Reinforcement Learning for Job-Shop Scheduling via Hybrid GNN Embeddings with Sophisticated State Representations

Haohong Xu, Qingxin Guo, Zhiming Dong, Guodong Zhao
National Frontiers Science Center for Industrial Intelligence and Systems Optimization, Northeastern University, China
Key Laboratory of Data Analytics and Optimization for Smart Industry (Northeastern University), Ministry of Education, China
Liaoning Engineering Laboratory of Data Analytics and Optimization for Smart Industry, Northeastern University
Liaoning Key Laboratory of Manufacturing System and Logistics Optimization, Northeastern University
Email: 2410376@stu.neu.edu.cn, guoqingxin@ise.neu.edu.cn, dongzm@neu.edu.cn, zhaoguodong@mail.neu.edu.cn

***Abstract*—Although neural optimization methods based on priority rule learning and graph isomorphic networks (GINs) have shown promise for the job shop scheduling problem (JSSP), traditional GINs still struggle to effectively capture the dynamic constraints and scheduling relationships when processing disjunctive graphs. This paper proposes an end-to-end algorithmic architecture that integrates a graph attention network (GAT) with a GIN. The architecture leverages the GIN to extract general graph structural features, while introducing the GAT to focus on temporal attributes and dynamic topological changes. The fused network is trained using the proximal policy optimization (PPO) algorithm to achieve adaptive optimization for dynamic job shop scheduling. Experimental results demonstrate that the proposed algorithm outperforms both traditional priority dispatching rules and advanced deep reinforcement learning methods, while also validating the effectiveness of the network fusion.**

***Index Terms*—Job-shop scheduling, fusion network, graph attention networks, graph isomorphism networks, deep reinforcement learning.**

I. Introduction

The Job-Shop Scheduling Problem (JSSP) is a crucial optimization problem in industrial production. It is widely applied in fields such as aerospace assembly [1], semiconductor chip manufacturing [2], and even the Internet of Things (IoT) [3]. In the practical operational processes of enterprises and factories, achieving highly efficient product manufacturing necessitates the judicious allocation of resources, operations, and machines. Concurrently, scheduling algorithms should also ensure good applicability when confronted with orders of varying scales and products involving different operational sequences. Consequently, there is a pressing need for models and algorithms capable of effectively solving JSSP by balancing both computational efficiency and generalization performance.

As a well-known NP-hard problem [4], the JSSP is characterized by a solution space that expands factorially with problem scale. While traditional exact methods such as mathematical programming [5] and branch-and-bound [6] theoretically guarantee optimality, they suffer from combinatorial explosion. For medium-to-large instances, these approaches incur prohibitive computational costs and are ill-suited for the responsiveness required in dynamic production. Consequently, contemporary research has pivoted towards heuristic approaches that deliver high-quality approximate solutions within actionable timeframes.

Approximate solving methods for JSSP primarily encompass two typical approaches: metaheuristic algorithms [7] and priority dispatching rules (PDRs) [8], [9]. Metaheuristic algorithms, such as Genetic Algorithms (GAs) [10], Tabu Search (TS) [11], and Simulated Annealing (SA) [12], explore the solution space through global search mechanisms, capable of yielding stable and reliable scheduling solutions. However, major drawbacks are that these algorithms incur high computational costs and lack the ability to learn or generalize from historical data. They must perform the search process from scratch for every new problem instance, leading to significant inefficiency in varying production environments. In contrast, PDRs can achieve rapid decision-making based on simple rules, enabling real-time responses to dynamic scheduling demands, which makes them more popular in practical production settings [13]. Nevertheless, because they only consider local information, the quality of their solutions can fluctuate significantly across different problem instances. Furthermore, designing effective PDRs is a cumbersome task, requiring substantial domain expertise [14].

In recent years, deep reinforcement learning (DRL) has demonstrated considerable advancements in solving JSSP in an end-to-end manner [15]–[17]. Even so, considerable scope for further enhancing algorithmic performance still exists. To address this challenge, researchers have proposed various innovative solutions, predominantly incorporating Graph Neural Networks (GNNs) to process the disjunctive graph representation of the problem [18]. However, when it comes to feature extraction from these disjunctive graphs, existing studies have consistently relied on the initial Graph Isomorphism Network (GIN) [19]. Although GIN is recognized for its strong ability to capture general graph features, it exhibits limited effectiveness in extracting specific attributes such as varying temporal information. Conversely, attention-based networks like Graph Attention Networks (GAT) [20] demonstrate superior spatial representation capabilities. Their ability to dynamically focus on relevant structural elements allows for better handling

of time-sensitive attributes and variations in local topology, thereby compensating for the limitations observed in global aggregation methods. Despite the evident complementarity between GIN and GAT, research exploring their integration remains scarce. The potential synergy between these two architectures has yet to be fully exploited in the context of solving JSSP.

