2025 International Conference on Intelligent Computing
July 26-29, Ningbo, China
https://www.ic-icc.cn/2025/index.php

ConDRL-JSP: A Contrastive and Reinforcement Learning-Based Framework for JSP

Jian Li[1,2],Shuhan Qi[1,2],Xinyu Xiao[1,2],Chao Xing[3] , Jiajia Zhang[1,2] and Xuan Wang[1,2]

[1] Harbin Institute of Technology, Shenzhen 518055, China
[2] Guangdong Provincial Key Laboratory of Novel Security Intelligence Technologies, Shenzhen 518055, China
[3] Shenzhen Zhice Technology Co., Ltd, Shenzhen 518055, China
{23s151055, 23b951021}@stu.hit.edu.cn
{shuhanqi,xingchao,zhangjiajia,wangxuan}@ cs.hitsz.edu.cn

Abstract. In fields such as manufacturing, the Job Shop Scheduling Problem (JSP), a classic NP-hard combinatorial optimization problem, is faced with challenges where traditional solution methods suffer from high computational complexity and poor adaptability, and construction-based methods have issues of low sample efficiency and weak generalization ability. To overcome these difficulties, this paper proposes the ConDRL-JSP framework, which innovatively integrates contrastive learning, reinforcement learning, and the curriculum learning strategy. The introduction of contrastive learning aims to address the problems of weak feature discrimination and limited data mining capabilities of traditional reinforcement learning in JSP. It enhances the model's ability to extract key features from complex scheduling states, enabling more efficient decision-making. The curriculum learning strategy dynamically adjusts the task difficulty based on the model's training performance. Starting with simple instances, it gradually progresses to complex scenarios, effectively improving the model's generalization ability and avoiding getting trapped in local optima.

Keywords: Job Shop Scheduling Problem ·Contrastive Learning, Reinforcement Learning, Curriculum Learning

1 Introduction

The Job Shop Scheduling Problem (JSP) is a classic NP-hard combinatorial optimization problem, which holds great significance in both the manufacturing and transportation industries. For example, in the production of automotive parts, reasonable JSP scheduling can optimize the job-machine allocation, enhance resource utilization, and reduce costs. In the transportation field, it can improve the operational efficiency of vehicles in logistics centers. Its core objective is to minimize the makespan by allocating jobs reasonably, so as to achieve the optimal allocation of resources.

Methods for solving the JSP can be mainly classified into three categories: traditional heuristic methods, construction-based methods, and methods focusing on optimization and adaptation strategies. Traditional heuristic methods rely on empirical

rules. For instance, the Shortest Processing Time (SPT) and the Most Work Remaining (MWKR) in the Priority Dispatching Rules (PDRs) [12] are simple and intuitive, but their performance is unstable across different problem instances. Exact algorithms (such as integer programming [1]) and heuristic methods (such as tabu search [10]) can obtain better solutions, yet they suffer from high computational complexity and are not applicable to large-scale JSP. The Efficient Active Search (EAS) strategy proposed by Hottung et al. [13] improves the construction method of combinatorial optimization through partial parameter updates. However, it has the issues of time-consuming search and insufficient ability to handle complex industrial constraints.

Construction-based methods often make use of machine learning techniques. Supervised learning methods, such as Pointer Networks [26] and Graph Convolutional Networks [19], rely on costly labeled data. Early unsupervised reinforcement learning methods (such as L2D [27]) have poor state representation and solution quality in complex scenarios. Corsini et al. [7] proposed a supervised learning task of predicting the quality of machine permutations, evaluated it using a MILP solver, and trained a deep learning model for application in the tabu search algorithm. Nevertheless, this method increases the algorithm's time cost, and its general applicability remains to be verified.

The construction methods based on reinforcement learning are constantly evolving. Tassel et al. [25] modeled the JSP as a single-agent reinforcement learning problem and constructed an environment using the Proximal Policy Optimization (PPO) algorithm. However, the PPO is prone to getting trapped in local optima, and the performance of the scheduling solutions is inferior to that of the optimal constraint solvers. The DGERD Transformer proposed by Chen et al. [5] combines the attention mechanism with disjunctive graph embedding, which can handle problems of any scale, but its ability to deal with complex constraints is limited. The adaptive scheduling framework proposed by Han et al. [11] based on the dueling double deep Q-network can directly learn scheduling strategies from high-dimensional inputs, but it has extremely high requirements for the quality of training data.

In terms of optimization solving and adaptation, the deep reinforcement learning heuristic algorithm based on Graph Neural Networks (GNN) proposed by Zhang et al. [28] relies on artificial rules, and its generalizability is limited. The NeuroLS model proposed by Falkner et al. [9] formalizes the local search component as a Markov Decision Process (MDPs), and the quality of the solution is sensitive to the initial conditions. The NeuRewriter model in the literature [6] is flexible in rewriting optimization problems but relies heavily on a large amount of prior knowledge. The POMO method proposed by Kwon et al. [16] solves combinatorial optimization problems through techniques such as multi-node exploration. However, in some problems, it is difficult to determine the optimal starting node, and its universality is insufficient. The hardness-adaptive curriculum learning method for the Traveling Salesman Problem proposed by Zhang et al. [30] has the defects of a single problem type, high training cost, and lack of sufficient verification with large-scale instances. Curriculum learning [14] faces the challenge of quantifying task difficulty in ultra-large-scale JSP, and the sample construction and target alignment mechanisms of contrastive learning in JSP also need further exploration.

Notably, contrastive learning has demonstrated remarkable effectiveness in various fields. For example, vanOord et al. [22] proposed Contrastive Predictive Coding (CPC), which compresses high-dimensional data, predicts the future with an autoregressive model, and uses InfoNCE loss for training. By maximizing mutual information between observations, CPC enables the model to extract useful features and has achieved excellent results in speech, image, natural language processing, and reinforcement learning tasks.

