbol	Definition	Description	Type	Unit
$\delta_{\text{split}}, \delta_{\text{merge}}$	Binary indicators for splitting or merging	1 if split/merge is applied, 0 otherwise	Decision variable	—
Γ_o	Limited-wait time threshold between consecutive operations	Maximum allowable waiting time between operation o and $o+1$	Parameter	minutes
$\tau_{o \to o+1}$	Transfer or setup time between consecutive operations	Time required to move or set up between operations	Parameter	minutes
$\text{WIP}_o(t)$	Work-in-process quantity at operation o and time t	Number of jobs waiting or being processed at operation o	State variable	lots
$\text{WIP}_{\text{cap}}(o)$	Maximum allowable WIP capacity for operation o	Buffer capacity limit at operation o	Parameter	lots
r_j	Release time (arrival time) of order j	Time at which order j becomes available for processing	Parameter	minutes
d_j	Due date of order j	Target completion time for order j	Parameter	minutes
$r_{j,o}^{\text{kit}}$	Material readiness time for operation o of order j	Time at which required materials are available	Parameter	minutes
$r_{j,o}^{\text{plan}}$	Planned release time for operation o of order j	Scheduled time to release operation o based on template	Parameter	minutes
EF_m	Energy factor of machine m	Energy consumption per unit operation	Parameter	kWh/unit
$P_{m,t}$	Power consumption of machine mmm at time t	Instantaneous power draw	Variable	kW
p_t^{TOU}	Time-of-Use electricity price at time t	TOU tariff rate	Parameter	NTD/kWh
φ_t	Carbon emission coefficient at time t	Grid emission factor (previously denoted as $\beta(t)$)	Parameter	kg CO_2/kWh
$\text{Idle}_m(t)$	Idle state indicator of machine mmm at time t	Binary: 1 if machine mmm is idle at time t, 0 otherwise	State variable	—
$F_1, F_2, F_3, F_4, F_5, F_6, F_7$	Objective function components	$F_1 = C_{max}$, $F_2 = \sum \max(0, C_j - d_j)$, F_3(energy cost), F_4(carbon), F_5(WIP dwell), F_6(idle time), F_7(material delay)	Function	—

Continued on next page

Symbol	Definition	Description	Type	Unit
C_{max}	Makespan	Completion time of the last job	Variable	minutes
$\mathrm{Sig}(j, o)$	Process signature of operation o of order j	Tuple {process, material, tool, quality_spec}	Attribute	—
λ^k	Weight vector for MOEA/D–DE decomposition	Weight for subproblem k	Parameter	—
F_{DE}	Differential evolution mutation factor	Scaling factor for DE mutation (previously denoted as F)	Parameter	—
CR	Crossover rate for MOEA/D–DE	Probability of crossover in DE	Parameter	—
N_{pop}	Population size for MOEA/D–DE	Number of candidate solutions	Parameter	—
G_{max}	Maximum number of generations for MOEA/D–DE	Stopping criterion for evolution	Parameter	—
θ_π	Policy network parameters in RL	Weights of Actor (policy) network	Variable	—
θ_V	Value network parameters in RL	Weights of Critic (value) network	Variable	—
w_1, w_2, w_3, w_4	Reward weight coefficients in RL	Weights for WIP, idle time, tardiness, and energy cost in reward function	Parameter	—
s_t	System state vector at time t	Input to RL policy network	Variable	—
a_t	Action selected by RL policy at time t	Output of RL policy (dispatch rule, split ratio, etc.)	Variable	—
r_t	Reward at time t	Immediate reward signal for RL	Variable	—
$\pi(a\|s_t)$	Policy function	Probability of selecting action a given state s_t	Function	—
$V(s_t)$	Value function	Estimated expected cumulative reward from state s_t	Function	—
ρ_{log}	Logistics availability ratio	Ratio of available AGVs or forklifts	Parameter	—
ζ_{conv}	Convergence tolerance threshold	Stopping condition for optimization and forecasting	Parameter	—

Continued on next page

Symbol	Definition	Description	Type	Unit
z^*	Reference point in Pareto optimization	Reference vector for MOEA/D–DE	Parameter	—
$g_k(x\|\lambda^k, z^*)$	Aggregation function for subproblem $k$in MOEA/D–DE	Tchebycheff or weighted sum function	Function	—

3.4. Dynamic lot splitting and compatible lob merging mechanism

To enhance production flexibility, the system dynamically restructures orders into smaller sub-lots or merged batches based on process signatures:

$$\text{Sig}(i, o) = \{\text{process}, \text{material}, \text{tool}, \text{quality_spec}\} \quad (3)$$

Jobs with identical process signatures are candidates for merging, while others may be split according to minimum batch-size requirements, due-date feasibility, WIP limits, and logistics capacity. This mechanism ensures load balance, reduced setup time, and improved responsiveness to dynamic events while minimizing late deliveries.

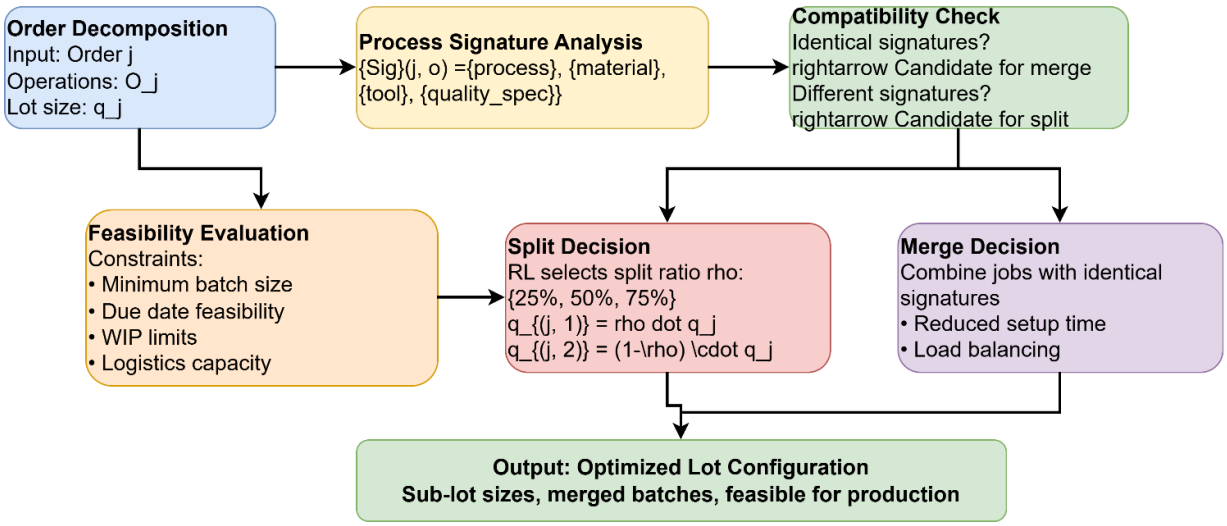


Figure 3. Mechanism of dynamic lot splitting and compatible job merging.

The figure illustrates how orders are decomposed, analyzed for compatibility, and evaluated for feasibility before merging or splitting. The resulting lot configurations are optimized for throughput and flexibility.

3.5. Upper-layer optimization using MOEA/D–DE

The upper-layer optimization uses MOEA/D–DE, which decomposes the global multi-objective problem into scalar subproblems via Tchebycheff aggregation. Each subproblem evolves through Differential Evolution (DE) to ensure convergence and diversity. Chromosome encoding includes:

$$x = \{route_i, machine_{i,o}, sequence_{i,o}, q_{i,b}, \text{WIP}_o, \Gamma_o\} \quad (4)$$

Algorithmic Steps: (1) Initialize population and reference vectors; (2) Evaluate objective vector $\mathbf{F}(x)$; (3) Apply DE operators: $v_i = x_{r1} + F_{DE}(x_{r2} - x_{r3})$; (4) Crossover and feasibility repair; (5) Update neighborhood and Pareto archive; and (6) Continue until convergence or horizon update.

Rationale for selecting MOEA/D–DE. MOEA/D–DE is selected for the upper layer for three reasons. First, the decision vector in Eq. (4) mixes discrete variables (route, machine, sequence) with integer or continuous variables (lot sizes, WIP limits, limited-wait thresholds); the DE operator provides strong global exploration on such mixed encodings and, together with the feasibility-repair step, preserves buffer and limited-wait feasibility. Second, with seven objectives, the problem is a many-objective one in which dominance-based methods such as NSGA-II lose selection pressure, because most solutions become mutually non-dominated; MOEA/D [19] replaces dominance ranking by the comparison of scalar subproblems and therefore retains a convergence-driving criterion. Third, every subproblem is tied to an explicit weight vector, so a planner's preference (e.g., a stronger weight on energy cost or tardiness) can be translated directly into the selection of a template for the lower layer.

Formal decomposition. Each of the N_{pop} subproblems minimizes the Tchebycheff aggregation function

$$g^{\mathrm{te}}(x|\lambda^k, z^*) = \max_{1\le j\le 7}\left\{\lambda_j^k \left| F_j(x) - {z_j}^* \right|\right\} \tag{5}$$

where $\lambda^k = \left({\lambda_1}^k, \ldots, {\lambda_7}^k\right)$ with ${\lambda_j}^k \geq 0$ and $\sum_j {\lambda_j}^k = 1$, and $z^* = ({z_1}^*, \ldots, {z_7}^*)$ is the reference (ideal) point, ${z_j}^*$ being the smallest F_j found so far and updated after every generation. Because the seven objectives have different units (minutes, currency, kg CO_2), each F_j is normalized to [0, 1] with the running ideal and nadir points before aggregation.

Weight vectors and neighborhood. The weight vectors are generated with the two-layer Das–Dennis simplex-lattice method. For seven objectives, a boundary layer with $H_1 = 3$ divisions yields $C(9,6) = 84$ vectors and an inner layer (scaled toward the simplex centre) with $H_2 = 2$ divisions yields $C(8,6) = 28$ vectors, giving $N_{\mathrm{pop}} = 112$ subproblems, which lies within the population range of Table 2 (80–120). The neighborhood of each subproblem consists of the $T = 10$ subproblems with the closest weight vectors (Euclidean distance). T is kept fixed during the search; the quantities adapted by the feedback mechanism are F_{DE}, CR, and λ^k (Eq. 9). Mating selection and replacement are restricted to the neighborhood, so each offspring updates only a few neighboring subproblems.