To bridge this gap, this paper proposes a novel method for solving the JSSP by combining a feature fusion network, based on GIN and GAT, with PPO reinforcement learning. The effective combination of GIN and GAT networks enhances the Graph Neural Network's ability to extract features from operation nodes, thereby enabling it to effectively capture complex dependencies between operations. Furthermore, the Markov Decision Process constructed by integrating the PPO algorithm consistently exhibited robust performance across multiple datasets of varying scales. The source code and datasets used in this study are available from the first author upon reasonable request.

II. RELATED WORK

Currently, extensive research has employed reinforcement learning methods to achieve end-to-end solutions for the standard JSSP. The core idea of most existing studies is to solve JSSP by training graph-based policy networks using reinforcement learning algorithms with disjunctive graph representations [15]. Following this line of research, Hottung et al. [21] incorporated an efficient active search strategy (EAS) to enhance algorithm performance by updating a subset of model parameters. Su et al. [22] utilized an evolution strategy (ES)-based learning algorithm instead of PPO to obtain a more stable and robust model. Yuan et al. [23] enriched the model with more effective information through the use of bidirectional scheduling features. In a parallel effort, Van et al. [24] and Liao et al. [25] separately enhanced the performance of reinforcement learning algorithms by employing offline learning and hierarchical learning techniques, respectively. Furthermore, Chen et al. [26] extended the algorithm's functionality to make it applicable to multi-objective tasks, which involve balancing cost, completion rate, and makespan. The aforementioned studies have explored various approaches to enhance the effectiveness of algorithms and models. Despite these algorithmic diversifications, the mechanism for extracting features from disjunctive graphs has remained largely uniform, predominantly relying on the standard Graph Isomorphism Network (GIN). While GIN is effective at aggregating global structural information, it often struggles to capture the fine-grained, varying temporal nuances required for high-precision scheduling.

Compared to GIN, the attention mechanism demonstrates superior spatial representation capabilities. It can effectively capture local relationships in complex tasks, exhibits heightened sensitivity to dynamic and temporal features, and possesses robust noise immunity. Leveraging these characteristics of the attention mechanism, researchers have made various attempts to address the JSSP. Chen et al. [27] proposed DGERD, a transformer-based construction policy for JSSPs. This method pioneers a novel paradigm that actively integrates graph representation learning with attention mechanisms to tackle combinatorial optimization problems, consequently boosting the algorithm's real-time decision-making capabilities. Similarly, Liao et al. [28] developed an attention-model-driven policy network capable of generating diverse scheduling solutions for JSSPs of various scales, critically, without requiring any expert knowledge. To capture dependencies among operations within identical jobs or machines, Lee et al. [29] developed the ARLS framework by integrating masked attention techniques. Furthermore, Yang et al. [30] proposed a reinforcement learning policy network that integrates Graph Attention Networks to effectively process disjunctive graph information, thereby enabling end-to-end decision-making. The advantages demonstrated by attention networks in solving JSSPs are substantiated by the research discussed.

As discussed, although GIN and attention-based networks like GAT offer individual merits in modeling and algorithmic problem-solving, studies specifically integrating their combined benefits to tackle JSSP are conspicuously limited. To this end, this paper proposes an algorithm that combines a feature fusion network with reinforcement learning to achieve an effective end-to-end solution for the JSSP.

III. METHOD

Fig. 1 presents the proposed framework, synergizing disjunctive graph modeling, hybrid GNN state encoding, and a PPO-based decision module [31]. Specifically, a fused GIN-GAT architecture is deployed to encode the graph topology into comprehensive nodal embeddings. These features serve as input to an Actor-Critic network, where action masking is applied to preclude infeasible actions. The agent iteratively assigns operations to machines until the schedule is finalized.