In view of the above problems, this paper proposes the ConDRL-JSP framework, which integrates contrastive learning, reinforcement learning, and curriculum learning. The contrastive-reinforcement learning integration module enhances the feature representation in complex scenarios, and the dynamic curriculum adjustment module based on the RASCL strategy [14] optimizes the learning process for problems of different scales. This framework overcomes the problems of poor generalization and low efficiency of existing methods. Experiments on the Taillard and Demirkol benchmark datasets show that the ConDRL-JSP framework can significantly shorten the makespan, improve the scheduling quality, obtain solutions close to the optimal, and has strong generalization ability, providing an effective solution for the practical application of JSP.

2 Model

In this section, we elaborate on the constraints of the problem and the principles of the method. We model the solution process of the Job Shop Scheduling Problem (JSP) as a Markov Decision Process (MDP). In this process, the solution is gradually constructed based on the inferred scheduling decisions, and contrastive learning continuously optimizes the state representation of samples during the training process. Finally, the training algorithm is introduced. The model schedules operations according to the scheduling policy π_θ, which is parameterized by a neural network and optimized through reinforcement learning.

The Job Shop Scheduling Problem (JSP) is a well-known combinatorial optimization challenge [23]. It involves assigning jobs ($J=\{J_1,J_2,\ldots,J_n\}$) with specific processing sequences and durations to machines ($M=\{M_1,M_2,\ldots,M_n\}$), aiming to minimize the makespan and achieve other optimization goals. Each job J_i is composed of multiple operation $O_{i1}\rightarrow O_{i2}\rightarrow\ldots\rightarrow O_{in_i}$, where the processing time of operation O_{ij} is P_{ij}, and the operations must be processed on a specific machine without interruption.

2.1 Mathematical Symbols

Table 1. Mathematical Symbols and Their Meanings. summarizes the mathematical symbols used in this paper and their corresponding meanings. This table serves as a quick reference for readers when encountering these symbols in the subsequent sections that describe the methods.

Table 1. Mathematical Symbols and Their Meanings.

Symbol	Meaning
$J=\{J_1,J_2,\ldots,J_n\}$	Set of jobs in JSP
$M=\{M_1,M_2,\ldots,M_m\}$	Set of machines in JSP
O_{ij}	j-th operation of job J_i
p_{ij}	Processing time of O_{ij}
s_t	State at decision-making moment t in MDP
a_t	Action at time t in MDP
A_t	Action space at time t in MDP
T	Transition function in MDP ($s_{t+1}=T(s_t,a_t)$)
$C_t(x,s_t,a_t)$	Cost function at time t in MDP
$\pi_\theta(a_t\|s_t)$	Policy network outputting a_t probability given s_t (θ parameter)
$J^\pi(\theta)$	Objective function for policy π (θ parameter)
E	Expectation operator
X	Set of all scheduling instances
∇_θ	Gradient w.r.t. θ
B	Training batch size
$\phi_\theta(x,s_t)$	Baseline by critic network (ϕ parameter)
L_{con}	Contrastive learning loss
L_{total}	Model's total loss function
λ	Contrastive loss weight in total loss
L_{actor}	Actor network policy gradient loss
L_{critic}	Critic network value evaluation loss
γ	Discount factor for R_t
$A(s_i,a_i)$	Advantage function for s_i, a_i
$Q(s_i,a_i)$	Action-value function for s_i, a_i
L	Set of curriculum learning difficulty levels ($L=\{l_0,\ldots,l_K\}$)
l	Current curriculum learning difficulty level
δ_{th}	Curriculum learning optimal gap threshold
$Makespan_{model}$	Model-obtained makespan
$Makespan_{opt}$	Optimal makespan

2.2 Constraints

Precedence Constraints: Operations within each job must be executed in the predetermined order. That is, only when the previous operation $O_{i,j-1}$ of job J_i is completed

can the subsequent operation O_{ij} start processing. Each job J_i must undergo operations in the order of $O_{i1}\rightarrow O_{i2}\rightarrow\ldots\rightarrow O_{in_i}$. This precedence constraint, as shown in **Fig. 1** in the flow of operation processing, has been extensively studied and is fundamental in JSP research [2,4].

Machine Exclusivity Constraints: Each machine can process only one job at a given moment; multiple jobs cannot be processed on one machine simultaneously. This constraint is a basic characteristic of the JSP problem [4,20,21] and is incorporated into the state representation and action-space definitions in our model.

Non-Preemption Constraints: Once a job starts processing, it cannot be interrupted or preempted before completion. This ensures the continuity of job processing and prevents production inefficiencies and disruptions to the production plan. This non-preemption constraint, as illustrated in the overall scheduling process in **Fig. 1**, is an important feature of JSP and has been discussed in relevant literature [4].

2.3 Markov Decision Process Formulation

In the ConDRL-JSP framework, we utilize the Markov Decision Process (MDP) to construct the decision-making framework and define its key elements.

State: The state s_t comprehensively describes the system status at the decision-making moment. It is a multi-dimensional composite information structure that encompasses machine utilization $M_{util}(t)$, job time $J_{times}(t)$, job earliest start time $J_{early}(t)$, job state $J_{state}(t)$, and priority $P(t)$. It can be expressed as:

$$s_t = \{M_{util}(t), J_{times}(t), J_{early}(t), J_{state}(t), P(t)\} \tag{1}$$

Action and Action Space: The action a_t represents the job chosen for scheduling from the set of currently executable operations. As the number of unfinished jobs decreases, the size of the action space A_t diminishes. The concept of action and action space within the Markov Decision Process (MDP) for scheduling has been explored in related works such as [29,18]. These studies define actions based on available operations in scheduling problems and analyze the evolution of the action space, which is similar to our approach in the ConDRL-JSP framework.

State Transition: Once the system executes action a_t, it moves from state s_t to s_{t+1} via the transition function T as shown in the equation:

$$s_{t+1} = T(s_t, a_t) \tag{2}$$

This transition involves job allocation and updates to machine and job states, as can be seen in the state-processing part of **Fig. 1**. The state transition description aligns with the common understanding in MDP-based scheduling applications, as discussed in [15,27], where it forms a crucial part of the scheduling model.