Many-objective considerations. With seven objectives, 112 weight vectors cover the Pareto front only sparsely. This is acceptable because the upper layer is not required to approximate the whole front: it needs to supply only a set of well-spread, high-quality trade-off templates from which the lower layer receives one (Section 3.6). Accordingly, the results in Section 4 are reported on the executed schedules (KPIs) rather than on front-quality indicators such as hypervolume.

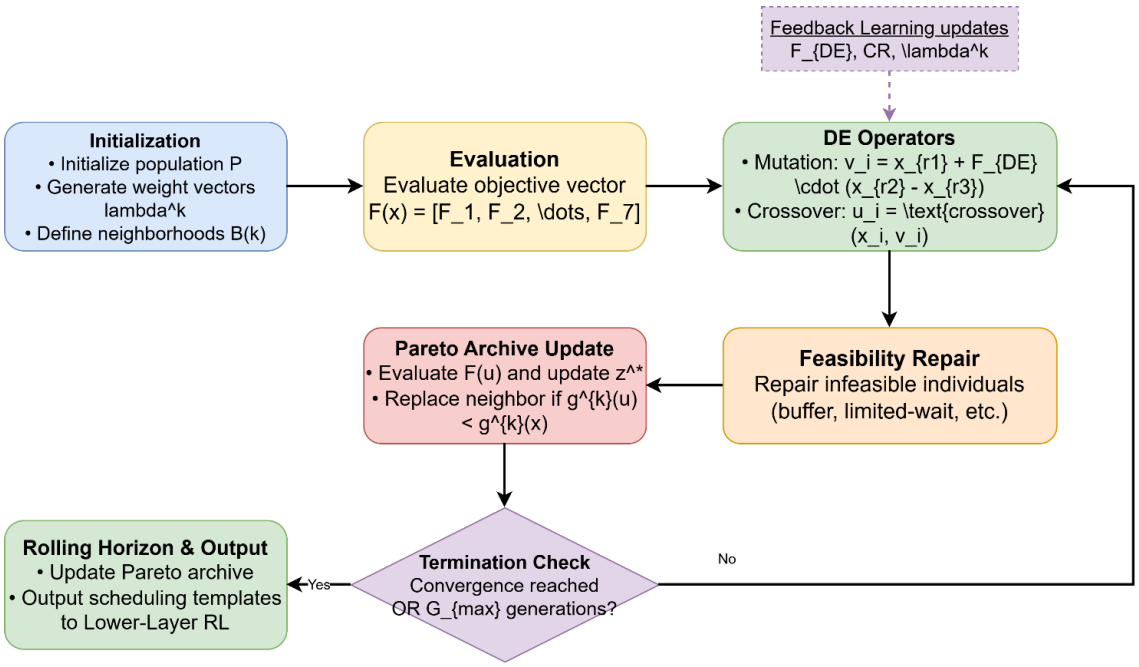


Figure 4. Upper-layer optimization workflow using MOEA/D–DE.

The figure depicts the stepwise workflow from initialization to Pareto front archiving and rolling-horizon optimization.

Table 2 summarizes the parameter configuration for forecasting (A-LSTM) and optimization (MOEA/D–DE and RL-based control) modules, ensuring convergence stability, computational efficiency, and consistency between predictive and optimization components.

Table 2. Parameter configuration for forecasting and optimization models.

Symbol	Parameter Name	Model / Module	Unit	Setting / Range	Description
p_t^{TOU}	Energy price	Forecasting (A-LSTM)	NTD/kWh	Dynamic (TOU-based)	Time-of-Use electricity tariff for each time slot t
φ_t	Carbon emission coefficient	Forecasting (A-LSTM)	kg CO_2/kWh	0.52–0.67	Grid emission factor based on local energy mix
H	Forecasting horizon	Forecasting (A-LSTM)	Time steps	12–24	Predictive look-ahead window (e.g., hourly intervals).
lr_f	Learning rate	Forecasting (A-LSTM)	—	0.001	Optimizer learning rate for LSTM network training.
F_{DE}	Differential evolution factor	MOEA/D–DE	—	0.5–0.9	Scaling factor controlling mutation strength

Continued on next page

Symbol	Parameter Name	Model / Module	Unit	Setting / Range	Description
CR	Crossover rate	MOEA/D–DE	—	0.7–0.95	Probability of combining parent and mutant individuals in DE.
N_{pop}	Population size	MOEA/D–DE	—	80–120	Number of candidate solutions evolved per generation.
$G_{\max}$	Maximum generations	MOEA/D–DE	—	100–300	Stopping criterion for MOEA/D–DE evolution.
λ^k	Weight vector	MOEA/D–DE	—	Adaptive (uniformly spread)	Weight for subproblem decomposition in MOEA/D–DE.
θ_π	Policy parameter (actor)	RL-based Control	—	Trainable	Parameter vector for policy function $\pi(a\|s_t)$
θ_V	Value function parameter (critic)	RL-based Control	—	Trainable	Parameter vector for value function $V(s_t)$
w_1, w_2, w_3, w_4	Reward weights	RL-based Control	—	0.2–0.4 (each)	Relative weights for WIP, idle time, tardiness, and energy cost objectives in reward function
Γ_o	Limited wait threshold	System constraint	min	3–10	Upper bound for inter-operation waiting time.
$\text{WIP}_{\text{cap}}(o)$	Buffer capacity	System constraint	lots	3–8	Maximum number of jobs allowed in WIP buffer at operation o
$\tau_{o\to o+1}$	Transfer time	Logistics module	min	2–6	Material transport duration between consecutive stages.
EF_m	Energy factor of machine mmm	Energy module	kWh/unit	Measured	Specific energy consumption per unit operation for machine mmm.
$\rho_{\log}$	Logistics availability ratio	Logistics module	—	0.75–1.00	Ratio of available AGVs or forklifts for transport tasks.
ζ_{conv}	Convergence tolerance	All models	—	10^{-4}	Stopping condition for optimization and forecasting convergence.

3.6. Lower-layer adaptive pull control via reinforcement learning

The lower-layer controller uses an Actor–Critic RL algorithm for event-driven scheduling. The design explicitly incorporates dynamic lot-splitting decisions into the action space to address the high-dimensional nature of sub-lot configuration under HFS constraints.

State space:

$$s_t = \{\mathrm{WIP}(o), \mathrm{BufferAge}(o), \mathrm{EnergyPrice}_t, \mathrm{Utilization}_m, \mathrm{LogisticsStatus}\} \quad (6)$$

Action space: (i) Dispatch rule selection (e.g., EDD, SPT, or learned composite rules); (ii) Lot split/merge decisions; (iii) Synchronization actions (e.g., holding or releasing jobs at buffer boundaries); and (iv) Job release control under TOU tariffs.

For lot splitting, the RL agent selects a split ratio from a discrete set $\{25\%, 50\%, 75\%\}$, which is then mapped to feasible sub-lot sizes subject to minimum batch-size, buffer capacity, and logistics constraints. Specifically, given an order with lot size q_i, the selected split ratio $\rho \in \{0.25, 0.50, 0.75\}$ determines the sub-lot quantities $q_{i,1} = \rho q_i$ and $q_{i,2} = (1-\rho)q_i$, with additional repair mechanisms applied to ensure integrality and feasibility. This discrete-action design enables the RL policy to effectively handle high-dimensional lot-splitting decisions while maintaining operational feasibility and interpretability.

Reward function:

$$r_t = -w_1 \cdot \mathrm{WIP}_t - w_2 \cdot \mathrm{Idle}_t - w_3 \cdot \mathrm{Tardness}_t - w_4 \cdot \mathrm{EnergyCost}_t \quad (7)$$

where w_1, w_2, w_3, w_4 are non-negative weight coefficients that balance the relative importance of WIP, machine idle time, tardiness, and energy cost in the short-term objective.

Policy update:

Standard Actor–Critic updates are applied, with policy and value network parameters θ_π and θ_V updated based on temporal-difference errors. The upper-layer MOEA/D–DE templates serve as reference trajectories that guide RL exploration and constraint satisfaction.

Interaction with the Pareto archive. The upper layer returns an archive A of non-dominated templates. At the beginning of each rolling horizon, a single template is selected according to a preference vector λ^{pref} supplied by the planner (by default, equal weights on the seven normalized objectives):

$$x_{\mathrm{tpl}} = \arg\min_{x \in \mathcal{A}} g^{\mathrm{te}}\left(x \middle| \lambda^{\mathrm{pref}}, z^*\right) \quad (8)$$

The RL policy is therefore trained and evaluated against one scalarized preference template at a time, and not against the whole front; its reward (Eq. 7) remains a weighted short-term scalar. The selected template serves the lower layer in three ways: It provides the planned release times r^{plan} and the machine/sequence reference used by the gating rules of Section 3.7; it provides the reference KPI levels that serve as the baseline for reward evaluation and for the deviation monitoring of Section 3.8; and it defines the lot-size and WIP bounds within which the policy may deviate.

State space. All state features in Eq. (6) are normalized to [0, 1] before entering the networks:

- $\mathrm{WIP}(o) \in [0,1]$: buffer occupancy of stage o, i.e., $\mathrm{WIP}_o(t) \Big/ \mathrm{WIP}_{\mathrm{cap}}(o)$, one element per stage;
- $\mathrm{BufferAge}(o) \in [0,1]$: buffer occupancy of stage o divided by the limited-wait threshold Γ_o (clipped at 1);

- $\text{EnergyPrice}_t \in [0,1]$: current TOU price, min–max normalized over the tariff range;
- $\text{Utilization}_m \in [0,1]$: current average utilization of machine m, one element per machine;
- $\text{LogisticStatus} \in [0,1]$: availability ratio ρ_{log} of AGVs/forklifts.

The state vector therefore has $d_s = 2|\mathcal{S}| + |\mathcal{M}| + 2$ dimensions, i.e., insert d_s of the simulated testbed ($|\mathcal{S}|$ stages, $|\mathcal{M}|$ machines).

