Research on Dynamic Energy-Efficient Scheduling of Flexible Job Shop Based on Deep Reinforcement Learning

Fanyao Gao

School of Business Administration, Liaoning Technical University, Huludao, Liaoning, 125105, China
E-mail: 1023892066@qq.com

Abstract. The Flexible Job Shop Scheduling Problem (FJSP) is a core challenge in manufacturing optimization under dynamic disturbances and energy constraints. This paper proposes a hierarchical deep reinforcement learning method for dynamic energy-efficient scheduling. The framework decomposes decisions into two layers: the upper agent selects an efficiency-priority, energy-priority, or balanced mode based on global workshop state; the lower agent performs job sequencing and machine allocation guided by the selected mode. Deep Q-Networks (DQN) are employed with a phased training strategy. Experiments under the high disturbance scenario show that the proposed method reduces makespan by 12.3% and energy consumption by 9.6% compared to single-agent DQN, and by 24.3% and 19.9% respectively compared to the SPT rule, verifying the effectiveness of the hierarchical architecture.

Keywords: Flexible job shop scheduling; Deep reinforcement learning; Hierarchical decision-making; Energy-efficient scheduling; Dynamic disturbances.

1 Introduction

Manufacturing production scheduling is crucial for both efficiency and energy consumption. The Flexible Job Shop Scheduling Problem (FJSP), characterized by complex machine allocation and operation sequencing, has become a core challenge[1]. With the deepening of green manufacturing concepts, energy-saving objectives have been integrated into scheduling optimization, forming Energy-Efficient FJSP (EEFJSP)[2]. In real-world production, dynamic disturbances such as urgent order insertions and machine breakdowns occur frequently, rendering pre-established schedules ineffective and exacerbating energy waste[3]. Dynamic disturbances not only disrupt the predefined production timetable but also introduce additional energy penalties that are often overlooked in static models. For example, machine breakdowns force unfinished operations to wait or restart, prolonging idle periods with non-negligible standby power consumption. Likewise, urgent order insertions cause frequent setup changes and rerouting, which increase non-processing energy usage such as material handling and tool adjustments. Empirical evidence from real-world workshops indicates that total energy con-

L. W. Koon et al. (eds.), *Proceedings of the 2026 7th International Conference on Management Science and Engineering Management (ICMSEM 2026)*, Advances in Economics, Business and Management Research 399,
https://doi.org/10.2991/978-94-6239-758-3_17

sumption under static schedules can exceed that of adaptive reactive policies by 20–30% in highly volatile environments. However, conventional mathematical programming and heuristic rules lack the computational agility to re-optimize in real time, while simple dispatching rules are myopic and incapable of balancing global makespan and energy trade-offs. Therefore, an intelligent approach that perceives disturbances online and adjusts scheduling decisions adaptively is urgently needed.

Traditional methods based on mathematical programming or heuristic rules struggle to respond to these dynamic changes in real-time[4]. Deep Reinforcement Learning (DRL) has shown great potential in dynamic scheduling due to its ability to learn adaptive policies through interaction[5].However, single-agent DRL couples macro-level task allocation with micro-level energy control within the same network, leading to action space explosion and training difficulties[6]. Hierarchical deep reinforcement learning, which decomposes decisions across different agent levels, offers a new approach to decoupling complex scheduling decisions[7]. However, few existing DRL-based approaches explicitly address the coupled dynamics of machine breakdowns and energy-aware dispatching under time-varying shop-floor states, let alone achieve interpretable mode switching. This motivates us to design a two-agent hierarchy that separates strategic prioritization from operational execution, enabling both adaptive response and transparent decision logic.Therefore, this paper proposes a hierarchical DRL framework to achieve collaborative optimization of production efficiency and energy consumption.

2 Problem Description and Modeling

2.1 Problem Description

Consider a flexible job shop with n jobs and m machines. Each job consists of several operations with precedence constraints. Each operation can be processed on multiple alternative machines with different processing times and energy rates. Two types of dynamic disturbances occur randomly: urgent order insertion and machine breakdown. The objective is to minimize makespan $C_{\max}$ and total energy consumption E_{total}.

2.2 Mathematical Model

Define the following symbols:

J_{ij} the i-th job, $i = 1, \ldots, n$;

O_{ij}: the j-th operation of job i;

M_k: the k-th machine, $k = 1, \ldots, m$;

p_{ijk}: processing time of O_{ij} on M_k;

e_{ijk}: energy consumption rate of O_{ij} on M_k during processing;

e_k^{idle}: idle energy consumption rate of M_k;

t_{ijk}^{start}, t_{ijk}^{end}: start and end times;

$x_{ijk} \in 0,1$: 1 if O_{ij} is assigned to M_k.