Reward and Cost Function: The cost function $C_t(x,s_t,a_t)$ is defined as:

$$C_t(x,s_t,a_t)=\tau(x,s_{t+1})-\tau(x,s_t) \tag{3}$$

which measures the increase in completion time between consecutive states. Treating this cost as the negative of the reward helps apply the "reward-to-go" technique and reduces the variance of the policy gradient. As studied in [17,15], such a cost function design and the use of the "reward-to-go" technique optimize the scheduling policy within the MDP framework.

Policy and Optimization: The policy $(\pi_\theta(a_t/s_t))$ is approximated by a neural network and outputs a probability distribution of actions based on the current state. We use the Reinforce algorithm to optimize the parameter θ with the goal of minimizing the expected completion time.

The objective function is:

$$J^{\pi}(\theta)=E_{x\sim X,a\sim\pi_\theta(\cdot|x,s_t)}[T^{\pi_\theta}(x)] \tag{4}$$

where

$$T^{\pi_\theta}(x)=\sum_t C_t(x,s_t,a_t) \tag{5}$$

The gradient of the objective function is:

$$\nabla_\theta\widehat{J^{\pi}}(\theta)\approx\frac{1}{B}\sum_b \sum_{t=1}^{n\cdot m} ((G(x,s_t)-b_\phi(x,s_t))\cdot\nabla_\theta log\pi_\theta(a_t|x,s_t)) \tag{6}$$

The critic network reduces the gradient variance by calculating the baseline $b_\phi(x,s_t)$, and its parameter ϕ is optimized by minimizing the equation:

$$L(\phi)=\frac{1}{B}\sum\sum_{t=1}^{n\cdot m} \|b_\phi(x,s_t)-G(x,s_t)\|^2 \tag{7}$$

The use of the Reinforce algorithm for policy optimization and the role of the critic network in reducing gradient variance are common techniques in reinforcement learning for scheduling problem [3,17], and are implemented in our ConDRL-JSP model as shown in **Algorithm 1**.

2.4 ConDRL-JSP Basic Network Architecture

In the following sections, we'll explain the key components of the ConDRL-JSP framework (**Fig. 1**). The State Representation Module captures problem states for further operations. The Shared Feature Layer extracts and shares features for information exchange. The Contrastive Learning Module refines state representations via contrastive learning to boost model performance. The Curriculum Learning Module enables the model to adapt to complex problems during training. The Loss Function integrates

these components, optimizing the training process for efficient and accurate learning.

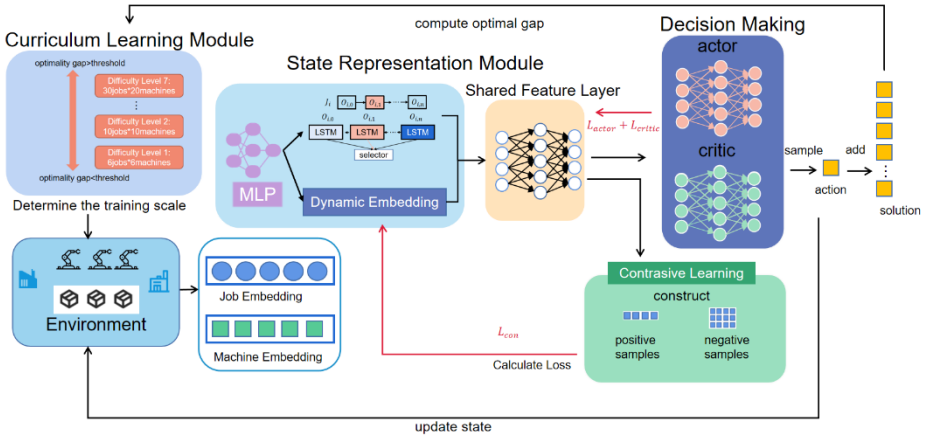

Fig. 1. ConDRL-JSP network architecture

State Representation Module The State Representation Module preprocesses job and machine states from the Environment (ENV) and comprises two components. For the Machine Embedding component, a linear projection is utilized to transform raw machine states, as described by the formula:

$$h_m = W_m s_t + b_m \tag{8}$$

Regarding the Operation Sequence Embedding with LSTM, the LSTM is employed to model the temporal relationships between operations, which can be expressed as:

$$h_t^{op} = LSTM\left(h_{t-1}^{op}, e\left(O_{i,t}\right)\right) \tag{9}$$

These features derived from the two components are then fed into the shared feature layer ϕ, resulting in the formation of a comprehensive state representation, as illustrated in **Fig. 1**.

Shared Feature Layer The core of the Shared Feature Layer is the component $\phi: S \rightarrow R^d$,, which integrates heterogeneous features $\{h_i\}_{i=1}^n$. The integration process is achieved through the following formula:

$$\phi(s) = \sigma(W_c[h_1 \oplus \cdots \oplus h_n] + b_c) \tag{10}$$

where $W_c \in R^{d \times \sum dim(h_i)}$ is a learnable parameter, $\sigma(\cdot)$ represents the activation function, and $\oplus$ denotes concatenation. The output of the shared feature layer, namely the state embedding s_{embed}, is obtained as follows:

$$s_{embed} = \phi(s) \tag{11}$$

This state embedding s_{embed} serves a dual purpose, being passed to both the Actor and Critic networks, and also acting as the input for the contrastive learning module.

