

# Job Shop Scheduling based on Hierarchical/Multi-agent Reinforcement Learning

①

Collaborative Agent FJSP

②

MARL JSP for Dynamic multi-agent  
manufacturing systems

③

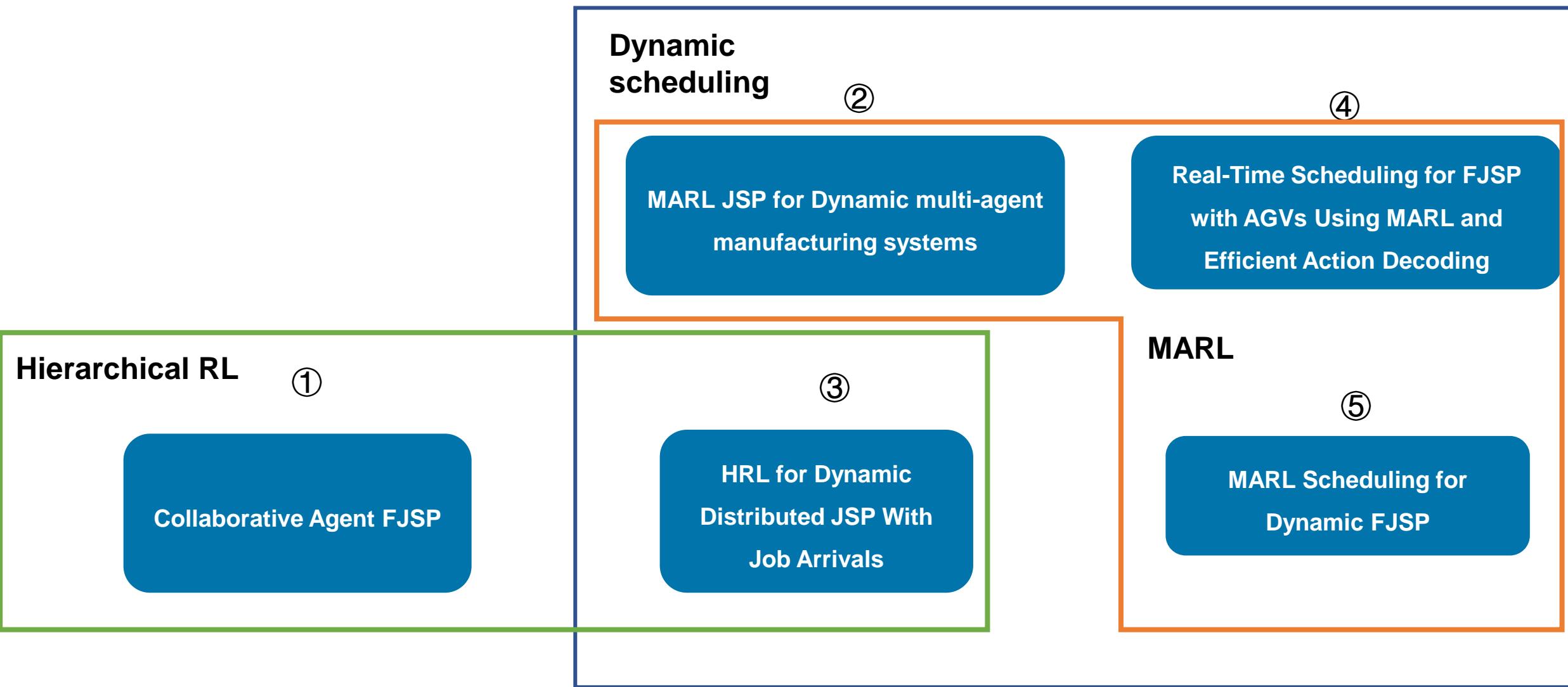
HRL for Dynamic  
Distributed JSP With  
Job Arrivals

④

Real-Time Scheduling for FJSP  
with AGVs Using MARL and  
Efficient Action Decoding

⑤

MARL Scheduling for Dynamic FJSP



## A novel collaborative agent reinforcement learning framework based on an attention mechanism and disjunctive graph embedding for **flexible job shop scheduling problem**

Wen quan Zhang, Fei Zhao, Yong Li, Chao Du, Xiaobing Feng, Xuesong Mei

**Publish Journal:** Journal of Manufacturing Systems(2024)



## Contents

- 01**  **Problem Description**
- 02**  **Method Design**
- 03**  **Experiment Results**
- 04**  **Conclusion**



## Contents

- 01**  Problem Description
- 02**  Method Design
- 03**  Experiment Results
- 04**  Conclusion

# Problem Description

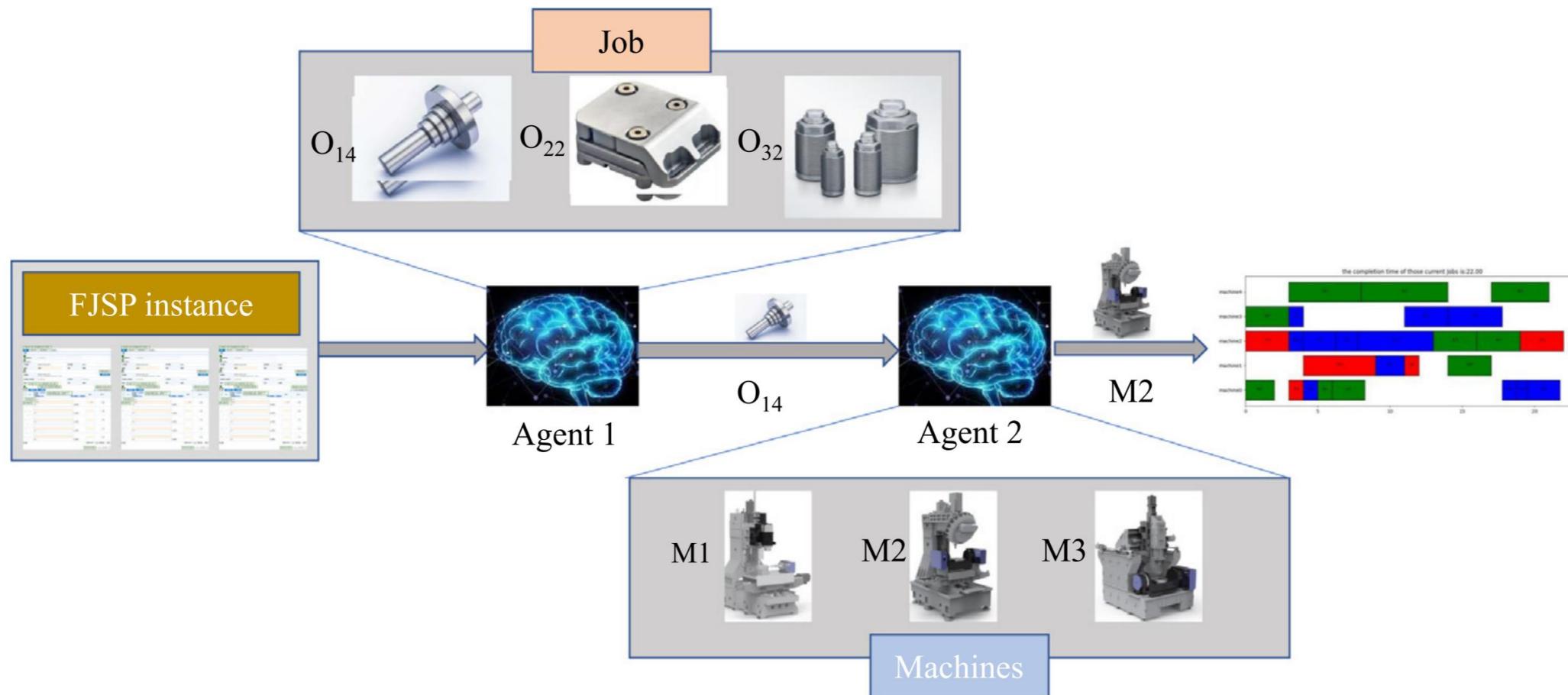
- A collaborative agent reinforcement learning (CARL) architecture was proposed, leveraging Graph Attention Network (GAT) and Transformer encoder for state representation, and trained with SAC and D5QN to optimize the Flexible Job Shop Scheduling Problem (FJSP).
- **Challenge:** Traditional state and action representations often lead to suboptimal solutions due to complexity and variability.
- **Method:** GAT was used to extract global state information, while the Transformer captured competitive relationships among machines, with SAC and D5QN optimizing the decision networks of agents.



## Contents

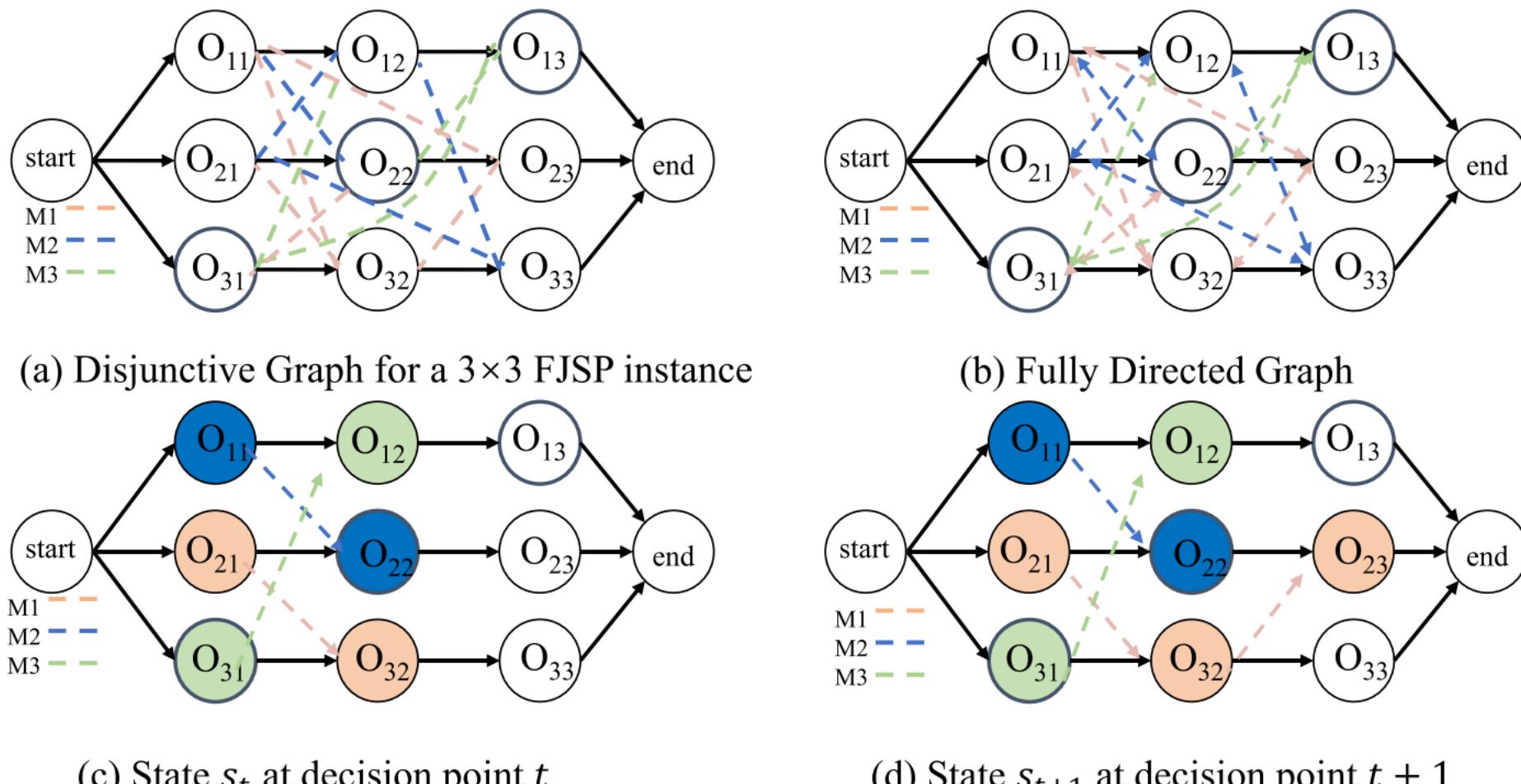
- 01**  Problem Description
- 02**  Method Design
- 03**  Experiment Results
- 04**  Conclusion

# Method Design



**Fig. 1.** CARL scheduling optimization model.

# Method Design



**Fig. 2.** Disjunctive graph model.

# Method Design

DQN

$$y_t = r_{t+1} + \gamma \max_{a'} Q^-(s_{t+1}, a'; \theta^-) \quad (5)$$

Double Deep Reinforcement Learning (DDQN)

$$y_t = r_{t+1} + \gamma Q' \left( s_{t+1}, \arg \max Q \left( s_{t+1}, a'; \theta \right); \theta' \right) \quad (6)$$

Noisy DQN (NDQN)

$$a_t = \arg \max_{a \in A} \tilde{Q} (s, a, \xi; \mu, \sigma)$$

Dueling DQN

$$Q(s, a) = V(s) + A(s, a)$$

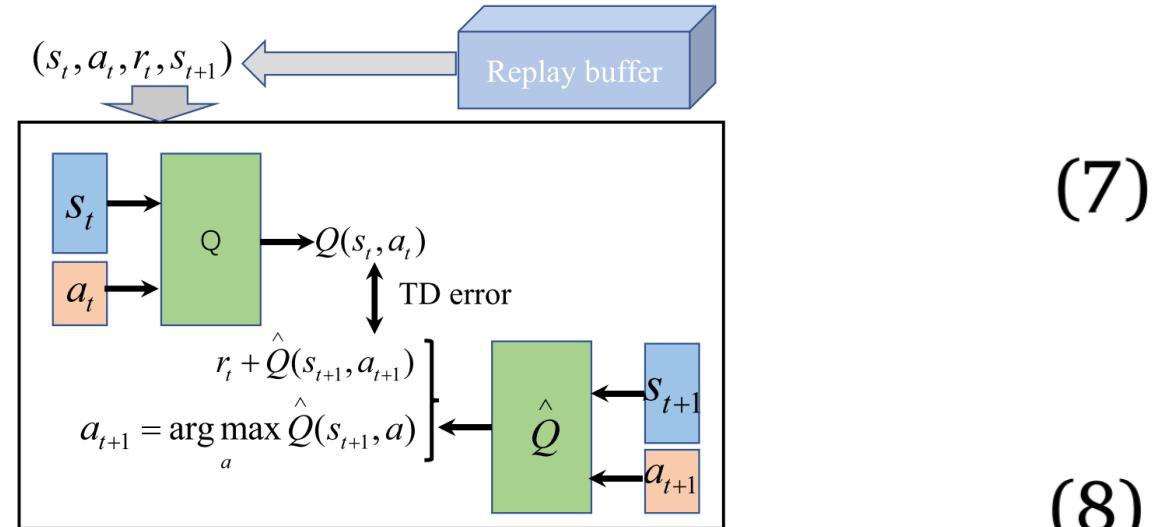


Fig. 3. Priority experience replay mechanism.

