LEARNING WITH RESOURCE-CAPACITY REASONING FOR FLEXIBLE JOB-SHOP SCHEDULING

Anonymous authors
Paper under double-blind review

ABSTRACT

Deep Reinforcement Learning (DRL) has emerged as a promising paradigm for the Flexible Job-Shop Scheduling Problem (FJSP) in recent years. However, existing DRL-based schedulers mainly rely on learned state representations to evaluate candidate decisions, with limited explicit reasoning about whether an action preserves sufficient resource capacity to avoid future bottlenecks and complete the remaining operations. In this paper, we introduce Resource-Capacity Reasoning (RCR), an encoder-agnostic framework that incorporates analytical resource-capacity information into DRL-based scheduling without modifying the underlying encoder. RCR constructs a Balanced-Capacity Descriptor (BCD) that characterizes how each candidate assignment changes residual capacity pressure and redistributes bottlenecks, augmenting the corresponding action representation used by the policy network. Based on the same capacity conditions, RCR further introduces a Completion-Loss Certificate (CLC) and incorporates it as an auxiliary penalty while preserving the original makespan-oriented reward. Extensive experiments on synthetic and public benchmarks demonstrate that RCR generally improves the solution quality of representative DRL-based scheduling methods. Moreover, RCR naturally extends to the Job-Shop Scheduling Problem (JSSP), confirming its applicability for different scheduling variants.

1 INTRODUCTION

Scheduling is a fundamental decision-making component in modern manufacturing, where effective allocation and sequencing of production tasks directly influence resource utilization, production efficiency, and system responsiveness (Rossit et al., 2019; Ngwu et al., 2026). Among classical scheduling formulations, the Flexible Job-Shop Scheduling Problem (FJSP) requires that each job follows a predefined sequence of operations while allowing each operation to be assigned to one of several eligible machines (Dauzère-Pérès et al., 2024). The Job-Shop Scheduling Problem (JSSP) can therefore be viewed as a special case of FJSP when machine assignments are fixed (Xiong et al., 2022). FJSP is NP-hard (Xie et al., 2019; Kotary et al., 2022) , making it computationally challenging for exact algorithms such as constraint programming (Da Col & Teppan, 2019; Meng et al., 2020) and for metaheuristic methods such as evolutionary algorithms (Gao et al., 2019).

Deep Reinforcement Learning (DRL) has recently emerged as an effective paradigm for learning constructive scheduling policies (Luo et al., 2021; Lei et al., 2022; Liu et al., 2022). Many studies represent scheduling states as disjunctive graphs (Błażewicz et al., 2000) to explicitly capture precedence constraints, machine conflicts, and relational dependencies under varying resource-availability conditions. By integrating specialized Graph Neural Networks (GNNs) (Wu et al., 2020; Zhou et al., 2020) or Transformer-based architectures (Vaswani et al., 2017), representative DRL-based methods learn expressive representations from structured scheduling states, improving feature extraction and decision quality across complex scheduling environments (Zhang et al., 2020; Song et al., 2023; Wang et al., 2024b; Xiao et al., 2026).

Resource-capacity information is fundamental to scheduling, as effective decisions must preserve sufficient future capacity to ensure completion of the remaining operations (Hu et al., 2026). In classical constraint-based scheduling, capacity reasoning provides a principled mechanism to evaluate whether cumulative processing demand within a time interval can be accommodated by the available resources (Derrien & Petit, 2014; Carlier et al., 2020). Related studies in constraint pro-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

gramming further highlight the value of reasoning about limited resources in scheduling problems (Godet et al., 2020). These insights provide a complementary perspective for DRL-based scheduling, where the capacity implications of scheduling decisions are typically captured only implicitly through learned representations. Representative DRL-based FJSP schedulers mainly focus on state representations, relational modeling, or policy optimization (Song et al., 2023; Wang et al., 2024b; Xiao et al., 2026), without directly characterizing how candidate assignments affect future job- and resource-level capacity and, ultimately, schedule completion. Such consequences must therefore be inferred from learned representations and reward signals.

To address this limitation, we propose Resource-Capacity Reasoning (RCR), an encoder-agnostic framework that introduces analytical resource-capacity information into DRL-based scheduling while leaving the underlying encoder unchanged. RCR constructs a Balanced-Capacity Descriptor (BCD) that characterizes how each candidate operation–machine assignment changes capacity pressure and redistributes scheduling bottlenecks, augmenting the corresponding action representation used by the policy network. Based on the same capacity conditions, RCR further introduces a Completion-Loss Certificate (CLC), which quantifies the capacity-induced completion loss and is incorporated as an auxiliary penalty while preserving the original makespan-oriented reward. By deriving BCD and CLC from a shared resource-capacity foundation, RCR provides consistent analytical guidance for both action evaluation and policy learning. Experimental results demonstrate that RCR improves the solution quality of representative DRL-based scheduling methods and generalizes effectively across FJSP and JSSP settings. Our main contributions are summarized as follows:

- We introduce RCR, an encoder-agnostic framework that incorporates resource-capacity information into DRL-based scheduling without modifying the underlying encoder and provides a unified analytical foundation for reasoning about candidate decisions and their downstream effects on schedule completion.
- We develop two complementary mechanisms from the same capacity conditions: BCD augments candidate-action representations with balanced capacity-pressure information, while CLC introduces a completion-loss-based auxiliary reward signal to guide policy learning without replacing the original makespan-oriented reward.
- We conduct extensive experiments with representative DRL-based scheduling methods on synthetic and public FJSP benchmarks. RCR generally improves solution quality, while JSSP results further demonstrate its applicability across scheduling variants.

2 Related Work

2.1 DRL-based Scheduling

Deep Reinforcement Learning (DRL) has emerged as a promising paradigm for solving FJSP (Du et al., 2022; Zhao et al., 2023; 2025) and JSSP (Zhang et al., 2020; Liu & Huang, 2023). Representative studies formulate scheduling states using disjunctive graphs (Brandimarte, 1993; Błażewicz et al., 2000), where operations and machines are represented as nodes and their structural relations as edges. Graph Neural Networks (GNNs) (Wu et al., 2020; Smit et al., 2025) and Transformer-based architectures (Vaswani et al., 2017; Tay et al., 2022) are widely employed to capture rich structural dependencies. For FJSP, Song et al. (2023) designed heterogeneous GNNs to model relational dependencies, while Wang et al. (2024b) introduced dual-attention networks. More recently, Xiao et al. (2026) adopted Transformer blocks with dot-product attention for both FJSP and JSSP. For policy optimization, algorithms such as Proximal Policy Optimization (PPO) (Schulman et al., 2017; Meng et al., 2023) and REINFORCE (Williams, 1992) are commonly used to optimize scheduling policies through cumulative rewards, forming an end-to-end learning framework (Wang et al., 2024a).

2.2 Capacity Reasoning in Scheduling

Capacity reasoning has long been studied in cumulative and resource-constrained scheduling. Energetic reasoning assesses whether processing demand within a time interval can be accommodated by available capacity, enabling feasibility detection and constraint propagation. Derrien & Petit (2014) identified relevant intervals for efficient energetic reasoning, while Carlier et al. (2020) developed an $O(n^2)$ time-bound adjustment algorithm. Related edge-finding methods further exploit resource

interactions to tighten activity bounds (Mercier & Van Hentenryck, 2008). Capacity-based reasoning has also been extended to RCPSP lower bounds (Haouari et al., 2012), resource-constrained machine scheduling (Godet et al., 2020), and integrated production settings (Hu et al., 2026). These studies establish capacity reasoning as an effective analytical tool, whereas such explicit reasoning remains relatively underexplored in DRL-based scheduling.

3 PRELIMINARY

3.1 FLEXIBLE JOB-SHOP SCHEDULING PROBLEM

The Flexible Job-Shop Scheduling Problem (FJSP) consists of a set of jobs $\mathcal{J} = \{J_1, \ldots, J_n\}$ and a set of machines $\mathcal{M} = \{M_1, \ldots, M_m\}$. Each job J_i contains an ordered sequence of operations $\{O_{i,1}, \ldots, O_{i,n_i}\}$. Operation $O_{i,j}$ can be processed by any machine M_k in its eligible machine set $\mathcal{M}_{i,j} \subseteq \mathcal{M}$, with processing time $p_{i,j,k} > 0$. A feasible schedule must satisfy three basic constraints: (1) each operation can start only after its predecessor in the same job has been completed; (2) each operation is assigned to exactly one eligible machine and processed without preemption; and (3) each machine can process at most one operation at any time. Let $C_{i,j}$ denote the completion time of $O_{i,j}$. The objective is to minimize the makespan $C_{\max} = \max_{i,j} C_{i,j}$, while satisfying all precedence, assignment, and machine-capacity constraints. The Job-Shop Scheduling Problem (JSSP) can be regarded as a special case of FJSP in which each operation is associated with a single predetermined machine, i.e., $|\mathcal{M}_{i,j}| = 1$. Therefore, JSSP removes machine-selection flexibility while retaining the same constraints and optimization objective.

3.2 RESOURCE-CAPACITY CONDITIONS

Given a partial schedule s_t at decision step t, the remaining operations must be completed using the residual processing capacity available before a reference horizon H, obtained from a reference schedule. Let $\mathcal{O}_t^r$ denote the set of remaining operations. For a resource set $\mathcal{G} \subseteq \mathcal{M}$ and a time interval $[l, u] \subseteq [0, H]$, let $D_{\mathcal{G}}^t(l, u)$ denote the processing demand that must be executed by machines in $\mathcal{G}$ within $[l, u]$, and let $C_{\mathcal{G}}^t(l, u)$ denote the corresponding available capacity. A necessary resource-capacity condition for completion within H is given by:

$$D_{\mathcal{G}}^t(l, u) \leq C_{\mathcal{G}}^t(l, u). \tag{1}$$

A violation of this condition indicates insufficient residual capacity over the considered resource-time interval under the current partial schedule. These conditions provide the analytical basis for the resource-capacity information used in our method.

