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

Learning Flexible Job Shop Scheduling with Bidirectional Cross-Attention Network via Deep Reinforcement Learning

Xiongxin Zha[1], Lisha Dong[2], Muhammad Sadiq[2], Qingling Zhu[1] (✉)

[1] School of Artificial Intelligence, Shenzhen University, Shenzhen, Guangdong, 518060, China
[2] Shenzhen institute of information technology, Shenzhen, Guangdong, 518060, China
2400671023@mails.szu.edu.cn, sadiq@sziit.edu.cn, zhuqingling@szu.edu.cn

Abstract. Neural combinatorial optimization (NCO) for solving scheduling problems has attracted more and more research interest because they do not rely on expert knowledge. However, existing NCO approaches face significant challenges in the flexible job shop scheduling problem (FJSP), because neural networks struggle to effectively capture the heterogeneous interactions among multiple machines and operations. To address this issue, we propose a bidirectional cross attention neural architecture trained by deep reinforcement learning. Our approach introduces a dual interaction mechanism to enable simultaneous learning of operation priorities and machine availability constraints. We demonstrate the effectiveness of this approach through extensive experiments, showing its superiority over classical network architectures on synthetic datasets.

Keywords: Deep Reinforcement Learning, Flexible Job Shop Scheduling Problem, Neural combinatorial optimization, Cross-Attention.

1 Introduction

The flexible job shop problem (FJSP) is a fundamental problem in operations research and industrial engineering [1]. The goal of FJSP is to find the optimal operations ordering and machine assignments to minimize an objective function such as makespan. The NP-hard nature of FJSP makes it extremely difficult to solve. Traditional solve methods usually rely on exact algorithms, heuristics and metaheuristics based on decades of research. For example, exact algorithms, including cutting plane and branch-and-bound, can find the optimal solution for the FJSP. However, the exponential time complexity of these algorithms makes it challenging to apply them to large-scale practical problems [2]. On the other hand, (meta)heuristic algorithms, like genetic algorithms and tabu searches, are capable of yielding high-quality solutions within a limited timeframe. Nevertheless, they do not ensure optimality [3,4]. Moreover, the development of efficient (meta)heuristic algorithms hinges on expert experience and domain knowledge [5,6,7].

In recent years, neural combinatorial optimization (NCO) methods have attracted growing attention. This is because they address the drawbacks of traditional methods and enable the learning of a general scheduling strategy [8]. Typically, these methods entail designing a sophisticated network architecture to represent a scheduling strategy. The policy network is commonly trained through either supervised learning or reinforcement learning. The supervised - learning approach utilizes numerous FJSP problem instances along with their optimal solutions as training data [9]. This method is simple and intuitive, however, the time cost of obtaining optimal solutions is substantial, particularly when dealing with large-scale instances. On the other hand, reinforcement learning-based methods formulate the FJSP as a Markov decision-making process, in which rewards are rationally devised to align with the objective function [10]. In contrast to supervised learning-based methods, reinforcement learning does not necessitate optimal labels. Nonetheless, its performance is constrained by the effectiveness of state representation [11]. Motivated by the fact that the job scheduling problem can be depicted as a disjunctive graph, a state representation method grounded in graph neural networks learns the partial solution of FJSP by representing it as such a disjunctive graph [12]. Nevertheless, the disjunctive graph fails to adequately capture the multi-associative relationships among operations and machines in FJSP, often rendering the redesign of the disjunctive graph based on domain knowledge [13]. Another state representation approach involves learning operations and machines as sequences without positional embeddings, leveraging the attention mechanism [14]. Although the attention-based representation method has yielded promising results thus far, it still lacks an efficient means of learning the heterogeneous relationships between operations and machines.

To address this issue, our contribution is to propose a bidirectional cross-attention neural architecture. It consists of two novel components: (1) operation-to-machine attention that dynamically identifies suitable machines for pending operations, and (2) machine-to-operation attention that evaluates workload compatibility from the machine's perspective. This network is trained by the Actor-Critic algorithm. Finally, we demonstrate the effectiveness of this network architecture by conducting comprehensive comparative experiments with traditional methods as well as classical NCO methods on synthetic datasets.

The rest of the paper is organized as follows. Section II describes the formal description of FJSP and reviews some NCO works related to FJSP. Section III details our proposed methodology. Section IV reports the experimental results. Section V summarizes the paper.