# Method Design

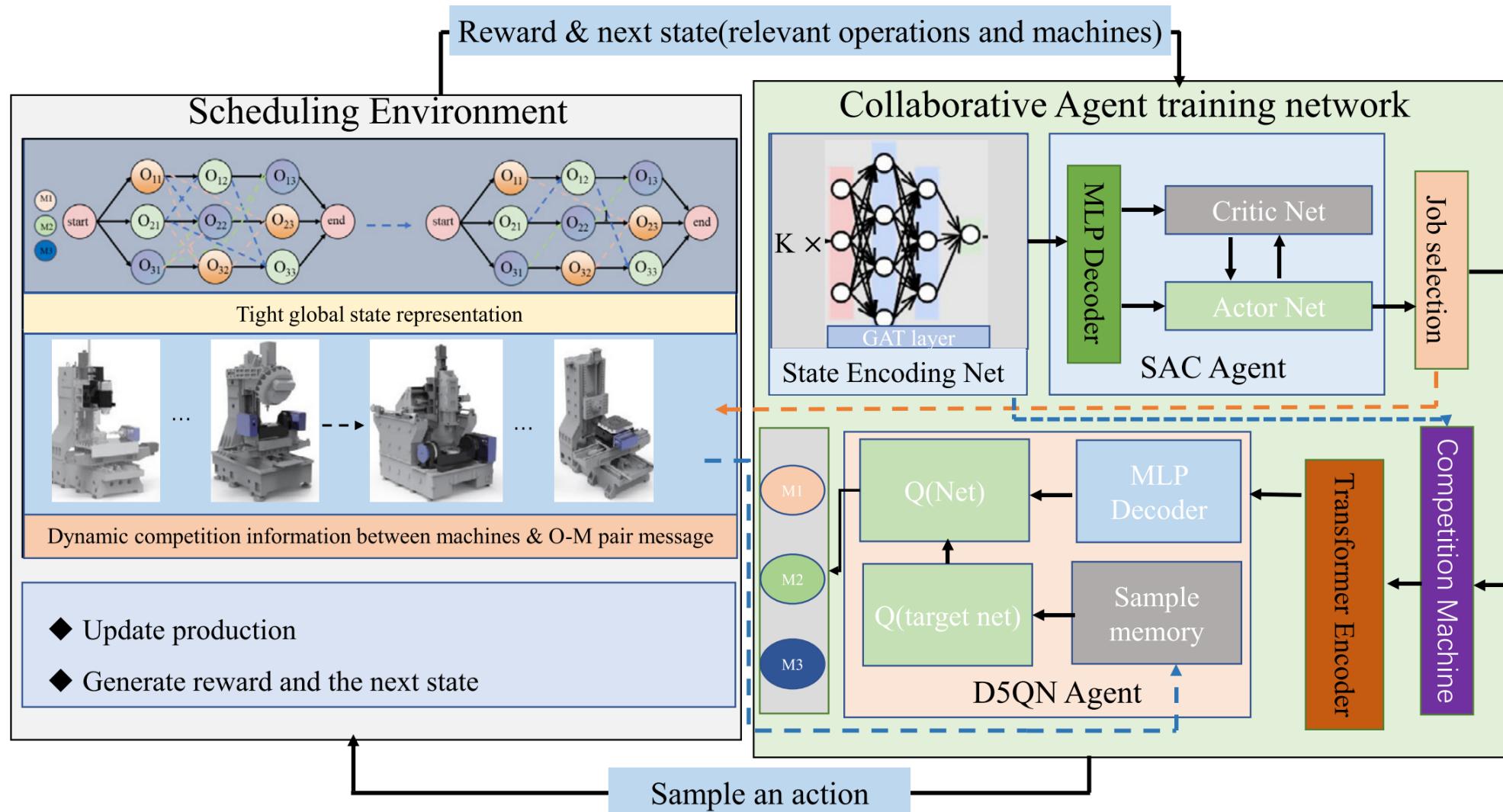


Fig. 4. Structural framework of the CARL algorithm for solving FJSP.

# Method Design

## MDP Formulation

---

- FJSP is formulated as a sequential decision-making problem requiring  $|O|$  decisions. The SAC agent selects an operation using sampling or greedy decoding, while the D5QN agent assigns it to a machine. This cooperative reinforcement learning problem is modeled as a Cooperative Markov Decision Process (coMDP)  $(S, A, P, r, \gamma)$  with a dual-layer action space  $A = A_o \times A_m$ .
- The state representation integrates global and local scheduling features. The global state  $s_o^t$  is derived from the disjunctive graph  $G(t) = (O, C \cup Du(t), D(t))$  classifying operations into completed, ongoing, and unplanned. The local state  $s_m^t$  represents machine statuses relevant to scheduling.
- At time  $t$ , an action  $a_t = (a_o^t, a_m^t)$  is selected, and the disjunctive arcs are updated, yielding new states  $s_o^{t+1}$  and  $s_m^{t+1}$ . The reward is defined as the makespan reduction between consecutive states. With a discount factor  $\gamma = 1$ , the cumulative reward is computed as the sum of rewards across all decision steps. Minimizing  $C_{max}$  is equivalent to maximizing the cumulative reward, making makespan reduction the agents' objective.
- The scheduling policy  $\pi(a_o, a_m | s)$  with parameters  $\theta = [\theta_o, \theta_m]$  guides decision-making, where  $\pi_{\theta_m}(a_o | s_t)$  and  $\pi_{\theta_m}(a_m | s_t, a_o)$  sample actions from operation and machine spaces  $A_o^t$  and  $A_m^t$ , respectively.

# Method Design

## Job operation encoder

---

$$e_{i,j,p} = \text{LeakyReLU} \left( \vec{\mathbf{a}}^\top \left[ \left( \mathbf{W} h_{O_{ij}} \right) \| \left( \mathbf{W} h_{O_{ip}} \right) \right] \right) \quad (15)$$

$$\alpha_{ijp} = \text{Softmax}(mask(e_{ijk})) \quad (16)$$

$$h'_{O_{ij}} = \sigma \left( \sum_{p=j-1}^{j+1} \alpha_{i,j,p} \mathbf{W} h_{O_{ip}} \right) \quad (17)$$

$$h^t_{O_{ij}} = \sigma \left( \frac{1}{K} \sum_{k=1}^K \sum_{p=j-1}^{j+1} \alpha_{i,j,p}^k \mathbf{W}^k h_{O_{ip}} \right) \quad (18)$$

$$h_{\mathcal{G}}^t = 1/\mathcal{O} \sum_{O_{ij} \in \mathcal{O}} h^t_{O_{ij}} \quad (19)$$

## Job operation decoder

---

$$sor_{t,O_{ij}}^o = MLP_{\theta_{\pi_o}}([h^t_{O_{ij}}, h_{\mathcal{G}}^t]), O_{ij} \in \{1, 2, \dots, O\} \quad (20)$$

# Method Design

## Machine encoder

$$P = \{M_k, M_q, \dots, M_v\}$$

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{(QK^T)}{\sqrt{d_k}}\right)V \quad (21)$$

$$\text{MultiHead}(Q, K, V) = [\text{head}_1, \dots, \text{head}_M] W^O \quad (22)$$

where  $\text{head}_i = \text{Attention} (QW_i^Q, KW_i^K, VW_i^V)$

## Machine decoder

$$sor_{t,M_P}^m = MLP_{\theta_{\pi_m}}([Z_{M_{P_i}}^t, h_{O-M}^t]), P_i \in P \quad (23)$$

$$a_t^m = \begin{cases} \arg \max_{a_{t,k}^m} (p_k (a_t^m)), \text{ random } < \varepsilon \\ \text{random.choice} (a_{t,k}^m), \text{ random } > \varepsilon \end{cases} \quad (24)$$

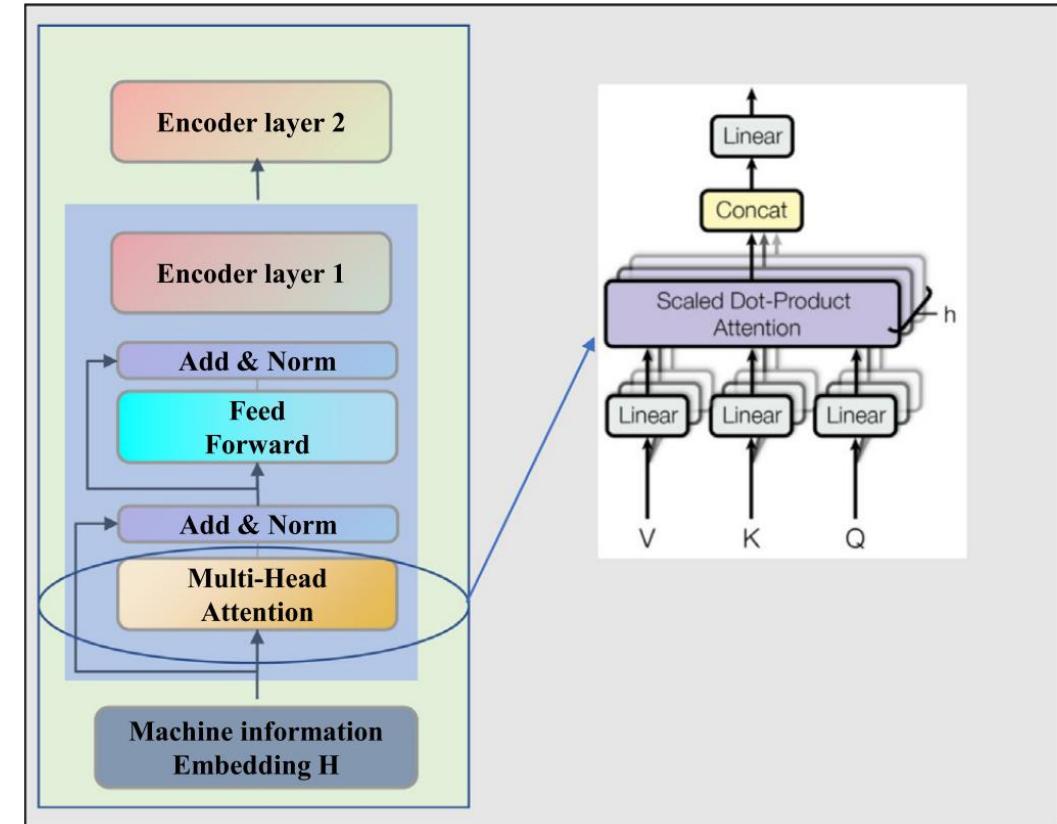


Fig. 5. Model competition machine information via a Transformer Encoder.

# Method Design

---

**Algorithm 1** Training CARL-FJSP Algorithm to dispatch

---

**Input:** Training episodes  $E_t$ , network update episodes  $E_s$ , batch size  $B$ ;empty replay buffer  $D$  and fixed validation data  $\varsigma_{val}$ ; SAC algorithm: Training the actor network  $\pi_{\theta_o}$  (parameter  $\theta_o$ ) and the critic network  $v_\phi$  (parameter  $\phi_1, \phi_2$ ). D5QN algorithm: Training the primary network  $Q_{\theta_m}$  (parameter  $\theta_m$ ), target network  $Q_{\theta_m^-}$  (parameters  $\theta_m^- = \theta_m$ ), execution network  $\pi_{\theta_m^{old}}$ , action selection probability parameter  $\varepsilon$ , Update network interval C.

**Output:**  $\pi_{\theta_o}$  (parameter  $\theta_o$ ),  $v_\phi$  (parameter  $\phi_1, \phi_2$ ),  $Q_{\theta_m}$  (parameter  $\theta_m$ )

```

1: Init  $\theta_m$  ,  $\theta_o$  ,  $\phi_1$  ,  $\phi_2$  ,  $\phi_o^{old} = \phi_o$  ,  $\theta_m^{old} = \theta_m$ ;
2: for  $e = 1, 2, \dots, E_t$  do
3:   Draw B FJSP Instances from GenerateInstance
4:   for  $b = 1, 2, \dots, B$  do
5:     for  $t = 1 \rightarrow T$  do
6:       Based on  $\pi_{\theta_o^{old}}$  using a greedy decision to execute  $a_{b,t}^o$  under  $s_{b,t}^o$  (c.f. Section 3.2) ;
7:       Based on  $\pi_{\theta_m^{old}}$  a probability  $\varepsilon$  to execute  $a_{b,t}^m$  according to  $s_{b,t}^m$  (c.f. Section 3.3) ;
8:       Observe reward  $r_{b,t}$  and next state  $s_{b,t+1}$ ;
9:        $y_{b,t} = \begin{cases} r_{b,t} + \gamma Q_{\theta_m^-} \left( s_{b,t+1}, \arg \max_{a'_{b,t}} Q_{\theta_m^{old}} \left( s_{b,t+1}, a'_{b,t}; \theta_m^{old} \right); \theta_m^- \right), \\ r_{b,t} \end{cases}$ ;
10:      Store  $(s_{b,t}, a_{b,t}, r_{b,t}, s_{b,t+1})$  in replay buffer D;
11:    end for
12:   end for
13:   for  $k = 1, 2, \dots, E_s$  do
14:     SAC cumulative LOSS:  $J_Q(\phi), J_\pi(\theta_i), J(\alpha)$  (c.f. Section 2.6);
15:     D5QN-loss:LossD5QN =  $\left( y_b - Q_{\theta_m} \left( s_{b,t}^m, a_{b,t}^m | \theta_m \right) \right)$ ;
16:     Every c steps reset  $Q_{\theta_m^-} = Q_{\theta_m^{old}}$  , soft update  $\phi_o = 0.8 * \phi_o + 0.2 * \phi_o^{old}$  ;
17:     Gradient descent updates all parameters;
18:   end for
19:   if Every Nr episodes then
20:     Resample B instances to form the training data;
21:   end if
22:   if Every  $N_{val}$  episodes then
23:     Validate  $\pi_\theta$  on  $\varsigma_{val}$ ;
24:   end if
25: end for

```

---



## Contents

- 01**  Problem Description
- 02**  Method Design
- 03**  Experiment Results
- 04**  Conclusion

# Experiment Results

## Dataset

Table 1

The top 6 results in hyperparameter tuning.

CARL-FJSP			Hyperparameters		1	2	3	4	5	6
Agent 1	GAT	number of attention heads	4	4	6	2	6	6		
		out feature dimension	8	8	16	8	32	16		
		dropout rate	0.9	0.9	0.95	0.9	0.9	0.95		
	SAC	learning rate	3e-4	3e-4	3e-3	3e-3	1e-3	3e-3		
		tau	5e-3	5e-3	5e-3	5e-3	5e-3	5e-3		
		decay ratio	0.9	0.95	0.95	0.95	0.95	1		
Agent 2	Transformer Encoder	hidden size	32	32	64	64	128	128		
		number of attention heads	4	4	8	2	4	8		
		number of encoder layers	2	2	2	3	3	3		
	D5QN	epsilon	0.8	0.8	0.85	0.8	0.8	0.8		
		gamma	1	1	1	1	1	1		
		learning rate	3e-4	3e-4	3e-3	3e-3	3e-4	3e-4		
Cmax			334.6	336.8	338.4	346.7	361.6	367.0		

# Experiment Results

## Baseline methods

**Table 2**

Heuristic scheduling rules.

Baseline rule	No.	Rule	Content
Operation selection rules	1	FIFO	select the earliest arriving job in the queue of a machine.
	2	MOPNR	select a job that has the greatest number of remaining operations to be done.
	3	LWKR	select a job that has the least total processing time remaining to be done
	4	SPT	select a job with the shortest processing time
	5	MWKR	select a job that has the most total processing time remaining to be done
Machine selection rules	1	SPT	select a machine with the shortest processing time for the operation of a job
	2	FIFO	select the earliest ready compatible machine.
	3	EET	select a machine that is idle at the earliest time

*RPD* (Relative Percent Deviation) measures the relative percentage deviation from the optimal values for different algorithms.

$$RPD = (C_{\max}/C_{\max}^{BS} - 1) \times 100\%$$

# Experiment Results

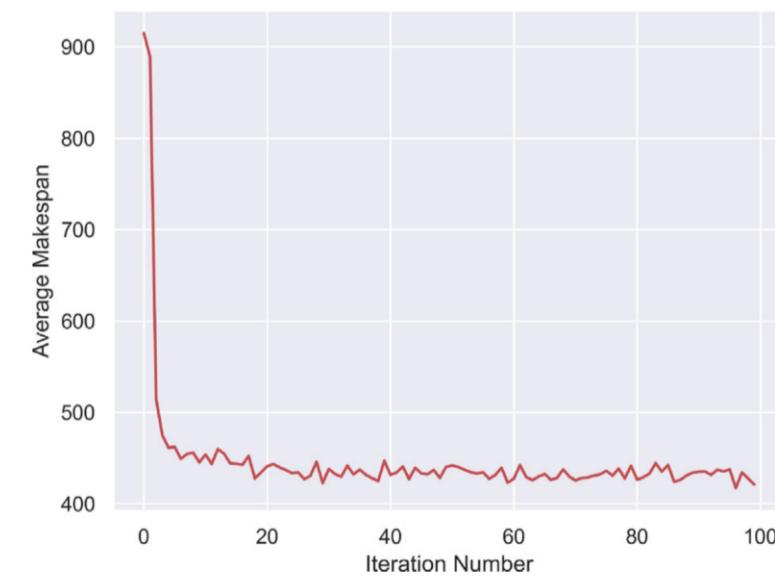


Fig. 6. Training curve on  $15 \times 10$  instances.