A. Disjunctive Graph mode

The JSSP is modeled as a disjunctive graph $G = (\mathcal{V}, \mathcal{C} \cup \mathcal{D})$, where $\mathcal{V}$ comprises all operation nodes alongside dummy start (S) and terminal (T) nodes. The edge set integrates directed conjunctive arcs $\mathcal{C}$, enforcing technological precedence within jobs, and undirected disjunctive arcs $\mathcal{D}$, representing resource contention on shared machines. As illustrated in Fig. 2, solving the problem entails orienting all arcs in $\mathcal{D}$ to resolve conflicts, with the objective of forming a Directed Acyclic Graph (DAG) that minimizes the critical path length from S to T.

B. Graph state representation

The entire process of this section is detailed within the state embedding component of Fig. 1. First, the state features undergo structural isomorphism enhancement via GIN to improve their discriminative power. Subsequently, the enhanced features are fed into a GAT, enabling attention weight learning to occur in a higher-quality feature space. This process ultimately achieves more intelligent operation priority evaluation and enhances the stability of feature extraction. Furthermore,

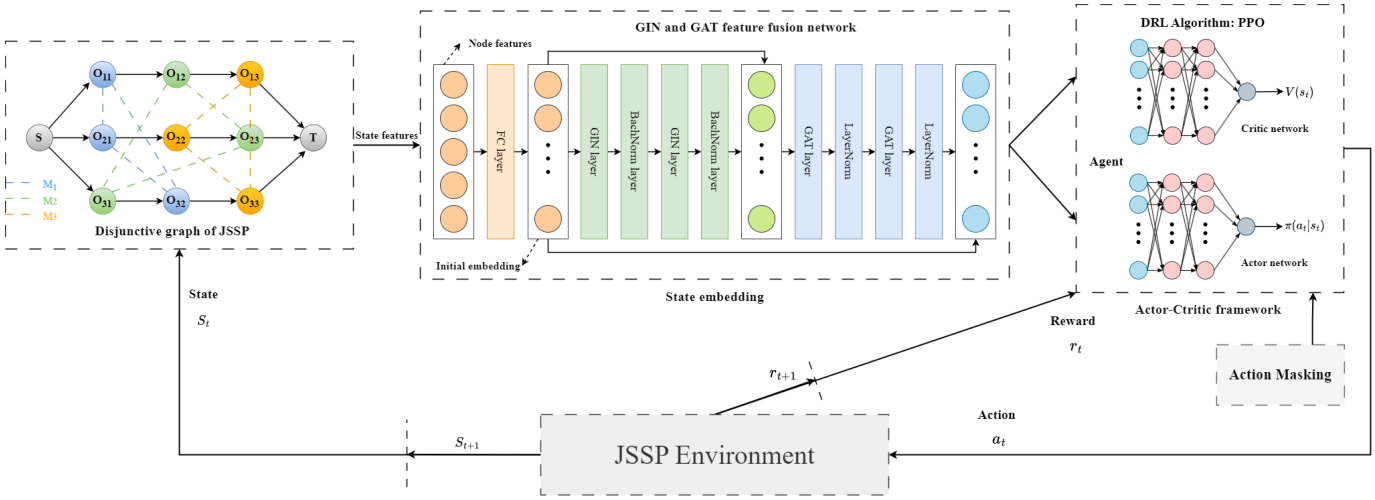

Fig. 1. Overview of the proposed method's framework.

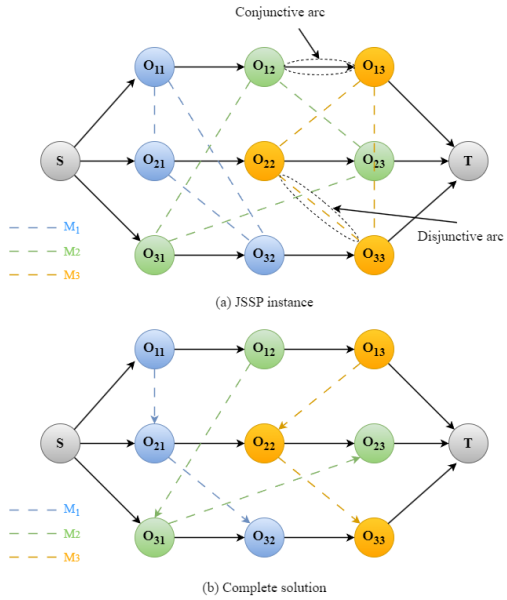

Fig. 2. The disjunctive graph for JSSP.

the effectiveness of our proposed fusion was validated through ablation experiments in the experimental section.