Constraints:

$\sum_k x_{ijk} = 1$(each operation assigned to one machine);
$t^{\text{end}}_{i,j-1,k} \le t^{\text{start}}_{ijk}$ (operation precedence);
A machine processes only one operation at a time;
Schedules adjust in real-time upon dynamic events.
Optimization objectives:

$$\min C_{\max} = \max_{i,j} t^{\text{end}}_{ij}$$

$$min\ E_{\text{total}} = \sum_k \left(\sum_{\text{processing}} e_{ijk} p_{ijk} + \int_{\text{idle}} e^{\text{idle}}_k dt \right)$$

Energy consumption includes both processing power and idle power. Processing power is calculated as energy rate multiplied by processing time; idle power accounts for standby periods between operations, which are particularly significant under dynamic disturbances.

2.3 Markov Decision Process Formulation

The scheduling problem is modeled as an MDP:

State s_t:machine statuses, pending operations, dynamic event flags.

Action a_t:divided into upper-level (strategy mode) and lower-level (operation selection + machine allocation).

Reward r_t:lower agent receives r_{step} per completed operation; upper agent receives weighted negative reward at each decision cycle based on average makespan and energy.

3 Hierarchical Deep Reinforcement Learning Method

3.1 Hierarchical Decision Framework

A two-layer agent architecture is designed. The upper agent perceives global workshop state and outputs one of three discrete strategy modes at fixed intervals: Efficiency Priority (η), Energy Priority (ε), or Balanced Mode (β). The lower agent receives the current mode and performs operation selection and machine allocation.

3.2 Upper-Level Agent Design

State Space: backlogged orders, average due date urgency, average machine load, urgent order flag, breakdown flag, current strategy duration.

Action Space: $\{\eta, \varepsilon, \beta\}$.

Decision Cycle:every N completed jobs.

Reward Function:

$$R_{\text{upper}} = -(\alpha \cdot \bar{C}_{\max} + \beta \cdot \bar{E}_{\text{total}})$$

where α, β are weight coefficients. The negative sign encourages maximizing performance.

3.3 Lower-Level Agent Design

The lower-level agent employs a two-step mechanism:

Operation Selection:selects one operation from pending operations based on priority scores.

Machine Allocation:assigns a machine to the selected operation based on scores considering processing time, energy, and current load.

State Space:current operation features, available machine statuses, and the upper-level strategy mode.

Action Space:priority scores for operations; allocation scores for machines.

Reward Function:

$$r_{\text{lower}} = r_{\text{step}} - \lambda \cdot (E_{excess})$$

where λ is a penalty coefficient, and E_{excess} represents energy exceeding a baseline threshold.

3.4 DQN-based Algorithm Implementation

Deep Q-Networks (DQN) train the upper and lower agents separately. The upper network takes global state and outputs Q-values for each mode. The lower network shares feature extraction before branching to operation and machine outputs:

$$L(\theta) = E_{(s,a,r,s')\sim D}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta^{-}) - Q(s, a; \theta)\right)^2\right]$$

where $\gamma = 0.95$, replay buffer capacity 10^5, target network updates every 500 steps, and ϵ-greedy exploration decays from 1.0 to 0.1.

A phased training strategy mitigates non-stationarity:

Phase 1: Fix upper strategy to uniform distribution, train lower network.

Phase 2: Fix lower network, train upper network for optimal switching timing.

Phase 3: Joint fine-tuning with reduced learning rate.

4 Experimental Design and Result Analysis

4.1 Simulation Environment Construction

A virtual flexible job shop is built using discrete event simulation (e.g., FlexSim) with 5 processes, each equipped with 3-5 machines. Processing times and energy rates are

based on literature. Dynamic events: urgent orders arrive with probability p_{new}; machine breakdowns occur with probability p_{fail}, with repair times uniformly distributed.

4.2 Comparison Methods and Parameter Settings

Comparison methods:

Dispatching Rules: SPT (Shortest Processing Time), EDD (Earliest Due Date);

Single-Agent DQN: no hierarchy, outputs joint actions;

Fixed-Weight Lower Agent: lower uses fixed weights ($\alpha = \beta = 0.5$), upper inactive.

Training parameters: $\gamma = 0.95$, learning rate 10^{-3}, replay buffer 10^5, batch size 64, ϵ decay from 1.0 to 0.1 over 5×10^4 steps.