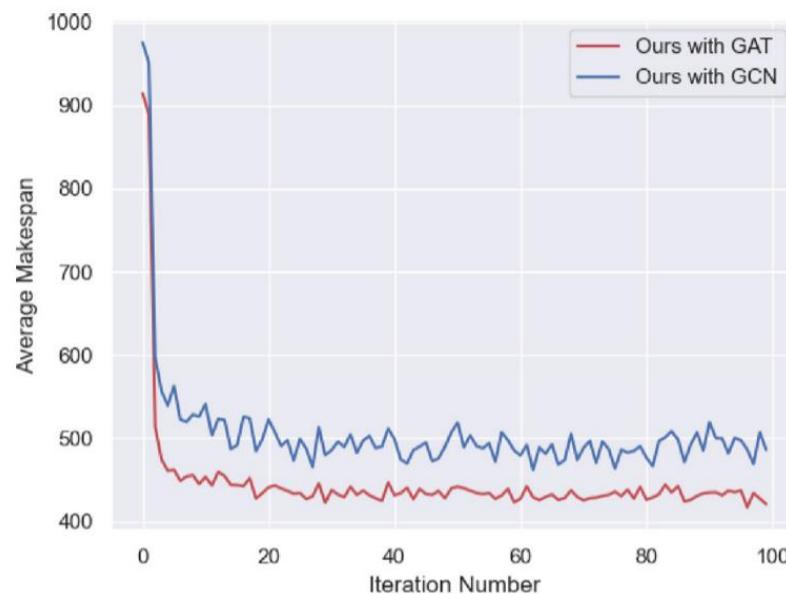


Fig. 7. Verification of the effectiveness of the GAT component.

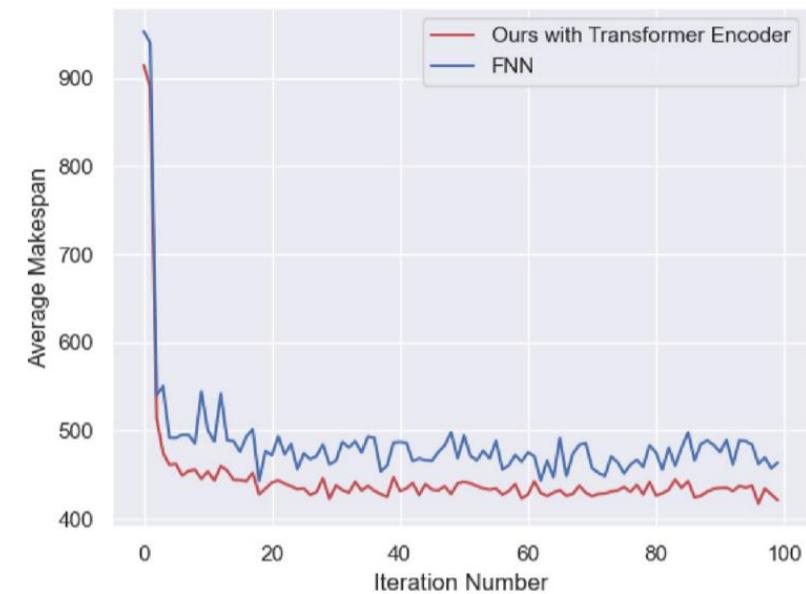


Fig. 8. Verification of the effectiveness of the Transformer Encoder component.

# Experiment Results

**Table 3**

Results of all methods on synthetic instances of training size.

Size	Ours	FIFO	SPT	MOPNR	MWKR	LWKR	FIFO	FIFO	MWKR	MOPNR	LWKR	[41]	[41]	OR-Tools	
		+FIFO	+SPT	+EET	+EET	+SPT	+SPT	+ EET	+ SPT	+SPT	+EET	20×5	20×10		
10×5	Obj	<b>398.6</b>	567.85	515.9	558.36	548.54	562.85	479.54	571.01	484.12	484.16	795.5	572.01	552.72	326.24(100%)
	Gap	<b>22.2%</b>	76.10%	58.42%	72.83%	69.61%	72.53%	46.99%	75.03%	48.39%	48.41%	143.84%	75.33%	69.42%	-
	Time(s)	2.03	0.052	0.057	0.058	0.059	0.016	0.019	<b>0.014</b>	0.016	0.016	0.015	0.324	0.318	8.945
10×10	Obj	<b>403.53</b>	718.34	542.6	696.98	690.7	618.25	486.03	698.5	476.36	485.54	1205.99	675.5	692.44	329.31(100%)
	Gap	<b>22.5%</b>	120.63%	66.19%	113.99%	112.19%	87.74%	47.59%	112.11%	44.65%	47.44%	266.22%	105.13%	110.27%	-
	Time(s)	4.87	0.073	0.086	0.085	0.086	0.034	0.041	<b>0.031</b>	0.035	0.036	0.034	0.63	0.623	2.40
15×5	Obj	<b>540.02</b>	802.46	676.61	801.62	793.81	731.35	656.06	806.81	665.57	664.71	1056.3	798.24	789.93	462.27(27%)
	Gap	<b>16.82%</b>	74.41	46.62%	74.40%	72.61%	58.21%	41.92%	74.53%	43.98%	43.79%	128.50%	72.68%	70.88%	-
	Time(s)	3.36	0.053	0.056	0.056	0.057	0.028	0.033	<b>0.023</b>	0.028	0.029	0.026	0.498	0.514	1438.4
15×10	Obj	<b>491.32</b>	868.76	701.98	840.73	868.76	784.61	610.08	874.68	593.15	612.14	1513.02	841.55	832	377.13(15%)
	Gap	<b>30.28%</b>	131.69	86.47	124.29	121.20%	108.05%	61.77%	131.93%	57.28%	62.32%	301.19%	123.15%	120.61%	-
	Time(s)	6.62	0.112	0.116	0.121	0.123	0.061	0.073	<b>0.051</b>	0.059	0.063	0.057	1.008	1.008	1582.4
20×5	Obj	<b>679.25</b>	1044.95	836.1	1054	1023.98	895.44	810.23	1050.38	818.08	813.92	1299.5	1030.44	1025.5	598.15(0%)
	Gap	<b>13.56%</b>	75.74%	40.05%	77.26%	72.36%	49.70%	35.46%	75.60%	36.77%	36.07%	117.25%	72.27%	71.45%	-
	Time(s)	4.54	0.072	0.09	0.089	0.09	0.043	0.049	<b>0.034</b>	0.041	0.045	0.04	0.644	0.644	1800

(%): The number of scheduling results that can be obtained within 1800s;

For OR-Tools, the solution and the ratio of optimally solved instances are reported.

# Experiment Results

**Table 4**

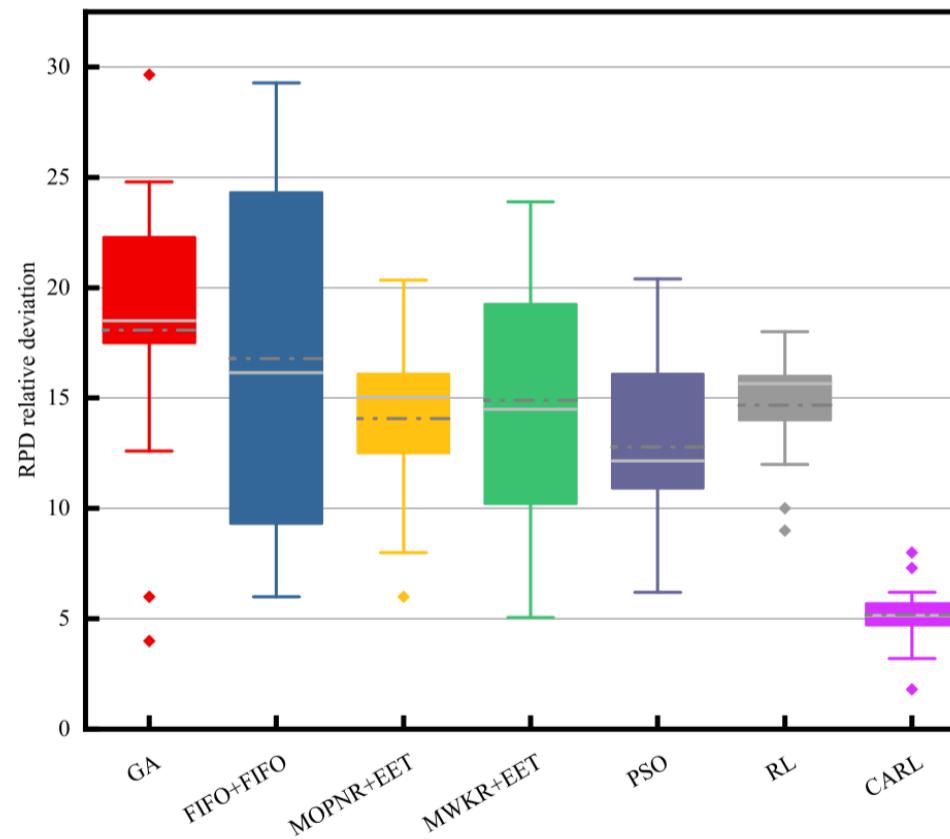
Results on the large-sized synthetic instances.

Size		Ours	FIFO	SPT	MOPNR	MWKR	LWKR	FIFO	FIFO	MWKR	MOPNR	LWKR	[41] 20×5	[41] 20×10	OR-Tools
			+FIFO	+SPT	+EET	+EET	+SPT	+ EET	+ SPT	+SPT	+EET	20×5	20×10		
20×10	Obj	<b>579.63</b>	1089.61	825.4	1055.46	1041.89	925.18	725.69	1082.34	712.86	724.71	791.18	1044.53	1045.1	456.58(1%)
	Gap	<b>26.95%</b>	139.55%	80.83%	132.05%	129.09%	102.63%	58.94%	137.05%	56.13%	58.73%	73.28%	128.77%	128.90%	-
	Time(s)	9.42	0.161	0.176	0.181	0.178	0.087	0.107	0.075	0.09	0.097	<b>0.016</b>	1.412	1.369	1783.6
30×20	Obj	<b>674.53</b>	1671.18	1158.75	1586.53	1573.13	1202.55	856.6	1672.89	832.09	852.87	3525.68	1574.22	1620.11	598.15(0%)
	Gap	<b>12.77%</b>	180.31%	94.06%	166.05%	163.86%	101.04%	43.21%	179.68%	39.11%	42.58%	489.43%	163.18%	170.85%	-
	Time(s)	31.85	0.812	0.833	0.84	0.842	0.4	0.507	<b>0.348</b>	0.415	0.457	0.398	4.076	4.206	1802

(%): The number of scheduling results that can be obtained within 1800s;

For OR-Tools, the solution and the ratio of optimally solved instances are reported.

# Experiment Results



**Fig. 9.** RPD comparison box chart of different algorithms on the mk dataset. (the statistics are computed from 100 simulations).

# Experiment Results

**Table 5**

Result comparison on Brandimarte's instances between CARL algorithm and Other algorithms.

Instance(size)	Ours	FIFO +FIFO	SPT +SPT	MOPNR +EET	MWKR +EET	LWKR +SPT	FIFO +SPT	FIFO + EET	MWKR + SPT	MOPNR +SPT	LWKR +EET	[41] 20×5	[41] 20×10	GA	PSO	UB
MK01(10×6)	Obj	44	48	59	49	46	76	64	47	53	61	78	55	48	42	46
	RPD	12.8%	23.1%	51.3%	25.6%	17.9%	94.9%	64.1%	20.5%	35.9%	56.4%	100.0%	41%	23.1%	7.7%	17.9%
MK02(10×6)	Obj	31	46	46	40	41	43	42	40	37	41	59	42	43	41	35
	RPD	19.2%	76.9%	76.9%	53.8%	57.7%	65.4%	61.5%	53.8%	42.3%	57.7%	126.9%	61.5%	65.4%	57.5%	34.6%
MK03(15×8)	Obj	207	220	303	216	213	362	338	207	342	330	325	232	213	238	212
	RPD	1.5%	7.8%	48.5%	5.9%	4.4%	77.5%	65.7%	1.5%	67.6%	61.8%	59.3%	13.7%	4.4%	16.7%	3.9%
MK04(15×8)	Obj	69	78	76	80	71	181	169	76	167	174	108	82	75	74	71
	RPD	15.0%	30.0%	26.7%	33.3%	18.3%	201.7%	181.7%	26.7%	178.3%	190.0%	80.0%	36.7%	25.0%	23.3%	18.3%
MK05(15×4)	Obj	177	186	226	191	186	279	251	194	280	269	250	205	185	188	185
	RPD	2.9%	8.1%	31.4%	11.0%	8.1%	62.2%	45.9%	12.8%	62.8%	56.4%	45.3%	19.2%	7.6%	9.3%	7.6%
MK06(10×10)	Obj	77	99	89	102	108	123	108	99	104	104	185	110	103	115	98
	RPD	32.8%	70.7%	53.4%	75.9%	86.2%	112.1%	86.2%	70.7%	79.3%	79.3%	219.0%	89.7%	77.6%	98.3%	69.0%
MK07(20×5)	Obj	151	214	214	219	212	230	232	212	217	217	273	215	214	183	176
	RPD	8.6%	54.0%	54.0%	57.6%	52.5%	65.5%	66.9%	52.5%	56.1%	56.1%	96.4%	54.7%	54.0%	31.7%	26.6%
MK08(20×10)	Obj	531	531	664	523	535	651	587	531	587	592	736	525	523	523	557
	RPD	1.5%	1.5%	27.0%	0.0%	2.3%	24.5%	12.2%	1.5%	12.2%	13.2%	40.7%	0.4%	0.0%	0.0%	6.5%
MK09(20×10)	Obj	334	372	448	342	355	511	466	349	456	462	616	424	333	361	345
	RPD	8.8%	21.2%	45.9%	11.4%	15.6%	66.4%	51.8%	13.7%	48.5%	50.5%	100.7%	38.1%	8.5%	17.6%	12.4%
MK10(20×15)	Obj	245	278	358	268	264	427	365	279	369	358	492	266	266	319	247
	RPD	24.4%	41.4%	81.7%	36.0%	34.0%	116.8%	85.3%	41.6%	87.3%	81.7%	149.7%	35.0%	35.0%	61.9%	25.4%
Average	Obj	186	207	248	203	203	288	262	203	261	261	312	216	200	208	197
	RPD	5.7%	19.7%	43.4%	17.3%	17.3%	66.5%	51.4%	17.3%	50.9%	50.9%	80.3%	24.9%	15.6%	20.2%	13.9%

UB column that is the best result from literature, and '\*' denotes the result is optimal

# Experiment Results

**Table 6**

Result comparison on Hurink's vdata instances between CARL algorithm and Other algorithms.