1) Initial embedding: Specifically, five distinct features are defined and normalized for each node. These raw features pass through a fully connected layer to create high-dimensional initial embeddings suitable for GIN processing.

Assuming that node $O_{ij} \in \mathcal{V}$ has an initial feature vector $\mathbf{x}_{ij}$, its initial embedding, denoted as $\mathbf{h}_{ij}^{(0)}$, satisfies the following formula:

$$\mathbf{h}_{ij}^{(0)} = \mathbf{x}_{ij}\mathbf{W} + \mathbf{b} \tag{1}$$

where $\mathbf{W}$ and $\mathbf{b}$ denote the weight matrix and bias vector of the fully connected layer, respectively.

2) Structural isomorphism enhancement: We leverage the Graph Isomorphism Network (GIN) [19] to capture the topological structure of the disjunctive graph G. For each node $O_{ij} \in \mathcal{V}$, the network generates a p-dimensional embedding through K message-passing iterations. The update rule for the k-th layer is defined as:

$$\mathbf{h}_{ij}^{(k)} = \text{MLP}^{(k)}\left((1+\epsilon^{(k)}) \cdot \mathbf{h}_{ij}^{(k-1)} + \sum_{O_{uv} \in \mathcal{N}(O_{ij})} \mathbf{h}_{uv}^{(k-1)}\right) \tag{2}$$

where $\mathbf{h}_{ij}^{(k)}$ denotes the feature embedding of node O_{ij} at iteration k. The learnable scalar $\epsilon^{(k)}$ balances the importance of the node's own features against the aggregated information from its neighbor set $\mathcal{N}(O_{ij})$, while $\text{MLP}^{(k)}$ is the Multi-Layer Perceptron for non-linear transformations in the k-th iteration.

To prevent over-smoothing, we limit the model to two layers with post-layer normalization. We also incorporate a residual connection to retain initial discriminative features, yielding the final GAT input:

$$\mathbf{z}_{ij}^{(0)} = \mathbf{h}_{ij}^{(K)} + \mathbf{h}_{ij}^{(0)} \tag{3}$$

3) Graph attention network: GAT's self-attention mechanism can deeply leverage the structured embeddings built by GIN, refining neighbor information through dynamic weighting to achieve more powerful, context-aware node representations. Within the GAT framework, node embeddings are iteratively refined across K message passing iterations. Specifically, the update mechanism for a node's embedding at iteration k is defined as follows:

$$\mathbf{z}_{ij}^{(k+1)} = \text{GAT}_{\text{layer}}^{(k+1)}\left(\mathbf{z}_{ij}^{(k)}, \left\{\mathbf{z}_{uv}^{(k)} \mid O_{uv} \in \mathcal{N}(O_{ij})\right\}\right) \tag{4}$$

where $\mathbf{z}_{ij}^{(k+1)}$ represents the updated node embedding. This layer aggregates the current state $\mathbf{z}_{ij}^{(k)}$ with the neighbor features, utilizing the adjacency set $\mathcal{N}(O_{ij})$.

The final node embedding $\mathbf{Z}_{ij}^{\text{out}}$ is derived by adding the initial embedding $\mathbf{h}_{ij}^{(0)}$ to the last GAT layer output $\mathbf{z}_{ij}^{(K)}$:

$$\mathbf{Z}_{ij}^{\text{out}} = \mathbf{z}_{ij}^{(K)} + \mathbf{h}_{ij}^{(0)} \tag{5}$$

This residual connection preserves original feature information and helps mitigate over-smoothing in deeper network layers.

C. Markovian Decision Process

Modeling the JSSP as a Markov Decision Process (MDP) involves defining a 5-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$. In this framework, $\mathcal{S}$ represents the current state of the JSSP, $\mathcal{A}$ specifies the set of possible actions, $\mathcal{P}$ governs the state transitions within the JSSP environment, $\mathcal{R}$ assigns immediate rewards for actions taken, and γ is the discount factor. These components are all intuitively presented in Fig. 1.

1) State: The state $s_t \in \mathbb{R}^{(n\times m)\times 5}$ is a feature matrix. It is constructed by stacking 5-dimensional feature vectors, x_{ij}, for each operation, with each x_{ij} vector encapsulating five distinct attributes.