4 METHODOLOGY

Resource-Capacity Reasoning (RCR) incorporates analytical resource-capacity information into DRL-based scheduling without modifying the encoder. Given a partial schedule, RCR evaluates residual completion requirements against a reference horizon H using job-path and resource-capacity conditions. Appendix G details the construction of H. These conditions inform two complementary RCR components. The Balanced-Capacity Descriptor (BCD) characterizes how each candidate operation–machine assignment changes capacity pressure and redistributes bottlenecks, augmenting the action representation. The Completion-Loss Certificate (CLC) measures violations of necessary completion conditions in successor states and adds an auxiliary penalty to the original makespan-oriented reward. Together, BCD and CLC provide analytical guidance for action evaluation and policy learning. The framework is illustrated in Figure 1.

4.1 BALANCED-CAPACITY DESCRIPTOR

The resource-capacity conditions introduced in Section 3.2 characterize whether residual processing requirements can be accommodated within a reference horizon H. However, they do not directly reveal how candidate assignments affect future capacity pressure. To provide action-specific information to the scheduling policy, we develop a Balanced-Capacity Descriptor (BCD). BCD evaluates

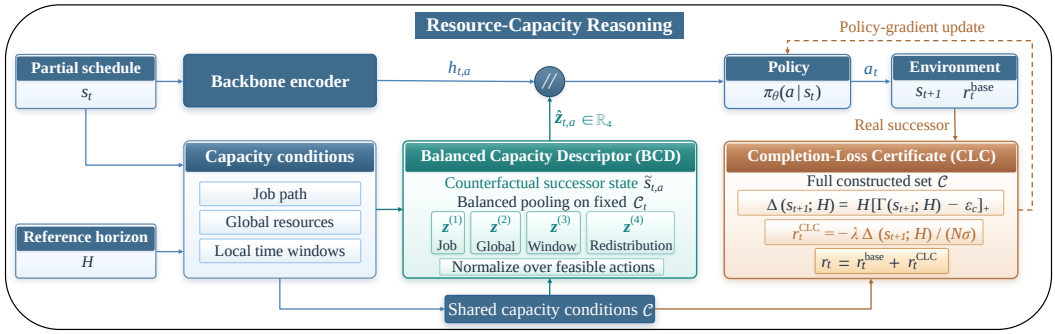

Figure 1: Overall framework of our RCR.

each feasible operation–machine assignment through counterfactual capacity analysis and summarizes its effect in four complementary dimensions. Importantly, BCD operates only on candidate-action representations without modifying the underlying state encoder.

Capacity-pressure construction For each remaining operation $O_q \in \mathcal{O}_t^r$, let $\underline{p}^q$ denote its minimum processing time over eligible machines. For job J_i, let R_i^t denote its current ready time and let W_i^t denote the minimum remaining processing requirement. We define the job-path pressure as:

$$W_i^t = \sum_{O_q \in \mathcal{O}_t^r \cap J_i} \underline{p}^q, \quad x_i^J(s_t; H) = \frac{R_i^t + W_i^t - H}{H}. \tag{2}$$

A positive value indicates that the job's earliest possible completion exceeds H. We form resource sets $\mathcal{G}$ from individual machines, the complete machine set, and selected eligibility-induced groups ranked by restricted workload per machine. We partition $[0, H]$ into B equal bins and use pairs of bin endpoints to define local intervals; $[0, H]$ serves as the global interval. For $O_q \in J_i$, let E_q^t and L_q^t denote its earliest start and latest completion under the current schedule and horizon H. We obtain E_q^t by adding the minimum processing times of unscheduled predecessors to R_i^t, and L_q^t by subtracting those of unscheduled successors from H. Following energetic reasoning (Carlier et al., 2020), the minimum processing amount of O_q required within $[l, u] \subseteq [0, H]$ is:

$$\omega_q^t(l, u) = \left[\min\left\{\underline{p}^q,\, u - l,\, E_q^t + \underline{p}^q - l,\, u - L_q^t + \underline{p}^q\right\}\right]_+, \tag{3}$$

where $[x]_+ = \max(x, 0)$. An operation contributes to resource set $\mathcal{G}$ when all its eligible machines belong to $\mathcal{G}$, since processing demand must then be accommodated by the set. Accordingly, the mandatory demand and available capacity introduced in the preliminary section are instantiated as:

$$D_{\mathcal{G}}^t(l, u) = \sum_{\substack{O_q \in \mathcal{O}_t^r \\ \mathcal{M}_q \subseteq \mathcal{G}}} \omega_q^t(l, u), \quad C_{\mathcal{G}}^t(l, u) = \sum_{M_k \in \mathcal{G}} \left[u - \max\{l, R_k^t\}\right]_+, \tag{4}$$

where R_k^t is the current ready time of machine M_k. Under the scheduling transition, previously scheduled operations and their processing intervals remain fixed, and any operation subsequently assigned to M_k starts no earlier than R_k^t. We distinguish global resource pressure over the complete horizon from local pressure over shorter intervals. Let $N_{\mathcal{G}}^t$ denote the number of remaining operations whose eligible-machine sets are contained in $\mathcal{G}$. Their normalized pressures are

$$x_{\mathcal{G}}^G(s_t; H) = \frac{D_{\mathcal{G}}^t(0, H) - C_{\mathcal{G}}^t(0, H)}{H|\mathcal{G}|}, \quad x_{\mathcal{G},[l,u]}^W(s_t; H) = \frac{D_{\mathcal{G}}^t(l, u) - C_{\mathcal{G}}^t(l, u)}{H \max\{N_{\mathcal{G}}^t, 1\}}. \tag{5}$$

Together, x^J, x^G, and x^W provide complementary views of job-path, global resource, and local time-window pressure. Positive pressures indicate violations of the corresponding necessary completion conditions, whereas negative values reflect residual slack.

Active constraints and balanced pooling The number of local time-window constraints can substantially exceed that of job or global resource constraints. A single softmax over all constraints would therefore bias the representation toward the largest family. We instead retain only active constraints. Specifically, completed jobs and resource sets without remaining restricted demand are excluded. For local time-window constraints, we define the earliest feasible action-start time as:

$$\phi_t = \min_{a=(O_q,M_k)\in\mathcal{A}_t} \max\left\{R^t_{i(q)}, R^t_k\right\}, \tag{6}$$

where $R^t_{i(q)}$ and R^t_k denote the ready times of the job containing O_q and machine M_k, respectively. Local windows whose right endpoints satisfy $u \le \phi_t$ are regarded as inactive and excluded. Let

$$\mathcal{C}_t = \mathcal{C}^J_t \cup \mathcal{C}^G_t \cup \mathcal{C}^W_t, \quad K_t = \sum_{k\in\{J,G,W\}} \mathbf{1}\left[|\mathcal{C}^k_t| > 0\right] \tag{7}$$

denote the active constraints and the number of nonempty families. For an active constraint $c \in \mathcal{C}^k_t$, we assign the balanced pressure weight as:

$$w^t_c = \frac{1}{K_t} \frac{\exp(x_c(s_t;H)/\tau)}{\sum_{d\in\mathcal{C}^k_t}\exp(x_d(s_t;H)/\tau)}, \quad c \in \mathcal{C}^k_t, \tag{8}$$

where $\tau > 0$ controls the concentration on high-pressure constraints. Thus, constraints compete only within the same family, while each active family receives equal total weight.

Proposition 1 (Cardinality-Invariant Weighting) *Under the balanced family weighting scheme, let K_t denote the number of nonempty active constraint families. Then, for any nonempty family $\mathcal{C}^k_t$, the aggregate weight assigned to that family is:*

$$\sum_{c\in\mathcal{C}^k_t} w^t_c = \frac{1}{K_t}, \qquad \sum_{c\in\mathcal{C}_t} w^t_c = 1. \tag{9}$$

Consequently, the aggregate weight allocated to each active family is invariant to its cardinality, preventing families with substantially more constraints from receiving disproportionate aggregate weight. The proof is provided in Appendix A.

Counterfactual-capacity descriptor For each feasible action $a \in \mathcal{A}_t$, we construct a counterfactual successor $\widetilde{s}_{t,a}$ without modifying the environment. To compare actions consistently, we hold the active constraint set $\mathcal{C}_t$ identified at s_t fixed across all successors. For $c \in \mathcal{C}_t$, let $\widetilde{x}_{c,a} = x_c(\widetilde{s}_{t,a}; H)$ and $\widetilde{w}_{c,a}$ denote the successor pressure and balanced weight. The vectors $\mathbf{w}^t = (w^t_c)_{c\in\mathcal{C}_t}$ and $\widetilde{\mathbf{w}}_{t,a} = (\widetilde{w}_{c,a})_{c\in\mathcal{C}_t}$ are the parent and counterfactual weight distributions, respectively. We define their divergence as $\delta^{\mathrm{KL}}_{t,a} = D_{\mathrm{KL}}(\mathbf{w}^t \| \widetilde{\mathbf{w}}_{t,a})$. The raw four-dimensional descriptor is defined as:

$$\mathbf{z}_{t,a} = \left[\sum_{c\in\mathcal{C}^J_t} w^t_c(\widetilde{x}_{c,a} - x^t_c), \quad \sum_{c\in\mathcal{C}^G_t} w^t_c(\widetilde{x}_{c,a} - x^t_c), \quad \sum_{c\in\mathcal{C}^W_t} w^t_c(\widetilde{x}_{c,a} - x^t_c), \quad \tau\delta^{\mathrm{KL}}_{t,a}\right]^\top, \tag{10}$$

where $x^t_c = x_c(s_t; H)$. We write $z^{(d)}_{t,a}$ for the d-th component of $\mathbf{z}_{t,a}$, with $d \in \{1,2,3,4\}$. Specifically, $d = 1$, 2, and 3 correspond to action-induced changes in job-path, global resource, and local time-window pressure, respectively, whereas $d = 4$ measures bottleneck redistribution through the divergence between the parent and counterfactual pressure distributions. Because the four components can exhibit different scales across instances and decision steps, each component is normalized over the currently feasible action set. For each dimension $d \in \{1,2,3,4\}$, we define:

$$\begin{aligned} \mu_{t,d} &= \frac{1}{|\mathcal{A}_t|}\sum_{a\in\mathcal{A}_t} z^{(d)}_{t,a}, \quad \sigma_{t,d} = \sqrt{\frac{1}{|\mathcal{A}_t|}\sum_{a\in\mathcal{A}_t}\left(z^{(d)}_{t,a} - \mu_{t,d}\right)^2}, \\ \widehat{z}^{(d)}_{t,a} &= \mathrm{clip}\left(\frac{z^{(d)}_{t,a} - \mu_{t,d}}{\max\{\sigma_{t,d}, \epsilon_z\}}, -B_z, B_z\right), \quad \widehat{\mathbf{z}}_{t,a} = \left[\widehat{z}^{(1)}_{t,a}, \widehat{z}^{(2)}_{t,a}, \widehat{z}^{(3)}_{t,a}, \widehat{z}^{(4)}_{t,a}\right]^\top \in \mathbb{R}^4. \end{aligned} \tag{11}$$

where $\epsilon_z > 0$ prevents numerical instability and $B_z > 0$ denotes the clipping bound. This state-wise normalization standardizes the four analytical components before their integration with the learned

action features while retaining action-specific variation. Finally, let $\mathbf{h}_{t,a}$ denote the candidate-action representation produced by the backbone. BCD augments only the policy-side representation:

$$\overline{\mathbf{h}}_{t,a} = [\mathbf{h}_{t,a} \| \widehat{\mathbf{z}}_{t,a}], \quad \pi_\theta(a|s_t) = \mathrm{softmax}_{a \in \mathcal{A}_t} \left(f_\theta(\overline{\mathbf{h}}_{t,a}) \right), \tag{12}$$

where f_θ denotes the policy scoring network. By concatenating $\widehat{\mathbf{z}}_{t,a}$ with $\mathbf{h}_{t,a}$, BCD provides explicit action-level resource-capacity information to the policy network without modifying the underlying encoder or depending on a specific policy-gradient objective.

4.2 Completion-Loss Certificate

Although BCD provides action-level resource-capacity information to the policy network, policy learning still relies on the backbone reward to assess executed decisions. Motivated by necessary-condition reasoning in cumulative scheduling (Baptiste et al., 1999), we derive a Completion-Loss Certificate (CLC) from the same resource-capacity conditions to provide an explicit completion-oriented learning signal. Unlike the active constraint subset used for BCD pooling, CLC evaluates the full set of constructed job-path, global resource, and local time-window conditions:

$$\mathcal{C} = \mathcal{C}^J \cup \mathcal{C}^G \cup \mathcal{C}^W, \qquad \mathcal{C}_t \subseteq \mathcal{C}. \tag{13}$$

where $\mathcal{C}$ denotes the full set of constructed valid capacity constraints. This distinction ensures that active filtering affects only the action representation without weakening the completion certificate.

Completion-loss construction For a partial schedule s_t, we reuse the normalized pressures defined in Section 4.1 and identify the most violated necessary completion condition as:

$$\Gamma(s_t; H) = \max_{c \in \mathcal{C}} x_c(s_t; H), \tag{14}$$

where $x_c(s_t; H)$ denotes the corresponding job-path, global resource, or local time-window pressure. Since $x_c(s_t; H) > 0$ indicates a violation of the associated necessary condition, $\Gamma(s_t; H)$ summarizes the strongest completion-critical capacity violation. We define the CLC as:

$$\Delta(s_t; H) = H \left[\Gamma(s_t; H) - \epsilon_c \right]_+, \tag{15}$$

where $\epsilon_c \geq 0$ is a small numerical tolerance. Multiplication by H restores the certificate to the scheduling-time scale after pressure normalization. Thus, $\Delta(s_t; H)$ quantifies the magnitude of the strongest certified completion-condition violation on the scheduling-time scale, rather than the actual makespan degradation.

Proposition 2 (Soundness of the Completion-Loss Certificate) *Let s_t be a partial schedule reached under the scheduling transition considered in this work, and let $H > 0$ be a fixed reference horizon. Let $\mathcal{F}_t(H)$ denote the set of feasible completions that preserve the operations already scheduled in s_t, follow the same transition, and have makespan at most H. Then*

$$\Delta(s_t; H) > 0 \quad \Longrightarrow \quad \exists\, c \in \mathcal{C} : x_c(s_t; H) > 0 \quad \Longrightarrow \quad \mathcal{F}_t(H) = \varnothing. \tag{16}$$

The converse does not necessarily hold: $\Delta(s_t; H) = 0$ only indicates that no tested capacity pressure exceeds the tolerance ϵ_c; it does not guarantee that $\mathcal{F}_t(H) \neq \varnothing$. A proof is provided in Appendix B.

Completion-oriented auxiliary learning signal CLC is evaluated on the real successor state s_{t+1} after executing action a_t, rather than on the counterfactual states used by BCD. Let N denote the number of operations, $\lambda \geq 0$ the penalty coefficient, and $\sigma > 0$ the reward scale used by the backbone. This auxiliary signal follows the general motivation of reward shaping (Ng et al., 1999) by providing additional guidance during policy learning. We define the CLC auxiliary signal as:

$$r_t^{\mathrm{CLC}} = -\frac{\lambda}{N\sigma} \Delta(s_{t+1}; H). \tag{17}$$

The normalization by N reduces sensitivity to problem size, while σ aligns the certificate penalty with the numerical scale of the backbone reward. RCR retains the original reward term of the backbone and adds a CLC-based auxiliary penalty:

$$r_t = r_t^{\mathrm{base}} + r_t^{\mathrm{CLC}}, \tag{18}$$

where r_t^{base} denotes the original reward under the same reward scaling convention. RCR retains the original backbone reward and adds a CLC-based auxiliary penalty for certified completion-condition violations. Together with BCD, it provides resource-capacity guidance at the action-evaluation and policy-learning levels without relying on a specific policy-gradient algorithm.

4.3 Training Algorithm

At each step t, RCR evaluates the capacity conditions of the state s_t and computes the normalized BCD for each feasible action $a \in \mathcal{A}t$. The BCD augments the backbone representations, enabling the policy $\pi\theta$ to select an operation–machine assignment. After executing the action, the environment transitions to s_{t+1}, where CLC evaluates violations of the necessary completion conditions. The resulting auxiliary penalty is combined with the original makespan-oriented reward to update the scheduling policy using the backbone's policy-gradient algorithm, such as PPO (Schulman et al., 2017) and REINFORCE (Williams, 1992). The reference horizon H and the constructed capacity constraints remain fixed throughout each episode. The training procedure is provided in Appendix C.

5 Experimental Results

5.1 Experimental Setup

Datasets We evaluate RCR on synthetic and public benchmarks for solution quality and generalization across sizes and distributions. Synthetic FJSP datasets comprise six configurations: 10×5, 20×5, 15×10, 20×10, 30×10, and 40×10. Models are trained separately on the four smaller sizes, while the model trained on 20×10 is evaluated on the two larger sizes without retraining. We consider two synthetic distributions: SD1 (Song et al., 2023) varies the number of operations per job, while SD2 (Wang et al., 2024b) uses an equal number of operations per job with a wider range of processing times. Public FJSP benchmarks include the Brandimarte (Brandimarte, 1993) and the Hurink Rdata, Edata, and Vdata groups (Hurink et al., 1994). We further evaluate RCR on the Taillard (Taillard, 1993) and DMU (Demirkol et al., 1998) JSSP benchmarks to assess its applicability across scheduling variants. Dataset details are provided in Appendix D.1.

Baselines and performance metrics We evaluate RCR using representative DRL-based scheduling backbones. For FJSP, we consider HGNN (Song et al., 2023), DANIEL (Wang et al., 2024b), and RESCHED (Xiao et al., 2026), while RESCHED also serves as the backbone for JSSP. RESCHED employs a Transformer-based architecture trained with REINFORCE, whereas HGNN and DANIEL use GNN-based architectures trained with PPO. The corresponding RCR-enhanced variants are RCR-HGNN, RCR-DANIEL, and RCR-RESCHED. We include the OR-Tools CP-SAT solver (Da Col & Teppan, 2019) as an optimization baseline. Following Xiao et al. (2026), we report the mean instance-wise relative gap to account for differences in instance scale. For synthetic instances, reference solutions are obtained using OR-Tools with a time limit of 1,800 s per instance. For public benchmarks, we adopt the upper bounds (UBs) from Behnke & Geiger (2012). Further details are provided in Appendix D.2.

Settings RCR introduces four additional hyperparameters: the BCD pooling temperature $\tau = 0.1$, the CLC penalty coefficient $\lambda = 0.3$, the number of time bins $B = 8$, and the maximum number of additional resource groups $\mathcal{G}_{\max} = 8$. We hold these settings fixed in the main comparisons while retaining each backbone's original configuration (Appendix D.3). We examine the sensitivity of RCR to these settings. We evaluate methods using greedy and sampling decoding. Greedy selects the highest-probability action, while sampling generates 100 samples and reports the minimum makespan. All experiments are conducted on a workstation equipped with a single NVIDIA RTX A10 GPU. Our source code is available at `https://anonymous.4open.science/r/RCR`.

5.2 Results on FJSP

Results on synthetic datasets Table 1 presents the results on the SD1 and SD2 datasets under greedy and sampling decoding. Models are trained on four instance sizes: 10×5, 20×5, 15×10, and 20×10. To evaluate cross-scale generalization, models trained on 20×10 instances are tested on 30×10 and 40×10 instances. RCR improves scheduling performance in **70/72** paired comparisons against the corresponding original backbones. Notably, RCR-HGNN substantially improves HGNN on SD2, enabling it to approach the performance of two recent competitive scheduling methods. RCR-RESCHED also achieves clear improvements on SD2, indicating that RCR remains beneficial even when integrated with a strong recent scheduler. Overall, these results demonstrate the effectiveness of RCR in improving scheduling quality and cross-scale generalization.

Table 1: Results on synthetic datasets. Bold highlights the better result in each pair.