Hybrid action space. The action a_t has four components: (i) a dispatching rule from {EDD, SPT, learned composite rule}; (ii) a lot-restructuring decision from {no split, merge, split with $\rho \in \{25\%, 50\%, 75\%\}$}; (iii) a synchronization decision from {hold, release}; and (iv) a TOU-aware job-release decision from {release now, defer}. Because the split ratio is discretized into three levels, the action space is a factored multi-discrete space and not a truly continuous–discrete hybrid. The Actor therefore uses a shared trunk with one categorical (softmax) head per component; the joint policy is $\pi(a|s) = \prod_c \pi_c(a_c|s)$) and its log-probability is the sum of the head log-probabilities, so that a standard Actor–Critic update with a single scalar critic $V(s)$ is unchanged. Infeasible choices are removed by action masking (e.g., splitting when a sub-lot would fall below the minimum batch size or exceed the buffer capacity), and the feasibility-repair operator rounds the sub-lot sizes $q_{i,1} = \rho q_i$ and $q_{i,2} = (1-\rho)q_i$ to integers that satisfy the minimum batch size, WIP_{cap}, and logistics limits. Moreover, if continuous split ratios were required, the same trunk could emit a parameterized-action output (a discrete choice plus a bounded continuous parameter); this variant is not used in the these experiments.

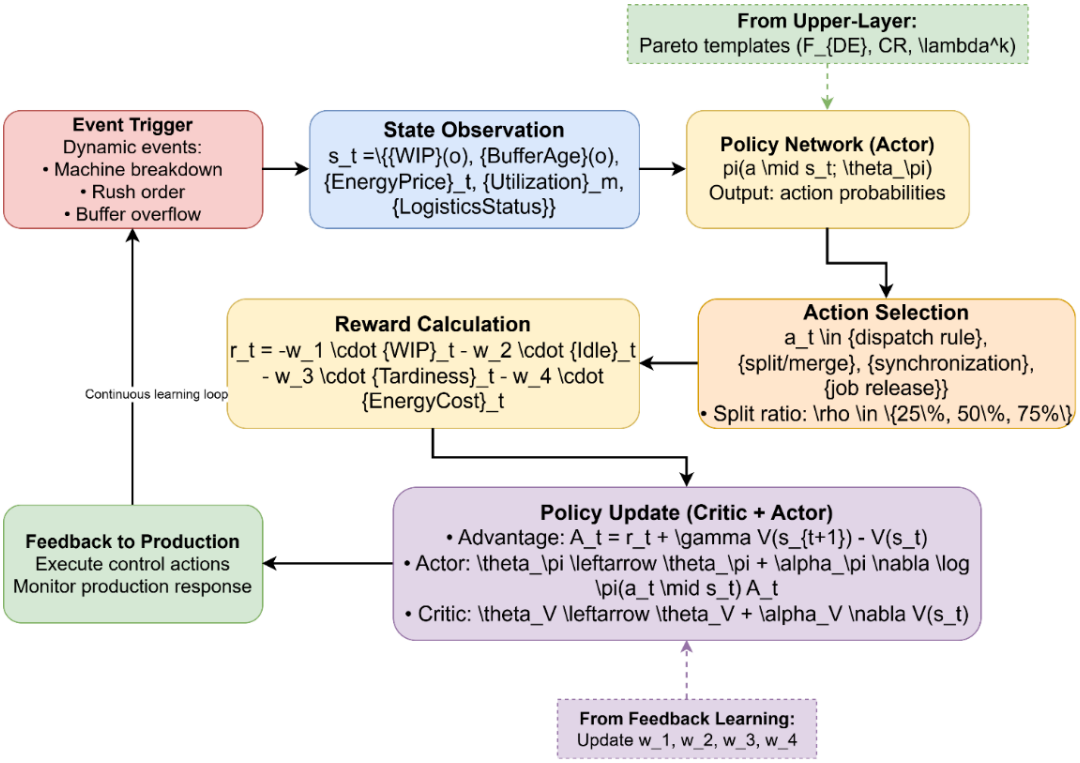


Figure 5. Lower-layer adaptive pull control via reinforcement learning

Figure 5 demonstrates the RL control cycle: Event Trigger→ State Observation→ Policy→ Action→ Reward→ Update→ Feedback.

3.7. Flow synchronization and feasibility control

Three gating mechanisms maintain synchronized flow and enforce feasibility:

(1) Material Kitting Gate: $S_{i,o} \geq r_{i,o}^{\mathrm{kit}}$, ensuring that operations commence only after material readiness.

(2) Buffer Capacity Gate: $\mathrm{WIP}_o(t) \leq \mathrm{WIP}_{\mathrm{cap}}(o)$, preventing buffer overflow and congestion.

(3) Limited-Wait Constraint: $0 \leq S_{i,o+1} - \left(C_{i,o} + \tau_{o\to o+1}\right) \leq \Gamma_{\mathrm{o}}$, bounding the waiting time between consecutive operations to satisfy process quality and technological requirements.

These gates collectively prevent congestion, starvation, and unstable flow between stages while ensuring that all scheduling decisions remain feasible with respect to shop-floor constraints.

3.8. Feedback Learning and Performance Monitoring

A continuous learning mechanism analyzes real-time KPIs and adjusts algorithmic parameters:

$$\{w_1, w_2, w_3, w_4\}_{\mathrm{RL}}, \{F_{\mathrm{DE}}, \mathrm{CR}, \lambda^k\}_{\mathrm{MOEA/D-DE}} \tag{9}$$

Performance metrics include tardiness, energy cost, WIP dwell time, and machine idle rate. The adaptation enhances learning stability and decision precision across layers.

Trigger conditions and update frequency:

To operationalize the feedback learning mechanism, the following trigger events and update frequencies are defined:

- Rolling-horizon re-optimization: Executed at fixed intervals (e.g., every 4 hours) to update scheduling templates based on the latest order backlog, machine status, and energy price forecasts.
- KPI-deviation trigger: If the deviation of key performance indicators (e.g., total tardiness, energy cost) between actual performance and the MOEA/D–DE template exceeds a predefined threshold (e.g., 15%), an immediate upper-layer re-optimization is triggered to regenerate scheduling templates.
- Disturbance-driven adaptation: Machine breakdowns, rush orders, or logistics disruptions first trigger lower-layer RL adaptation for immediate response. Subsequently, the resulting KPI changes are incorporated into the next rolling-horizon optimization cycle to adjust MOEA/D–DE and RL parameters.

This structured feedback protocol ensures that the push–pull architecture can continuously self-improve over time, balancing long-term strategic optimization with short-term operational adaptability under dynamic and uncertain manufacturing conditions.

Invocation procedure. At every decision epoch, the monitor computes the relative deviation between the actual KPI and the KPI predicted by the selected template, for $\kappa \in \{\text{total tardiness}, \text{energy cost}\}$ (extendable to WIP dwell time and idle rate):

$$\Delta_\kappa = \frac{\left|\mathrm{KPI}_\kappa^{\mathrm{act}} - \mathrm{KPI}_\kappa^{\mathrm{tpl}}\right|}{\mathrm{KPI}_\kappa^{\mathrm{tpl}}} \tag{10}$$

The upper layer is invoked when 4 hours have elapsed since the last optimization or when $\max_\kappa \Delta_\kappa \geq 15\%$. Upon invocation, (1) the current order backlog, machine states, and price forecast are

collected; (2) MOEA/D–DE is run on the residual problem; (3) a new template is selected from the new archive by Eq. (4b); and (4) the RL reward weights and the MOEA/D–DE parameters F_{DE}, CR, and λ^k are updated according to Eq. (9). Between invocations only the lower layer acts, so the extra computation introduced by the feedback mechanism consists of the monitoring step (a few arithmetic operations per epoch) and one MOEA/D–DE run per trigger; the measured overhead is reported in Section 4.9.

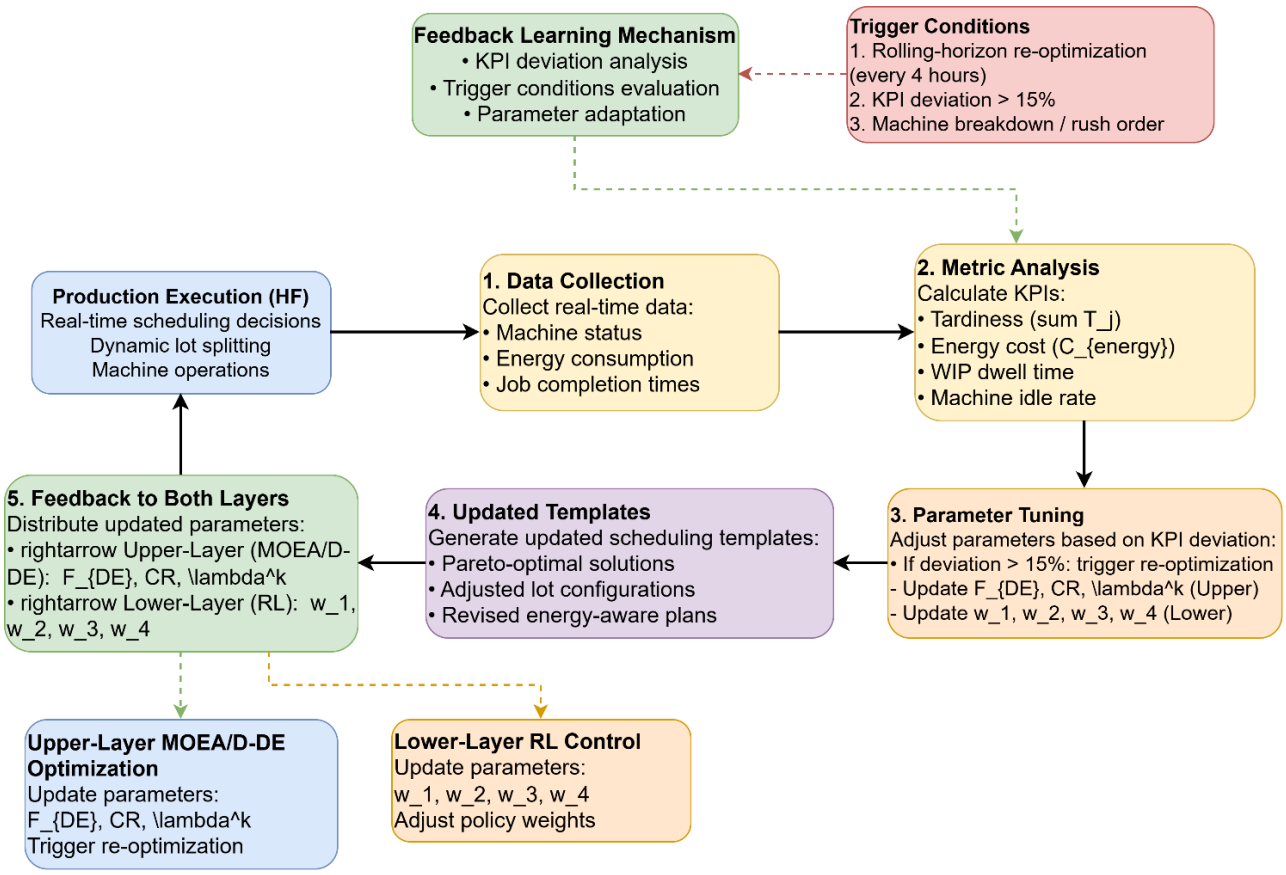


Figure 6. Feedback learning and performance monitoring mechanism.