- **Processing Time:** The fixed duration required to complete operation O_{ij}.
- **Readiness Status:** Readiness Status: A binary indicator (1 if ready, 0 if not) showing whether operation O_{ij} has met all its prerequisites and is available for processing.
- **Scheduled Status:** A binary indicator (1 if scheduled, 0 if not) showing whether operation O_{ij} has already been assigned to a machine.
- **Available Machines Count:** The total number of machines currently idle and available in the system.
- **Job Assignment:** The unique identifier (index) of the job J_i to which operation O_{ij} belongs.

2) Action: To enhance decision-making efficiency and solution quality, action masking is employed to restrict the agent's selection to only feasible operations [32]. An operation is deemed "unassignable" and consequently masked if its preceding operation is incomplete, the required machine is unavailable, or it has already been scheduled. Action a_t is selected via stochastic sampling during training and greedy decoding during validation. For testing, we utilize parallel stochastic sampling to obtain the minimize makespan.

3) Transition: State transitions, managed by the environment simulator, are influenced by both agent actions and elapsed time, moving the system from state s_t to s_{t+1}. When the agent schedules operation O_{ij} in s_t, its status becomes 'scheduled', $O_{i(j+1)}$'s arrival time is set to C_{ij}, and job J_i's remaining operations decrease. Consequently, any other 'assignable' operations requiring the now-occupied machine M_{ij} become 'non-assignable'. Concurrently, disjunctive arc directions are also updated.

4) Reward: The goal is to learn the dispatching process step by step to minimize the makespan. Hence, we define the reward as the makespan increment. Specifically, we aim to minimize the makespan increment from state s_t to state s_{t+1}. By progressively seeking the smallest possible increment at each step, we achieve the objective of minimizing the overall makespan. Finally, by negating this increment, we derive the following reward function:

$$R(s_t, a_t) = H(s_t) - H(s_{t+1}) \tag{6}$$

Here, $H(\cdot)$ serves as a measure for the lower bound of the current makespan.

D. Learning Algorithm

We train the policy network using Proximal Policy Optimization (PPO) [31], which ensures update stability via a clipped surrogate objective. The core objective function is defined as:

$$L^t_{CLIP}(\theta) = \hat{E}_t \left[\min \left(R_t(\theta)\hat{A}_t, \text{clip}(R_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right] \tag{7}$$

where $R_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ denotes the probability ratio between new and old policies, $\hat{A}_t$ is the advantage estimation, and ϵ represents the clipping coefficient.

To balance exploration and value estimation accuracy, we maximize the following composite objective:

$$L_t(\theta, \phi) = c_p L^t_{CLIP}(\theta) + c_e L^E_t(\theta) - c_v L^V_t(\phi) \tag{8}$$

where $L^E_t(\theta)$ and $L^V_t(\phi)$ represent the entropy bonus and critic's mean squared error, respectively, weighted by coefficients c_p, c_e, and c_v.

IV. EXPERIMENT

In this section, we experimentally evaluate the performance of the proposed algorithm. To assess its effectiveness and the model's generalization capability, we will train the model on a single data scale and subsequently test it on datasets of varying scales.

A. Datasets

For our experiments, the training and validation datasets consist of randomly generated Job Shop Scheduling Problem (JSSP) instances. All these synthetic instances are uniformly sized at 15x15 (15 jobs and 15 machines); specifically, 2000 training instances and 1000 validation instances were generated. The primary test set employed is the standard Taillard's benchmark dataset [33], which is composed of 80 instances distributed across eight distinct problem scales. Our experimental protocol involves training the model exclusively on the 15x15 instances and subsequently evaluating its performance and zero-shot generalization capabilities on all eight problem scales from the Taillard benchmark.

B. Baselines

We evaluated the efficacy of our proposed algorithm by contrasting it with eight varied baseline methods. These include two state-of-the-art end-to-end neural techniques: L2D [15] and RL-GNN [34]. The remaining six baselines are established Priority Dispatching Rules (PDRs), classified by their decision scope. At the operation level, we employ Shortest Processing Time (SPT). The job-level rules include Longest Total Processing Time (LTPT), Least/Most Operation Remaining (LOR/MOR), and Most Work Remaining (MWKR). Finally, Least Queue Next Operation (LQNO) is selected for the machine level. In order to maintain experimental consistency, the L2D algorithm was re-implemented using our framework, with the graph embedding module being the only variant.