2 Background and Related Work

2.1 Flexible Job Shop Scheduling Problem

The FJSP involves scheduling a set of jobs J, where each job $i \in J$ comprises a sequence of operations O_i. Each operation $j \in O_i$ must be processed on a machine k selected from its eligible machine set M_{ij}, with a processing time p_{ijk}. Decision variables include binary machine assignment x_{ijk}(1 if operation j of job i uses machine k), and timing variables s_{ij}(start time) and c_{ij}(completion time) for each operation. The objective of this problem is to determine the decision variables to minimize the maximum completion time of all jobs, while respecting the following constraints: 1) the operations of job $i \in J$ must be processed in the sequential order defined by O_i; 2) each operation O_{ij} must be assigned to exactly one machine from its compatible set M_{ij}; and 3) no two operations assigned to the same machine k may have overlapping processing time intervals.

2.2 NCO with Graph Neural Networks

The Flexible Job Shop Scheduling Problem (FJSP) has garnered significant attention in recent years, particularly with the integration of advanced computational techniques such as Graph Neural Networks (GNNs) and Deep Reinforcement Learning (DRL). Zhang et al. [15] showed the potential of GNN in the field of scheduling problems by modeling the solution construction process of the job scheduling problem as a disjunctive graph and training it with deep reinforcement learning. Song et al. [16] extended this idea to FJSP, where they added machines as graph nodes to the disjunctive graph to achieve a unified representation of operations and machines. Ho et al. [17] proposed a novel method termed “residual scheduling,” which emphasizes the removal of irrelevant machines and jobs. There are also approaches that consider redesigning the graph structure or employing advanced GNNs. Echeverria et al. [18] designed a new graph structure with jobs, machines, and operations as graph nodes, and designed six types of edges, which significantly enhances the state representation. Wan et al. [19] considered the heterogeneous relationship between operations and machines in FJSP, and used the meta-path-based heterogeneous graph neural network as a state representation network. However, these methods are far from the optimal because the graph structure cannot fully characterize the FJSP.

2.3 NCO with Attention Networks

Due to the remarkable effect of the attention mechanism in routing problems, some studies have recently started to try to use the attention mechanism as a representation method for FJSP [20]. Shao et al. [21] achieved the first joint representation of multi-dimensional spatio-temporal information in a dynamic scheduling environment by designing a multi-channel state matrix (containing operation, machine load, and residual operation features) with a hierarchical structure. On this basis, Xu et al. [22] adopts the Transformer architecture to construct a compact state space, and its innovative dual-attention module establishes dependencies across time steps at the operation level and the machine level respectively, which effectively solves the gradient decay problem in long range scheduling decisions. Notably, the dual attention architecture proposed by Wang et al. [23] achieves decoupled feature learning between the operations and machine streams through the operation message attention block and the machine message attention block. Based on [23], Zhao et al. [24] used a dynamic attention-based feature extraction method to process the attention coefficients of operation nodes and machine nodes separately to better capture the competitive relationship. However, most of the existing attention-based representation methods focus on inter-operations or inter -machines, and have not yet explicitly constructed the dynamic association relationship between operations and machines.

3 Proposed Method

In this section, we present the main research approaches. Firstly, we model the FJSP as a Markov decision-making process (MDP). Secondly, the bidirectional cross-attention network architecture, the actor network and the critic network are described in detail. Finally, deep reinforcement learning strategy is introduced. Fig. 1 illustrates the overall workflow of our method.

3.1 MDP Formulation

To train the network using deep reinforcement learning, we need to formulate the FJSP as a MDP. The following sections describe the five key components of the MDP in detail.

State. Based on the characteristics of the FJSP, we propose a compact state representation consisting of three types of entities: operations, machines, and compatible operation-machine pairs. Given a decision time step t, each operation can be classified into one of three distinct statuses: completed, pending, or unscheduled. Since completed operations are meaningless for current and future decisions, we remove them from s_t. The remaining set of operations we define as the decision-relevant operations $O_u(t)$. Each pertinent operation $O_{ij} \in O_u(t)$ can be represented as a feature vector $h(O_{ij}) \in \mathbb{R}^{10}$. We define $M_u(t)$ as the set of related machines, where each machine $M_k \in M_u(t)$ is described by a feature vector $h(M_k) \in \mathbb{R}^8$. The feature vector $h(O_{ij}, M_k) \in \mathbb{R}^8$ corresponds to each pair of compatible operation-machines $(O_{ij}, M_k) \in A(t)$. The meaning of the specific elements in each feature vector and the formula can be found in [23]. As scheduling proceeds, more and more operations are completed resulting in a decreasing state space.