	Methods		SD1						SD2					
			10 × 5	20 × 5	15 × 10	20 × 10	30 × 10	40 × 10	10 × 5	20 × 5	15 × 10	20 × 10	30 × 10	40 × 10
	OR-Tools	Ref.	96.32	188.15	143.53	195.98	274.67	365.96	326.24	602.04	377.17	464.16	692.26	998.39
Greedy	HGNN	Gap (%)	16.03	12.27	16.33	10.15	14.01	13.75	71.42	76.79	115.26	126.12	123.57	108.12
Greedy	RCR-HGNN	Gap (%)	**11.89**	**10.22**	**11.33**	**9.03**	**13.18**	**12.78**	**27.22**	**35.52**	**56.78**	**32.32**	**12.08**	**12.49**
Greedy	DANIEL	Gap (%)	10.87	5.03	12.42	1.31	2.50	1.52	25.68	**11.52**	57.16	31.58	11.95	-1.67
Greedy	RCR-DANIEL	Gap (%)	**9.76**	**4.13**	**11.47**	**0.18**	**1.96**	**0.74**	**24.85**	11.71	**55.53**	**30.10**	**10.22**	**-3.28**
Greedy	RESCHED	Gap (%)	12.25	4.63	6.51	0.48	2.69	1.64	16.36	9.87	18.14	14.18	6.30	-4.58
Greedy	RCR-RESCHED	Gap (%)	**9.19**	**3.02**	**5.10**	**-0.95**	**1.47**	**0.82**	**11.69**	**5.43**	**15.39**	**10.82**	**2.43**	**-7.55**
Sampling	HGNN	Gap (%)	9.66	10.31	12.13	9.64	13.49	13.49	49.71	60.70	101.52	114.15	111.51	99.26
Sampling	RCR-HGNN	Gap (%)	**5.42**	**8.88**	**8.83**	**8.57**	**12.37**	**12.05**	**11.93**	**19.11**	**40.88**	**24.35**	**11.22**	**11.06**
Sampling	DANIEL	Gap (%)	5.57	2.46	6.79	-1.03	1.67	1.14	12.57	**4.66**	38.70	19.13	4.80	-6.60
Sampling	RCR-DANIEL	Gap (%)	**4.08**	**1.58**	**6.03**	**-1.70**	**1.03**	**0.39**	**11.35**	4.81	**36.76**	**18.14**	**3.99**	**-7.54**
Sampling	RESCHED	Gap (%)	5.98	2.33	3.09	-1.55	1.51	1.10	6.39	3.68	9.81	7.90	1.40	-7.61
Sampling	RCR-RESCHED	Gap (%)	**4.38**	**1.51**	**2.24**	**-2.45**	**0.69**	**0.33**	**5.01**	**1.64**	**8.32**	**6.02**	**-0.38**	**-9.22**

Table 2: Results on public benchmarks.

	Methods		Brandimarte	Edata	Rdata	Vdata
	UB	Ref.	172.50	1028.88	934.28	919.50
	OR-Tools	Gap (%)	1.50	-0.03	0.11	-0.01
Greedy	HGNN	Gap (%)	28.52	15.53	11.15	4.25
Greedy	RCR-HGNN	Gap (%)	**11.77**	**15.51**	**10.09**	**3.19**
Greedy	DANIEL	Gap (%)	**13.58**	16.33	11.42	3.28
Greedy	RCR-DANIEL	Gap (%)	14.82	**15.65**	**11.25**	**3.01**
Greedy	RESCHED	Gap (%)	**9.08**	15.48	10.18	3.48
Greedy	RCR-RESCHED	Gap (%)	10.81	**13.63**	**7.96**	**2.46**
Sampling	HGNN	Gap (%)	18.56	8.17	5.57	1.32
Sampling	RCR-HGNN	Gap (%)	**8.85**	**8.09**	**5.50**	**0.75**
Sampling	DANIEL	Gap (%)	9.53	9.08	**4.95**	0.69
Sampling	RCR-DANIEL	Gap (%)	**8.51**	**8.58**	5.01	**0.51**
Sampling	RESCHED	Gap (%)	6.61	8.13	5.04	0.82
Sampling	RCR-RESCHED	Gap (%)	**6.31**	**7.53**	**3.97**	**0.63**

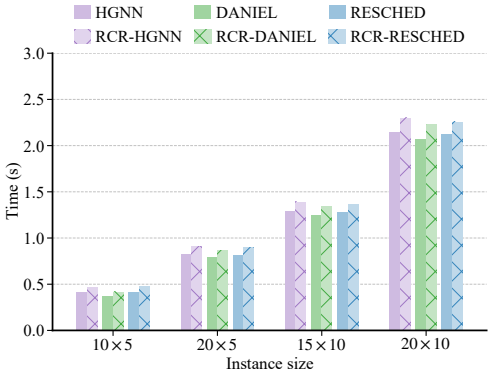

Figure 2: Inference time of all methods.

Results on public benchmarks. Table 2 reports the results on the Brandimarte and Hurink benchmarks under greedy and sampling decoding. RCR improves scheduling performance in **21/24** paired comparisons with the corresponding original backbones, demonstrating its effectiveness across established benchmark instances. These improvements indicate that incorporating resource-capacity information can enhance scheduling decisions beyond those derived from learned state representations alone. General gains in backbones, decoding strategies, and instance distributions demonstrate the general applicability of our framework to improve scheduling decisions.

5.3 Analyses

Inference efficiency As shown in Fig. 2, we evaluate the inference time of all methods on the SD1 dataset using greedy decoding, reporting the average over five independent runs per method. RCR introduces a modest computational overhead due to additional resource-capacity reasoning, while improving scheduling quality. This trade-off indicates that the additional computation provides useful decision-making information without substantially increasing inference time.

Multiple random seeds To evaluate the robustness of RCR under stochastic training variations, we conduct experiments with multiple random seeds in Appendix E. The general performance trends across independent runs and representative benchmark settings indicate that the observed improvements are not tied to a particular initialization or training trajectory. These results provide further evidence for the reproducibility and stability of RCR, demonstrating that its effectiveness is preserved under different stochastic training conditions and across repeated training runs.

Table 3: Ablation results for RCR-RESCHED.

Variants		10×5	20×5	15×10	20×10
RCR-RESCHED	Gap (%)	**9.19**	**3.02**	**5.10**	**-0.95**
w/o BCD	Gap (%)	10.83	4.32	6.51	0.44
w/o CIW	Gap (%)	10.42	3.97	5.80	-0.39
w/o CLC	Gap (%)	9.94	4.26	6.11	0.38
RESCHED	Gap (%)	12.25	4.63	6.51	0.48

Table 4: Sensitivity results of τ.

Variants		10×5	20×5	15×10	20×10
$\tau = 0.1$	Gap (%)	**9.19**	**3.02**	**5.10**	**-0.95**
$\tau = 0.2$	Gap (%)	9.67	3.90	5.45	-0.04
$\tau = 0.3$	Gap (%)	9.88	4.02	6.04	-0.01
$\tau = 0.4$	Gap (%)	10.21	4.78	6.67	0.31
$\tau = 0.5$	Gap (%)	10.56	4.50	6.48	0.50

Table 5: Results on Taillard Benchmark for JSSP. UB is the best-known upper bound.

Methods		15×15	20×15	20×20	30×15	30×20	50×15	50×20	100×20
UB	Ref.	1233.9	1361.3	1617.1	1771.2	1919.4	2783.8	2834.4	5369.6
OR-Tools	Gap (%)	0.1	0.2	0.7	2.1	2.8	0.0	2.8	3.9
RESCHED	Gap (%)	15.7	19.7	16.3	**21.5**	22.5	16.1	15.6	9.6
RCR-RESCHED	Gap (%)	**14.9**	**17.6**	**16.0**	21.7	**22.2**	**15.3**	**15.1**	**9.5**

Ablation study Table 3 presents ablation results for RCR-RESCHED on SD1 under greedy decoding. We independently remove the Balanced Capacity Descriptor (w/o BCD), Cardinality-Invariant Weighting (w/o CIW), and Completion-Loss Certificate (w/o CLC), using the original backbone as a reference. Removing any component degrades scheduling performance relative to the complete RCR framework, supporting the complementary contributions of all three components. Appendix F.1 provides additional fine-grained ablations of local time windows, resource groups, and bottleneck redistribution, alongside ablation results on other backbones.

Sensitivity study Table 4 examines the sensitivity of RCR-RESCHED to the BCD pooling temperature τ on SD1 using the greedy strategy, with $\tau = 0.1$ achieving the best performance. The temperature controls the emphasis of family-balanced weighting on high-pressure constraints. Smaller τ prioritizes critical constraints but may overlook broader capacity patterns, while larger τ distributes weight more uniformly, potentially diluting bottleneck information. The results suggest that $\tau = 0.1$ balances bottleneck emphasis with comprehensive capacity assessment. Sensitivity analyses of the remaining hyperparameters and other RCR-enhanced variants are provided in Appendix F.2.

5.4 Results on JSSP

Table 5 presents results on the Taillard JSSP benchmarks (Taillard, 1993). RESCHED (Xiao et al., 2026) and its RCR-enhanced variant, RCR-RESCHED, are trained on 10×10 instances and directly evaluated on larger instances ranging from 15×15 to 100×20 under greedy decoding. RCR improves RESCHED on **7/8** evaluated instance sizes, supporting its cross-scale generalization beyond the training distribution. These results suggest that resource-capacity reasoning provides transferable scheduling guidance across different problem scales without scale-specific retraining or adaptation, even under substantial increases in problem size. Together with the FJSP results, these findings support the applicability of RCR across scheduling variants. Additional results on the DMU benchmarks (Demirkol et al., 1998) are provided in Appendix H.

6 Conclusion

This work introduces Resource Capacity Reasoning (RCR), a framework that incorporates analytical resource-capacity information into DRL-based scheduling without modifying backbone encoders. RCR consists of two complementary components: the Balanced Capacity Descriptor (BCD), which provides action-level capacity guidance through counterfactual analysis, and the Completion-Loss Certificate (CLC), which provides completion-oriented reward guidance based on necessary capacity conditions. Extensive experiments on FJSP and JSSP benchmarks show that RCR generally improves multiple DRL backbones across scales, distributions, and decoding strategies. These results highlight the effectiveness of integrating analytical knowledge into DRL-based scheduling.