Decision Making Module The Decision Making Module is mainly composed of the Actor and Critic networks. The Actor network generates a probability distribution over scheduling actions via the policy function $\pi(a|\phi(s))$. Given the state representation $\phi(s) \in R^d$ from the shared feature layer, the Actor first computes unnormalized action scores using the formula:

$$f\left(a\middle|\phi(s)\right)=\boldsymbol{W}_2 ReLU(\boldsymbol{W}_1\phi(s)+\boldsymbol{b}_1)+\boldsymbol{b}_2 \tag{12}$$

Here, $\boldsymbol{W}_1 \in R^{h\times d}$ and $\boldsymbol{W}_2 \in R^{A\times h}$ are learnable parameters, with h denoting the hidden layer dimension and A representing the action space size. To handle the constraints of the JSP, a binary mask vector $\boldsymbol{m} \in \{0,1\}^A$ is introduced to invalidate illegal actions. Consequently, the probability distribution is calculated as:

$$\pi\left(a\middle|\phi(\boldsymbol{s})\right)=\frac{m(a)\cdot exp\left(f\left(a\middle|\phi(\boldsymbol{s})\right)\right)}{\sum_{a'} m(a')\cdot exp\left(f\left(a'\middle|\phi(\boldsymbol{s})\right)\right)} \tag{13}$$

As depicted in **Fig. 1**, the Actor samples specific actions through a "sample" operation. Subsequently, the environment executes these actions and updates its state. This iterative process continues until all operations are scheduled, ultimately yielding a complete JSP solution.

The Critic network, conversely, evaluates the state quality using the value function $V\left(\phi(\boldsymbol{s})\right)$, which is computed as:

$$V\left(\phi(\boldsymbol{s})\right)=\boldsymbol{w}_2^T ReLU(\boldsymbol{w}_1^T\phi(\boldsymbol{s})+c_1)+c_2 \tag{14}$$

where $\boldsymbol{w}_1 \in R^{h\times d}$ and $\boldsymbol{w}_2 \in R^h$ are parameters specific to the Critic. The output of the Critic is then used to compute the advantage function:

$$A(s,a)=R(s,a)+\gamma V\left(\phi(s')\right)-V\left(\phi(s)\right) \tag{15}$$

This advantage function signal guides the parameter updates of the Actor network via the policy gradient theorem:

$$\nabla_\theta J(\theta)=E\left[\nabla_\theta log\,\pi\left(a\middle|\phi(s)\right)\cdot A(s,a)\right] \tag{16}$$

Contrastive Learning Module As a key part, the contrastive learning module boosts scheduling decision performance. Integrated into each action-selection step of the scheduling process (**Algorithm 1**), it optimizes state representation via well-designed procedures, supporting scheduling decisions.

The contrastive learning module takes as input the state embedding s_{embed} from the state representation and shared feature layers, which describes the current scheduling

state. When the Actor network does forward propagation for action probability distribution, contrastive features are generated. s_{embed} is fed into the contrastive projection head (a series of linear layers and activation functions), outputting the contrastive feature c_{feat}, as per the formula below:

$$c_{feat}=P(s_{embed}) \tag{17}$$

where $P(\cdot)$ represents the mapping operation of the contrastive projection head. As demonstrated in **Algorithm 1**, these generated contrastive features form the basis for constructing positive and negative samples and calculating the contrastive loss.

Algorithm 1 ConDRL-JSP Single Level Training Algorithm

Require: learning rates α,β, contrastive loss weight λ, temperature hyperparameter τ = 0.1, difficulty level l

Initialize policy network π_θ, shared feature extractor ϕ
for each training batch **do**
  for each time step t **do**
    Sample action a_t from $\pi_\theta(a|s_t)$
    Execute action a_t in state s_t to get s_{t+1} and r_t
    Store (s_t,a_t,r_t,s_{t+1})
  end for
  Calculate cumulative discounted return $R_t=\sum_{k=t}^{T}\gamma^{k-t}r_k$
  Calculate target advantage A^{target}
  Calculate actor loss: $L_{actor}=-\frac{1}{N}\sum log\pi_\theta(a|s)\cdot A^{target}$
  for each sample s_i in the batch **do**
    Set s_i as positive sample and other samples as negative samples
  end for
  Calculate similarity matrix $S_{ij}=\frac{cos(\phi(s_i),\phi(s_j))}{\tau}$
  Calculate contrastive loss: L_{con} (see Equation 19)
  Calculate total loss: $L_{total}=L_{actor}+L_{critic}+\lambda\cdot L_{con}$
  Update parameters: $\theta,\phi\leftarrow\theta-\alpha\nabla L_{total}$
end for

Positive and negative samples are constructed based on the contrastive features at different time steps. In each scheduling cycle, contrastive features c_{feat} at multiple time steps are collected. Each batch of contrastive features cf_{batch} is reshaped and normalized to obtain cf_{norm}. Subsequently, the contrastive relationship is established by calculating the cosine similarity matrix *sim_matrix* among different samples. If the i-th and j-th feature vectors in cf_{norm} are z_i and z_j respectively, then the cosine similarity is:

$$sim(z_i, z_j) = \frac{z_i \cdot z_j}{\|z_i\| \|z_j\|} \tag{18}$$

In the constructed similarity matrix *sim_matrix*, the diagonal elements (z_i, z_i) are positive sample pairs, corresponding to the features of the same state at different time steps. The off-diagonal elements (z_i, z_j) $(i \neq j)$ constitute negative sample pairs. Just as in **Algorithm 1**, this method of sample construction helps the model learn the differences between various states, thus optimizing the state representation.

The contrastive learning loss adopts the calculation method of the INFONCE loss. By utilizing the constructed similarity matrix *sim_matrix*, the cross-entropy loss function is applied for calculation. Let N be the number of samples, and the labels $labels$ be the index sequence ranging from 0 to $N - 1$. The calculation formula for the contrastive learning loss L_{con} is:

$$L_{con} = -\frac{1}{N} \sum_{i=0}^{N-1} log \frac{exp(sin(z_i, z_i)/\tau)}{\sum_{j=0}^{N-1} exp(sin(z_i, z_j)/\tau)} \tag{19}$$

where τ is the temperature parameter, which is used to adjust the sharpness of the softmax function and control the intensity of contrastive learning. By minimizing L_{con}, the model continuously optimizes the parameters of the contrastive projection head, making the contrastive features c_{feat} more discriminative and enhancing the quality of the state representation. This is also an integral part of the overall parameter update process in **Algorithm 1**.

Algorithm 2 ConDRL-JSP Curriculum Learning strategy

Require: Set of difficulty levels $L=\{l_0, \ldots, l_K\}$, optimal gap threshold δ_{th} Initialize the policy network π_{θ}.