Instances(size)		Ours	FIFO +FIFO	SPT +SPT	MOPNR +EET	MWKR +EET	LWKR +SPT	FIFO +SPT	FIFO + EET	MWKR + SPT	MOPNR +SPT	LWKR +EET	[41] 20×5	[41] 20×10	UB
la1-5(10×5)	Obj	<b>544.4</b>	576.1	631.9	559.6	555.4	803.0	790.0	613.4	788.0	602.0	749.6	580.4	550.4	507*
	RPD	<b>7.4%</b>	13.6%	24.6%	10.4%	9.5%	58.4%	55.8%	21.0%	55.4%	18.7%	47.9%	14.5%	8.6%	
la6-10(15×5)	Obj	822.2	850.8	973.0	838.4	823.3	1263.	1146.0	842.8	1196.0	861.8	1061.4	845.6	<b>821.8</b>	794*
	RPD	3.6%	7.2%	22.5%	5.6%	3.7%	659.1%	44.3%	6.1%	50.6%	8.5%	33.7%	6.5%	<b>3.5%</b>	
la11-15(20×5)	Obj	<b>1062.8</b>	1089.6	1181.1	1080.6	1063.5	1464.6	1283.4	1090	1398.6	1094.8	1355.6	1072.8	1064	1040.8*
	RPD	<b>2.1%</b>	4.7%	13.5%	3.8%	2.2%	40.7%	23.3%	4.7%	34.4%	5.2%	30.2%	3.1%	2.2%	
la16-20(10×10)	Obj	<b>695</b>	716.6	728.7	719.1	714.0	1339.2	1002.6	716.4	1360	737.6	1138.0	715.6	703.2	679.8*
	RPD	<b>2.2%</b>	5.4%	7.2%	5.8%	5.0%	97.0%	47.5%	5.4%	100.1%	8.5%	67.4%	5.3%	3.4%	
la21-25(15×10)	Obj	858.4	874.1	981.3	847.9	866.9	1607.8	1232.6	894.4	1702.8	936.8	1277.2	848.2	<b>825.2</b>	777.2
	RPD	10.4%	12.5%	26.3%	9.1%	11.5%	106.9%	58.6%	15.1%	119.1%	20.5%	64.3%	9.1%	<b>6.2%</b>	
la26-30(20×10)	Obj	1100	1141.4	1376.7	1101.2	1119.8	2169.4	1603.6	1119.8	2136.6	1175.	1538.8	1107.8	<b>1085.4</b>	1054.2
	RPD	4.3%	8.3%	30.6%	4.5%	6.2%	105.8%	52.1%	6.2%	102.7%	411.5%	46.0%	5.1%	<b>3.0%</b>	
la31-35(30×10)	Obj	<b>1580.6</b>	1619.4	1786.7	1595.0	1589.5	2598.0	2136.0	1611.2	2639.6	1657.2	2322.4	1588.2	1614.0	1552.2*
	RPD	<b>1.8%</b>	4.3%	15.1%	2.8%	2.4%	67.4%	37.6%	3.8%	70.1%	6.8%	49.6%	2.3%	4.0%	
la36-40(15×15)	Obj	<b>970.8</b>	995.0	1003.7	986.4	986.8	1991.4	1589.6	1002.4	2039.2	1087.8	1601.2	983.6	983.4	950.8*
	RPD	<b>2.1%</b>	4.6%	5.6%	3.7%	3.8%	109.4%	67.2%	5.4%	114.5%	14.4%	68.4%	3.4%	3.4%	
Average	Obj	<b>954</b>	980.0	1083.0	966.0	961.8	1655	1348	986.0	1658.0	1019.0	1381.0	968	959	920
	RPD	<b>3.7%</b>	7.6%	17.8%	5.1%	4.6%	79.9%	46.5%	7.3%	80.3%	10.8%	50.1%	5.2%	3.8%	

UB column that is the best result from literature, and '\*' denotes the result is optimal

# Experiment Results

**Table 7**

Result comparison on Hurink's rdata instances between CARL algorithm and Other algorithms.

Instances(size)		Ours	FIFO +FIFO	SPT +SPT	MOPNR +EET	MWKR +EET	LWKR +SPT	FIFO +SPT	FIFO + EET	MWKR + SPT	MOPNR +SPT	LWKR +EET	[41] 20×5	[41] 20×10	UB
la1-5(10×5)	Obj	<b>549.8</b>	585.6	676.8	576	594.8	893.6	677	598.4	784.6	804.4	781.4	617.2	574.8	507.6
	RPD	<b>8.30%</b>	15.40%	33.30%	13.50%	17.20%	76.00%	33.40%	17.90%	54.60%	58.50%	53.90%	21.60%	13.20%	
la6-10(15×5)	Obj	<b>838.2</b>	865.6	1012	855.7	841.2	1256.4	1002.8	856.8	1090.4	1036.4	1129.8	878.8	839.6	794.2
	RPD	<b>5.50%</b>	9.00%	27.40%	7.70%	5.90%	58.20%	26.30%	7.90%	37.30%	30.50%	42.30%	10.70%	5.70%	
la11-15(20×5)	Obj	<b>1093</b>	1137.6	1223.7	1126.2	1095	1500.6	1327.4	1147	1354.4	1343.4	1416.4	1121.2	1105.4	1041
	RPD	<b>5.00%</b>	9.30%	17.60%	8.20%	5.20%	44.10%	27.50%	10.20%	30.10%	29.00%	36.10%	7.70%	6.20%	
la16-20(10×10)	Obj	<b>823</b>	864.1	887.8	860	842.4	1294.6	1063.6	830.2	1030.4	1042.8	1303.2	875.8	884.4	697
	RPD	<b>18.10%</b>	24.00%	27.40%	23.40%	20.90%	85.70%	52.60%	19.10%	47.80%	49.60%	87.00%	25.70%	26.90%	
la21-25(15×10)	Obj	958.2	1037.6	1140.9	1003.8	987.3	1816.4	1334.4	1043	1261.4	1398.2	1654.4	1004.6	<b>947.6</b>	807.2
	RPD	18.70%	28.50%	41.30%	24.40%	22.30%	125.00%	65.30%	29.30%	56.30%	73.20%	105%	24.50%	<b>17.40%</b>	
la26-30(20×10)	Obj	1182.4	1252	1441	1224.6	1191	2095.8	1772	1236	1585.4	1646.4	2068	1247.6	<b>1175.6</b>	1062
	RPD	11.30%	17.90%	35.70%	15.30%	12.10%	97.30%	66.90%	16.40%	49.30%	55.00%	94.70%	17.50%	<b>10.70%</b>	
la31-35(30×10)	Obj	1651	1692.2	1910	1656.3	1642	2650.6	2050.2	1683.4	2217	2111.8	2578	1637.4	<b>1612.8</b>	1553
	RPD	6.30%	9.00%	23.00%	6.70%	5.70%	70.70%	32.00%	8.40%	42.80%	36.00%	66.00%	5.40%	<b>3.90%</b>	
la36-40(15×15)	Obj	<b>1198.8</b>	1262	1311	1231	1231	2095.6	1612.8	1273	1579	1603.4	2132.4	1259.8	1212	1013
	RPD	<b>18.30%</b>	24.60%	29.40%	21.50%	21.50%	106.90%	59.20%	21.50%	55.90%	58.30%	111%	24.40%	19.60%	
Average		<b>1037</b>	1087	1200	1043	1053	1700	1355	1083	1363	1373	1633	1080	1044	934
		<b>11.00%</b>	16.30%	28.40%	11.70%	12.70%	82.00%	45.10%	15.90%	45.90%	47.00%	74.80%	15.60%	11.80%	

UB column that is the best result from literature

# Experiment Results

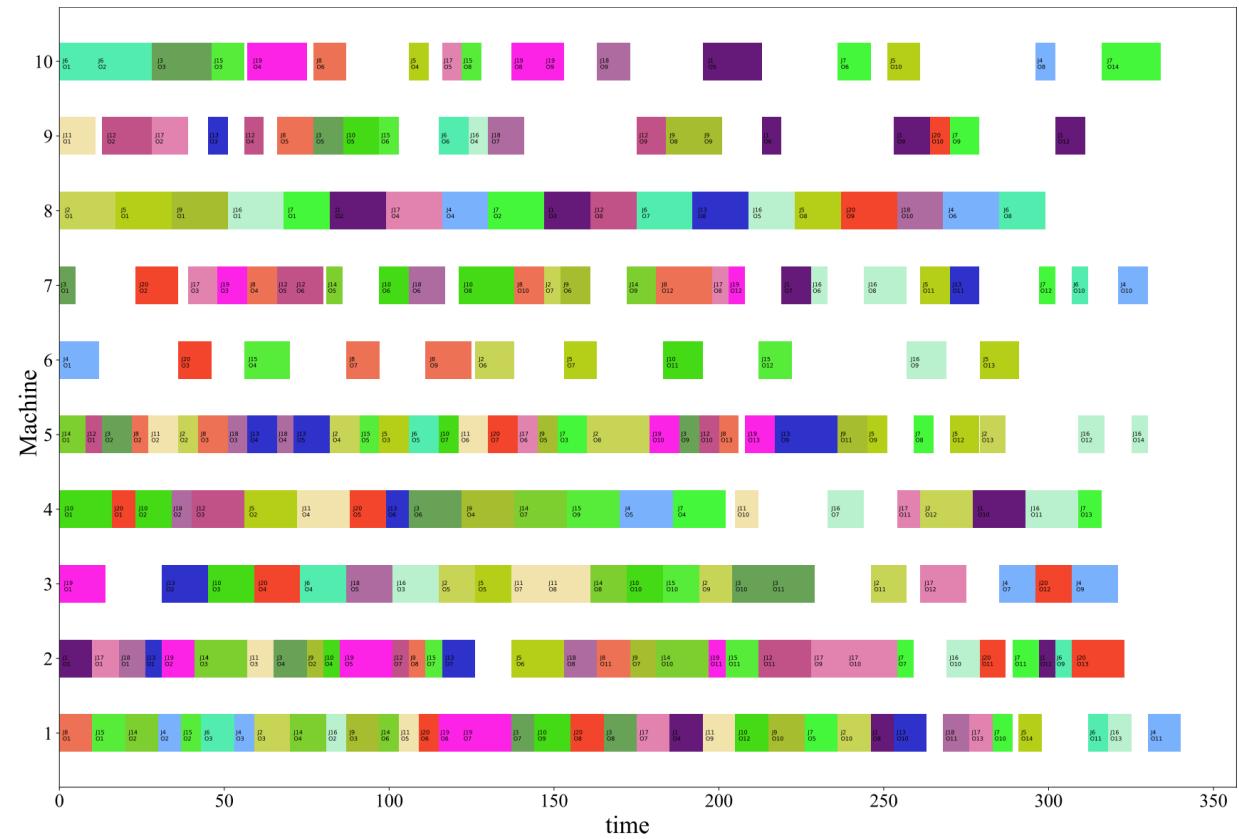


Fig. 10. The Gantt charts of our method on the mk09 dataset.

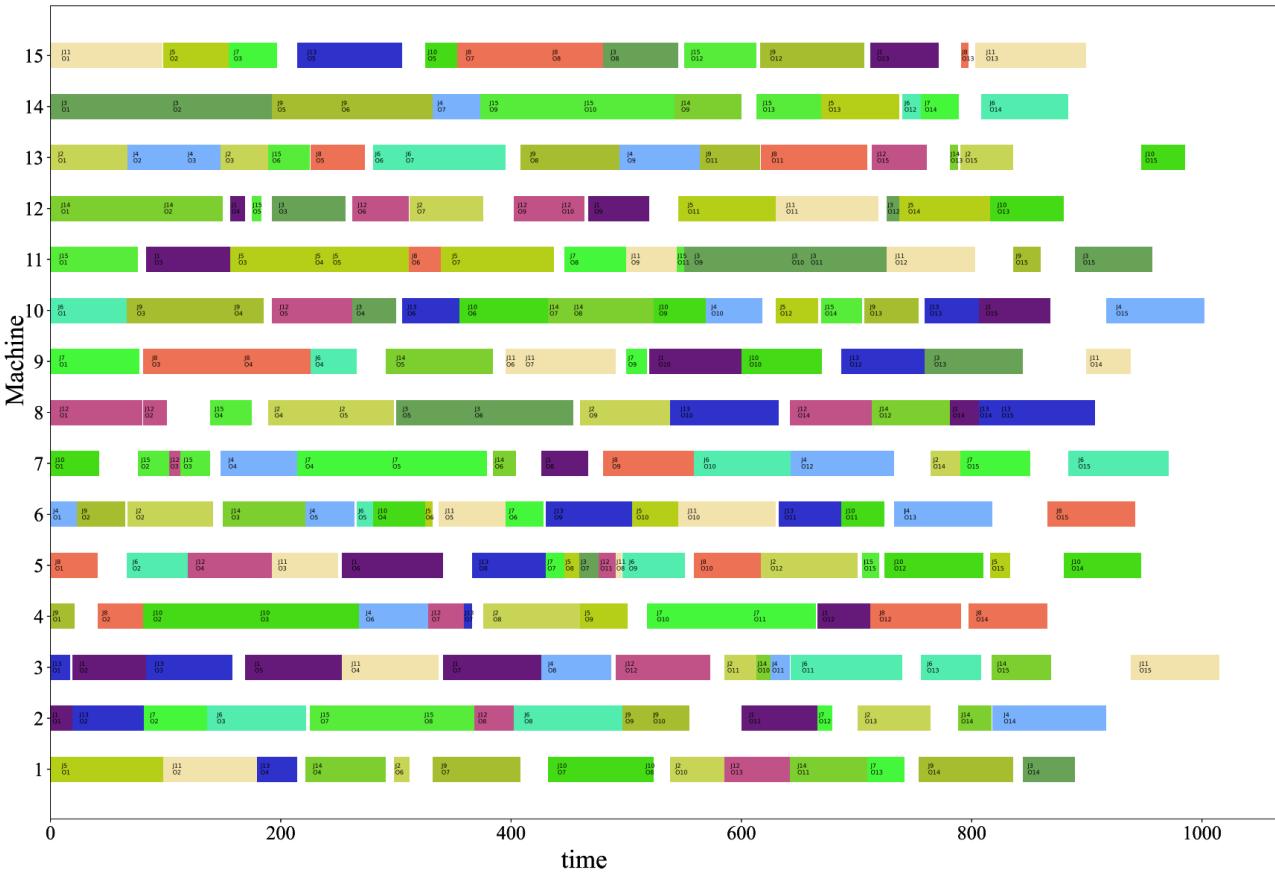


Fig. 11. The Gantt charts of our method on the vdata40 dataset.

# Experiment Results

**Table 8**

Result comparison on DPpaulii's instances between CARL algorithm and dispatching rules.