AI USE STATEMENT

In this work, we used generative AI tools solely for polishing the manuscript to improve grammar, clarity, and overall presentation readability. We have not used generative AI tools for any other tasks with required and recommended disclosure. All AI-assisted content has been carefully reviewed and verified by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We emphasize reproducibility throughout this work. Detailed experimental configurations, implementation settings, and evaluation protocols are provided in Section 5.1 and Appendix D. Anonymous source code is released with the submission to facilitate independent verification and reproduction. Together, these resources provide sufficient information to reproduce the proposed method, experimental results, and comparisons reported in this paper.

REFERENCES

Ph Baptiste, Claude Le Pape, and Wim Nuijten. Satisfiability tests and time-bound adjustmentsfor cumulative scheduling problems. *Annals of Operations research*, 92(0):305–333, 1999.

Dennis Behnke and Martin Josef Geiger. Test instances for the flexible job shop scheduling problem with work centers. 2012.

Jacek Błażewicz, Erwin Pesch, and Małgorzata Sterna. The disjunctive graph machine representation of the job shop scheduling problem. *European Journal of Operational Research*, 127(2): 317–331, 2000.

Paolo Brandimarte. Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations research*, 41(3):157–183, 1993.

Jacques Carlier, Eric Pinson, Abderrahim Sahli, and Antoine Jouglet. An o (n2) algorithm for time-bound adjustments for the cumulative scheduling problem. *European Journal of Operational Research*, 286(2):468–476, 2020.

Giacomo Da Col and Erich C Teppan. Industrial size job shop scheduling tackled by present day cp solvers. In *International Conference on Principles and Practice of Constraint Programming*, pp. 144–160. Springer, 2019.

Stéphane Dauzère-Pérès, Junwen Ding, Liji Shen, and Karim Tamssaouet. The flexible job shop scheduling problem: A review. *European Journal of Operational Research*, 314(2):409–432, 2024.

Ebru Demirkol, Sanjay Mehta, and Reha Uzsoy. Benchmarks for shop scheduling problems. *European Journal of Operational Research*, 109(1):137–141, 1998.

Alban Derrien and Thierry Petit. A new characterization of relevant intervals for energetic reasoning. In *International conference on principles and practice of constraint programming*, pp. 289–297. Springer, 2014.

Yu Du, Junqing Li, Chengdong Li, and Peiyong Duan. A reinforcement learning approach for flexible job shop scheduling problem with crane transportation and setup times. *IEEE Transactions on Neural Networks and Learning Systems*, 35(4):5695–5709, 2022.

Kaizhou Gao, Zhiguang Cao, Le Zhang, Zhenghua Chen, Yuyan Han, and Quanke Pan. A review on swarm intelligence and evolutionary algorithms for solving flexible job shop scheduling problems. *IEEE/CAA Journal of Automatica Sinica*, 6(4):904–916, 2019.

Arthur Godet, Xavier Lorca, Emmanuel Hebrard, and Gilles Simonin. Using approximation within constraint programming to solve the parallel machine scheduling problem with additional unit resources. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 1512–1519, 2020.

Mohamed Haouari, Anis Kooli, and Emmanuel Néron. Enhanced energetic reasoning-based lower bounds for the resource constrained project scheduling problem. *Computers & Operations Research*, 39(5):1187–1194, 2012.

Xuejun Hu, Yuhao Wang, Mario Vanhoucke, Zhongbao Zhou, and Jianjiang Wang. Integrated capacity planning and multi-project scheduling considering resource transfer and idleness. *International Journal of Production Research*, 64(9):3445–3468, 2026.

Johann Hurink, Bernd Jurisch, and Monika Thole. Tabu search for the job-shop scheduling problem with multi-purpose machines. *Operations-Research-Spektrum*, 15(4):205–215, 1994.

James Kotary, Ferdinando Fioretto, and Pascal Van Hentenryck. Fast approximations for job shop scheduling: A lagrangian dual deep learning method. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pp. 7239–7246, 2022.

Kun Lei, Peng Guo, Wenchao Zhao, Yi Wang, Linmao Qian, Xiangyin Meng, and Liansheng Tang. A multi-action deep reinforcement learning framework for flexible job-shop scheduling problem. *Expert Systems with Applications*, 205:117796, 2022.

Chien-Liang Liu and Tzu-Hsuan Huang. Dynamic job-shop scheduling problems using graph neural network and deep reinforcement learning. *IEEE transactions on systems, man, and cybernetics: systems*, 53(11):6836–6848, 2023.

Renke Liu, Rajesh Piplani, and Carlos Toro. Deep reinforcement learning for dynamic scheduling of a flexible job shop. *International Journal of Production Research*, 60(13):4049–4069, 2022.

Shu Luo, Linxuan Zhang, and Yushun Fan. Real-time scheduling for dynamic partial-no-wait multiobjective flexible job shop by deep reinforcement learning. *IEEE Transactions on Automation Science and Engineering*, 19(4):3020–3038, 2021.

Leilei Meng, Chaoyong Zhang, Yaping Ren, Biao Zhang, and Chang Lv. Mixed-integer linear programming and constraint programming formulations for solving distributed flexible job shop scheduling problem. *Computers & industrial engineering*, 142:106347, 2020.

Wenjia Meng, Qian Zheng, Gang Pan, and Yilong Yin. Off-policy proximal policy optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 9162–9170, 2023.

Luc Mercier and Pascal Van Hentenryck. Edge finding for cumulative scheduling. *INFORMS Journal on Computing*, 20(1):143–153, 2008.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pp. 278–287. Citeseer, 1999.

Chinyere Ngwu, Ying Liu, and Rui Wu. Reinforcement learning in dynamic job shop scheduling: a comprehensive review of ai-driven approaches in modern manufacturing. *Journal of Intelligent Manufacturing*, 37(3):1093–1108, 2026.

Daniel Alejandro Rossit, Fernando Tohmé, and Mariano Frutos. Industry 4.0: smart scheduling. *International Journal of Production Research*, 57(12):3802–3813, 2019.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Igor G Smit, Jianan Zhou, Robbert Reijnen, Yaoxin Wu, Jian Chen, Cong Zhang, Zaharah Bukhsh, Yingqian Zhang, and Wim Nuijten. Graph neural networks for job shop scheduling problems: A survey. *Computers & Operations Research*, 176:106914, 2025.

Wen Song, Xinyang Chen, Qiqiang Li, and Zhiguang Cao. Flexible job-shop scheduling via graph neural network and deep reinforcement learning. *IEEE Transactions on Industrial Informatics*, 19(2):1600–1610, 2023.

Eric Taillard. Benchmarks for basic scheduling problems. *european journal of operational research*, 64(2):278–285, 1993.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *ACM Computing Surveys*, 55(6):1–28, 2022.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Rongkai Wang, Yiyang Jing, Chaojie Gu, Shibo He, and Jiming Chen. End-to-end multitarget flexible job shop scheduling with deep reinforcement learning. *IEEE Internet of Things Journal*, 12(4):4420–4434, 2024a.

Runqing Wang, Gang Wang, Jian Sun, Fang Deng, and Jie Chen. Flexible job shop scheduling via dual attention network-based reinforcement learning. *IEEE Transactions on Neural Networks and Learning Systems*, 35(3):3091–3102, 2024b.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

Xiangjie Xiao, Cong Zhang, Wen Song, and Zhiguang Cao. RESCHED: Rethinking flexible job shop scheduling from a transformer-based architecture with simplified states. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=s5pWbwf2tk.

Jin Xie, Liang Gao, Kunkun Peng, Xinyu Li, and Haoran Li. Review on flexible job shop scheduling. *IET collaborative intelligent manufacturing*, 1(3):67–77, 2019.

Hegen Xiong, Shuangyuan Shi, Danni Ren, and Jinjin Hu. A survey of job shop scheduling problem: The types and models. *Computers & Operations Research*, 142:105731, 2022.

Cong Zhang, Wen Song, Zhiguang Cao, Jie Zhang, Puay Siew Tan, and Xu Chi. Learning to dispatch for job shop scheduling via deep reinforcement learning. *Advances in neural information processing systems*, 33:1621–1632, 2020.

Linlin Zhao, Jiaxin Fan, Chunjiang Zhang, Weiming Shen, and Jing Zhuang. A drl-based reactive scheduling policy for flexible job shops with random job arrivals. *IEEE Transactions on Automation Science and Engineering*, 21(3):2912–2923, 2023.

Peng Zhao, You Zhou, Di Wang, Zhiguang Cao, Yubin Xiao, Xuan Wu, Yuanshu Li, Hongjia Liu, Wei Du, Yuan Jiang, et al. Dual operation aggregation graph neural networks for solving flexible job-shop scheduling problem with reinforcement learning. In *Proceedings of the ACM on Web Conference 2025*, pp. 4089–4100, 2025.

Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI open*, 1:57–81, 2020.