Set $k \leftarrow 0$ and the current difficulty level $l \leftarrow l_0$
while $k \leq K$ **do**
  Update $\{g(l) | l \in L\}$ (based on the optimal gap from the test set)
  if $max(g(l)) \leq \delta_{th}$ **then**
    $k \leftarrow k+1, L \leftarrow L \cup \{l_k\}$
  end if
  Sample l from the distribution $softmax(1/g(l))$
  Call ConDRL-JSP Single Level Training Algorithm with difficulty level l
end while
return π_{θ}

Curriculum Learning Strategy The ConDRL-JSP framework employs a Reinforcement Adaptive Stepped Curriculum Learning (RASCL)[14] strategy to tackle the Job Shop Scheduling Problem (JSP), as illustrated in **Algorithm 2**. RASCL is designed to

manage difficulty levels dynamically and progressively introduce more complex problem instances as the model's performance improves.

The difficulty levels are defined in a set $L=\{l_0,...,l_K\}$, where each level corresponds to a different problem size or complexity. Training begins at the easiest level l_0 and progresses as the model's performance improves. The optimal gap threshold δ_{th} is used to assess when the model has sufficiently mastered the current difficulty level. If the model reaches this threshold, it moves on to a more challenging level.

The curriculum learning strategy helps the model learn from simple to complex tasks, ensuring that the training process is both efficient and balanced. By using this approach, the model avoids local optima and becomes more capable of solving larger and more complex instances of the JSP.

Joint Loss Function The joint loss function integrates the requirements of policy optimization, value evaluation, and feature contrast. It is composed of three main components:

Policy Gradient Loss($\mathrm{L}_{\mathbf{actor}}$*)* This component maximizes the cumulative discounted return R_t to optimize the scheduling strategy. The actor loss is computed as:

$$L_{actor}=-\frac{1}{N}\sum log\,\pi_\theta(a|s)\cdot A^{target} \tag{20}$$

where A^{target} is the target advantage.

Value Evaluation Loss($\mathrm{L}_{\mathbf{critic}}$*)* The critic network is optimized to minimize the squared difference between the estimated and target advantages. The loss function is:

$$L_{critic}=\frac{1}{B}\sum_{b}\sum_{t=1}^{n\cdot m}\|b_\phi(x,s_t)-G(x,s_t)\|^2 \tag{21}$$

Contrastive Loss($\mathrm{L}_{\mathbf{con}}$*)* The contrastive loss is computed using the InfoNCE loss function, which helps improve the state representation by learning from positive and negative samples. The contrastive loss formula is shown in Equation 19, which was presented in the previous description of the contrastive learning module.

The total loss function is a combination of these three components:

$$L_{total}=L_{actor}+L_{critic}+\lambda L_{con} \tag{22}$$

The weight λ adjusts the contribution of the contrastive loss, enabling the model to optimize for both scheduling quality and generalization performanceIntroduction

3 Experiments

3.1 Dataset and Setting

In this study, multiple datasets are used to comprehensively evaluate the performance of the model. The synthetic dataset is selected as the training set, covering the scales from 6×6 to 30×20 (number of jobs $\times$ number of machines), which are used for the training and optimization of the model. Two standard datasets are adopted as the test set:

Taillard benchmark It contains 80 classic instances, with the scale ranging from 15×15 to 100×20. This benchmark has been widely used in the literature to evaluate job-shop scheduling algorithm [24], providing a reliable basis for comparing the performance of different models across various scales.

Demirkol benchmark It contains 80 instances, with the scale ranging from 10×10 to 30×20. Similar to the Taillard benchmark, the Demirkol benchmark [8] is also commonly employed in job-shop scheduling research to test the generalization ability of models for problems of different sizes.

These instances are widely used in the field of job-shop scheduling and can effectively test the generalization ability of the model for problems of different scales.

The core task of the experiment is to optimize the job-shop scheduling scheme to minimize the average Makespan. This indicator measures the longest time from the start to the completion of all jobs and is a key criterion for evaluating the quality of the scheduling strategy. To accurately evaluate the gap between the model's solution and the optimal solution, the average optimality gap(GAP) is used as an auxiliary indicator, and its definition is:

$$Gap=\left(\frac{Makespan_{model}}{Makespan_{opt}}-1\right)\times 100\% \tag{23}$$

where $\mathrm{Makespan}_{\mathrm{opt}}$ is the optimal solution or the best known solution generated by Google OR-Tools (with the time limit set to 1800 seconds). The smaller the Gap value is, the closer the model's solution is to the theoretical optimal solution.

3.2 Baseline

To verify the effectiveness of the proposed method, the following comparison methods are selected:

RASCL[14]: A baseline method that adopts a reinforcement adaptive curriculum learning strategy. It improves the robustness of the model by dynamically adjusting the difficulty level and performs well in JSP benchmark tests. In this experiment, the results of the RASCL method are reproduced based on its paper, and in the Taillard and Demirkol benchmark tests, the sampling inference strategy proposed in the paper**[14]** is used to ensure the consistency and fairness of the comparison.

L2D[27]: An end-to-end deep reinforcement learning method proposed by **[27]**. It models JSP as a disjunctive graph, uses a graph isomorphism network (GIN) to capture

dynamic states, and learns the scheduling strategy through policy gradient optimization. In this experiment, the results in its paper are directly used for comparison.

Traditional Heuristic Methods[12]:

Shortest Processing Time (SPT) prioritizes jobs with the shortest processing times to improve system throughput and reduce overall makespan.Maximum Work Remaining (MWKR) focuses on jobs with the most remaining work to prevent bottlenecks and ensure steady progress. Fast Descent/Dynamic Weighted Round-Robin (FDD/WKR) makes greedy decisions and uses weighted rotation to adapt to different scheduling situations. Minimum Operation Processing Number Rule (MOPNR) centers on jobs with the fewest operations to simplify scheduling and enhance resource efficiency.