Action. The action space at a decision time step t is the set of compatible operation-machines pairs $A(t)$, which gets smaller and smaller as sequential decisions are made.

State Transition. After taking an action, each state feature vector is updated accordingly. The updated operations, machines, and compatible operation-machines pair feature vectors represent the new state s_{t+1} and indicate the completion of the current action.

Reward. To align with the optimization objective of FJSP, we use the difference in the partial scheduling maximum time span between s_t and s_{t+1}. We set the discount factor γ to 1 so that the cumulative reward is equal to the negative makespan. Therefore, minimizing the objective function is equivalent to maximizing the cumulative reward.

Policy. The policy $\pi(a_t|s_t)$ computes the probability distribution of the action space $A(t)$ in state s_t. The deep reinforcement learning algorithm continuously optimises the parameters of the bidirectional cross-attention network, the action network and the critic network by sampling and evaluating the actions a_t to maximize the expectation of the cumulative reward.

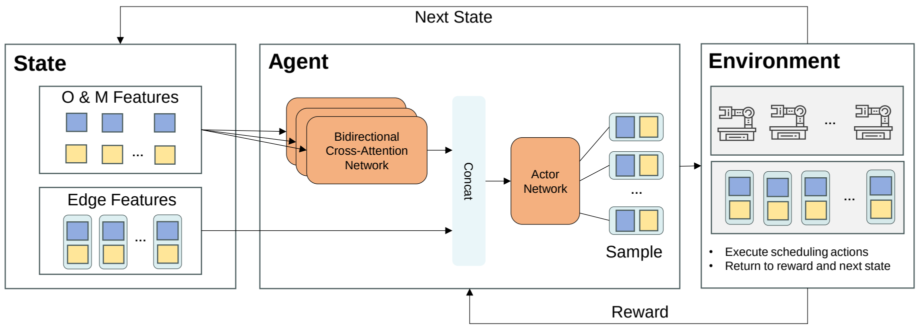
Next State
State
O & M Features
Edge Features
Agent
Bidirectional Cross-Attention Network
Concat
Actor Network
Sample
Environment
Execute scheduling actions
Return to reward and next state
Reward

Fig. 1. The overall workflow of our method.

3.2 Policy Network

State representation learning is crucial for the effectiveness of deep reinforcement learning. In the MDP Formulation subsection, we introduced the set of feature vectors of states, however the underlying representations between them need to be learnt by neural networks. Since the job scheduling problem can be represented as a disjunctive graph and graph neural networks are naturally good at learning graph-structured data, graph neural networks are often used as state representation networks. However, a disjunctive graph can only represent operations and their relationships, and cannot represent the complex heterogeneous interactions between multiple operations and machines in the FJSP. The self-attention mechanism, on the other hand, is able to identify the important elements by learning the interrelationships of the elements within the sequences, and thus it is adopted to characterize the set of operations and machines. Compared to CNNs and MLPs that can only handle fixed sequence lengths, self-attention can handle sequences of different lengths, which is important for the FJSP because real-world scenarios have different scales.

However, self-attention networks have difficulty dealing with the complex connectivity relationships between operations and machines. Unidirectional cross-attention can only capture dependencies from a single perspective, leading to information asymmetry and suboptimal local solutions. For example, operation-to-machine attention can learn an operation's preference for machines but fails to reflect the machine's selection constraints. Moreover, selecting machines solely from the operation's perspective may overlook the machine's global state, causing the scheduling strategy to converge to suboptimal solutions. However, a bidirectional interaction mechanism can simultaneously model two critical aspects: 1) dynamically evaluating an operation's suitability for machines, and 2) assessing operation compatibility from the machine's perspective. This design effectively resolves the information asymmetry and local optimization issues inherent in unidirectional attention methods. Therefore, we propose a model based on a bidirectional cross-attention network.