A Proof of Proposition 1

Proposition 1 (Cardinality-Invariant Weighting) *Under the balanced family weighting scheme, let K_t denote the number of nonempty active constraint families. Then, for any nonempty family $\mathcal{C}_t^k$, the aggregate weight assigned to that family is:*

$$\sum_{c\in\mathcal{C}_t^k} w_c^t = \frac{1}{K_t}, \qquad \sum_{c\in\mathcal{C}_t} w_c^t = 1. \tag{19}$$

Proof Let $\mathcal{K}_t$ denote the index set of nonempty active constraint families, with $K_t = |\mathcal{K}_t| \geq 1$. The active constraint set is partitioned into disjoint families, such that $\mathcal{C}_t = \bigcup_{k\in\mathcal{K}_t} \mathcal{C}_t^k$. Under the family-balanced weighting scheme, the weight of each constraint $c \in \mathcal{C}_t^k$ is defined as:

$$w_c^t = \frac{1}{K_t} \frac{\exp(x_c^t/\tau)}{\sum_{c'\in\mathcal{C}_t^k} \exp(x_{c'}^t/\tau)}, \tag{20}$$

where $\tau > 0$ is the pooling temperature. Since $\mathcal{C}_t^k$ is nonempty, its softmax normalization term is strictly positive. Summing the weights within family k gives:

$$\sum_{c\in\mathcal{C}_t^k} w_c^t = \frac{1}{K_t} \frac{\sum_{c\in\mathcal{C}_t^k} \exp(x_c^t/\tau)}{\sum_{c'\in\mathcal{C}_t^k} \exp(x_{c'}^t/\tau)} = \frac{1}{K_t}.$$

Because the nonempty active families form a disjoint partition of $\mathcal{C}_t$, the total weight is:

$$\sum_{c\in\mathcal{C}_t} w_c^t = \sum_{k\in\mathcal{K}_t} \sum_{c\in\mathcal{C}_t^k} w_c^t = \sum_{k\in\mathcal{K}_t} \frac{1}{K_t} = 1. \tag{21}$$

Thus, each nonempty active family receives the same aggregate weight $1/K_t$, independently of its number of constraints. This establishes the claimed family-level allocation property.

B Proof of Proposition 2

Proposition 2 (Soundness of the Completion-Loss Certificate) *Let s_t be a partial schedule reached under the scheduling transition considered in this work, and let $H > 0$ be a fixed reference horizon. Let $\mathcal{F}_t(H)$ denote the set of feasible completions that preserve the operations already scheduled in s_t, follow the same transition, and have makespan at most H. Then*

$$\Delta(s_t; H) > 0 \quad \Longrightarrow \quad \exists\, c \in \mathcal{C} : x_c(s_t; H) > 0 \quad \Longrightarrow \quad \mathcal{F}_t(H) = \varnothing. \tag{22}$$

The converse does not necessarily hold: $\Delta(s_t; H) = 0$ does not guarantee that $\mathcal{F}_t(H) \neq \varnothing$.

Proof Suppose that $\widehat{S} \in \mathcal{F}_t(H)$. For each job J_i, its remaining operations require at least W_i^t processing time after its current ready time R_i^t. Therefore, $R_i^t + W_i^t \leq H$, and the corresponding job-path pressure is nonpositive. Consider any constructed resource set $\mathcal{G}$ and interval $[l, u] \subseteq [0, H]$. Each remaining operation counted in $D_{\mathcal{G}}^t(l, u)$ must be processed on a machine in $\mathcal{G}$ and contributes at least $\omega_q^t(l, u)$ units of processing within $[l, u]$. Previously scheduled operations and their processing intervals remain fixed. Moreover, under the scheduling transition, every operation subsequently assigned to M_k starts no earlier than R_k^t. Hence, the total time available to remaining operations on machines in $\mathcal{G}$ within $[l, u]$ is at most $C_{\mathcal{G}}^t(l, u)$. It follows that

$$D_{\mathcal{G}}^t(l, u) \leq C_{\mathcal{G}}^t(l, u). \tag{23}$$

The same argument applies to the global interval $[0, H]$. Since all pressure denominators are positive, every constructed job-path, global-capacity, and local-capacity pressure must satisfy

$$x_c(s_t; H) \leq 0, \qquad \forall c \in \mathcal{C}. \tag{24}$$

Algorithm 1 Training Procedure of Resource Capacity Reasoning (RCR)

Input: Training instances $\mathcal{D}_{\mathrm{tra}}$, training iterations I, a backbone scheduling policy and its original policy-gradient optimizer, and RCR hyperparameters
Output: Trained scheduling policy π_θ

for $i = 1, 2, \ldots, I$ **do**
  Sample a batch of instances from $\mathcal{D}_{\mathrm{tra}}$ according to the backbone's training procedure.
  Initialize the environments, rollout policy π_{roll}, and trajectory buffer $\mathcal{T} \leftarrow \emptyset$.
  for each instance in the batch **do**
    Construct a feasible reference schedule and set H to its makespan.
    Construct fixed constraints $\mathcal{C} = \mathcal{C}^J \cup \mathcal{C}^G \cup \mathcal{C}^W$ from resource groups and time windows.
    Initialize s_0 and set $t \leftarrow 0$.
    while s_t is nonterminal **do**
      Obtain the feasible action set $\mathcal{A}_t$ and evaluate the current constraint pressures.
      Identify the active constraint set $\mathcal{C}_t \subseteq \mathcal{C}$ and compute its family-balanced weights $\mathbf{w}^t$.
      for each $a \in \mathcal{A}_t$ **do**
        Construct the non-mutating counterfactual successor $\widetilde{s}_{t,a}$.
        Compute the four-dimensional BCD $\mathbf{z}_{t,a}$ using the fixed parent support $\mathcal{C}_t$.
      end for
      Standardize and clip the BCD descriptors across $\mathcal{A}_t$ to obtain $\widehat{\mathbf{z}}_{t,a}$.
      Augment the backbone action representations with $\widehat{\mathbf{z}}_{t,a}$.
      Select a_t using π_{roll} and execute it in the original environment.
      Observe the real successor s_{t+1} and original reward r_t^{base}.
      Compute the CLC $\Delta_{t+1} \leftarrow \Delta(s_{t+1}; H)$ over the complete constraint set $\mathcal{C}$.
      Form the augmented reward $r_t \leftarrow r_t^{\mathrm{base}} - \lambda \Delta_{t+1}/(N\sigma)$.
      Append the transition, selected action, r_t, and current-state BCD descriptors to $\mathcal{T}$.
      $s_t \leftarrow s_{t+1}$; $t \leftarrow t + 1$.
    end while
  end for
  Compute returns and optional advantages via backbone estimation.
  Update the backbone parameters using its original policy-gradient optimizer on $\mathcal{T}$.
end for

By definition,

$$\Gamma(s_t; H) = \max_{c \in \mathcal{C}} x_c(s_t; H), \qquad \Delta(s_t; H) = H[\Gamma(s_t; H) - \epsilon_c]_+, \tag{25}$$

where $H > 0$ and $\epsilon_c \geq 0$. If $\Delta(s_t; H) > 0$, then $\Gamma(s_t; H) > \epsilon_c \geq 0$. Thus, some $c^* \in \mathcal{C}$ satisfies $x_{c^*}(s_t; H) > 0$, contradicting Eq. 24. Therefore, $\mathcal{F}_t(H) = \varnothing$. Conversely, $\Delta(s_t; H) = 0$ implies only that $x_c(s_t; H) \leq \epsilon_c$ for every $c \in \mathcal{C}$. When $\epsilon_c > 0$, small positive violations remain possible. Even when $\epsilon_c = 0$, satisfying the constructed necessary conditions does not establish the existence of a feasible completion within H.

C Training Algorithm

Algorithm 1 integrates RCR into a scheduling backbone over I training iterations. For each instance, a feasible reference schedule determines H, while resource groups and time windows define the fixed constraint set $\mathcal{C}$. At state s_t, the active subset $\mathcal{C}_t$ and family-balanced weights $\mathbf{w}^t$ are identified. For each feasible action $a \in \mathcal{A}_t$, a non-mutating counterfactual successor $\widetilde{s}_{t,a}$ yields four-dimensional BCD $\mathbf{z}_{t,a}$ on the fixed parent support. The descriptors are standardized and clipped across feasible actions before augmenting backbone action representations. The rollout policy π_{roll} selects a_t, yielding the real successor s_{t+1} and original reward r_t^{base}. Unlike BCD, CLC evaluates the complete constraint set on s_{t+1} to obtain Δ_{t+1}. Its penalty, scaled by λ, operation count N, and reward factor σ, produces augmented reward r_t. Transitions and current-state BCD descriptors are stored in $\mathcal{T}$. Returns, applicable advantages, and updates follow the backbone's original policy-gradient procedure. RCR introduces counterfactual capacity reasoning and a certificate-based auxiliary reward without replacing the backbone encoder or policy-gradient optimizer. BCD compares feasible actions using a fixed parent support, whereas CLC evaluates the realized successor against the complete constraint set. This enables consistent RCR integration across scheduling backbones.

D Experimental Details

D.1 Dataset Details

For synthetic FJSP datasets, we follow the generation procedures adopted in prior studies (Song et al., 2023; Wang et al., 2024b). Given an instance with n jobs and m machines, processing times and operation configurations are sampled from predefined distributions. Specifically, SD1 samples processing times uniformly from $[1, 20]$ and assigns each job a number of operations uniformly drawn from $[0.8m, 1.2m]$. In contrast, SD2 samples processing times from the broader range $[1, 99]$ and assigns each job exactly m operations, resulting in scheduling instances with different structural and processing-time characteristics.

For public FJSP benchmarks, we evaluated the Brandimarte dataset (mk01–mk10) (Brandimarte, 1993) and three Hurink groups, including Rdata, Edata, and Vdata, with 40 instances in each group (Hurink et al., 1994). These benchmarks cover instances with 10–30 jobs and 4–15 machines, where the processing times of operations are distributed between 1 and 99.

For JSSP evaluation, we use the Taillard (Taillard, 1993) and DMU (Demirkol et al., 1998) benchmarks. The Taillard instances range from 15×15 to 100×20, while the DMU instances range from 20×15 to 50×20. These datasets cover diverse instance scales and distributions, enabling a comprehensive evaluation of the proposed method's generalization capability across different scheduling scenarios and problem formulations.

D.2 Baseline Details

D.2.1 FJSP Baselines

For FJSP, we select three representative DRL-based scheduling methods as backbones, including HGNN (Song et al., 2023), DANIEL (Wang et al., 2024b), and RESCHED (Xiao et al., 2026). HGNN and DANIEL are GNN-based scheduling approaches that employ PPO for policy optimization, while RESCHED adopts a Transformer-based architecture trained with REINFORCE. These methods represent different neural architectures and policy-gradient optimization paradigms, providing a comprehensive evaluation of the general applicability of RCR. We integrate RCR into each backbone without modifying its original encoder architecture, policy optimization algorithm, or training procedure, resulting in RCR-HGNN, RCR-DANIEL, and RCR-RESCHED. This design allows RCR to serve as an independent resource-capacity reasoning module that complements existing representation learning frameworks. In addition, the OR-Tools CP-SAT solver (Da Col & Teppan, 2019) is included as an optimization-based reference.