Instances Size	Ours	FIFO +FIFO	SPT +SPT	MOPNR +EET	MWKR +EET	LWKR +SPT	FIFO +SPT	FIFO + EET	MWKR + SPT	MOPNR +SPT	LWKR +EET	
10×5	DPpaulii1a	2980	3113	<b>2852</b>	3078	3179	3983	3123	3189	3313	3263	3718
	DPpaulii2a	<b>2483</b>	2526	2772	2973	2651	4388	3254	2839	3039	3236	4244
	DPpaulii3a	<b>2366</b>	2398	3128	2580	2432	4396	3016	3344	3105	3164	3760
	DPpaulii4a	3055	3000	<b>2889</b>	3018	3106	3917	3126	3102	3136	3206	3878
	DPpaulii5a	2422	<b>2419</b>	2485	2769	2602	3895	3020	3146	2977	2959	4141
	DPpaulii6a	<b>2330</b>	2502	3054	2527	2446	4233	3139	2993	3109	3157	3792
15×8	DPpaulii7a	2854	<b>2768</b>	2974	3052	3047	4073	3200	3087	3121	3149	3812
	DPpaulii8a	2327	<b>2319</b>	2632	2454	2452	3771	2943	3312	2933	3207	3999
	DPpaulii9a	<b>2171</b>	2240	2521	2368	2257	4321	3183	3087	3011	3263	3720
	DPpaulii10a	2819	2840	<b>2808</b>	3000	3018	4015	3110	3312	3202	3098	4145
	DPpaulii11a	2352	<b>2260</b>	2596	2544	2455	3734	3032	3104	3291	3038	3682
	DPpaulii12a	<b>2147</b>	2236	2255	2361	2233	3872	2720	2957	2855	2796	4016
20×10	DPpaulii13a	<b>2836</b>	2924	3487	2967	2838	4368	3095	3168	3054	3193	4482
	DPpaulii14a	<b>2351</b>	2374	2716	2578	2441	3930	3550	3315	3220	3165	4065
	DPpaulii15a	<b>2301</b>	2400	2505	2435	2306	4259	3339	3298	3181	3249	4482
	DPpaulii16a	<b>2815</b>	3050	2989	2843	2852	3899	3202	3178	3012	3078	3731
	DPpaulii17a	<b>2392</b>	2467	2441	2479	2416	4167	3200	3252	3301	3092	4209
	DPpaulii18a	<b>2242</b>	2328	2677	2422	2290	4025	2897	2976	2891	2978	4826

# Experiment Results

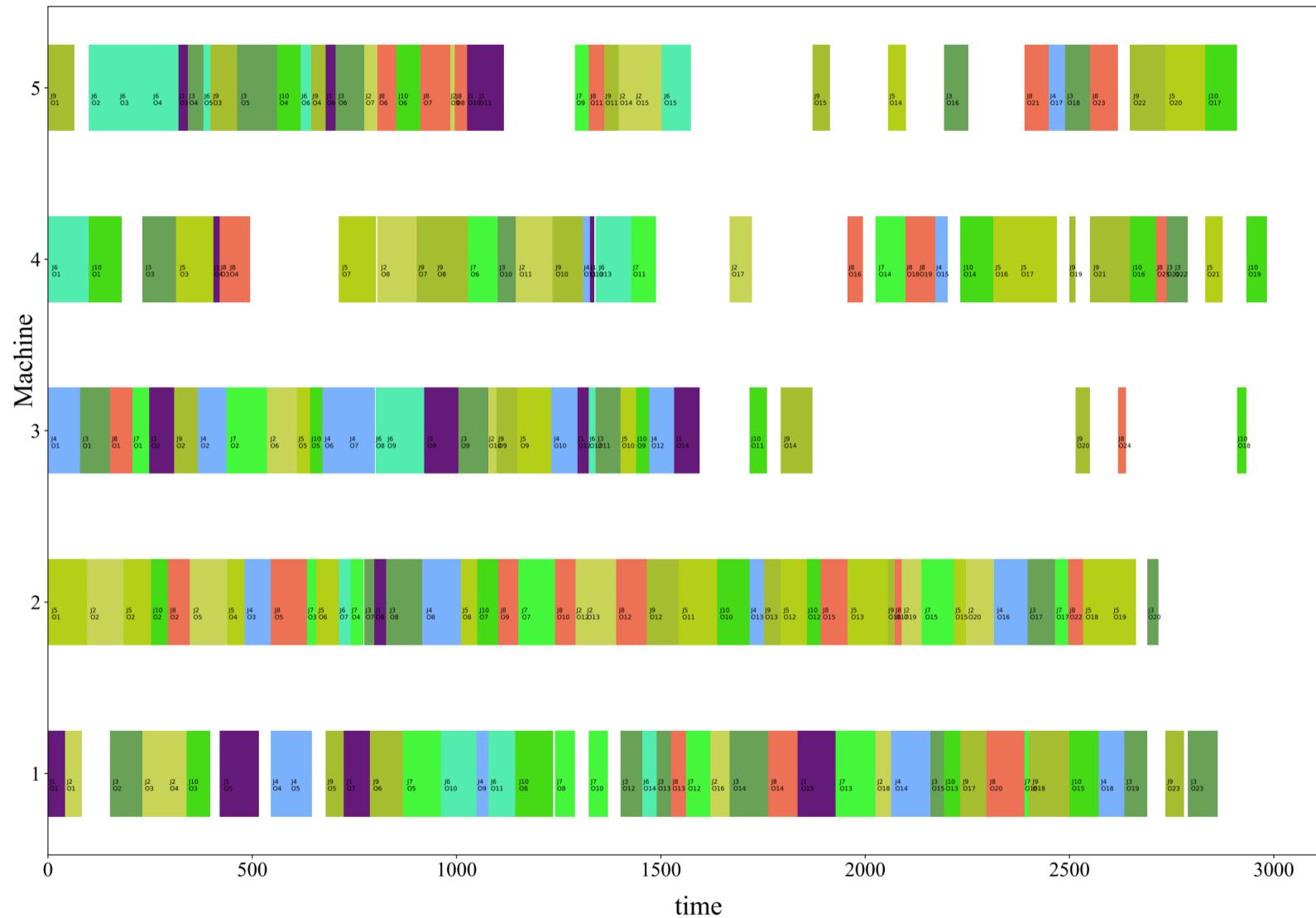


Fig. 12. The Gantt charts of our method on the DPpaullii1a dataset.



## Contents

- 01**  Problem Description
- 02**  Method Design
- 03**  Experiment Results
- 04**  Conclusion

# Conclusion

①

This study proposes the CARL model to tackle FJSP, integrating cooperative DRL for operation selection and machine allocation, achieving a 55.52% performance improvement on large-scale instances without specific training while demonstrating strong generalization and robustness.

②

Future research may explore innovative disjunctive graph embeddings and advanced machine modeling, alongside diverse RL agents such as SARSA, DQN, PPO, and A2C for job and machine selection.

## Dynamic job shop scheduling based on deep reinforcement learning for multi-agent manufacturing systems

Yi Zhang, Haihua Zhu \*, Dunbing Tang, Tong Zhou, Yong Gui

**Publish Journal:** Robotics and Computer-Integrated Manufacturing (2022)



## Contents

- 01**  **Problem Description**
- 02**  **Method Design**
- 03**  **Experiment Results**
- 04**  **Conclusion**



## Contents

- 01**  Problem Description
- 02**  Method Design
- 03**  Experiment Results
- 04**  Conclusion

# Problem Description

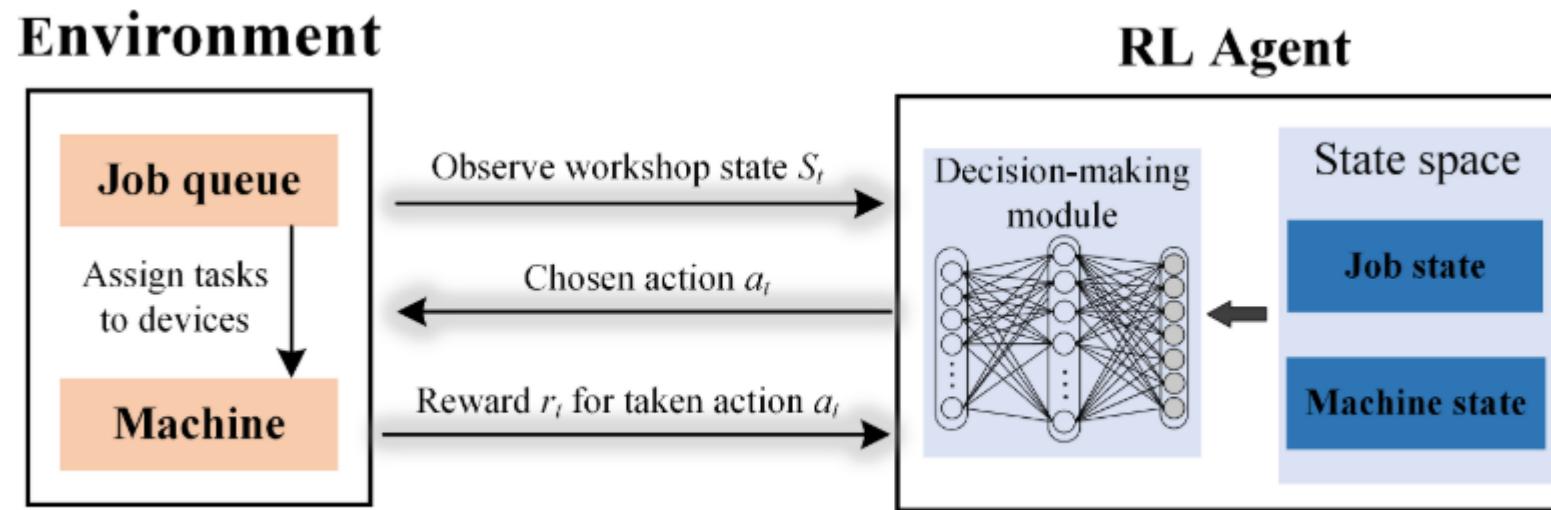
- This study proposes a multi-agent manufacturing system based on deep reinforcement learning to enhance the dynamic response and self adjustment capabilities of workshops to personalized orders.
- **Challenge:** Traditional scheduling methods struggle to maintain efficient operation in dynamic workshop environments with uncertainties such as random job insertions and equipment failures.
- **Method:** The system utilizes an improved Contract Network Protocol (CNP) to coordinate agent competition and cooperation, and optimizes the decision-making ability of the AI scheduler through PPO algorithm to achieve autonomous scheduling and real-time adjustment.



## Contents

- 01**  Problem Description
- 02**  Method Design
- 03**  Experiment Results
- 04**  Conclusion

# Method Design



**Fig. 1.** The design of RL-based scheduling method in a smart workshop.

# Method Design

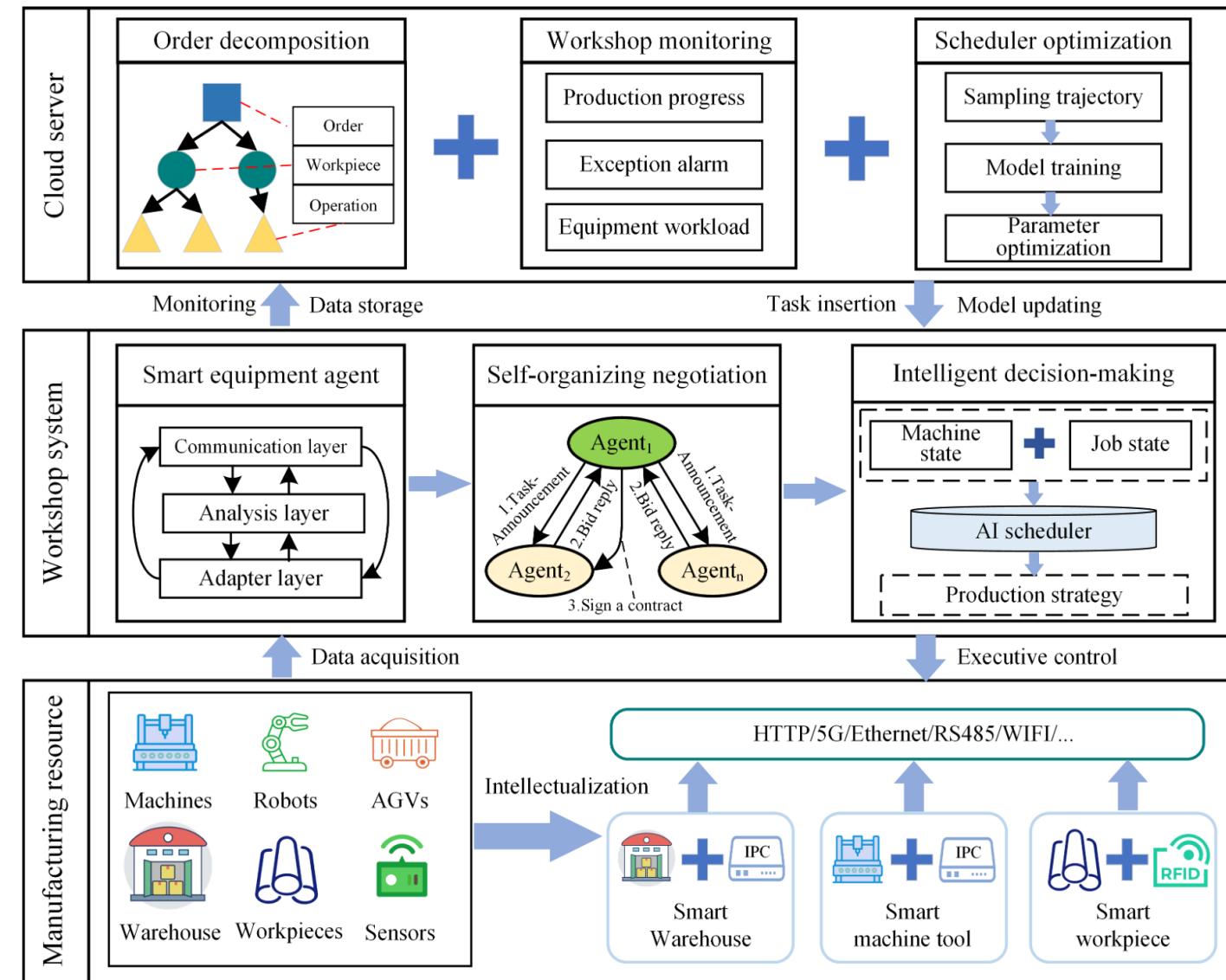


Fig. 2. The architecture of multi-agent manufacturing system.

# Method Design

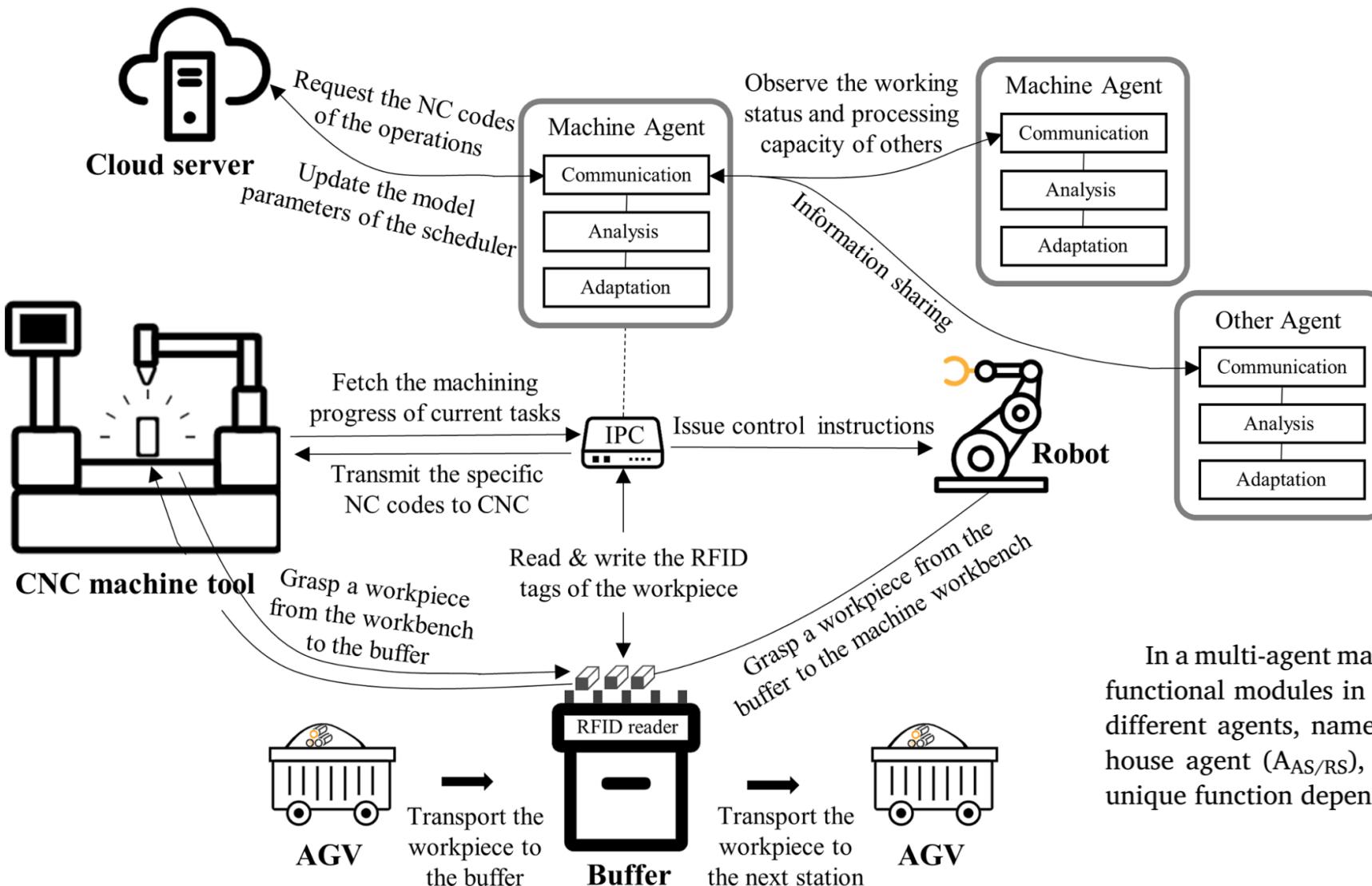


Fig. 3. The operation logic of a machine agent.