The general architecture of the model is shown in Fig. 2. It utilizes two attention channels to deal with operations and machines separately. These channels are called the operation feature extractor and the machine feature extractor, respectively. The operation feature extractor processes the operation feature vectors through a stack of self-attention layer and cross-attention layer. In each operation feature extractor pair, we process operations so that 1.) in the first layer, an operation pays attention to each other only with its predecessor and successor, and performs a mask on the attention coefficients of the nonexistent predecessor and successor, and 2.) in the second layer, operations are used as queries, machines are used as keys and values, and each operation computes the attention coefficients only with its machines that can process it.

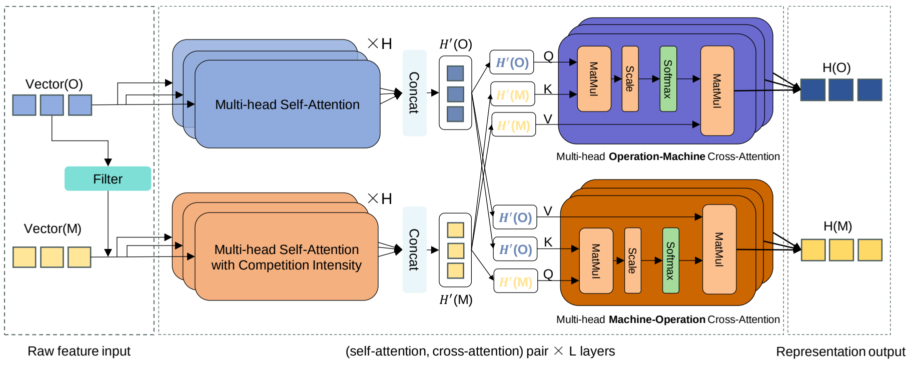
Vector(O)
Vector(M)
Filter
×H
Multi-head Self-Attention
×H
Multi-head Self-Attention with Competition Intensity
Concat
Concat
H′(O)
H′(M)
H′(O)
H′(M)
H′(M)
Q
K
V
H′(O)
H′(O)
H′(M)
V
K
Q
MatMul
Scale
Softmax
MatMul
Multi-head Operation-Machine Cross-Attention
Multi-head Machine-Operation Cross-Attention
H(O)
H(M)
Raw feature input
(self-attention, cross-attention) pair × L layers
Representation output

Fig. 2. The general architecture of Bidirectional Cross-Attention Network. The network takes Vector(O) and Vector(M) as inputs. The outputs of two self-attention modules are cross fed into the "Multi-head Operation-Machine Cross-Attention" and "Multi-head Machine-Operation Cross-Attention" modules respectively. Finally, the network generates two output vectors, H(O) and H(M).

Similarly, a stack of self-attention layer and cross-attention layer is utilized by the machine feature extractor to process the machine feature vectors. In each machine feature extractor pair, we process machines so that 1.) in the first layer, a machine pays attention to each other only with machines that are in a competitive relationship, where competitive relationship means that there is an overlap in the operations that these machines can process, and 2.) in the second layer, machines are used as queries, operations are used as keys and values, and each machine computes the attention coefficients only with operations that it can process. Details about the self-attention layer can be referred to the practice in [23].

Bidirectional Cross-Attention Network. Since self-attention only focuses on the isomorphic relationships between feature vectors within the operation or machine, the goal of cross-attention is to automatically mine the heterogeneous edge relationships between the operation and machine to provide more adequate decision-relevant information for the action network. The self-attention module of the operation feature extractor can be referred to [23]. The inputs to the operation-to-machine cross-attention module are the operation feature matrix $O \in \mathbb{R}^{N\times d_o}$ and the machine feature matrix $M \in \mathbb{R}^{M\times d_m}$ from the self-attention layer, where O and M are the number of operations and machines, respectively, and d_o and d_m are the feature dimensions. The operation features are mapped as query vector Q by query transformation matrix W_Q, and the machine features are mapped as key vector K and value vector V by key matrix W_K and value matrix W_V, respectively: $Q = OW_Q$, $K = MW_K$, $V = MW_V$, where $W_Q \in \mathbb{R}^{d_o\times d_{out}}$, $W_K, W_V \in \mathbb{R}^{d_m\times d_{out}}$, and d_{out} is the output dimension. Then, the scaled dot product attention score matrix $S \in \mathbb{R}^{N\times \mathrm{M}}$ is computed for each operation with each machine and the nonlinearity is enhanced by applying LeakyReLU. The invalid connections are masked by a dynamic mask matrix $A \in \{0,1\}^{\mathrm{N}\times\mathrm{M}}$, the invalid position scores are set to negative infinity, and the attention weights α are subsequently obtained by normalizing each row:

$$\alpha = softmax\left(LeakyReLU\left(\frac{QK^T}{\sqrt{d_{out}}}\right)\right) \tag{1}$$