D.2.2 JSSP Baselines

For JSSP, we consider RESCHED (Xiao et al., 2026) as a representative learning-based baseline. RESCHED employs a Transformer-based architecture with REINFORCE training. Its RCR-enhanced variants, RCR-RESCHED, is constructed by incorporating the proposed resource-capacity reasoning module while preserving the original backbone designs and optimization frameworks. All baseline methods follow their original implementation settings.

D.3 Configuration Details

Table 6 summarizes the training configurations of the evaluated backbone methods. HGNN and DANIEL employ PPO for FJSP, while RESCHED uses REINFORCE with a Transformer-based architecture. For each method, we follow its original training configuration, including the number of iterations, parallel batch size, learning rate, and coefficients of the policy, value, and entropy terms. We also retain its discount factor: $\gamma = 1.0$ for HGNN and DANIEL, and $\gamma = 0.99$ for RESCHED. Each RCR variant uses the same configuration as its corresponding backbone; only the proposed capacity-based action features and auxiliary reward are added. Thus, the paired comparisons hold the policy-gradient algorithm and listed training settings fixed, allowing changes in scheduling performance to be assessed without backbone-specific retuning.

Table 6: Training configurations of the evaluated backbone methods. RCR-enhanced variants retain the original training strategies and hyperparameter settings of their respective backbones.

Configuration	RCR-HGNN	RCR-DANIEL	RCR-RESCHED
Training algorithm	PPO	PPO	REINFORCE
Training iterations	1000	1000	2000
Parallel batch size	20	20	50
Learning rate	2×10^{-4}	3×10^{-4}	6×10^{-4}
Policy coefficient	1	1	1
Value coefficient	0.5	0.5	N/A
Entropy coefficient	0.01	0.01	N/A
Discount factor	1.0	1.0	0.99

Table 7: Mean $\pm$ standard deviation on synthetic FJSP datasets across five random seeds.

Methods		SD1		SD2	
		10×5	20×10	10×5	20×10
HGNN	Gap (%)	16.03 ± 3.25	10.15 ± 1.88	71.42 ± 3.47	126.12 ± 4.21
RCR-HGNN	Gap (%)	$\mathbf{11.89 \pm 1.64}$	$\mathbf{9.03 \pm 0.74}$	$\mathbf{27.22 \pm 2.02}$	$\mathbf{32.32 \pm 2.28}$
DANIEL	Gap (%)	10.87 ± 1.02	1.31 ± 1.82	25.68 ± 1.23	31.58 ± 0.94
RCR-DANIEL	Gap (%)	$\mathbf{9.76 \pm 0.53}$	$\mathbf{0.18 \pm 1.06}$	$\mathbf{24.85 \pm 0.76}$	$\mathbf{30.10 \pm 0.68}$
RESCHED	Gap (%)	12.25 ± 2.06	0.48 ± 1.02	16.36 ± 2.95	14.18 ± 2.34
RCR-RESCHED	Gap (%)	$\mathbf{9.19 \pm 1.38}$	$\mathbf{-0.95 \pm 0.73}$	$\mathbf{11.69 \pm 1.57}$	$\mathbf{10.82 \pm 1.20}$

Table 8: Mean $\pm$ standard deviation on public FJSP benchmarks across five random seeds.

Methods		Brandimarte	Edata	Rdata	Vdata
HGNN	Gap (%)	28.52 ± 4.88	15.53 ± 2.29	11.15 ± 2.43	4.25 ± 1.21
RCR-HGNN	Gap (%)	$\mathbf{11.77 \pm 2.45}$	$\mathbf{15.51 \pm 0.61}$	$\mathbf{10.09 \pm 1.05}$	$\mathbf{3.19 \pm 0.63}$
DANIEL	Gap (%)	$\mathbf{13.58 \pm 1.28}$	16.33 ± 1.97	11.42 ± 1.43	3.28 ± 0.65
RCR-DANIEL	Gap (%)	14.82 ± 0.72	$\mathbf{15.65 \pm 1.09}$	$\mathbf{11.25 \pm 0.55}$	$\mathbf{3.01 \pm 0.45}$
RESCHED	Gap (%)	$\mathbf{9.08 \pm 1.28}$	15.48 ± 2.28	10.18 ± 2.71	3.48 ± 0.79
RCR-RESCHED	Gap (%)	10.81 ± 0.87	$\mathbf{13.63 \pm 1.23}$	$\mathbf{7.96 \pm 1.45}$	$\mathbf{2.46 \pm 0.51}$

E MULTIPLE RANDOM SEEDS

Tables 7 and 8 report the performance of three backbones and their RCR-enhanced variants as mean $\pm$ standard deviation across five independent random seeds. We evaluate synthetic and public FJSP datasets, including 10×5 and 20×10 instances from SD1 and SD2, together with the Brandimarte and Hurink benchmarks. This evaluation covers different instance scales, data distributions, and public benchmarks, providing an assessment of training stability. Across these settings, the RCR-enhanced variants exhibit relatively small standard deviations, with particularly consistent results for RCR-DANIEL and RCR-RESCHED. Although RCR-HGNN shows larger variation on some SD2 instances, its performance remains substantially improved over the original backbone. More importantly, the overall performance trends remain consistent across runs and benchmark settings rather than being driven by a particular random initialization. These results provide further evidence for the reproducibility and robustness of RCR while confirming that the main conclusions remain valid when accounting for run-to-run variance.

Table 9: Fine-grained ablation results for RCR-RESCHED on SD1.

Variants		10×5	20×5	15×10	20×10
RCR-RESCHED	Gap (%)	**9.19**	**3.02**	**5.10**	**-0.95**
w/o LW	Gap (%)	10.02	3.98	6.08	-0.14
w/o RG	Gap (%)	9.91	4.21	6.32	0.27
w/o KL	Gap (%)	9.87	4.01	5.83	0.08
w/o BCD	Gap (%)	10.83	4.32	6.51	0.44
w/o CIW	Gap (%)	10.42	3.97	5.80	-0.39
w/o CLC	Gap (%)	9.94	4.26	6.11	0.38

Table 10: Ablation results for RCR-HGNN.

Variants		10×5	20×5	15×10	20×10
RCR-HGNN	Gap (%)	**11.89**	**10.22**	**11.33**	**9.03**
w/o BCD	Gap (%)	13.57	11.32	12.05	10.33
w/o CIW	Gap (%)	12.86	10.92	12.42	10.05
w/o CLC	Gap (%)	12.14	11.97	12.38	10.08
HGNN	Gap (%)	16.03	12.27	16.33	10.15

F Experimental Analyses

F.1 Ablation study

Fine-grained ablation results Table 9 presents fine-grained ablation results for RCR-RESCHED on four SD1 instance sizes under greedy decoding. We examine local time windows (LW), additional resource groups (RG), and KL-based bottleneck redistribution (KL). Specifically, w/o LW removes local time-window constraints from BCD and CLC, while w/o RG excludes eligibility-induced groups but retains singleton-machine and all-machine groups. w/o KL sets the fourth BCD component to zero, preserving the policy input dimension. w/o CIW replaces family-balanced weighting with a single softmax over active constraints, while w/o CLC removes the auxiliary penalty. All variants share the same backbone configuration and test instances. These ablations assess temporal capacity reasoning, resource-group modeling, bottleneck redistribution, family-balanced weighting, and completion-oriented reward guidance.

Ablation study on RCR-HGNN and RCR-DANIEL Tables 10 and 11 present ablation results for RCR-HGNN and RCR-DANIEL under greedy decoding. We evaluate variants on SD1, including 10×5, 20×5, 15×10 and 20×10 instances. Removing individual components generally degrades scheduling performance across backbones and instance scales. RCR-DANIEL slightly underperforms the w/o CLC variant on 20×5, although the difference is marginal. Similar degradation patterns across both GNN-based backbones indicate that RCR is not architecture-specific. These results support the transferability of resource-capacity reasoning across different instance scales.

Table 11: Ablation results for RCR-DANIEL.

Variants		10×5	20×5	15×10	20×10
RCR-DANIEL	Gap (%)	**9.76**	4.13	**11.47**	**0.18**
w/o BCD	Gap (%)	11.45	5.98	12.64	0.74
w/o CIW	Gap (%)	10.72	5.29	11.91	0.89
w/o CLC	Gap (%)	11.23	**4.02**	12.43	1.01
DANIEL	Gap (%)	10.87	5.03	12.42	1.31

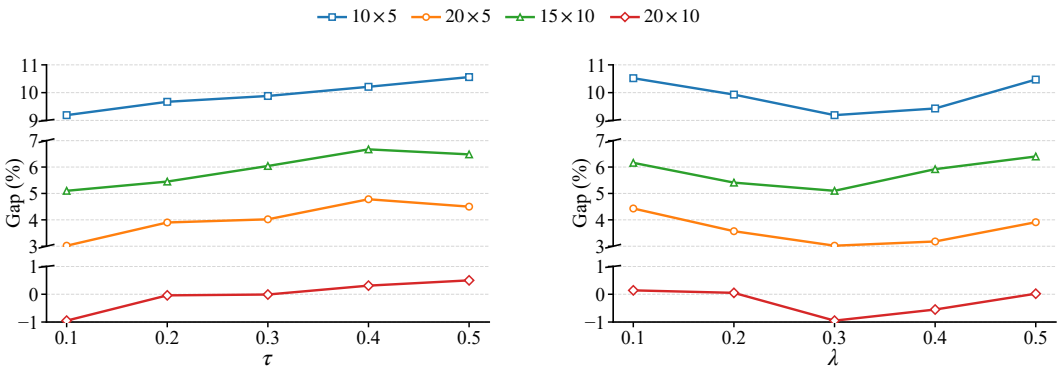

Figure 3: Sensitivity results of τ and λ.

Table 12: Sensitivity analysis of λ.