TABLE I
AVERAGE GAP ON TAILLARD'S BENCHMARK

Method	TA 15×15	TA 20×15	TA 20×20	TA 30×15	TA 30×20	TA 50×15	TA 50×20	TA 100×20
SPT	56.85%	63.83%	65.27%	67.63%	67.29%	50.33%	55.29%	39.83%
LTPT	53.02%	58.54%	58.02%	52.08%	63.75%	42.83%	48.01%	31.33%
LOR	63.63%	72.95%	73.09%	68.79%	75.15%	51.31%	64.21%	40.19%
MOR	26.24%	30.75%	27.94%	29.29%	34.47%	22.26%	24.44%	12.59%
MWKR	26.97%	28.48%	28.57%	31.25%	34.21%	23.25%	24.74%	12.90%
LQNO	50.09%	61.53%	60.10%	53.71%	61.93%	36.12%	50.26%	29.50%
L2D	20.13%	25.99%	23.63%	27.91%	31.08%	18.92%	23.25%	12.91%
RL-GNN	20.10%	24.90%	29.20%	24.70%	32.00%	15.90%	21.30%	**9.20%**
Ours	**17.15%**	**21.43%**	**20.65%**	**23.45%**	**26.64%**	**14.56%**	**19.72%**	9.91%

For the PDR baselines, we enhanced their performance by incorporating techniques like left-shift heuristics, which aim to schedule each operation at the earliest possible time, thus leading to more persuasive experimental findings. In evaluating experimental results, the performance of various algorithms is assessed by averaging the gap for each instance. This gap, also referred to as the relative scheduling error [34], is computed as follows:

$$\text{Gap} = \left(\frac{C_{\max}}{C^*_{\max}} - 1 \right) \times 100\% \tag{9}$$

Here, $C_{\max}$ is the makespan from the algorithm, while $C^*_{\max}$ is the optimal makespan, treated as an upper bound from the best result in existing literature.

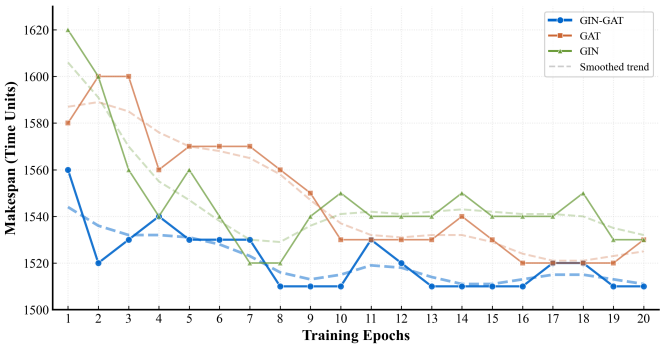

Fig. 3. Training convergence curves.

C. Training details

Experiments were executed on an Intel Xeon Gold 6248R CPU and NVIDIA A30 GPU. The architecture features 2-layer GIN/GAT encoders (128 hidden units, ReLU) and 8 attention heads. Training spanned 20 epochs using Adam ($lr = 2.0 \times 10^{-5}$), with PPO settings including a clip ratio of 0.2 and loss coefficients of 1.0 (policy), 0.5 (value), and 0.01 (entropy)

D. Experimental result

Table I reports the average optimality gaps on the Taillard benchmark. The proposed method outperforms conventional heuristics and DRL baselines (L2D, RL-GNN) on 7 out of 8 problem scales. Most notably, on the challenging 20×20 instances, our approach reduces the optimality gap from 29.20% (RL-GNN) to 20.65%, a substantial margin of approximately 8.5%. While RL-GNN performs slightly better on the largest 100×20 set, our framework is more effective across all remaining scales. These results suggest that the feature fusion strategy successfully encodes complex scheduling dependencies that single-architecture models fail to capture.

Convergence Analysis: Fig. 3 compares the learning curves of the proposed framework with the standalone GIN and GAT baselines. The GIN-GAT model converges rapidly, reducing the makespan from 1560 to approximately 1510 within the initial epochs. In contrast, the GAT and GIN variants plateau at inferior solutions (approx. 1520 and 1540, respectively). The stability of the proposed curve suggests that fused features provide a robust state representation, avoiding the premature convergence observed in single-architecture embeddings.