The normalized attention weight α is weighted and summed with the value vector V to obtain the updated operation feature O':

$$O' = \alpha V \tag{2}$$

To capture multi-dimensional interactions, we extend to a multi-head mechanism. The module contains H independent attention headers, each with independent header parameters. The output of each header is aggregated by concated：

$$MultiHead(O, M) = Concat(head_1, \dots, head_H) \tag{3}$$

where $head_h$ is the output of the h^{th} head. The concat preserves the multi-subspace information and the output dimension is $H \times d_{out}$, which finally introduces nonlinearity through the ELU activation function.

For the machine feature extractor, the framework of the machine-to-operation cross-attention layer is similar to the above module, while the machine self-attention layer employs customized attention coefficients based on the intensity of competition. Specifically, we define the existence of a competitive relationship between a machine M_k and another machine M_q as the existence of an overlapping set Occur_{kq} of operations that can be processed by these two machines. We define the competitive intensity C_{kq} to be equal to the sum of the feature vectors of all the overlapping set Occur_{kq} operations. Finally, the attention coefficients for machine M_k and machine M_q are computed as follows：

$$\mu_{k,q} = a^T LeakyReLU\left(\left[\left(W^1 h_{M_k}\right) \middle\| W^2 h_{M_q} \middle\| \left(W^2 c_{kq}\right)\right]\right) \tag{4}$$

where a^T, W^1, W^2 are all linear layers.

After obtaining the attention coefficients, softmax normalization is applied to $u_{k,q}$ to derive the weights $\alpha_{k,q}$, which quantify the influence of machine M_q on the decision-making of M_k:

$$\alpha_{k,q} = \frac{\exp\left(u_{k,q}\right)}{\sum_{q' \in \mathcal{N}_k} \exp\left(u_{k,q'}\right)} \tag{5}$$

where $\mathcal{N}_k$ represents the set of machines that have competitive relationships with machine M_k.

Weighted aggregation of competing machines' features is performed to generate the new feature h'_{M_k} for machine M_k, which is then used for subsequent scheduling decisions:

$$h'_{M_k} = ELU\left(\sum_{\mathrm{q} \in \mathcal{N}_k} \alpha_{k,q} W^1 \, h_{M_k}\right) \tag{6}$$

Actor network and training. For each feasible action $a_t \in A(t)$, we concat the average pooling of the operation representations, the average pooling of the machine representations, and the edge features and input them to the MLP, outputting the probability of the action:

$$\pi_\theta(a_t | s_t) = softmax\left(MLP_\theta\left(\left[h_{O_{ij}}^{(G)} \middle\| h_{M_k}^{(G)} \middle\| h\left(O_{ij}, M_k\right)\right]\right)\right) \tag{7}$$

The critic network uses MLPs with the same architecture but different parameters. Training is done using the PPO algorithm [25], which starts by initializing the policy and critic networks. Subsequently, trajectories are gathered, and advantages are computed using the critic network. The policy network is updated by maximizing a clipped advantage-weighted probability ratio. Concurrently, the critic network is also updated to enhance its estimation of value functions. Both of these updates occur over several epochs, utilizing the data from the collected trajectories.

4 Experiments

4.1 Datasets

We define FJSP instances with different number of jobs and machines as (n, m). Our experiments use synthetic data generated by [16]. The number of compatible machines for each operation and the processing time are sampled from U(1, m) and U(1, 99), respectively. Depending on whether the number of operations is the same for each operation, we divide the synthetic data into two groups. We define that the number of operations of each job in SD1 is sampled randomly, and the specific sampling method can be found in [16]. On the other hand, the SD2 dataset indicates that the number of operations in each job is the same and equal to the number of machines. For each group, we set up 6 scales, which are (10,5), (20,5), (15,10), (20,10), (30,10), and (40,10). 200 different instances are generated under each scale, of which 100 are used as validation sets and the remaining as test sets. We only trained the models under 4 small scale settings, thus a total of 8 models were trained and the training instances were randomly generated.