Variants		10×5	20×5	15×10	20×10
$\lambda = 0.3$	Gap(%)	**9.19**	**3.02**	**5.10**	**-0.95**
$\lambda = 0.1$	Gap(%)	10.52	4.43	6.16	0.14
$\lambda = 0.2$	Gap(%)	9.93	3.57	5.41	0.05
$\lambda = 0.4$	Gap(%)	9.43	3.18	5.92	-0.55
$\lambda = 0.5$	Gap(%)	10.47	3.91	6.40	0.02

F.2 Sensitivity Study

We conduct sensitivity analyses of RCR-RESCHED on SD1 under greedy decoding to evaluate the impact of the additional hyperparameters introduced by RCR. We investigate four parameters: the BCD pooling temperature τ, the CLC penalty coefficient λ, the number of time bins B, and the maximum number of additional resource groups $\mathcal{G}_{\max}$. These parameters regulate the concentration of capacity weighting, the strength of completion-oriented guidance, the temporal resolution of capacity reasoning, and the granularity of resource-group construction. The backbone architecture, optimization strategy, and training configurations remain unchanged across all experiments, ensuring that the observed variations primarily reflect the effects of RCR-specific hyperparameters.

We present the sensitivity analysis of τ in the main text. Figure 3 visualizes the effects of τ and λ, while Table 12 reports the results for λ. The results show that $\tau = 0.1$ achieves the best performance, suggesting that concentration on high-pressure capacity constraints facilitates identification of critical bottlenecks. As τ increases, the weighting distribution becomes more uniform across constraints, which may dilute capacity-critical signals and reduce BCD effectiveness in distinguishing favorable actions. For λ, the best performance is obtained at $\lambda = 0.3$, indicating that a moderate CLC penalty provides completion-oriented guidance while preserving the makespan objective. Smaller values may provide insufficient guidance, whereas larger values can overemphasize the auxiliary objective and introduce optimization bias. These observations show that weighting and penalty strength are important for balancing capacity guidance with scheduling decisions.

Table 13 and Table 14 report sensitivity results for B and $\mathcal{G}_{\max}$. The setting $B = 8$ achieves the best performance, indicating that suitable temporal resolution captures capacity variations while limiting discretization noise. Fewer bins may obscure temporal pressure patterns, whereas more bins increase granularity and reasoning complexity. Similarly, $\mathcal{G}_{\max} = 8$ performs best on three of four scales, except on 10×5, where $\mathcal{G}_{\max} = 4$ performs slightly better. Overall, $\mathcal{G}_{\max} = 8$ balances resource interaction modeling and stable capacity representations. Insufficient groups may overlook resource relationships, while excessive grouping can fragment capacity information. RCR remains effective across parameter configurations, indicating moderate sensitivity. The selected settings provide balanced resource-capacity reasoning without backbone-specific retuning.

Table 13: Sensitivity analysis of B.

Variants		10×5	20×5	15×10	20×10
$B = 8$	Gap(%)	**9.19**	**3.02**	**5.10**	**-0.95**
$B = 4$	Gap(%)	9.82	3.07	6.16	-0.44
$B = 12$	Gap(%)	9.43	3.76	5.35	-0.02
$B = 16$	Gap(%)	10.02	3.91	6.67	-0.88
$B = 32$	Gap(%)	9.66	4.05	6.80	-0.11

Table 14: Sensitivity analysis of $\mathcal{G}_{\max}$.

Variants		10×5	20×5	15×10	20×10
$\mathcal{G}_{\max} = 8$	Gap(%)	9.19	**3.02**	**5.10**	**-0.95**
$\mathcal{G}_{\max} = 4$	Gap(%)	**9.10**	3.33	5.81	-0.44
$\mathcal{G}_{\max} = 12$	Gap(%)	9.79	3.30	5.30	-0.80
$\mathcal{G}_{\max} = 16$	Gap(%)	9.64	3.49	6.11	-0.02
$\mathcal{G}_{\max} = 32$	Gap(%)	9.95	3.58	6.05	0.17

Table 15: Sensitivity analysis of τ and λ for RCR-DANIEL.

τ						λ					
Variants		10×5	20×5	15×10	20×10	Variants		10×5	20×5	15×10	20×10
$\tau = 0.1$	Gap(%)	9.76	**4.13**	**11.47**	**0.18**	$\lambda = 0.3$	Gap(%)	**9.76**	**4.13**	**11.47**	**0.18**
$\tau = 0.2$	Gap(%)	10.77	5.02	12.79	0.54	$\lambda = 0.1$	Gap(%)	10.23	5.64	12.45	0.92
$\tau = 0.3$	Gap(%)	**9.63**	5.22	11.98	0.87	$\lambda = 0.2$	Gap(%)	11.04	4.92	12.29	0.95
$\tau = 0.4$	Gap(%)	10.80	4.71	12.53	0.56	$\lambda = 0.4$	Gap(%)	9.83	5.88	12.50	1.29
$\tau = 0.5$	Gap(%)	10.34	5.93	12.86	1.56	$\lambda = 0.5$	Gap(%)	11.20	5.49	12.15	1.44

We further analyze the sensitivity of τ and λ for RCR-DANIEL under greedy decoding on SD1. Since RCR-DANIEL employs a GNN-based scheduler optimized with PPO, this experiment complements the sensitivity analysis of RCR-RESCHED and provides additional evidence for the robustness of RCR across different backbones. As shown in Table 15, $\tau = 0.1$ achieves the best performance on three of the four evaluated instance scales, while $\lambda = 0.3$ consistently performs best across all scales. The only exception occurs on the 10×5 instances, where $\tau = 0.3$ marginally outperforms $\tau = 0.1$. Overall, these results indicate that the selected hyperparameters provide stable resource-capacity guidance and transfer effectively across different scheduling backbones.

G Reference-Horizon Sensitivity and Test-Time Cost

The reference horizon H is initialized to the makespan of a feasible reference schedule and defines the time windows and capacity pressures used by RCR. Because these quantities affect BCD action features, we assess the sensitivity of scheduling performance to changes in H.

We evaluate RCR-RESCHED with greedy decoding on the SD1 and SD2 datasets. For each instance, Shortest Processing Time (SPT) and Most Work Remaining (MWKR) generate two deterministic feasible reference schedules on separate copies of the scheduling environment. We set the default horizon to $H_0 = \min\{C_{\max}^{\mathrm{SPT}}, C_{\max}^{\mathrm{MWKR}}\}$. To assess sensitivity to the reference horizon, we evaluate the checkpoints trained with H_0 using $H_\alpha = \alpha H_0$ for $\alpha \in \{1.0, 1.1, 1.2\}$, without retraining. Since $\alpha \geq 1$, the selected reference schedule remains feasible under every tested horizon. For each value of α, we update the horizon-dependent time windows and capacity pressures while keeping the model parameters, test instances, and greedy decoding fixed.

Table 16: Sensitivity of RCR-RESCHED to the reference horizon $H_\alpha = \alpha H_0$ and its test-time cost. The first three rows report mean instance-wise gaps under greedy decoding without retraining. The final two rows report per-instance times at $\alpha = 1.0$; t_{test} includes t_H.

Setting		SD1		SD2	
		10×5	20×10	10×5	20×10
$\alpha = 1.0$	Gap (%)	**9.19**	**-0.95**	**11.69**	**10.82**
$\alpha = 1.1$	Gap (%)	10.02	-0.17	12.34	11.68
$\alpha = 1.2$	Gap (%)	10.32	0.05	11.90	11.24
t_H	s	0.12	0.44	0.11	0.43
t_{test}	s	0.48	2.26	0.49	2.29

Table 17: Results on DMU Benchmark for JSSP. UB is the best-known upper bound.

Methods		20×15	20×20	30×15	30×20	40×15	40×20	50×15	50×20
UB	Ref.	3023.8	3472.6	3879.0	4248.4	4871.2	5240.9	5950.6	6227.3
RESCHED	Gap (%)	**23.7**	22.2	27.8	28.3	26.5	**29.2**	26.3	31.8
RCR-RESCHED	Gap (%)	**23.7**	**22.1**	**26.4**	**28.1**	**25.3**	30.4	**25.6**	**31.7**

At test time, H_0 is generated before the first decision, and H_α remains fixed throughout the episode. The horizon conditions BCD action features; the CLC penalty is applied only during training. Table 16 shows that $\alpha = 1.0$ achieves the lowest gap on all four datasets; the maximum increase is 1.13 percentage points. Here, t_H is the per-instance wall-clock time required to generate H_0, and t_{test} is the per-instance time for the complete greedy test episode, including horizon generation.

H Performance on the DMU JSSP Benchmarks

The main text reports the performance of RCR on the Taillard JSSP benchmarks (Taillard, 1993). Table 17 presents additional results for RESCHED (Xiao et al., 2026) and its RCR-enhanced variant, RCR-RESCHED, on the DMU benchmarks (Demirkol et al., 1998) under greedy decoding. Both models are trained on 10×10 instances and directly evaluated on larger DMU instances ranging from 20×15 to 50×20 without retraining. RCR improves RESCHED on six of the eight evaluated instance sizes and matches its performance on 20×15. On 40×20, however, the gap increases from 29.2% to 30.4%. Despite this exception, the overall results support the transferability of resource-capacity reasoning beyond the training scale under substantial changes in instance size and distribution. Together with the Taillard and FJSP results, these findings provide further evidence of the applicability of RCR across benchmark distributions and scheduling variants.

I Future Work

Future work will strengthen the expressiveness and theoretical foundations of resource-capacity reasoning. Adaptive selection of resource groups and time intervals could improve detection of bottlenecks unobserved under fixed constraint sets. Such extensions should preserve certificate soundness and common constraint support across candidate actions. Another direction is to study reference-horizon dependence and develop reference horizon updates that yield stable, informative action descriptors. We will also analyze how cumulative CLC penalties influence policy preferences and identify conditions under which preserving completion feasibility improves makespan. Finally, extending RCR to uncertain processing times, machine disruptions, and shared resources will require capacity conditions accounting for uncertainty and evolving availability, alongside systematic evaluation of generalization across scheduling settings and policy architectures.