In a multi-agent manufacturing system, the machine, workpiece, and functional modules in the workshop are abstracted and constructed as different agents, namely, machine agent ( $A_M$ ), part agent ( $A_p$ ), warehouse agent ( $A_{AS/RS}$ ), logistics agent ( $A_{AGV}$ ). Each agent has its own unique function depending on its characteristics.

# Method Design

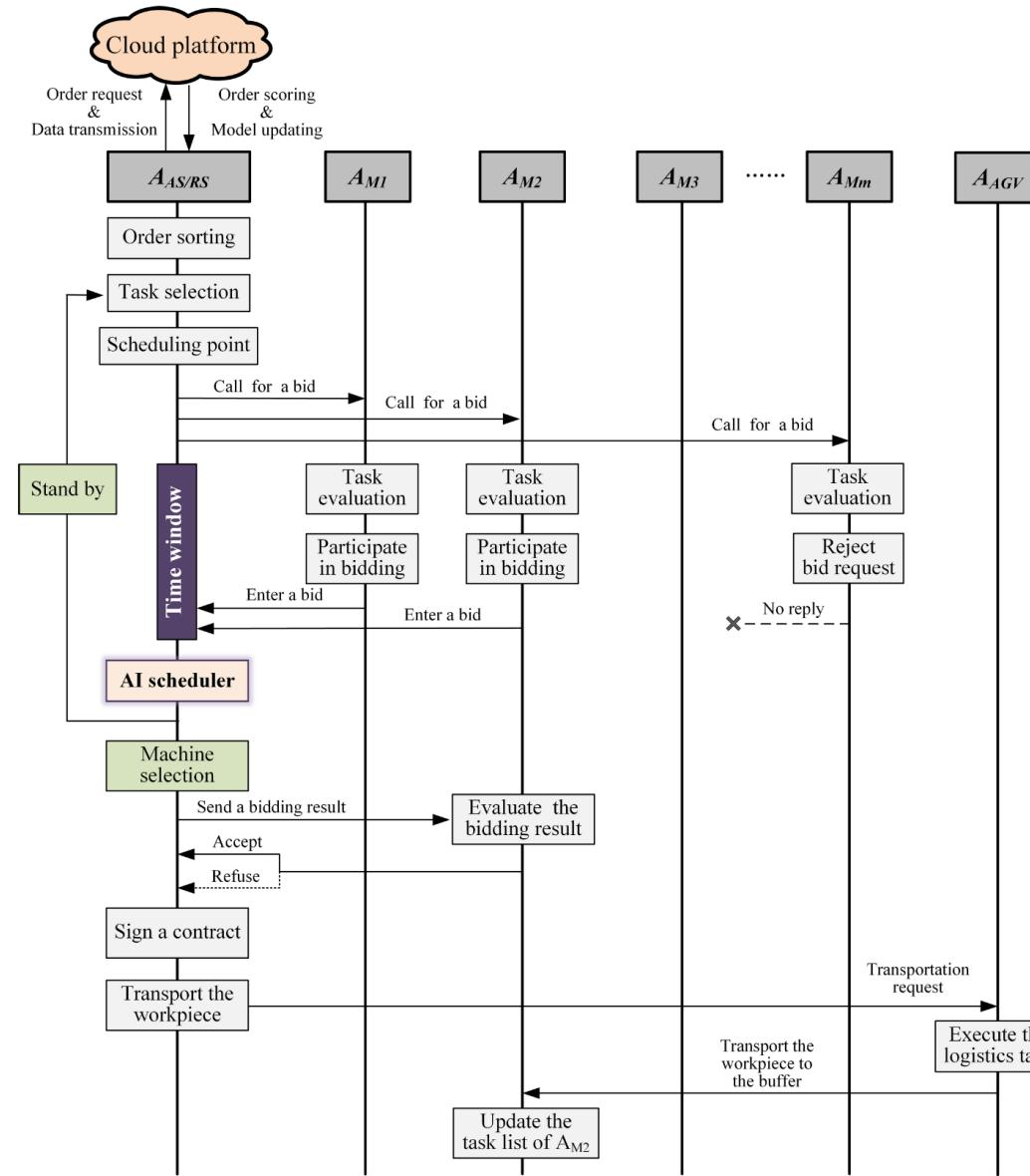


Fig. 4. Self-organizing negotiation process among multiple agents.

# Method Design

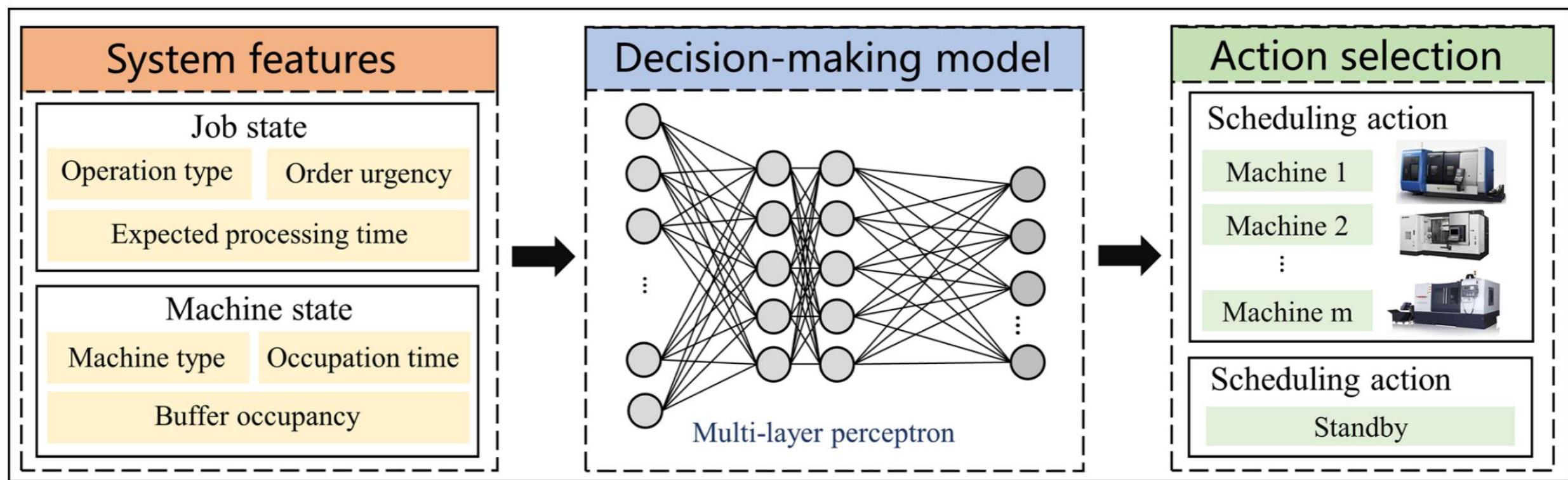


Fig. 5. Adaptive decision-making process of AI scheduler.

# Method Design

## State Space

State space is an accurate representation of workshop environment characteristics and the key basis for AI scheduler to make decisions. At the scheduling moment  $t$ , the state space  $S_t$  of workshop environment consists of operation attributes  $O_t$  of the workpiece  $J_i$  to be scheduled and working status  $M_t$  of all available machining tools, and can be described as  $S_t = \{O_t, M_t\}$ . The operation attributes of the workpiece to be scheduled  $O_t = \{O_{1t}, O_{2t}, O_{3t}\}$  are composed of processing technology type  $O_{1t}$ , ideal operating time  $O_{2t}$  and order urgency  $O_{3t}$ , and the dimension of job state is 3.  $O_{1t}$  denotes the numerical expression of operation types (i.e., 1: turning, 2: milling).  $O_{2t}$  indicates the workload time required to complete the operation. Order urgency  $O_{3t}$  indicates the difference between due time and current timepoint  $t$ , as shown in Eq. (6).

$$O_{3t} = \frac{\hat{T}_{ij}^{(c)} - t}{\sum_{j < N_i} T_{ij}} \quad (6)$$

## Action Space

$A_t$ . The action space is a collection of all optional production actions  $A_t = \{a_0, a_1, \dots, a_m\}$  for AI scheduler in a workshop environment, as shown in Fig. 5. In this paper, there are two kinds of actions within  $A_t$ : standby “ $a_0$ ”, and machine allocation “ $a_1, \dots, a_m$ ”. The standby action indicates

## Reward function design

$$r_t = \begin{cases} -50 & \text{invalid action} \\ w_1 \times r_{tard} - w_2 \times r_{load} + w_3 \times r_{profit} & \text{order to machine or standby} \end{cases} \quad (7)$$

$$r_{tard} = \frac{\hat{T}_{ij}^{(c)} - T_{ij}^{(c)}}{\tilde{T}_{ij}} \quad (8)$$

Where  $\hat{T}_{ij}^{(c)}$  and  $T_{ij}^{(c)}$  are the expected completion time and actual completion time of operation  $O_{ij}$ , and  $\tilde{T}_{ij}$  is the nominal operating time of operation  $O_{ij}$ . If the actual completion time of operation is earlier than expected,  $r_{tard}$  will be positive. The goal of AI scheduler is to maximize the value of  $r_{tard}$ .

$$LD_m = \int_{t_0}^t \sum_{ij} K_m^{(S)} \times T_{ij} \quad (9)$$

Where  $LD_m$  represents the cumulative workload time of machine  $M_m$  from timestep  $t_0$  to  $t$ .

$$r_{load} = \sqrt{\frac{\sum_{m \in c} (LD_m - \bar{LD})^2}{M_c}} \quad (10)$$

Where  $r_{load}$  indicates the workload deviation between isomorphic machine tools,  $\bar{LD}$  represents the average workload time of isomorphic machine tool., and  $M_c$  denotes the number of types of machine tools.

$$r_{profit} = e - K_m^{(E)} \quad (11)$$

Where  $r_{profit}$  indicates the production profit of the work order.

# Method Design

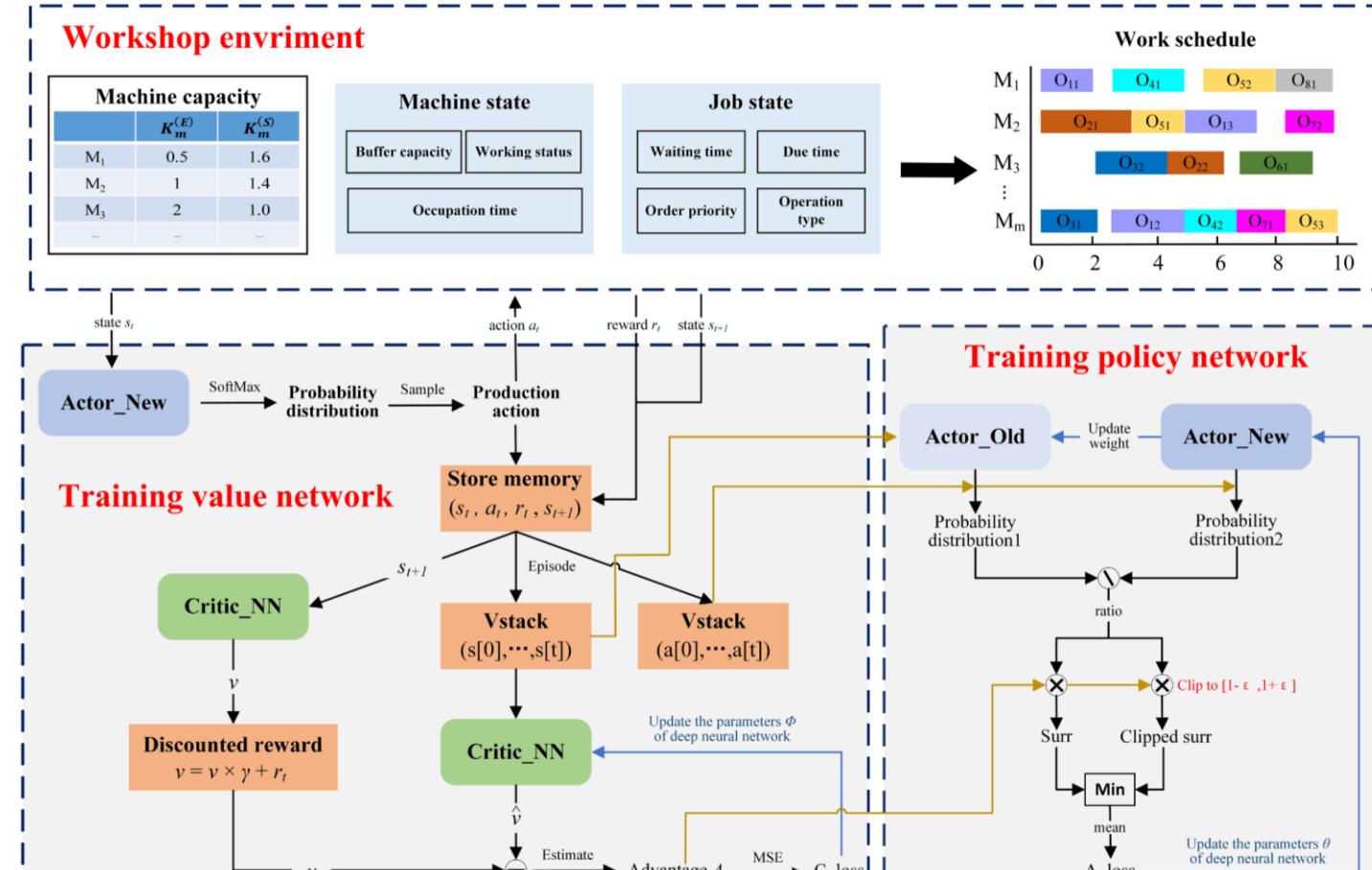


Fig. 6. Self-learning mechanism of decision-making model.

Table. 2

Adaptive optimization method of AI scheduler based on DRL.

- 
- Algorithm 1: the pseudo-code of adaptive optimization method of AI scheduler based on DRL
- 1: Initialization: hyper parameter  $\alpha_p$ ,  $\alpha_v$ ,  $\gamma$ ,  $\epsilon$  and the network parameter of PPO algorithm
  - 2: Generate a simulated job shop environment for training according to predefined criteria
  - 3: For episode number  $e = 1, 2, \dots, N$  do
  - 4: Reset the simulated workshop environment
  - 5: For epoch number  $t = 1, 2, \dots, T$  do
  - 6: AI scheduler observes the state  $S_t$  of workshop environment: operation characteristics of the workpiece to be scheduled, running status of machine sets
  - 7: AI scheduler run policy  $\pi(\theta)$  to choose a production action  $a_t$  based on the workshop state  $S_t$
  - 8: The workshop environment feeds back a composite reward value  $r_t$  to AI scheduler after the action  $a_t$  executed, and makes a translation to a new state  $S_{t+1}$
  - 9: Store the  $(s_t, a_t, r_t)$  in the trajectory buffer  $(s_1, a_1, r_1, \dots, s_\tau, a_\tau, r_\tau)$
  - 10: End for
  - 11: AI scheduler runs critic network and estimates the value of advantage function  $\hat{A}_t$
  - 12: For epoch number  $i = 1, 2, \dots, M$  do
  - 13: Sample mini-batches from the trajectory buffer
  - 14: Update  $\theta$  by a gradient method to optimize the loss function  $J_{ppo}(\theta)$
  - 15: End for
  - 16: For epoch number  $j = 1, 2, \dots, B$  do
  - 17: Sample mini-batches from the trajectory buffer
  - 18: Update  $\emptyset$  by a gradient method to optimize the loss function  $L_{BL}(\emptyset)$
  - 19: End for
  - 20: Replace the parameters of actor network  $\pi(\text{old}) \leftarrow \pi(\theta)$
  - 21: End for
-



## Contents

- 01**  Problem Description
- 02**  Method Design
- 03**  Experiment Results
- 04**  Conclusion

# Experiment Results

Physical workshop



Digital workshop

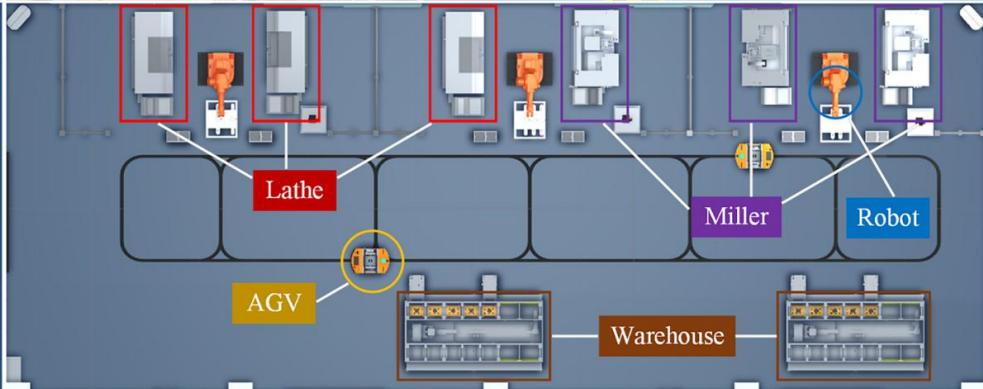


Fig. 7. The overall layout of a smart workshop.