The figure illustrates the closed feedback cycle: data collection→ metric analysis→ parameter tuning→ updated templates→ feedback to both layers.

4. Experimental results and analysis

In this section, we present a comprehensive evaluation of the proposed HP²-RL framework through simulation experiments based on a multi-stage HFS configuration with finite buffers, parallel heterogeneous machines, and dynamic lot splitting. The experiments are designed to assess the framework's performance in terms of production efficiency, energy performance, environmental impact, and system robustness under stochastic disturbances. Comparative analyses are conducted against rule-based dispatching, static MOEA/D–DE optimization, and pure RL baselines.

4.1. Experimental environment and benchmark settings

The experimental testbed simulates a dynamic hybrid flow shop manufacturing system characterized by heterogeneous machines, multi-stage processing routes, and limited buffer capacities. The system consists of multiple processing stages, each equipped with parallel machines having different processing speeds and energy consumption profiles. Work-in-process (WIP) buffers are explicitly constrained at each stage to reflect realistic shop-floor congestion conditions.

To assess adaptability under uncertainty, multiple order arrival and disturbance scenarios are designed, including rush orders, machine breakdowns, and energy price peaks. These events are triggered dynamically, forcing real-time rescheduling decisions in the pull-control layer. Time-of-use (TOU) electricity tariffs, demand charges, and carbon emission factors are incorporated to evaluate the framework's energy-aware and low-carbon capabilities.

Table 3 summarizes the experimental parameter configuration, including production system settings, order characteristics, disturbance patterns, energy and carbon parameters, and optimization/control hyperparameters.

Table 3. Experimental parameter configuration.

Category	Parameter	Description
Production System	Number of machines	Heterogeneous machines across stages
	Processing stages	Multi-stage hybrid flow shop
	Buffer capacity	Finite WIP limits between stages
Orders	Order types	Regular orders, rush orders
	Arrival pattern	Static + dynamic arrivals
Disturbances	Machine failures	Random breakdown and repair
	Energy events	Peak price / demand charge intervals
	Logistics	Limited transport capacity
Energy	Electricity pricing	Time-of-use tariff
	Demand charge	Peak power-based cost
Carbon	Emission factor	CO_2 per unit energy
Optimization	MOEA/D–DE	Population size, mutation factor
Control	RL parameters	Learning rate, discount factor

Model verification and validation. The simulation model and the scheduling decisions are verified and validated in four steps. (1) Constraint verification: Every schedule or action produced by any method is checked against the machine-capacity constraint, the buffer-capacity gate $\mathrm{WIP}_o(t) \leq \mathrm{WIP}_{\mathrm{cap}}(o)$, the limited-wait window of Eq. (2), and the material-kitting gate of Section 3.7; infeasible decisions are repaired by the feasibility-repair operator or rejected, so that all reported KPIs are computed on feasible schedules. (2) Metric consistency: All indicators in Table 4 are computed by the same evaluation module from the executed event log and are consistent with objectives F_1–F_7 of Section 3.3. (3) Fair and repeatable comparison: All methods are run on identical scenarios (order arrivals, breakdown times, and price events) generated from the same random seeds; each method is run 30 times with independent seeds, and results are reported as mean values accompanied by the statistical tests of Section 4.8. (4) Behavioral validation: The learned policy is examined through the

training curves (Section 4.5), the sensitivity tests on carbon caps and price multipliers (Section 4.6), and the disturbance-response tests (Section 4.7), which verify that the controller behaves plausibly beyond the nominal scenario.

4.2. Performance metrics and mathematical definitions

To provide a multifaceted evaluation of the HP²-RL framework, we employ a comprehensive set of metrics covering production efficiency, energy performance, environmental impact, and system robustness. All metrics are defined consistently with the objective formulations in Section 3.3.

Table 4. Performance metrics summary and mathematical definitions.

Category	Metric	Symbol	Mathematical Definition
Production Performance	Makespan	$C_{\max}$	$C_{\max} = \max_{j \in J}\{C_j\}$
	Total Tardiness	$\sum T_j$	$T_j = \max(0, C_j - d_j)$ $\sum T_j = \sum_{j=1}^{N} \max(0, C_j - d_j)$
	Average Flow Time	$\bar{F}$	$\bar{F} = \frac{1}{\lvert J \rvert} \sum_{j \in J} (C_j - r_j)$
	Average WIP Level	WIP	$\text{WIP} = \frac{1}{T} \int_0^T N(t) dt$
	Machine Utilization	U_m	$U_m = \frac{\sum_j P_{jm}}{T_m^{\text{available}}}$
Energy Performance	Total Energy Consumption	E_{tot}	$E_{\text{tot}} = \sum_m \sum_t P_m(t) \Delta t$
	Peak Power Demand	P_{peak}	$P_{\text{peak}} = \max_t \sum_m P_m(t)$
	Total Energy Cost	C_{energy}	$C_{\text{energy}} = \sum_k E_k \cdot \pi_k + \pi_d \cdot P_{\text{peak}}$
Environmental Impact	Carbon Emissions	CE	$CE = E_{\text{tot}} \cdot \psi$
	Carbon Penalty Cost	C_{carbon}	$C_{\text{carbon}} = \max(0, CE - CE^{\text{cap}}) \cdot \lambda_c$
System Robustness	Recovery Time	RT	$RT = t^{\text{stable}} - t^{\text{dist}}$
	Schedule Deviation	SD	$SD = \sum_{j \in J} \left\lvert S_j^{\text{plan}} - S_j^{\boldsymbol{exec}} \right\rvert$
Learning Performance (RL)	Episode Reward	R	$R = \sum_{t=1}^{T} \gamma^t r_t$
	Reward Function	r_t	$r_t = -(w_1 \cdot \text{WIP}_t + w_2 \cdot E_t + w_3 \cdot \text{Tardiness}_t + w_4 \cdot D_t)$
	Action Response Time	ART	$ART = t^{\text{action}} - t^{\text{event}}$

4.3. Comparative analysis of scheduling performance

The initial phase of the experiments involves benchmarking HP²-RL against three baseline approaches: (i) rule-based dispatching (EDD), (ii) static MOEA/D–DE optimization without real-time adaptation, and (iii) pure reinforcement learning (Pure RL) without upper-layer optimization guidance.

To ensure a fair and reproducible comparison, all methods share the same simulator, scenarios, random seeds, and evaluation module, and their configurations are summarized in Table 5. In particular, the static MOEA/D–DE baseline uses the same optimizer settings as the upper layer of HP²-RL but is executed open-loop, and the pure-RL baseline uses the same Actor–Critic architecture and training budget as the lower layer of HP²-RL without upper-layer guidance, so that the differences between methods are attributable to the proposed components and not to different tuning effort.

Table 5. Configuration of HP²-RL and the baseline methods.

Method	Upper layer (push)	Lower layer (pull)	Handling of dynamic events	Feedback
Rule-based (EDD)	None	Earliest-due-date dispatching; each released job goes to the earliest-available eligible machine	Rules re-applied at every event; jobs of a failed machine are re-dispatched by the same rule	None
Static MOEA/D–DE	MOEA/D–DE with the same encoding (Eq. 4), $N_{\text{pop}} = 112$, $T = 10$, and the same F_{DE}/CR ranges as HP²-RL; solved once at $t = 0$ on the known orders; template selected by Eq. (8)	None (open-loop execution of the template)	Right-shift repair after breakdowns; rush orders appended by EDD; no re-optimization	None
Pure RL	None	Same Actor–Critic architecture, state, action space, reward, and 300 training episodes as HP²-RL, but without template reference	Event-driven decisions by the learned policy	None
HP²-RL (proposed)	MOEA/D–DE with rolling-horizon re-optimization	Actor–Critic guided by the selected template	Policy reacts immediately; upper layer re-invoked at the next trigger	4-h / 15% triggers

As shown in Figure 7, the HP²-RL framework demonstrates significant superiority in efficiency and resource utilization. Under normal operating conditions, HP²-RL achieved a makespan of 1240.5 minutes, outperforming the rule-based approach by approximately 14.5%. In high-disturbance

scenarios, the advantage of HP²-RL becomes even more pronounced, maintaining a high machine utilization of 0.862, whereas static optimization methods (MOEA/D–DE) experienced a significant drop in efficiency due to their inability to adapt to real-time shop-floor changes. Pure RL, while more adaptive than static optimization, exhibited slower convergence and higher variance, resulting in suboptimal long-term performance.

These results confirm that the hierarchical push–pull architecture effectively combines the global search capability of MOEA/D–DE with the real-time responsiveness of RL, yielding superior performance in deterministic and stochastic environments.