4.3 Experimental Results Analysis

Three disturbance scenarios are tested: no disturbance, low ($p_{\text{new}} = 0.05, p_{\text{fail}} = 0.02$), and high ($p_{\text{new}} = 0.15, p_{\text{fail}} = 0.05$). Each scenario runs 20 independent trials.

Convergence Analysis: The hierarchical method converges at approximately 2×10^5 steps with stable training, while single-agent DQN shows significant fluctuations, indicating reduced learning difficulty.

Performance Comparison: Table 1 shows results under high disturbance. All values are normalized using the hierarchical method as baseline.

Table 1. Performance comparison under high disturbance scenario.

Method	Makespan	Total Energy
SPT	1.52	1.41
EDD	1.48	1.38
Single-Agent DQN	1.31	1.25
Fixed-Weight Lower	1.28	1.19
Hierarchical Method	1.15	1.13

The hierarchical method reduces makespan by 12.3% and energy by 9.8% compared to single-agent DQN; compared to SPT, reductions are 24.3% and 19.9% respectively. A Wilcoxon signed-rank test confirms statistical significance ($p < 0.01$).

Upper-Level Strategy Analysis: When breakdown rate is high, the upper agent selects efficiency-priority mode; when backlog is low, energy-priority mode is preferred. This demonstrates dynamic adaptability.

Sensitivity Analysis: Varying α, β weights shows the hierarchical method consistently ranks among the top two, confirming robustness to weight selection.

5 Conclusion and Future Work

This paper proposes a hierarchical deep reinforcement learning method for dynamic energy-efficient scheduling in flexible job shops. The two-layer framework decouples

macro-level strategy selection from micro-level execution, effectively addressing the coupling between task allocation and energy control. Simulation experiments demonstrate that the method outperforms traditional dispatching rules and single-agent DQN in both makespan and total energy consumption across multiple disturbance scenarios. Statistical analysis confirms the significance of improvements. The proposed phased training strategy proves effective in stabilizing the learning process, and the upper agent's adaptive mode switching demonstrates interpretable behavior aligned with operational conditions.

Future work will extend this approach in several directions. First, we will incorporate additional disturbance types, such as tool wear, order cancellations, and quality defects, to test the robustness of the hierarchical framework. Second, we plan to investigate graph neural networks to capture inter-operation precedence and machine compatibility as graph-structured features, which may improve state representation and generalization across different job sizes. Third, multi-agent reinforcement learning with communication among machines could be explored to further decentralize decision-making. Fourth, we aim to apply transfer learning to reduce retraining time when migrating the policy to new shop configurations. Finally, real-world data from a printed circuit board assembly line will be used to validate the method in a physical setting, addressing the gap between simulation and practice. We also acknowledge that the current reward design relies on hand-tuned weights; adaptive weight adjustment based on real-time energy prices or carbon intensity would be a valuable enhancement.

References

1. Dauzère-Pérès S, Ding J, Shen L, et al. The flexible job shop scheduling problem: A review[J]. European Journal of Operational Research, 2024, 314(2): 409-432.
2. Yue L, Wang H, Mumtaz J, et al. Energy-efficient scheduling of a two-stage flexible printed circuit board flow shop using a hybrid Pareto spider monkey optimisation algorithm[J]. Journal of Industrial Information Integration, 2023, 31: 100412.
3. Zhou Q, Hu X, Peng S, et al. Scheduling Optimization of Printed Circuit Board Micro-Hole Drilling Production Line Based on Complex Events[J]. Processes, 2023, 11(11): 3073.
4. Wang H, Cheng J, Liu C, et al. Multi-objective reinforcement learning framework for dynamic flexible job shop scheduling problem with uncertain events[J]. Applied Soft Computing, 2022, 131: 109717.
5. Liu R, Piplani R, Toro C. Deep reinforcement learning for dynamic scheduling of a flexible job shop[J]. International Journal of Production Research, 2022, 60(13): 4049-4069.
6. LEI K, GUO P, WANG Y, et al. Large-Scale Dynamic Scheduling for Flexible Job-Shop With Random Arrivals of New Jobs by Hierarchical Reinforcement Learning[J]. IEEE Transactions on Industrial Informatics, 2024, 20(1): 1007-1018.
7. Luo S, Zhang L, Fan Y. Dynamic multi-objective scheduling for flexible job shop by deep reinforcement learning[J]. Computers & Industrial Engineering, 2021, 159: 107489.