## LAN Port:

- data transmission
- info communication

## PLC:

- execution control
- logical operation

## IPC:

- agent programs
- protocol analysis
- negotiation interaction
- decision-making

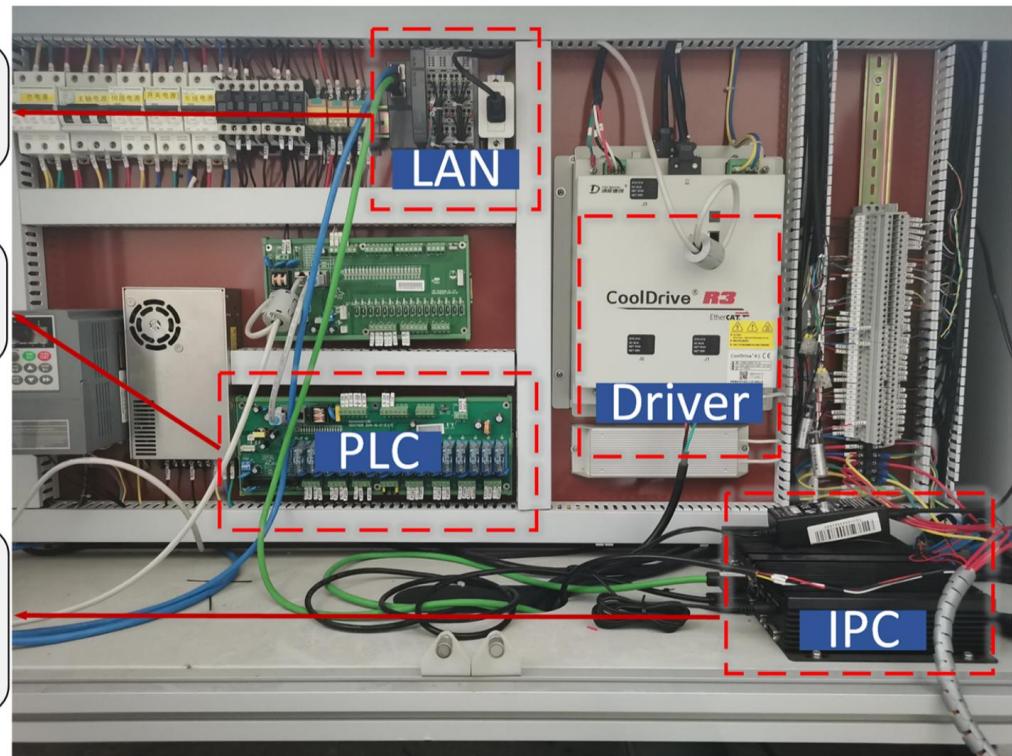


Fig. 8. The internal view of a smart machine.



Job type	Process route	Available machines
Shaft	Turning → Milling	(M <sub>1</sub> , M <sub>2</sub> , M <sub>3</sub> ), (M <sub>4</sub> , M <sub>5</sub> , M <sub>6</sub> )
Flange	Turning → Milling	(M <sub>1</sub> , M <sub>2</sub> , M <sub>3</sub> ), (M <sub>4</sub> , M <sub>5</sub> , M <sub>6</sub> )
Plate	Milling	(M <sub>4</sub> , M <sub>5</sub> , M <sub>6</sub> )

Fig. 9. The process route of jobs and available machines of operations.

# Experiment Results

**Table. 4**

Test case.

Index	Job type	Arrival time	Due time	Ideal workload time
1	Shaft	10	151	35,12
2	Flange	45	171	32,10
3	Plate	70	199	43
4	Plate	95	209	38
5	Plate	125	266	47
6	Flange	150	282	30,14
7	Shaft	175	307	33,11
8	Plate	190	340	50
9	Flange	200	368	39,17
10	Plate	210	297	29
11	Flange	235	373	33,13
12	Plate	260	392	44
13	Plate	270	384	38
14	Shaft	295	412	29,10
15	Flange	320	425	26,9
16	Shaft	350	464	28,10
17	Plate	370	475	35
18	Plate	385	496	37
19	Shaft	430	571	31,16
20	Flange	495	651	34,18

# Experiment Results

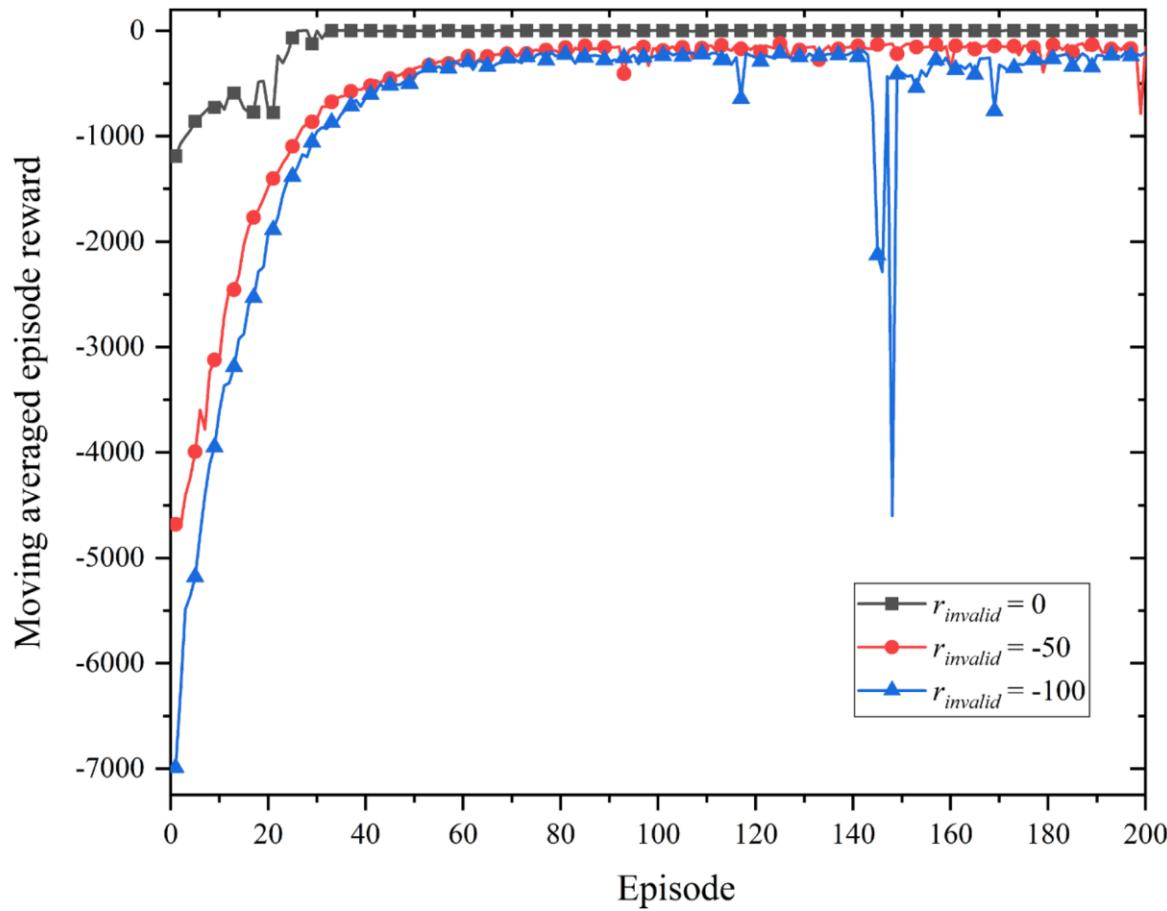


Fig. 10. The learning curves of accumulated reward under three kinds of reward settings.

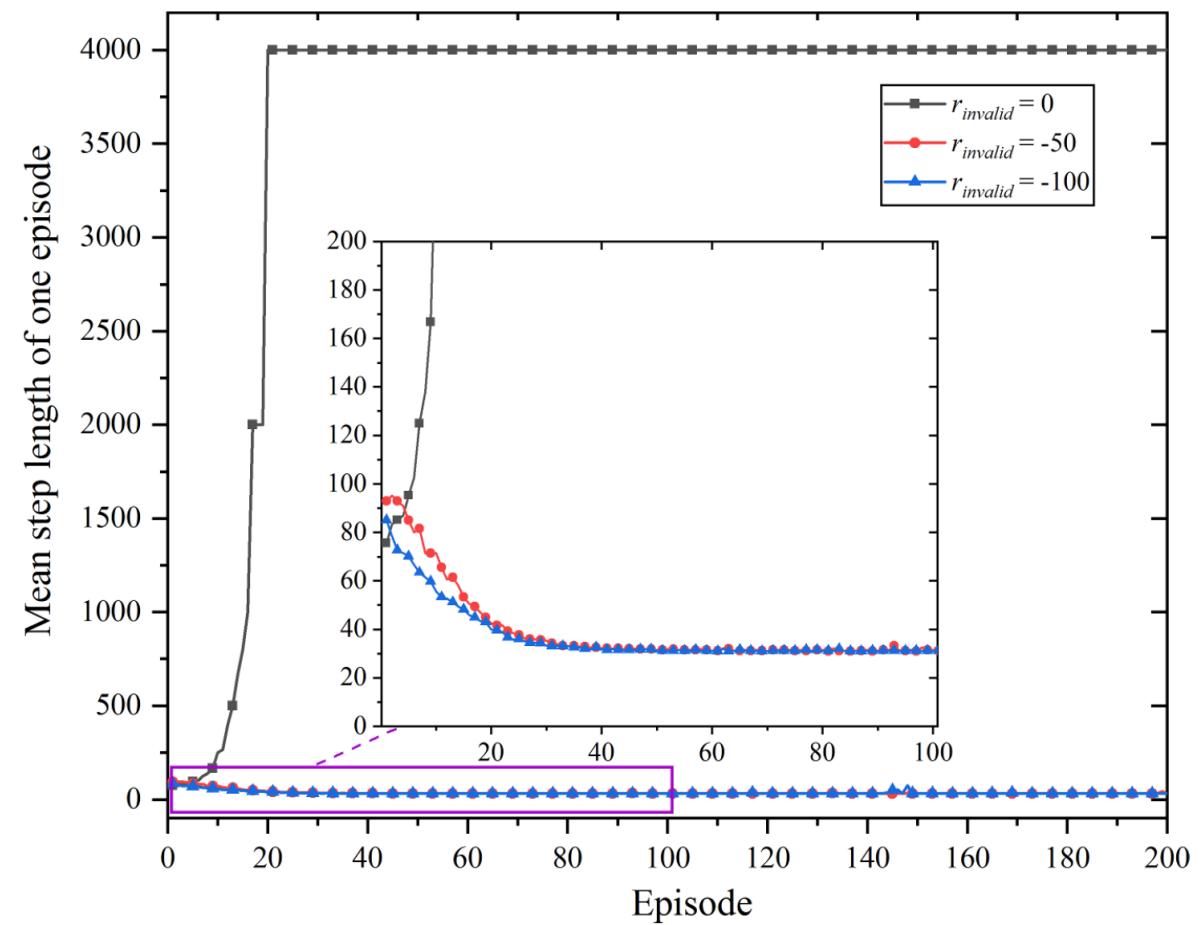


Fig. 11. The learning curves of step length under three kinds of reward settings.

# Experiment Results

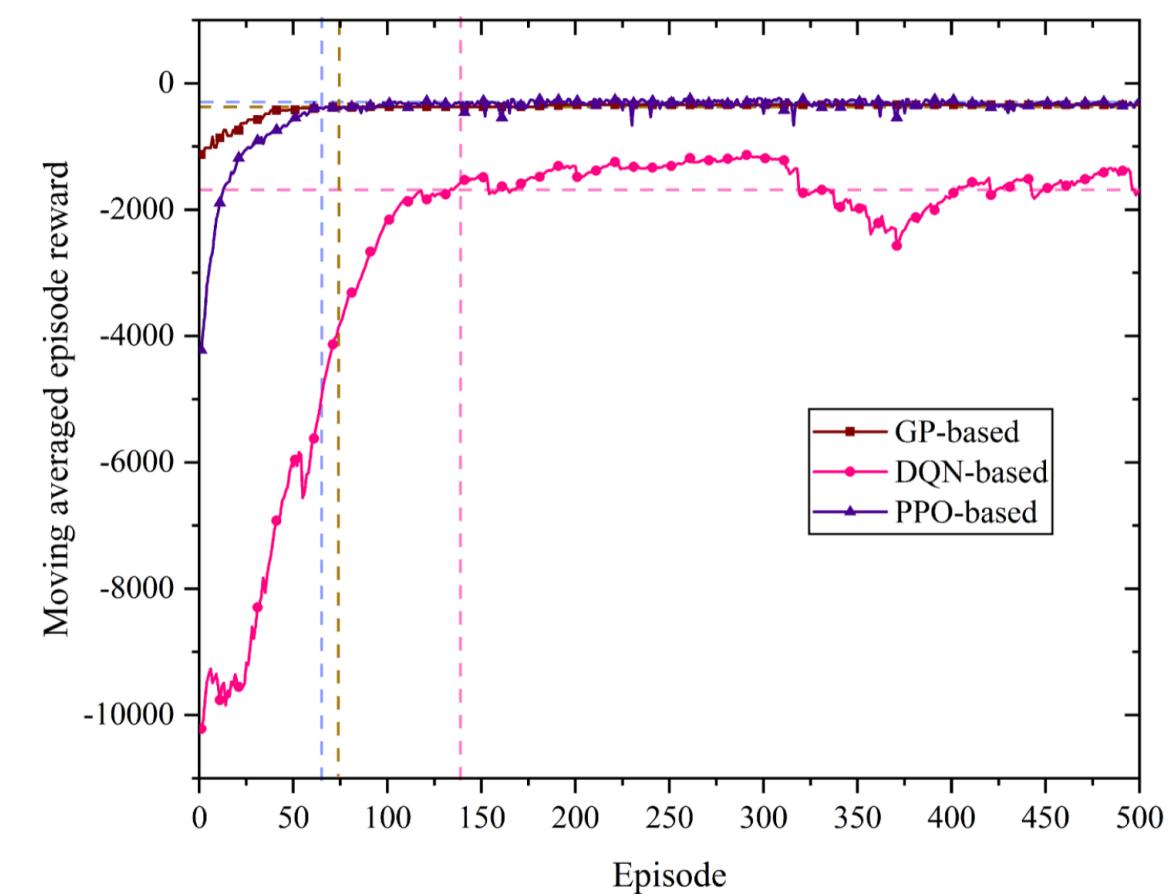


Fig. 12. The learning curve of three different methods in training process.