Exact Solver: The CP-SAT model of Google OR-Tools is used as the exact solver, with a time limit of 1800 seconds, to generate the optimal solution as the benchmark.

3.3 Scheduling Quality Comparison

Tables 2 and 3 show performance comparisons on Taillard and Demirkol benchmarks.Objective indicates the average makespan for a given problem size; and Gap, the average difference (in percent) to the upper bound known for the instances.

Table 2. Results on Taillard's instances

Instance		SPT	FDD/WKR	MWKR	MOPNR	L2D	RASCL	**ConDRL**
15 * 15	Obj.	1546.1	1808.6	1464.3	1481.3	1547.4	1389.5	**1346.1**
	Gap	25.81%	47.15%	19.15%	20.53%	25.92%	13.07%	**9.54%**
20 * 15	Obj.	1813.5	2054	1683.6	1686.7	1774.7	1575.8	**1533.5**
	Gap	32.87%	50.57%	23.35%	23.55%	30.02%	15.45%	**12.35%**
20 * 20	Obj.	2049.1	2387.2	1969.8	1968.3	2128.1	1860.2	**1793**
	Gap	26.70%	47.61%	21.81%	21.71%	31.58%	15.02%	**10.86%**
30 * 15	Obj.	2419.3	2590.8	2214.8	2195.8	2378.8	2127.1	**2044.3**
	Gap	35.15%	45.02%	23.91%	22.83%	32.89%	18.83%	**14.20%**
30 * 20	Obj.	2619.1	3045	2439	2433.6	2603.9	2370.4	**2269**
	Gap	34.43%	56.30%	25.17%	24.94%	33.65%	21.67%	**16.46%**
50 * 15	Obj.	3441	3736.3	3240	3254.5	3393.8	3176.7	**3057.6**
	Gap	24.05%	34.77%	16.86%	17.37%	22.35%	14.53%	**10.23%**
50 * 20	Obj.	3569.8	4022.1	3352.8	3346.9	3593.9	3266.6	**3150.5**
	Gap	25.52%	41.50%	17.95%	17.68%	26.37%	14.86%	**10.78%**
100 * 20	Obj.	6139	6620.7	5812.2	5856.9	6097.6	5794.1	**5666.5**
	Gap	14.41%	23.39%	8.31%	9.15%	13.64%	7.98%	**5.61%**

A comprehensive analysis of **Table 2** and **Table 3** clearly demonstrates that the ConDRL method significantly outperforms other methods across diverse instances. In both the Taillard and Demirkol benchmarks, across various combinations of jobs and machines, the ConDRL method consistently yields lower average makespan values, as reflected by the objective values presented in the tables.

Regarding solution accuracy, the ConDRL method showcases a smaller average optimality gap (Gap) in most instances. For instance, in the "100×20" instance from the Taillard benchmark and the "20×20" instance from the Demirkol benchmark, ConDRL's solutions closely approach the theoretical optimum, highlighting its ability to achieve optimized resource allocation with high precision.

The instances covered in these two tables span a wide range of scales, and throughout, ConDRL demonstrates consistent and excellent performance, underscoring its broad applicability. This adaptability to different production scenarios indicates that ConDRL is not confined to specific problem types. Moreover, regardless of the instance size, ConDRL consistently maintains a clear edge over other methods in terms of both makespan and relative optimal gap. Its stable performance and consistent superiority across multiple metrics collectively showcase its outstanding comprehensive advantages.

Table 3. Results on DMU's instances

Instance		SPT	FDD/WKR	MWKR	MOPNR	L2D	RASCL	**ConDRL**
20 * 15	Obj.	4951.5	4666.3	4909.9	4513.2	4215.3	3723	**3467.4**
	Gap	63.65%	54.22%	62.27%	49.16%	39.32%	23.05%	**14.60%**
20 * 20	Obj.	5960.5	5298.2	5489	5052.3	4804.5	4302.8	**3966.8**
	Gap	71.62%	52.55%	58.05%	45.47%	38.34%	23.89%	**14.22%**
20 * 20	Obj.	6306.2	6016.5	6252.9	5742.8	5557.9	5050.7	**4804.5**
	Gap	62.33%	54.87%	60.96%	47.83%	43.07%	30.01%	**23.67%**
30 * 15	Obj.	7036	6827.3	6925	6491.9	5967.4	5510.9	**5101.4**
	Gap	65.28%	60.38%	62.67%	52.50%	40.18%	29.46%	**19.84%**
30 * 20	Obj.	7601.2	7420	7484.2	7105.5	6663.9	6220.6	**5673.2**
	Gap	55.80%	52.09%	53.40%	45.64%	36.59%	27.50%	**16.28%**
40 * 15	Obj.	8538.1	8210.9	8460.9	7870.7	7375.8	6875.5	**7228.3**
	Gap	62.60%	56.37%	61.13%	49.89%	40.46%	30.93%	**37.65%**
40 * 20	Obj.	8975.4	9150.2	8906	8436.5	8179.4	7655.6	**7186.3**
	Gap	50.57%	53.50%	49.40%	41.53%	37.21%	28.43%	**20.55%**
50 * 20	Obj.	10132.8	9899.6	9807	9408	8751.6	8340.8	**7753**
	Gap	61.94%	58.22%	56.74%	50.36%	39.87%	33.30%	**23.91%**

3.4 Ablation Studies

To verify the effectiveness of the contrastive learning module in the ConDRL-JSP framework, we carried out ablation experiments. Under the curriculum learning setting, we compared the performance of the full ConDRL-JSP model (with the contrastive learning module) and a baseline model, which was based on the RASCL architecture but lacked the contrastive learning module. The experiment monitored the changes in the average makespan of the models on medium-scale (30×15, 30×20) and unseen large-scale (50×15, 50×20) instances every 100 training rounds.

We applied exponential moving average (EMA) with a smoothing coefficient ($\alpha = 0.9$) to the raw makespan data to observe model performance trends during training. For both RASCL and ConDRL, we plotted two curves: an unsmoothed curve showing real-time performance fluctuations and a smoothed curve to highlight long-term trends. This EMA processing reduced training noise, allowing for an intuitive comparison of the stability and effectiveness of different scheduling optimization methods.