4.2 Configuration

We set up lightweight model parameters and training process. In terms of model architecture, our proposed policy network has only three layers, i.e., two self-attention layers and one cross-attention layer. Each layer of the attention mechanism uses 4 attention heads, and the output dimension of each attention head in the first layer is 32, and the output dimension of each attention head in the second and third layers is 8. Both the action network and the critic network contain only two 64-dimensional hidden layers. For the PPO parameters, the policy coefficient, value coefficient, and entropy coefficient in the loss function are set to 1, 0.5, and 0.05, respectively. The GAE parameter λ and the discount factor γ are set to 0.98 and 1, respectively. The network is updated 4 epochs per episode using the Adam optimizer, with a learning rate $lr = 2 \times 10^{-4}$. All experiments were performed on a machine equipped with an Intel Xeon Gold 6226R CPU and a single NVIDIA GeForce RTX 4090 GPU.

4.3 Baselines and performance metrics

In order to evaluate our approach more comprehensively, the performance of the proposed model will be compared with heuristic algorithms, exact algorithms, and classical learning-based algorithms. Specifically, the heuristic algorithms include FIFO, MOPNR, MWKR, SPT. We define FIFO, MOPNR, MWKR, SPT as Priority Dispatch Rules (PDR). The exact algorithms are executed by OR-tools under a time constraint of 1800 seconds. The classical learning-based algorithms include HGNN [16] and DANIEL [23], which are representative of the GNN-based and attention-based methods, respectively. For the performance metrics, we simply take the makespan given by OR-tools as a reference and calculate the relative gap between the C_{max} of each algorithm and the results of OR-Tools. The calculation method is:

$$gap = \left(\frac{C_{max}}{C_{max}^{OR}} - 1\right) \times 100\% \tag{8}$$

4.4 Results

In Table. 1, we report the comparative performance of each method in the same distribution and same size scenarios. For all synthetic data test sets of all problem sizes, our method outperforms all the PDRs and classical learning-based methods while maintaining the same level of inference time. At the same time, our model steadily outperforms [16,23], which suggests that the bidirectional cross-attention module can further enhance the representation learning capability of the policy network. In the best case, it achieves a maximum improvement of 96.87% over [16], and a maximum of 3.01% over [23]. In the worst case, our model also manages to achieve an improvement of 0.90% over [23]. In contrast to the PDRs, our method obtains far better results without a significant increase in the solving time.

Table. 2 reports the comparative performance of each method in a large-scale scenario with the same distribution. For the SD1 dataset, our method reduces the optimality gap to 1.42% for the (30,10)-scale problem, outperforming HGNN (13.97%) and DANIEL (2.48%), all while maintaining a fast response time of 2.40 seconds. When scaled to (40,10), the gap further decreases to 0.52%. In the more challenging SD2 dataset, our method controls the gap at 9.80% for the (30,10)-scale problem, significantly better than HGNN (123.00%) and DANIEL (11.89%). Particularly noteworthy is the achievement of a negative gap of -4.83% at the (40,10) scale, surpassing OR-Tools' reference solution. These results show that our approach balances solution quality and efficiency, outperforming both learning-based baselines and exact methods under time constraints.

Table 1. Performance Comparison of Same Distribution and Same Scale Scenarios.

Size			Top PDR	Learning-based Solver			OR-Tools
				HGNN [16]	DANIEL [23]	Ours	
SD1	(10,5)	Gap	17.56%	15.94%	10.79%	**9.89%**	96.32
		Time	0.16s	0.45s	0.21s	0.39s	
	(20,5)	Gap	11.50%	12.26%	5.00%	**2.84%**	188.15
		Time	0.32s	0.81s	0.53s	0.78s	
	(15,10)	Gap	19.31%	16.30%	12.37%	**10.83%**	143.53
		Time	0.50s	1.22s	0.96s	1.17s	
	(20,10)	Gap	10.27%	10.10%	1.29%	**0.12%**	195.98
		Time	0.71s	1.77s	1.13s	1.59s	
SD2	(10,5)	Gap	57.67%	69.70%	25.18%	**23.96%**	326.24
		Time	0.16s	0.46s	0.24s	0.41 s	
	(20,5)	Gap	38.85%	75.91%	11.50%	**9.91%**	602.04
		Time	0.33s	0.94s	0.62s	0.98 s	
	(15,10)	Gap	86.41%	114.09%	56.75%	**54.01%**	377.17
		Time	0.51s	1.50s	1.06s	1.29 s	
	(20,10)	Gap	78.63%	125.31%	31.45%	**28.44%**	464.16
		Time	0.70s	1.95s	1.35s	1.96 s	

Table 2. Performance Comparison with Distributed Large-Scale Scenarios.