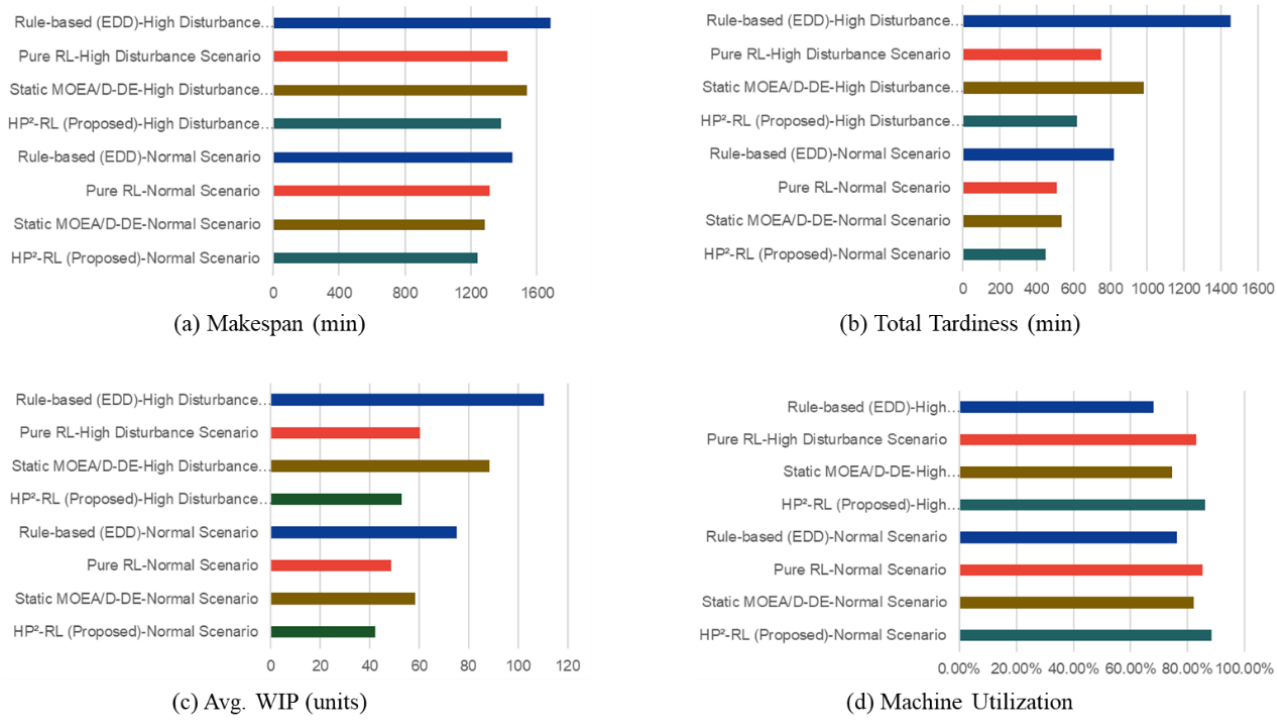


Figure 7. Comparative performance across scheduling scenarios.

4.4. Evaluation of energy efficiency and sustainability metrics

Integrating sustainability into the scheduling logic is one of our primary objectives for this study. As such, we analyze the framework's ability to minimize energy expenditure, energy cost, and carbon intensity under TOU pricing and carbon emission constraints.

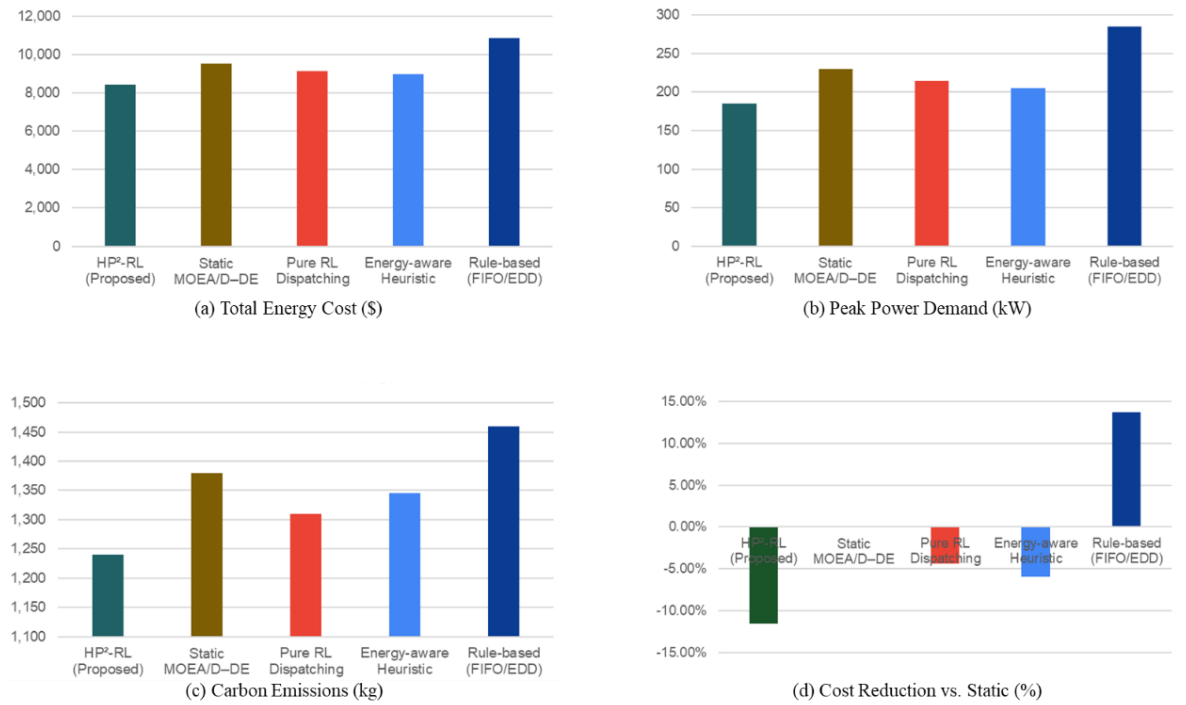


Figure 8. Analysis of total energy cost and peak power demand.

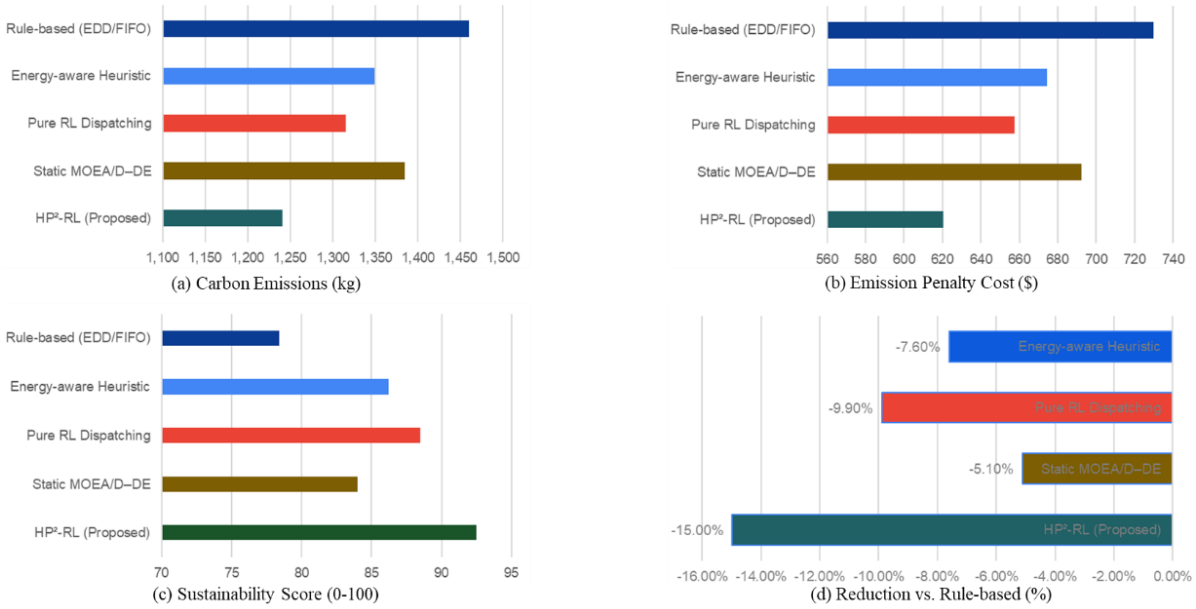


Figure 9. Carbon emissions and sustainability score assessment.

According to the data in Figures 8 and 9, HP²-RL achieved a total energy cost of $8,450, representing an 11.5% reduction compared to static optimization. This improvement is attributed to the RL agent's ability to learn and avoid peak-tariff periods through energy-aware job release and lot-splitting decisions. Furthermore, the framework attained the highest sustainability score (92.5) with the lowest carbon footprint (1,241 kg), validating its effectiveness as a tool for green manufacturing.

These findings indicate that explicitly incorporating energy and carbon considerations into the optimization and learning layers enables substantial reductions in operational costs and environmental impact without compromising production efficiency.

4.5. Training dynamics and algorithm convergence

The stability of the reinforcement learning component is crucial for industrial deployment. We monitor the reward trajectories and variance during the training phase to assess convergence behavior and learning stability.

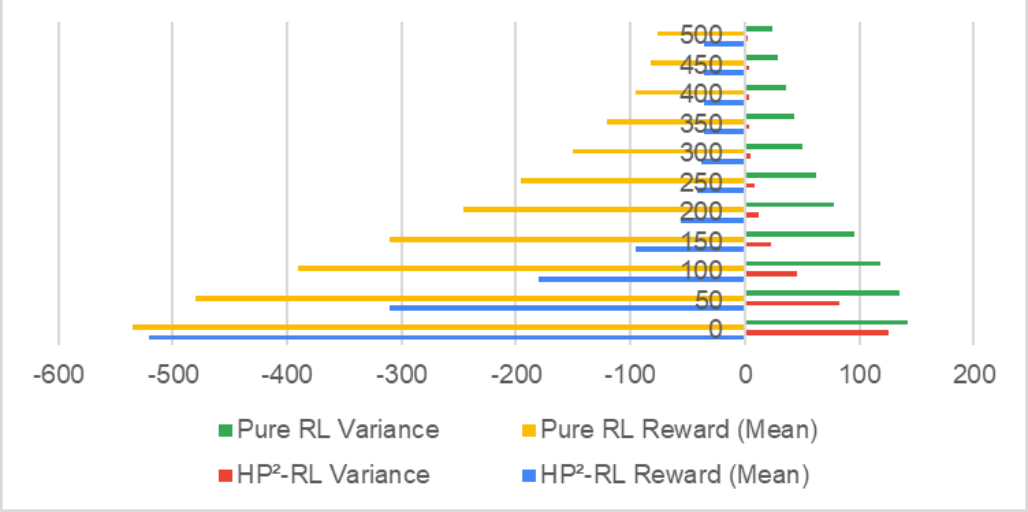


Figure 10. Training dynamics and algorithm convergence.