Analysis of Results on Medium-Scale Instances The changes in the average makespan of the two models with the number of training rounds on medium-scale instances (30×15 and 30×20) are shown in **Fig. 2**.

In the early training stage, all models' average makespan decreased significantly, but method differences were minor. This is because models focused on learning basic features and scheduling patterns, and the advantages of modules like contrastive learning weren't fully realized.As training rounds increased, the smoothed ConDRL method gradually showed outstanding advantages. In the 30×15 instance, the downward trend of ConDRL (Smoothed)'s average makespan became more stable, and the gap with RASCL (Smoothed) widened, proving ConDRL's superiority in optimizing scheduling strategies and reducing the average makespan.

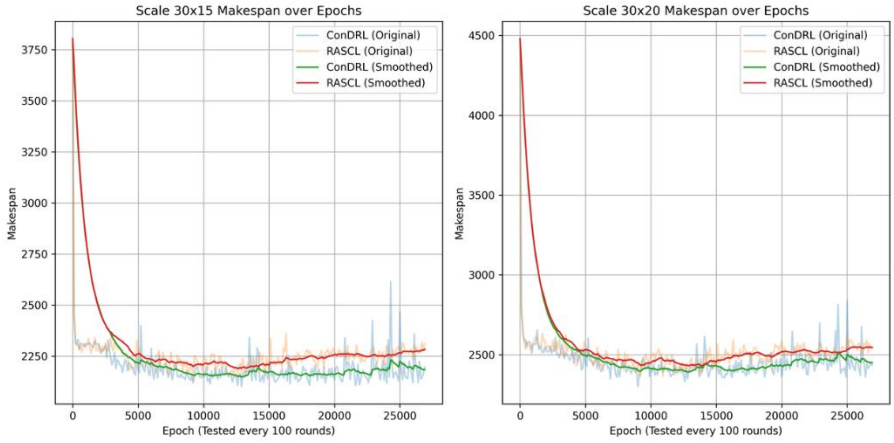

Fig. 2. Makespan comparison on medium-scale instances

In the 30×20 instance, after 5000 training rounds, ConDRL (Smoothed) showed a significantly more pronounced decreasing trend in the average makespan compared to RASCL (Smoothed), fully demonstrating that the ConDRL method can more effectively optimize scheduling schemes and reduce the average makespan when dealing with larger-scale scheduling problems.

This indicates that, although RASCL (Smoothed) performs well in some instances, the ConDRL (Smoothed) method consistently exhibits more stable and superior scheduling capabilities across all instances. With the introduction of the contrastive learning module, ConDRL is able to more effectively learn the feature differences between different jobs and machines, thereby generating better scheduling plans, especially show-

ing higher performance in medium-scale and large-scale scheduling problems. By enhancing the discriminative ability of state features, ConDRL is better able to capture key information in job-shop scheduling, make more reasonable scheduling decisions, and ultimately reduce the average makespan, demonstrating its significant advantage in scheduling optimization tasks.

Analysis of Results on Large-Scale Instances For the unseen large-scale instances (50×15 and 50×20), the performance differences between the two models were more significant, as shown in **Fig. 3**.

For the large-scale instances (50×15 and 50×20), we observe that under the RASCL curriculum learning strategy, ConDRL (Smoothed), integrated with the contrastive learning module, demonstrates a significant advantage over RASCL (Smoothed).

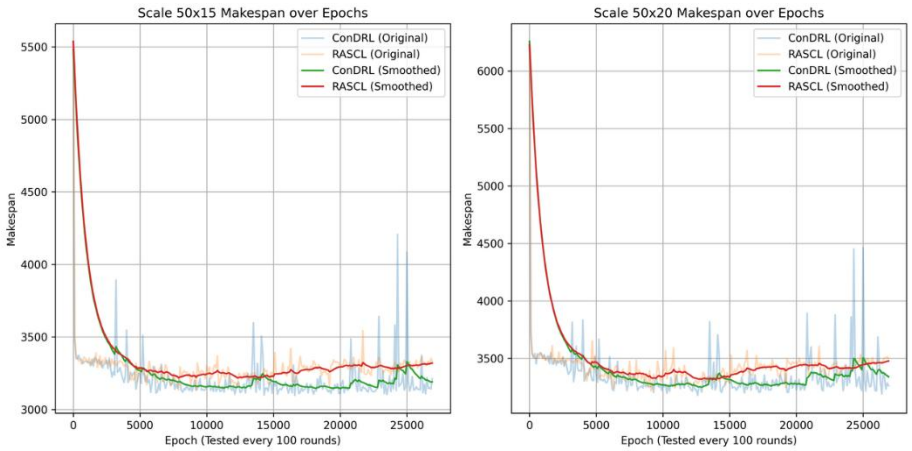

Fig. 3. Makespan comparison on large-scale instances

In the early stages of training, the average makespan of all models shows a significant downward trend. However, as training progresses, RASCL encounters difficulties when faced with large-scale problems. The model's ability to capture complex scheduling patterns gradually weakens, and as the complexity of the problem increases, the rate of decrease in the average makespan slows down, with some fluctuations indicating that the models are getting stuck in local optima. In contrast, ConDRL, aided by the contrastive learning module, continuously improves its performance and becomes more effective in distinguishing subtle differences between state features. By leveraging contrastive learning, ConDRL enhances the ability to learn discriminative representations between jobs and machines, resulting in more refined scheduling solutions.

The contrastive learning module enables ConDRL to focus on learning invariant features that remain consistent across different instances, which helps the model generalize better to unseen data. As a result, ConDRL (Smoothed) and ConDRL (Original) demonstrate a more stable and sustained decrease in the average makespan across the 50×15 and 50×20 instances, surpassing the results of both RASCL (Smoothed) and RASCL

(Original). Contrastive learning enhances the model's ability to focus on relevant features while ignoring noise or irrelevant information, ensuring that even as the problem complexity increases, ConDRL can continue to improve its scheduling efficiency.