Size			Top PDR	Learning-based Solver			OR-Tools
				HGNN [16]	DANIEL [23]	Ours	
SD1	(30,10)	Gap	13.93%	13.97%	2.48%	**1.42%**	274.67
		Time	1.09 s	2.84s	2.07s	2.40s	
	(40,10)	Gap	13.35%	13.72%	1.50%	**0.52%**	365.96
		Time	1.50 s	3.81s	2.98s	3.20s	
SD2	(30,10)	Gap	59.77%	123.00%	11.89%	**9.80%**	692.26
		Time	1.10 s	2.93s	2.81s	3.12s	
	(40,10)	Gap	35.93%	103.58%	-3.59%	**-4.83%**	998.39
		Time	1.50 s	3.93s	4.05s	4.47s	

To validate the necessity of the proposed bidirectional cross-attention mechanism, we conducted systematic ablation experiments. As shown in Table. 3, the experiments compared the performance of different attention configurations across two problem scales (SD1 and SD2), yielding the following conclusions: 1.) The complete model (Ours) achieved optimal performance in all tasks, with significantly lower Gap values than other variants (e.g., only 0.12% for SD1(20,10), representing a 0.26-1.17 percentage point reduction compared to single cross-attention versions). This conclusively demonstrates the importance of simultaneously modeling bidirectional interactions between operators (OP) and machines (MC); 2.) Scalability analysis revealed that while runtime increased with problem size (SD1 from 0.39s to 1.59s), the improvement in Gap was more substantial (e.g., a 9.77 percentage point reduction from SD1(10,5) to (20,10)), indicating the mechanism's particular suitability for complex scenarios. These results consistently verify the critical role of bidirectional cross-attention in enhancing model performance.

Table 3. Performance Comparison with Different Attention Network Configurations.

Size			without cross-Att	only OP-cross-MC	only MC-cross-OP	Ours	OR-Tools
SD1	(10,5)	Gap	10.81%	10.52%	10.61%	**9.89%**	274.67
		Time	0.22s	0.32s	0.24s	0.39s	
	(20,10)	Gap	1.29%	0.38%	0.51%	**0.12%**	365.96
		Time	1.13s	1.42s	1.22s	1.59s	
SD2	(10,5)	Gap	25.22%	24.87%	25.02%	**23.96%**	692.26
		Time	0.25s	0.34s	0.28s	0.41s	
	(20,10)	Gap	31.62%	29.41%	29.51%	**28.44%**	998.39
		Time	1.38s	1.83s	1.44s	1.96s	

5 Conclusions

The FJSP is a classic combinatorial optimization problem in fields such as manufacturing and cloud computing. Due to the inadequacy of current attention-based mechanisms for state representation, this paper proposes a bidirectional cross-attention network to learn the heterogeneous relationship between operations and machines. Through extensive experiments on synthetic data, this paper demonstrates that our model can further improve the performance stably without significantly increasing the inference time. In the future, we hope to generalize this model to the dynamic and complex constraint variant FJSP.