Figure 10 illustrates that HP²-RL reaches a stable convergence state (mean reward ≈ −38.5) within approximately 300 episodes. Compared to Pure RL, the hybrid approach exhibits significantly lower variance, indicating that the upper-layer MOEA/D–DE optimization effectively guides the lower-layer RL exploration, thereby narrowing the search space and accelerating the learning process. Pure RL, by contrast, requires more episodes to converge and displays higher fluctuation in reward values, reflecting greater exploration noise and less structured policy updates.

These results demonstrate that the push–pull architecture not only improves final performance but also enhances learning efficiency and stability, which are critical for practical deployment in dynamic manufacturing environments.

4.6. Sensitivity analysis under economic and environmental constraints

To assess the robustness of the framework against external fluctuations, sensitivity tests are performed on carbon constraints and electricity price volatility. Multiple scenarios are constructed by varying carbon emission caps and applying electricity price multipliers (1.0×, 1.25×, 1.5×) to simulate stricter environmental regulations and more volatile energy markets.

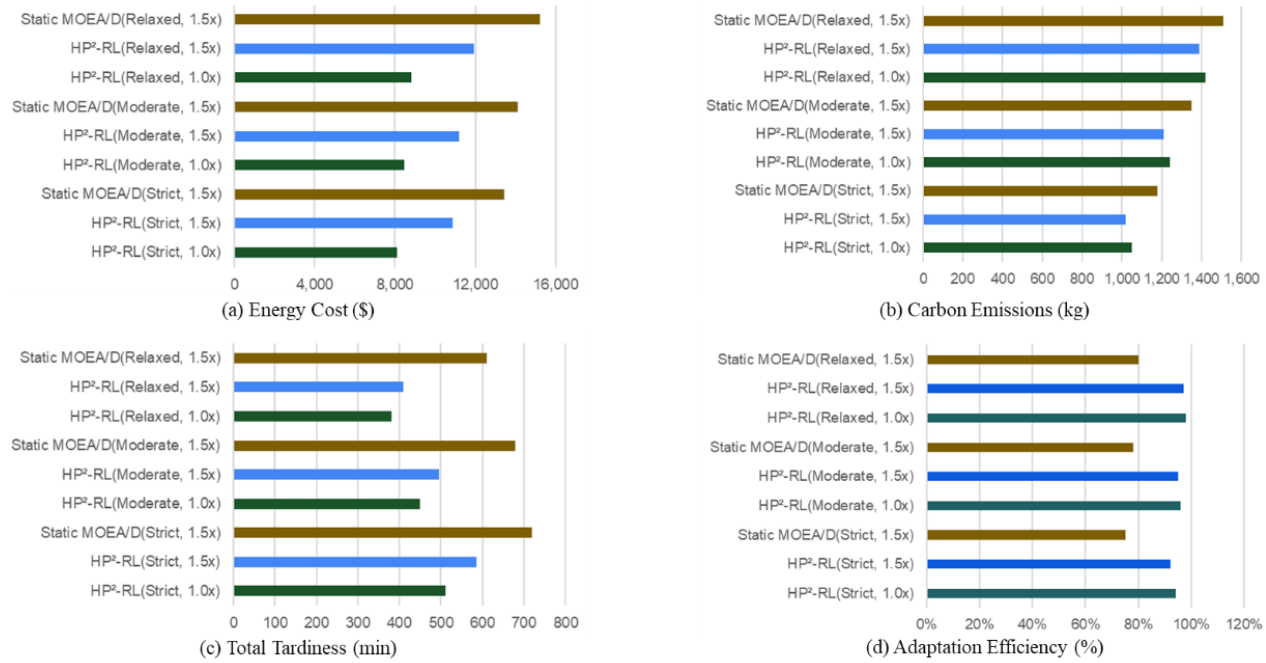


Figure 11. System adaptation efficiency under varying constraints and price multipliers.

The results in Figure 11 demonstrate that even under strict carbon constraints and a 1.5× electricity price multiplier, HP²-RL maintained a high adaptation efficiency of 0.92. This suggests that the proposed framework can dynamically balance production targets with evolving economic and regulatory requirements, leveraging the flexibility provided by dynamic lot splitting and energy-aware dispatching.

4.7. Resilience and disturbance response analysis

For the final evaluation, we focus on the system's ability to recover from unexpected shop-floor events, such as rush orders and machine failures. Recovery time, stability index, and performance degradation rate are used as key metrics to quantify resilience.

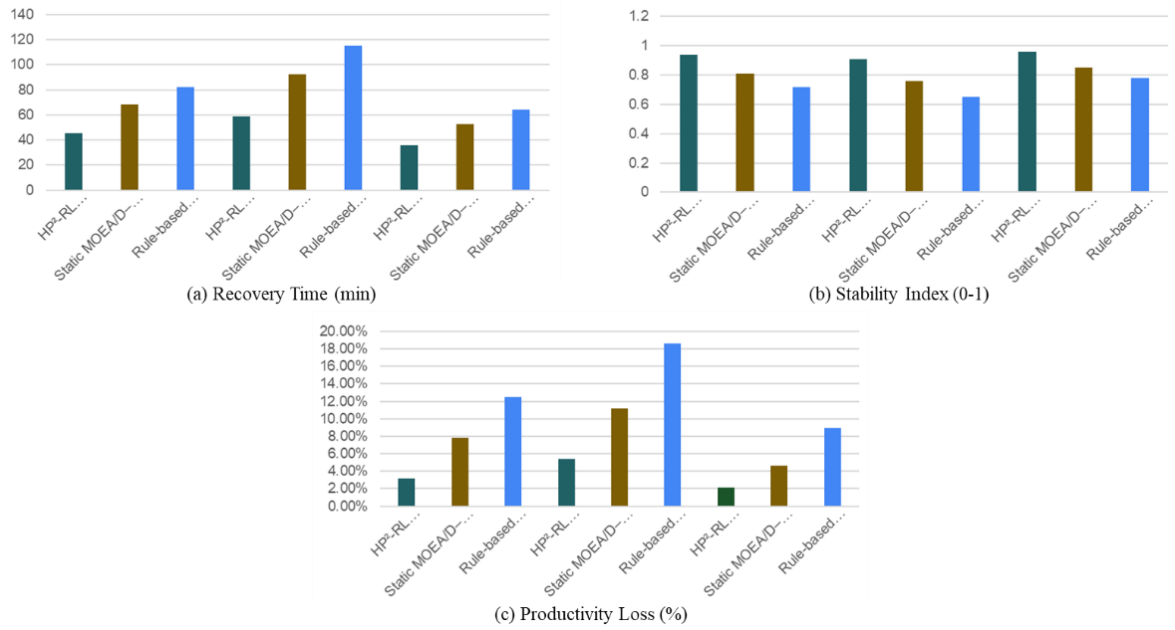


Figure 12. Recovery time and stability index across disturbance types.

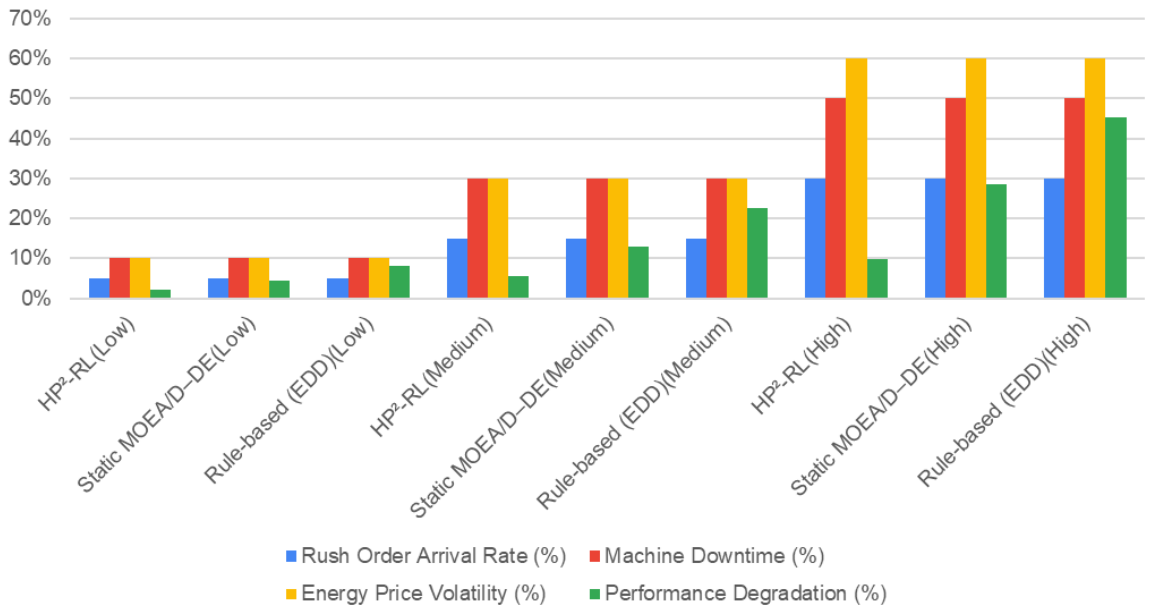


Figure 13. Performance degradation rate relative to disturbance intensity.

As depicted in Figure 12, HP²-RL exhibits superior resilience, with a recovery time for machine failures (58.6 min) that is significantly shorter than the rule-based baseline (115 min). Furthermore, Figure 13 shows that as disturbance intensity increases, the performance degradation rate of HP²-RL remains remarkably stable (under 10%), whereas the static and rule-based methods suffer from exponential efficiency losses. This confirms that the proposed hybrid architecture provides the necessary flexibility to maintain high performance in volatile manufacturing environments.