TABLE II
ABLATION STUDY ON FEATURE FUSION STRATEGY

Inst.	Size	Opt	Makespan ($C_{\max}$)		
			GIN	GAT	GIN-GAT
ta01	15 × 15	1231	1414	1415	**1385**
ta02	15 × 15	1244	1403	1456	**1381**
ta11	20 × 15	1357	1692	1635	**1626**
ta12	20 × 15	1367	1650	1641	**1547**
ta21	20 × 20	1642	1961	2062	**1912**
ta22	20 × 20	1600	1916	1913	**1830**
ta31	30 × 15	1764	2182	2112	**2099**
ta32	30 × 15	1784	**2269**	2310	2277
ta41	30 × 20	2006	2509	2555	**2473**
ta42	30 × 20	1939	2504	**2433**	2491
ta51	50 × 15	2760	3269	3194	**3089**
ta52	50 × 15	2756	3253	3227	**3041**
ta61	50 × 20	2868	3441	3529	**3422**
ta62	50 × 20	2869	3504	3476	**3475**
ta71	100 × 20	5464	6117	6177	**5868**
ta72	100 × 20	5181	5693	5676	**5623**

E. Ablation Study

To validate the fusion strategy, we conducted ablation experiments (Table II) using identical hyperparameters. Models were trained on generated 15×15 instances and evaluated on the Taillard dataset. Table II reports the results for representative instances, where 'Opt' denotes the best-known or optimal solution. The GIN-GAT network consistently outperforms the individual GIN and GAT modules across most scales. It is noteworthy that our GIN implementation follows the structure in Fig. 1, distinct from the L2D approach. As evident from the results in Table II, the proposed GIN-GAT fusion network consistently achieved superior performance across a majority

of instances, thereby underscoring the effectiveness of our feature fusion approach.

V. CONCLUSION

This paper addresses the combinatorial complexity of JSSP by proposing a novel Reinforcement Learning framework with hybrid graph embeddings. By integrating disjunctive graph modeling with a fused GIN-GAT architecture, the proposed method effectively captures complex topological dependencies and inter-operation constraints. Experimental results demonstrate that this hybrid feature extraction significantly enhances representation power, enabling the algorithm to outperform standard dispatching rules and state-of-the-art DRL baselines. Future work will explore integrating diverse graph neural architectures to further augment embedding capacity and generalization performance.

ACKNOWLEDGMENT

This research was supported by the Major Program of National Natural Science Foundation of China (72192830, 72192831), the 111 Project (B16009).

REFERENCES

[1] H. Wang, B. R. Sarker, J. Li, and J. Li, "Adaptive scheduling for assembly job shop with uncertain assembly times based on dual q-learning," *International Journal of Production Research*, vol. 59, no. 19, pp. 5867–5883, 2021.

[2] S. Knopp, S. Dauzère-Pérès, and C. Yugma, "A batch-oblivious approach for complex job-shop scheduling problems," *European Journal of Operational Research*, vol. 263, no. 1, pp. 50–61, 2017.

[3] C. Liao, J. Chen, D. Gao, X. Huang, S. Liu, and D. Qian, "Jrlao: Joint resource and latency-aware sfc optimized orchestration in nfv-enabled networks," *IEEE Transactions on Cognitive Communications and Networking*, 2025.

[4] M. R. Garey, D. S. Johnson, and R. Sethi, "The complexity of flowshop and jobshop scheduling," *Mathematics of Operations Research*, vol. 1, no. 2, pp. 117–129, 1976.

[5] A. S. Manne, "On the job-shop scheduling problem," *Operations Research*, vol. 8, no. 2, pp. 219–223, 1960.

[6] Z. Lomnicki, "A "branch-and-bound" algorithm for the exact solution of the three-machine scheduling problem," *Journal of the Operational Research Society*, vol. 16, no. 1, pp. 89–100, 1965.

[7] F. Glover and M. Laguna, "Tabu search. handbook of combinatorial optimization," *Handbook of Combinatorial Optimization*, pp. 2093–2229, 1998.

[8] R. Haupt, "A survey of priority rule-based scheduling," *Operations-Research-Spektrum*, vol. 11, no. 1, pp. 3–16, 1989.

[9] S. Nguyen, M. Zhang, M. Johnston, and K. C. Tan, "A computational study of representations in genetic programming to evolve dispatching rules for the job shop scheduling problem," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 5, pp. 621–639, 2013.

[10] D. Hutter, M. Hellwig, and T. Steinberger, "An interior-point genetic algorithm with restarts for flexible job shop scheduling problems," in *2024 IEEE Congress on Evolutionary Computation (CEC)*, 2024, pp. 01–09.

[11] C. D. Geiger, K. G. Kempf, and R. Uzsoy, "A tabu search approach to scheduling an automated wet etch station," *Journal of Manufacturing Systems*, vol. 16, no. 2, pp. 102–116, 1997.