References

1. Dauzère-Pérès, S., Ding, J., Shen, L., et al.: The flexible job shop scheduling problem: A review. European Journal of Operational Research 314(2), 409–432 (2024)
2. Festa, P.: A brief introduction to exact, approximation, and heuristic algorithms for solving hard combinatorial optimization problems. In: 2014 16th International Conference on Transparent Optical Networks (ICTON), pp. 1–20. IEEE, 2014
3. Pezzella, F., Morganti, G., Ciaschetti, G.: A genetic algorithm for the flexible job - shop scheduling problem. Computers & Operations Research 35(10), 3202–3212 (2008)
4. Saidi-Mehrabad, M., Fattahi, P.: Flexible job shop scheduling with tabu search algorithms. The International Journal of Advanced Manufacturing Technology 32, 563–570 (2007)
5. Li, J.Q., Pan, Q.K., Suganthan, P.N., et al.: A hybrid tabu search algorithm with an efficient neighborhood structure for the flexible job shop scheduling problem. The International Journal of Advanced Manufacturing Technology 52, 683–697 (2011)
6. Xie, J., Li, X., Gao, L., et al.: A new neighbourhood structure for job shop scheduling problems. International Journal of Production Research 61 (7), 2147–2161 (2023)
7. Tamssaouet, K., Dauzère-Pérès, S.: A general efficient neighborhood structure framework for the job - shop and flexible job - shop scheduling problems. European Journal of Operational Research 311(2), 455–471 (2023)
8. Chung, K.T., Lee, C.K.M., Tsang, Y.P.: Neural combinatorial optimization with reinforcement learning in industrial engineering: a survey. Artificial Intelligence Review 58(5), 130 (2025)
9. Yao, S., Lin, X., Wang, J., et al.: Rethinking Supervised Learning based Neural Combinatorial Optimization for Routing Problem. ACM Transactions on Evolutionary Learning (2024)
10. Nazari, M., Oroojlooy, A., Snyder, L., Takác, M.: Reinforcement learning for solving the routing routing problem. Advances in Neural Information Processing Systems 31 (2018)
11. Henderson, P., Islam, R., Bachman, P., et al.: Deep reinforcement learning that matters. In: Proceedings of the AAAI Conference on Artificial Intelligence, 32(1) (2018)
12. Park, J., Chun, J., Kim, S.H., et al.: Learning to schedule job - shop problems: representation and policy learning using graph neural network and reinforcement learning. International Journal of Production Research 59(11), 3360–3377 (2021)
13. Lv, Q.H., Chen, J., Chen, P., et al.: Flexible Job - shop Scheduling Problem with parallel operations using Reinforcement Learning: An approach based on Heterogeneous Graph Attention Networks. Advances in Production Engineering & Management 19(2), 157–181 (2024)

14. Xu, S., Li, Y., Li, Q.: A deep reinforcement learning method based on a transformer model for the flexible job shop scheduling problem. Electronics 13(18), 3696 (2024)
15. Zhang, C., Song, W., Cao, Z., et al.: Learning to dispatch for job shop scheduling via deep reinforcement learning. Advances in Neural Information Processing Systems 33, 1621–1632 (2020)
16. Song, W., Chen, X., Li, Q., et al.: Flexible job-shop scheduling via graph neural network and deep reinforcement learning. IEEE Transactions on Industrial Informatics 19(2), 1600–1610 (2022)
17. Ho, Kuo-Hao, et al.: Residual scheduling: A new reinforcement learning approach to solving job shop scheduling problem. IEEE Access 12, 14703–14718 (2024)
18. Echeverria, I., Murua, M., Santana, R.: Solving the flexible job-shop scheduling problem through an enhanced deep reinforcement learning approach. arXiv preprint arXiv:2310.15706 (2023)
19. Wan, Lanjun, et al.: Flexible job shop scheduling via deep reinforcement learning with meta - path - based heterogeneous graph neural network. Knowledge Based Systems 296, 111940 (2024)
20. Bogyrbayeva, Aigerim, et al.: Machine learning to solve vehicle routing problems: A survey. IEEE Transactions on Intelligent Transportation Systems 25(6), 4754–4772 (2024)
21. Shao, C., Yu, Z., Ding, H., et al.: A reinforcement learning method for flexible job shop scheduling based on multi - head attention and deep residual network. Computers and Electrical Engineering 123, 110044 (2025)
22. Xu, S., Li, Y.W., Li, Q.Y.: A deep reinforcement learning method based on a transformer model for the flexible job shop scheduling problem. Electronics 13(18), 3696 (2024)
23. Wang, R., Wang, G., Sun, J., et al.: Flexible job shop scheduling via dual attention network - based reinforcement learning. IEEE Transactions on Neural Networks and Learning Systems 35(3), 3091–3102 (2023)
24. Zhao, Y., Chen, Y., Wu, J., et al.: Dual Dynamic Attention Network for Flexible Job Scheduling with Reinforcement Learning. In: 2024 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. IEEE, 2024
25. Schulman, J., Wolski, F., Dhariwal, P., et al.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)