This indicates that, under the RASCL curriculum learning strategy, ConDRL integrated with the contrastive learning module is better suited to tackle large-scale scheduling problems. By generating more discriminative features, ConDRL can optimize scheduling plans more effectively, reduce the average makespan, and demonstrate superior learning and generalization capabilities in complex large-scale scenarios.Introduction

4 Conclusion

This research introduced the ConDRL-JSP framework to address the NP-hard JSP in manufacturing. The framework adopted a RASCL-inspired curriculum learning strategy during training, which improved generalization across different-scale JSP instances.Experiments on synthetic and Taillard & Demirkol benchmark datasets showed the framework's superiority. It achieved lower average makespan and closer-to-optimal solutions across scales, with strong generalization. Ablation experiments validated the contrastive learning module's importance, especially in medium and large-scale problems.Future work will adapt the framework to more complex real-world manufacturing scenarios and integrate emerging techniques like advanced graph neural networks or refined self-supervised learning to enhance performance and generalization.

References

1. Balas, E.: An additive algorithm for solving linear programs with zero - one variables. Operations Research 13.4, 517–546 (1965)
2. Balas, E.: Machine sequencing via disjunctive graphs: An implicit enumeration algorithm. Operations Research 17.6, 941–957 (1969)
3. Bello, I., et al.: Neural combinatorial optimization with reinforcement learning. arXiv preprint, arXiv:1611.09940 (2016)
4. Blazewicz, J., Pesch, E., Sterna, M.: The disjunctive graph machine representation of the job shop scheduling problem. European Journal of Operational Research 127.2, 317–331 (2000)
5. Chen, R., Li, W., Yang, H.: A deep reinforcement learning framework based on an attention mechanism and disjunctive graph embedding for the job - shop scheduling problem. IEEE Transactions on Industrial Informatics 19.2, 1322–1331 (2022)
6. Chen, X., Tian, Y.: Learning to perform local rewriting for combinatorial optimization. Advances in neural information processing systems 32 (2019)
7. Corsini, A., Calderara, S., Dell'Amico, M.: Learning the Quality of Machine Permutations in Job Shop Scheduling. IEEE Access 10, 99384–99396 (2022)
8. Demirkol, E., Mehta, S., Uzsoy, R.: Benchmarks for shop scheduling problems. European Journal of Operational Research 109.1, 137–141 (1998)
9. Falkner, J. K., et al.: Learning to control local search for combinatorial optimization. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer, 2022, pp. 361–376

10. Glover, F., Laguna, M.: Tabu search. Springer, 1998
11. Han, B. - A., Yang, J. - J.: Research on adaptive job scheduling problems based on dueling double DQN. Ieee Access 8, 186474–186495 (2020)
12. Haupt, R.: A survey of priority rule - based scheduling. Operations Research Spektrum 11.1, 3–16 (1989)
13. Hottung, A., Kwon, Y. - D., Tierney, K.: Efficient active search for combinatorial optimization problems. arXiv preprint arXiv:2106.05126 (2021)
14. Iklassov, Z., et al.: On the study of curriculum learning for inferring dispatching policies on the job shop scheduling. Proceedings of the Thirty - Second International Joint Conference on Artificial Intelligence (IJCAI - 23), 2023
15. Kool, W., van Hoof, H., Welling, M.: Attention, learn to solve routing problems!. arXiv preprint, arXiv:1803.08475 (2018)
16. Kwon, Y. - D., et al.: POMO: Policy optimization with Multiple Optima for Reinforcement Learning. 34th Conference on Neural Information Processing Systems (NeurIPS 2020), 2020
17. Liu, C. - L., Chang, C. - C., Tseng, C. - J.: Actor - critic deep reinforcement learning for solving job shop scheduling problems. IEEE Access 8, 71752–71762 (2020)
18. Nazari, M., et al.: Reinforcement learning for solving the vehicle routing problem. In: Advances in Neural Information Processing Systems. Vol. 31, 2018, pp. 9839–9849
19. Nowak, A., et al.: A note on learning algorithms for quadratic assignment with graph neural networks. STAT 1050, 22 (2017)
20. Nowicki, E., Smutnicki, C.: A Fast Taboo Search Algorithm for the Job Shop Problem. Management Science 42, 1185–1199 (1996)
21. Nowicki, E., Smutnicki, C.: An Advanced Tabu Search Algorithm for the Job Shop Problem. Journal of Scheduling 8, 149–157 (2005)
22. van Oord, A., Li, Y., Vinyals, O.: Representation learning with contrastive predictive coding. arXiv preprint, arXiv:1807.03748 (2018)
23. Rinnoy Kan, A. H. G.: Machine scheduling problems: classification, complexity and computations. Springer Science Business Media, 2012
24. Taillard, E.: Benchmarks for basic scheduling problems. European Journal of Operational Research 64.2, 278–285 (1993)
25. Tassel, P., Gebser, M., Schekotihin, K.: A Reinforcement Learning Environment For Job - Shop Scheduling. arXiv preprint, arXiv:2104.03760 (2021)
26. Vinyals, O., Fortunato, M., Jaitly, N.: Pointer networks. In: Advances in Neural Information Processing Systems 28, 2015, pp. 2692–2700
27. Zhang, C., et al.: Learning to dispatch for job shop scheduling via deep reinforcement learning. In: Advances in Neural Information Processing Systems. Vol. 33, 2020, pp. 1621–1632
28. Zhang, C., et al.: Deep reinforcement learning guided improvement heuristic for job shop scheduling. arXiv preprint arXiv:2211.10936 (2022)
29. Zhang, W., Dietterich, T. G.: A reinforcement learning approach to job - shop scheduling. Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI - 95), 1995, pp. 1114–1120
30. Zhang, Z., et al.: Learning to Solve Travelling Salesman Problem with Hardness - Adaptive Curriculum. The Thirty - Sixth AAAI Conference on Artificial Intelligence (AAAI - 22), 2022