4.8. Ablation study and statistical significance analysis

To identify which component makes the key contribution, and to justify the combination of components, four variants are created by removing mechanisms from HP²-RL: (i) w/o Feedback: the template is generated once and no rolling-horizon or KPI-triggered re-optimization and no parameter adaptation is performed; (ii) w/o Lot-Splitting: the split/merge actions are removed from the action space and from the chromosome; (iii) w/o Push (Pure RL): the upper layer, and hence the template and the feedback, is removed; and (iv) w/o Pull (Static Opt): the RL controller, lot splitting, and feedback are removed, and the static template is executed open-loop. All variants use the same scenarios, seeds, and hyperparameters as the full framework and are run 30 times independently. Differences with respect to the full HP²-RL are tested with the paired two-sided Wilcoxon signed-rank test at the 0.05 level, with samples paired by scenario and seed.

Table 6. Ablation study and Wilcoxon signed-rank test results (mean of 30 independent runs).

Variant	Push	Pull	Lot split	Feed-back	Makespan (min)	Energy cost ($)	Recovery time (min)	Wilcoxon p vs. full
Full HP²-RL	✓	✓	✓	✓	1240.5	8450	58.6	—
w/o Feedback	✓	✓	✓	✗	1295.2	8920	78.2	0.0021
w/o Lot-Splitting	✓	✓	✗	✓	1340.8	9150	85.4	0.0004
w/o Push (Pure RL)	✗	✓	✓	✗	1380.0	9100	92.1	0.0001
w/o Pull (Static Opt)	✓	✗	✗	✗	1310.0	9550	115.0	0.0001

Table 6 shows that removing any single component degrades makespan, energy cost, and recovery time, and that all differences are statistically significant ($p < 0.05$); the conclusions are unchanged under a Bonferroni correction for the four comparisons ($\alpha = 0.0125$). Removing the upper layer (pure RL) causes the largest makespan increase (+11.2%), and removing the lower layer (static optimization) causes the largest energy-cost increase (+13.0%) and almost doubles the recovery time (+96%), which shows that global templates and real-time control address different aspects of performance. Removing dynamic lot splitting raises makespan by 8.1%, energy cost by 8.3%, and recovery time by 45.7%, and removing the feedback loop raises them by 4.4%, 5.6%, and 33.4%, respectively; the feedback loop is thus the smallest contributor to steady-state efficiency but still matters for disturbance recovery. No single component reproduces the performance of the combination, which supports the chosen design. The corresponding Wilcoxon test of HP²-RL against the EDD baseline gave p = insert value.

4.9. Computational complexity and practical overhead

The cost of one upper-layer run is $O\left(G_{\max} \cdot N_{\text{pop}} \cdot (C_{\text{eval}} + 7T)\right)$, where C_{eval} is the cost of decoding and evaluating one chromosome on the seven objectives and $7T$ accounts for the neighborhood updates with the Tchebycheff function. The lower layer requires one forward pass of the

Actor per decision, whose cost does not depend on the number of waiting jobs, because the state vector has fixed dimension. The measured overheads are summarized in Table 7.

With a 4-hour rolling horizon, a re-optimization of about 42.5 s occupied less than 0.3% of the horizon, and an inference time below 3.8 ms per decision was negligible compared with the minute-scale transfer and processing times of the shop floor (Table 2). KPI-triggered re-optimizations cost the same per run, and the monitoring step was negligible. The framework was therefore suitable for real-time dispatching, and the additional cost of the feedback mechanism was limited to the periodic or triggered MOEA/D–DE runs.

Table 7. Computational overhead of HP²-RL.

Item	Setting	Measured cost
Hardware / software	Intel Core i9 CPU, NVIDIA RTX 4090 GPU; Python/PyTorch	—
Upper-layer re-optimization (one MOEA/D–DE run)	N_{pop} = 112, G_{max} within 100–300	42.5 ± 3.2 s
Lower-layer training (offline, once)	300 episodes	insert wall-clock time
Lower-layer inference (per scheduling decision)	Single forward pass of the Actor	< 3.8 ms
Memory footprint	Networks and replay buffer	< 1.2 GB
Feedback monitoring	Eq. (7a) per decision epoch	O(1), a few arithmetic operations

4.10. Why the proposed algorithm is effective

The experimental evidence indicates five mechanisms behind the performance of HP²-RL. (1) Guided exploration: The template confined the RL policy to a feasible, energy-aware region of the state–action space, which explained the lower reward variance and the convergence within about 300 episodes in Figure 10 compared with pure RL; without it (w/o Push), makespan and energy cost deteriorated by 11.2% and 7.7% (Table 6). (2) Preserved selection pressure in seven dimensions: Decomposition with Tchebycheff aggregation kept every subproblem comparable, so the upper layer still delivered well-spread, high-quality templates where dominance ranking would stall; these templates carried global TOU- and carbon-aware load shifting that a purely myopic policy could not discover. (3) Run-time adaptivity: The pull layer absorbed breakdowns and rush orders immediately, which was why the recovery time was 58.6 min compared with 115 min for rule-based dispatching (Figure 12) and why the degradation rate remained below 10% as disturbance intensity grew (Figure 13). (4) Structural flexibility: Dynamic lot splitting and merging relieved bottlenecks and buffers and removing it cost 8.1% in makespan and 45.7% in recovery time. (5) Template refresh: The KPI-triggered feedback realigned the template with the actual shop state, preventing the drift that otherwise lengthened recovery by 33.4%. Because each mechanism covered a weakness of the others, the benefits were complementary rather than redundant. These explanations are empirical and derive from a simulated testbed; a formal convergence analysis and validation on real production data remain topics for future work.

5. Conclusion

In this research, we developed and validated a (HP²-RL framework for multi-objective dynamic scheduling in HFS environments with finite buffers, dynamic lot splitting, and energy–carbon constraints. By integrating global MOEA/D–DE optimization with real-time Actor–Critic RL control, we addressed the critical trade-offs between production efficiency, energy conservation, carbon mitigation, and system resilience.

The key findings of this study are summarized as follows:

- Superior multi-objective performance: Experimental results demonstrated that HP²-RL consistently outperformed traditional rule-based dispatching, static MOEA/D–DE optimization, and pure RL baselines. Under normal conditions, it achieved a makespan of 1240.5 minutes with high machine utilization (0.885), confirming that the upper-layer optimization provides high-quality strategic guidance for the shop floor.
- Excellence in green manufacturing: The framework showed significant capabilities in energy management. By leveraging the RL agent's ability to shift loads during peak-tariff periods and optimize lot-splitting configurations, the system reduced total energy costs by 11.5% relative to static MOEA/D–DE and achieved the highest sustainability score (92.5) among all compared methods.
- Enhanced system resilience: One of the most significant contributions was the framework's robustness against stochastic disturbances. HP²-RL maintained a stability index of 0.91 even during machine failures, with a recovery time nearly 50% faster than rule-based approaches. This confirms that the pull mechanism can effectively absorb shop-floor volatility without compromising global targets.
- Accelerated learning and convergence: By utilizing optimization templates from the MOEA/D–DE layer as reference trajectories, the RL training process was significantly stabilized. Convergence was reached within 300 episodes, a substantial improvement over pure RL methods, which often suffer from high variance and slow exploration.
- Verified contributions and practical feasibility: Ablation experiments showed that the push layer, the pull layer, dynamic lot splitting, and the feedback loop each contributed significantly (Wilcoxon signed-rank test, $p < 0.05$, 30 independent runs), while a re-optimization took about 42.5 s and an inference less than 3.8 ms, which supports real-time deployment.

Overall, the proposed HP²-RL framework provides a systematic and implementable approach for integrating multi-objective optimization, reinforcement learning, and energy–carbon considerations in dynamic HFS scheduling, offering a viable pathway toward smart, low-carbon, and resilient manufacturing systems.

While the HP²-RL framework demonstrates promising results, several avenues for future research remain:

- Multi-agent RL scalability: Researchers could explore the transition from a centralized RL controller to a Multi-Agent Reinforcement Learning (MARL) architecture, where each machine or production cell acts as an autonomous agent. This extension would enable the framework to handle even larger and more complex shop-floor scales while preserving decentralization and scalability.
- Integration with Digital Twin (DT) technology: Incorporating high-fidelity Digital Twin data could further enhance the predictive capabilities of the HP²-RL framework, enabling look-ahead scheduling that anticipates disturbances before they occur. Real-time synchronization between the DT and the HP²-RL controller would enable more accurate state estimation and proactive decision-making.
- Expansion to circular economy objectives: Future versions of the framework could include

metrics related to remanufacturing, material recycling, and product end-of-life handling, aligning the scheduling logic with broader circular economy goals beyond energy and carbon reduction.

- Human–robot collaboration (HRC): Considering the increasing role of human operators in smart factories, integrating human factors, such as fatigue levels, skill heterogeneity, and variable manual processing times, into the scheduling state space would be a valuable extension. This would facilitate more realistic and human-centric scheduling policies in mixed human–robot production environments.
- Theoretical convergence analysis: While empirical results demonstrate rapid convergence and stability, future work could include formal convergence analysis and sample complexity bounds for the proposed push–pull architecture under specific assumptions on the environment dynamics and reward structure.

Availability of supporting data

The data supporting the findings of this study are generated through simulation experiments based on the proposed HP²-RL scheduling framework. The datasets and simulation configurations used in the current study are available from the corresponding author upon reasonable request.

Use of Generative-AI tools declaration

The author declares that no Generative AI or AI-assisted technologies were used in the writing, data analysis, or technical implementation of this research work.

Conflict of interest

The author declares no conflict of interest.