[12] L. Tang, Z. Li, and J.-K. Hao, "Solving the single-row facility layout problem by k-medoids memetic permutation group," *IEEE Transactions on Evolutionary Computation*, vol. 27, no. 2, pp. 251–265, 2023.

[13] L. Tang, F. Li, and Z.-L. Chen, "Integrated scheduling of production and two-stage delivery of make-to-order products: Offline and online algorithms," *INFORMS J. on Computing*, vol. 31, no. 3, p. 493–514, 2019.

[14] L. Mönch, J. W. Fowler, and S. J. Mason, *Production planning and control for semiconductor wafer fabrication facilities: modeling, analysis, and systems*. Springer Science & Business Media, 2012, vol. 52.

[15] C. Zhang, W. Song, Z. Cao, J. Zhang, P. S. Tan, and X. Chi, "Learning to dispatch for job shop scheduling via deep reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1621–1632, 2020.

[16] L. Tang, T. Li, Y. Meng, and J. Liu, "Searching in symmetric solution space for permutation-related optimization problems," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 47, no. 8, pp. 7036–7052, 2025.

[17] T. Li, Y. Meng, and L. Tang, "Scheduling of continuous annealing with a multi-objective differential evolution algorithm based on deep reinforcement learning," *IEEE Transactions on Automation Science and Engineering*, vol. 21, no. 2, pp. 1767–1780, 2024.

[18] I. G. Smit, J. Zhou, R. Reijnen, Y. Wu, J. Chen, C. Zhang, Z. Bukhsh, Y. Zhang, and W. Nuijten, "Graph neural networks for job shop scheduling problems: A survey," *Computers & Operations Research*, vol. 176, p. 106914, 2025.

[19] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[20] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[21] A. Hottung, Y.-D. Kwon, and K. Tierney, "Efficient active search for combinatorial optimization problems," *arXiv preprint arXiv:2106.05126*, 2021.

[22] C. Su, C. Zhang, D. Xia, B. Han, C. Wang, G. Chen, and L. Xie, "Evolution strategies-based optimized graph reinforcement learning for solving dynamic job shop scheduling problem," *Applied Soft Computing*, vol. 145, p. 110596, 2023.

[23] E. Yuan, S. Cheng, L. Wang, S. Song, and F. Wu, "Solving job shop scheduling problems via deep reinforcement learning," *Applied Soft Computing*, vol. 143, p. 110436, 2023.

[24] Z. Liao, Q. Li, Y. Dai, and Z. Zhang, "Learning to schedule job-shop problems via hierarchical reinforcement learning," in *2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2022, pp. 3222–3227.

[25] J. v. Remmerden, Z. Bukhsh, and Y. Zhang, "Offline reinforcement learning for learning to dispatch for job shop scheduling," *Machine Learning*, vol. 114, no. 8, p. 191, 2025.

[26] S. Chen, Y. Tian, and L. An, "Multi-objective order scheduling via reinforcement learning," *Algorithms*, vol. 16, no. 11, p. 495, 2023.

[27] R. Chen, W. Li, and H. Yang, "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*, vol. 19, no. 2, pp. 1322–1331, 2022.

[28] Z. Liao, J. Chen, and Z. Zhang, "Solving job-shop scheduling problem via deep reinforcement learning with attention model," in *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*. Springer, 2023, pp. 201–212.

[29] J. Lee, S. Kee, M. Janakiram, and G. Runger, "Attention-based reinforcement learning for combinatorial optimization: Application to job shop scheduling problem," *arXiv preprint arXiv:2401.16580*, 2024.

[30] S. Yang, "Using attention mechanism to solve job shop scheduling problem," in *2022 2nd international conference on consumer electronics and computer engineering (ICCECE)*. IEEE, 2022, pp. 59–62.

[31] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[32] P. Tassel, M. Gebser, and K. Schekotihin, "A reinforcement learning environment for job-shop scheduling," *arXiv preprint arXiv:2104.03760*, 2021.

[33] E. Taillard, "Benchmarks for basic scheduling problems," *European Journal of Operational Research*, vol. 64, no. 2, pp. 278–285, 1993.

[34] J. Park, J. Chun, S. H. Kim, Y. Kim, and J. Park, "Learning to schedule job-shop problems: representation and policy learning using graph neural network and reinforcement learning," *International Journal of Production Research*, vol. 59, no. 11, pp. 3360–3377, 2021.