References

1. F. Tao, J. F. Cheng, Q. L. Qi, M. Zhang, F. Y. Sui, Digital twin-driven product design, manufacturing and service with big data, *Int. J. Adv. Manuf. Technol.*, **94** (2017), 3563–3576. https://doi.org/10.1007/s00170-017-0233-1
2. L. R. Lu, Q. W. Deng, Y. Y. Hu, Q. Luo, S. C. Gao, J. X. Zhang, Deep reinforcement learning for dynamic scheduling in distributed heterogeneous flexible job shop with integrated inventory allocation and product delivery, *Eng. Appl. Artif. Intell.,* **162** (2025), 112681. https://doi.org/10.1016/j.engappai.2025.112681
3. Y. J. Qu, X. G. Ming, Z. W. Liu, X. Y. Zhang, Z. T. Hou, Smart manufacturing systems: state of the art and future trends, *Int. J. Adv. Manuf. Technol.*, **103** (2019), 3751–3768. https://doi.org/10.1007/s00170-019-03754-7
4. M. Füchtenhans, C. H. Glock, Energy-efficient job shop scheduling considering processing speed and incentive-based programs, *IFAC-PapersOnLine*, **55** (2022), 192–197. https://doi.org/10.1016/j.ifacol.2022.09.389
5. G. Mouzon, M. B. Yildirim, J. Twomey, Operational methods for minimization of energy consumption of manufacturing equipment, *Int. J. Prod. Res.*, **45** (2007), 4247–4271. https://doi.org/10.1080/00207540701450013

6. R. Li, L. Wang, W. Gong, J. F. Chen, Z. X. Pan, Y. Wu, et al., Evolutionary computation and reinforcement learning integrated algorithm for distributed heterogeneous flowshop scheduling, *Eng. Appl. Artif. Intell.*, **135** (2024), 108775. https://doi.org/10.1016/j.engappai.2024.108775
7. Y. Tang, L. Shen, S. Han, Low-carbon flexible job shop scheduling problem based on deep reinforcement learning, *Sustainability,* **16** (2024), 4544. https://doi.org/10.3390/su16114544
8. M. L. Pinedo, *Scheduling: Theory, algorithms, and systems*, Springer Cham, 2022. https://doi.org/10.1007/978-3-031-05921-6
9. D. Ouelhadj, S. Petrovic, A survey of dynamic scheduling in manufacturing systems, *J. Sched.*, **12** (2009), 417–431. https://doi.org/10.1007/s10951-008-0090-8
10. R. Ruiz, J. A. Vázquez-Rodríguez, The hybrid flow shop scheduling problem, *Eur. J. Oper. Res.*, **205** (2010), 1–18. https://doi.org/10.1016/j.ejor.2009.09.024
11. P. Godinho, F. G. Branco, Adaptive policies for multi-mode project scheduling under uncertainty, *Eur. J. Oper. Res.*, **216** (2012), 553–562. https://doi.org/10.1016/j.ejor.2011.08.010
12. R. L. Haupt, S. E. Haupt, *Practical genetic algorithms*, Wiley, 2003. https://doi.org/10.1002/0471671746
13. M. F. Ashby, *Materials and the environment,* ScienceDirect*,* 2013. https://doi.org/10.1016/C2010-0-66554-0
14. Y. Dong, H. Zhang, C. Wang, X. Zhou, Soft actor-critic DRL algorithm for interval optimal dispatch of integrated energy systems with uncertainty in demand response and renewable energy, *Eng. Appl. Artif. Intell.*, **127** (2024), 107230. https://doi.org/10.1016/j.engappai.2023.107230
15. A. Mencaroni, P. Leyman, B. Raa, S. D. Vuyst, D. Claeys, Towards net-zero manufacturing: Carbon-aware scheduling for GHG emissions reduction, *J. Clean. Prod.*, **529** (2025), 146787. https://doi.org/10.1016/j.jclepro.2025.146787
16. K. Gao, Y. Huang, A. Sadollah, L. Wang, A review of energy-efficient scheduling in intelligent production systems, *Complex Intell. Syst.*, **6** (2020), 237–249. https://doi.org/10.1007/s40747-019-00122-6
17. L. Glomb, F. Liers, F. Rösel, A rolling-horizon approach for multi-period optimization, *Eur. J. Oper. Res*., **300** (2022), 189–206. https://doi.org/10.1016/j.ejor.2021.07.043
18. K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, *IEEE Trans. Evol. Comput*., **6** (2002), 182–197. https://doi.org/10.1109/4235.996017
19. Q. Zhang, H. Li, MOEA/D: A multiobjective evolutionary algorithm based on decomposition, *IEEE Trans. Evol. Comput.*, **11** (2007), 712–731. https://doi.org/10.1109/TEVC.2007.892759
20. C. A. C. Coello, G. B. Lamont, D. A. V. Veldhuizen, *Evolutionary algorithms for solving multi-objective problems*, Springer, 2007. https://doi.org/10.1007/978-0-387-36797-2
21. K. Tamssaouet, S. Dauzère-Pérès, S. Knopp, A. Bitar, C. Yugma, Multiobjective optimization for complex flexible job-shop scheduling problems, *Eur. J. Oper. Res.*, **296** (2022), 87–100. https://doi.org/10.1016/j.ejor.2021.03.069
22. G. Fu, T. Yu, W. Li, Q. Deng, B. Yang, A decomposition-based multiobjective optimization evolutionary algorithm with adaptive weight generation strategy, *Math. Probl. Eng*., **2021** (2021), 2764558. https://doi.org/10.1155/2021/2764558
23. W. Chen, J. Wang, G. Yu, Y. Hu, Energy-efficient hybrid flow-shop scheduling under time-of-use and ladder electricity tariffs, *Appl. Sci.*, **12** (2022), 6456. https://doi.org/10.3390/app12136456

24. D. Schweisgut, A. Benoit, Y. Robert, H. Meyerhenke, Carbon-aware workflow scheduling with fixed mapping and deadline constraint, *Proc. Int. Conf. Parallel Process.*, 2025, 627–637. https://doi.org/10.1145/3754598.3754673
25. M. S. Cavallieri, E. Viles, J. R. Montoya-Torresb, Sustainable production scheduling: insights from real-life applications, *Prod. Plan. Control.*, **36** (2025), 1595–1622. https://doi.org/10.1080/09537287.2024.2389879
26. B. Wang, Q. Li, Rolling horizon procedure for large-scale job-shop scheduling problems, *IEEE Int. Conf. Autom. Logist*., 2007, 18–21. https://doi.org/10.1109/ICAL.2007.4338679
27. R. Yu, D. Oron, Single-machine scheduling with fixed energy recharging times to minimize the number of late jobs and the number of just-in-time jobs: A parameterized complexity analysis, *Eur. J. Oper. Res.,* **324** (2025), 40–48. https://doi.org/10.1016/j.ejor.2025.01.007
28. B. Kocot, P. Czarnul, J. Proficz, Energy-aware scheduling for high-performance computing systems: A survey, *Energies*, **16** (2023), 890. https://doi.org/10.3390/en16020890
29. R. S. Sutton, A. G. Barto, *Reinforcement learning: An introduction*, MIT Press, 2018.
30. Z. Wang, M. Gombolay, Learning scheduling policies for multi-robot coordination with graph attention networks, *IEEE Robot. Autom. Lett.,* **5** (2020), 4509–4516. https://doi.org/10.1109/LRA.2020.3002198
31. Z. Luo, C. Jiang, L. Liu, X. Zheng, H. Ma, F. Dong, Deep-reinforcement-learning-based production scheduling in industrial internet of things, *IEEE Internet Things J.*, **10** (2023), 19725–19739. https://doi.org/10.1109/JIOT.2023.3283056
32. M. Monaci, V. Agasucci, G. Grani, An actor-critic algorithm with policy gradients to solve the job shop scheduling problem using deep double recurrent agents, *Eur. J. Oper. Res.,* **312** (2024), 910–926. https://doi.org/10.1016/j.ejor.2023.07.037
33. P. Stöckermann, S. Feudel, A. Immordino, N. Hayen, T. Altenmüller, M. Gebser, et al., Reinforcement learning based dispatching solutions in semiconductor manufacturing: a literature review on validation and deployment, *Prod. Manuf. Res.*, **13** (2025), 2582472. https://doi.org/10.1080/21693277.2025.2582472
34. J. Jang, D. Klabjan, H. Liu, N. S. Patel, X. Li, B. Ananthanarayanan, et al., Scalable multi-agent reinforcement learning for factory-wide dynamic scheduling in semiconductor manufacturing, *Eng. Appl. Artif. Intell.,* **161** (2025), 112168. https://doi.org/10.1016/j.engappai.2025.112168
35. F. Eberle, F. Fischer, J. Matuschke, N. Megow, On index policies for stochastic minsum scheduling, *Oper. Res. Lett.,* **47** (2019), 213–218. https://doi.org/10.1016/j.orl.2019.03.007
36. B. Wang, H. Wang, Q. Yan, E. Ma, Multi-agent reinforcement learning-aided evolutionary algorithm for a many-objective distributed hybrid flow shop scheduling problem, *Swarm Evol. Comput.*, **97** (2025), 101991. https://doi.org/10.1016/j.swevo.2025.101991
37. Z. Pan, D. Lei, L. Wang, A knowledge-based two-population optimization algorithm for distributed energy-efficient parallel machines scheduling, *IEEE Trans. Cybern.,* **52** (2022), 5051–5063. https://doi.org/10.1109/TCYB.2020.3026571
38. H. Wang, B. R. Sarker, J. Li, J. Li, Adaptive scheduling for assembly job shop with uncertain assembly times based on dual Q-learning, *Int. J. Prod. Res.,* **59** (2021), 5867–5883. https://doi.org/10.1080/00207543.2020.1794075

39. F. Zhao, Z. Fu, L. Wang, H. Sang, A heterogeneous graph reinforcement learning framework with question-aware neighborhood aggregation and interoption prompt attention for dynamic flexible job shop scheduling problem, *IEEE Trans. Ind. Inform.,* **22** (2026), 2863–2874. https://doi.org/10.1109/TII.2025.3646962
40. S. Chen, X. Wang, Y. Wang, X. Gu, A knowledge-driven many-objective algorithm for energy-efficient distributed heterogeneous hybrid flowshop scheduling with lot-streaming, *Swarm Evol. Comput.*, **91** (2024), 101771. https://doi.org/10.1016/j.swevo.2024.101771



AIMS Press

© 2026 the Author(s), licensee AIMS Press. This is an open access article distributed under the terms of the Creative Commons Attribution License (https://creativecommons.org/licenses/by/4.0)