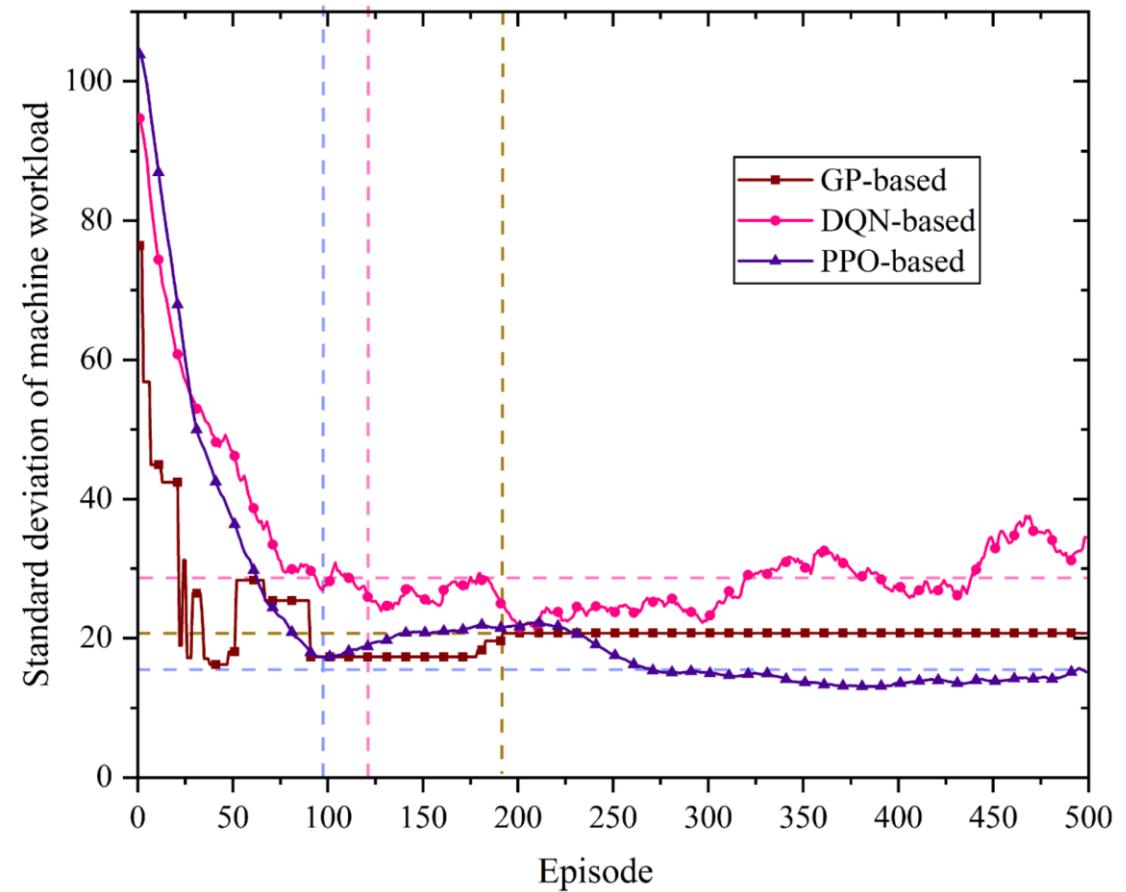


Fig. 13. The learning curve of workload balance between isomorphic machine tools.

# Experiment Results

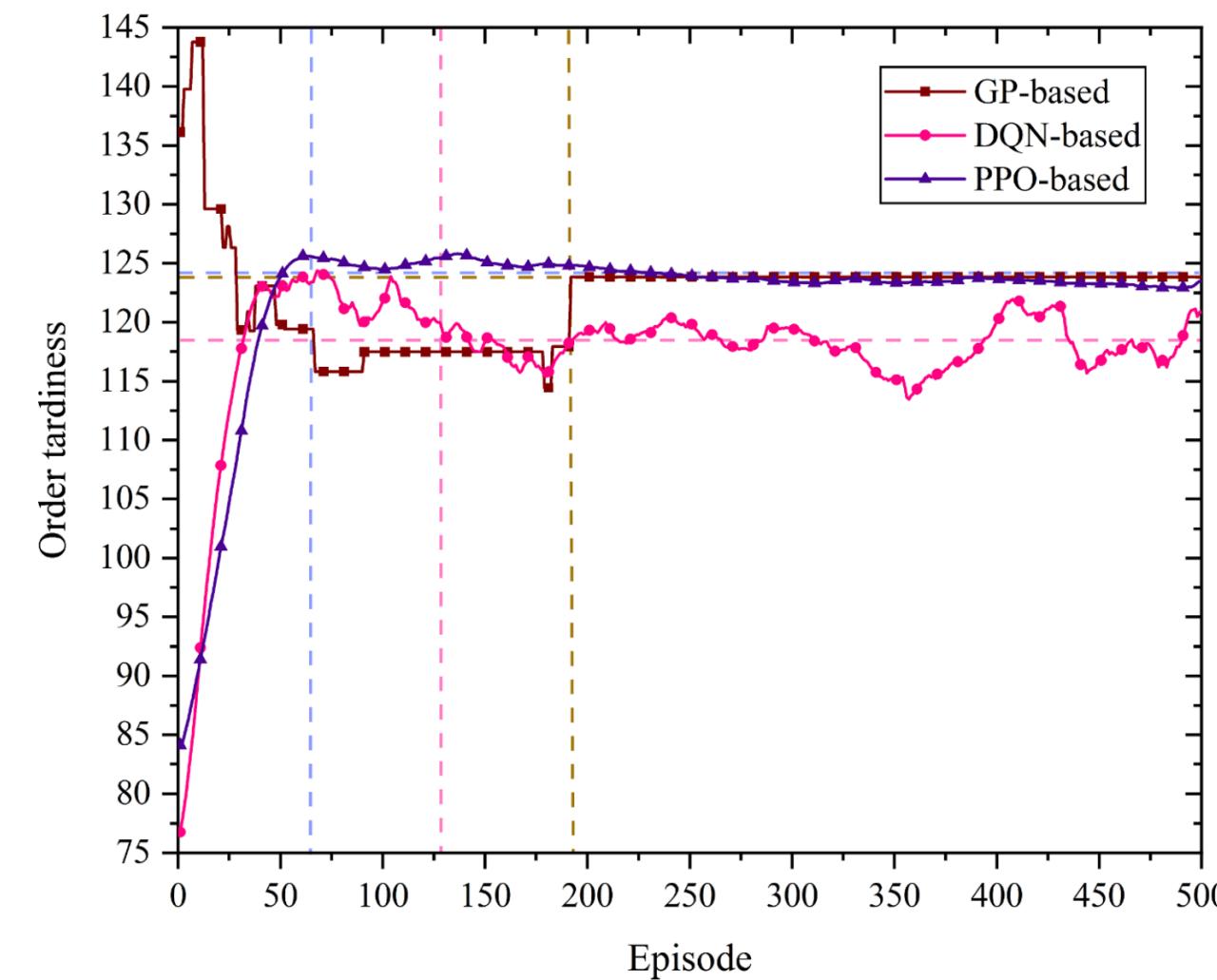


Fig. 14. The learning curve of order tardiness by three different methods.

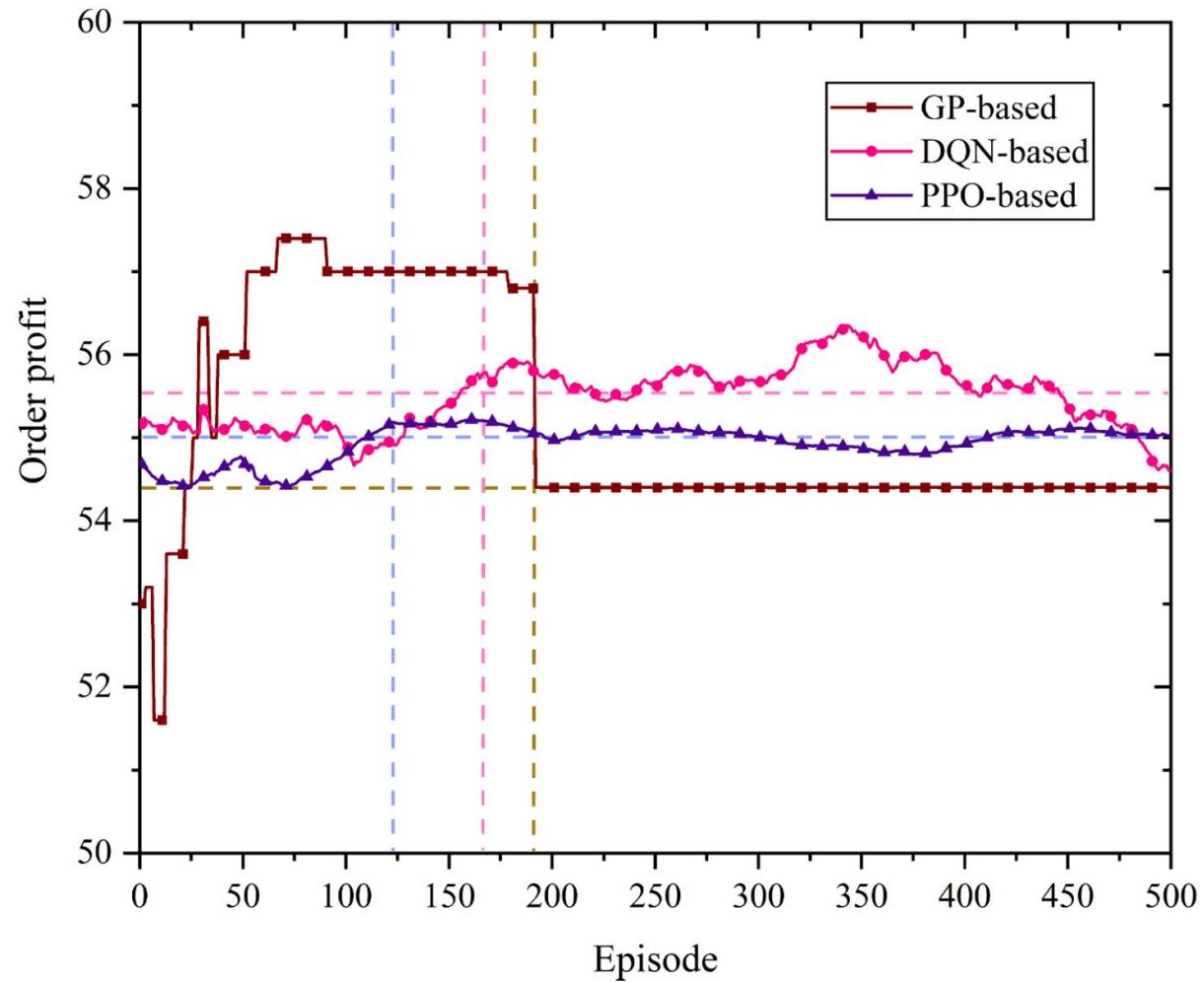


Fig. 15. The learning curve of order profit by three different methods.

# Experiment Results

**Table. 6**

Comparison of results on test case by five scheduling methods.

	GA-based	SPT+FIFO	GP-based	DQN-based	PPO-based
Tardiness	125.3	146.8	124.9	121.9	123.0
Workload balance	15.6	79.6	20.8	43.9	15.3
Profit	55.6	50.4	54.4	54.4	55.8
Evaluation value	165.3	117.6	158.5	132.4	163.5
Architecture	☆	☆	★	☆	☆
Execution time (s)	28.23	6.33	1.32	6.52	1.06

**Table. 7**

Comparison of results on test case under new order insertion.

	GA-based	SPT+FIFO	GP-based	DQN-based	PPO-based
Tardiness	123.0	145.8	122.9	124.2	117.6
Workload balance	20.1	71.8	23.1	29.2	17.7
Profit	54.9	50.8	54.8	55.2	56.6
Evaluation value	157.8	124.8	154.6	150.2	156.5
Architecture	☆	☆	★	☆	☆
Execution time (s)	54.22	6.35	1.34	6.56	1.07

**Table. 8**

Comparison of results on test case under machine failure.

	GA-based	SPT+FIFO	GP-based	DQN-based	PPO-based
Tardiness	122.5	143.8	122.4	122.2	121.6
Workload balance	28.6	60.0	31.7	35.6	31.1
Profit	55.1	51.6	55.0	55.4	56.0
Evaluation value	149.0	135.4	145.7	142.0	146.5
Architecture	☆	☆	★	☆	☆
Execution time (s)	42.53	6.52	1.42	6.67	1.08



## Contents

- 01**  Problem Description
- 02**  Method Design
- 03**  Experiment Results
- 04**  Conclusion

# Future Work

## Contributions

This paper constructs a multi-agent manufacturing system with an AI scheduler based on deep reinforcement learning to improve dynamic responsiveness and production efficiency for personalized orders.

## Future Work

Future research will focus on expanding the system's scale, enhancing the state space design, and incorporating various dispatching rules.

## A Hierarchical Multi-Action Deep Reinforcement Learning Method for Dynamic Distributed Job-Shop Scheduling Problem With Job Arrivals

Jiang-Ping Huang , Liang Gao , Senior Member, IEEE, and Xin-Yu Li , Member, IEEE

**Publish Journal:** IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING (2025)



## Contents

- 01**  **Problem Description**
- 02**  **Method Design**
- 03**  **Experiment Results**
- 04**  **Conclusion**



## Contents

- 01**  Problem Description
- 02**  Method Design
- 03**  Experiment Results
- 04**  Conclusion

# Problem Description

- This paper addresses the Distributed Job-shop Scheduling Problem (DJSP) with job arrivals using a Multi-action Deep Reinforcement Learning (MDRL) method, incorporating a hierarchical action space and Graph Neural Network (GNN) for scheduling optimization.
- **Challenges:** The main challenge is managing job arrival uncertainties and designing a reward function and state transition that can effectively handle dynamic disturbances in the scheduling process.
- **Method:** The MDRL approach utilizes Proximal Policy Optimization (PPO) with two actor-critic frameworks to enable intelligent decision-making, outperforming other methods in both experimental and industrial settings.



## Contents

- 01**  Problem Description
- 02**  Method Design
- 03**  Experiment Results
- 04**  Conclusion

# Method Design

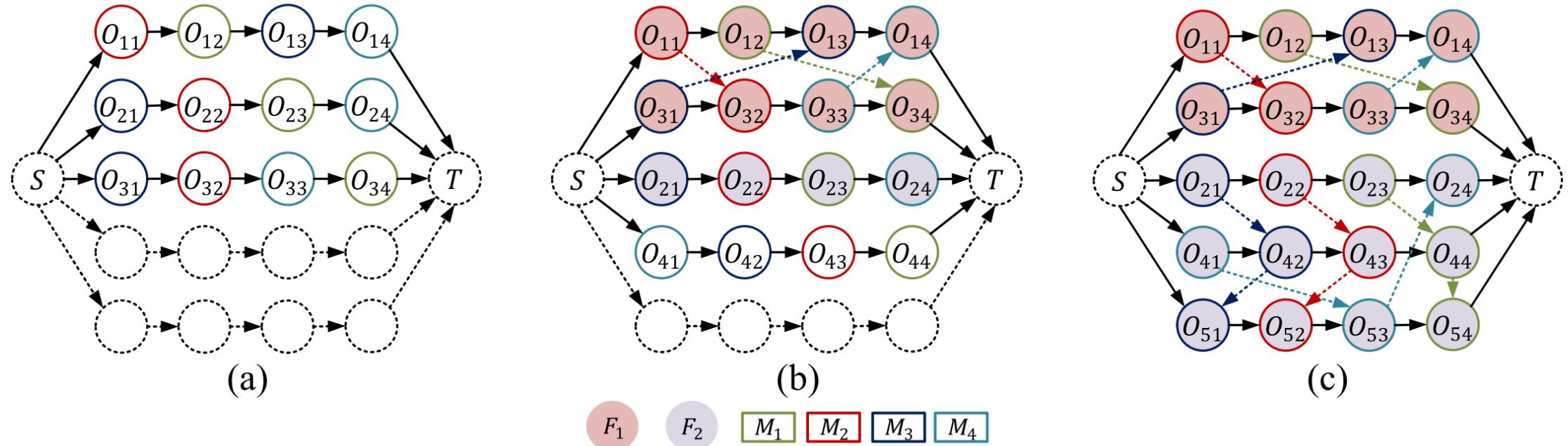
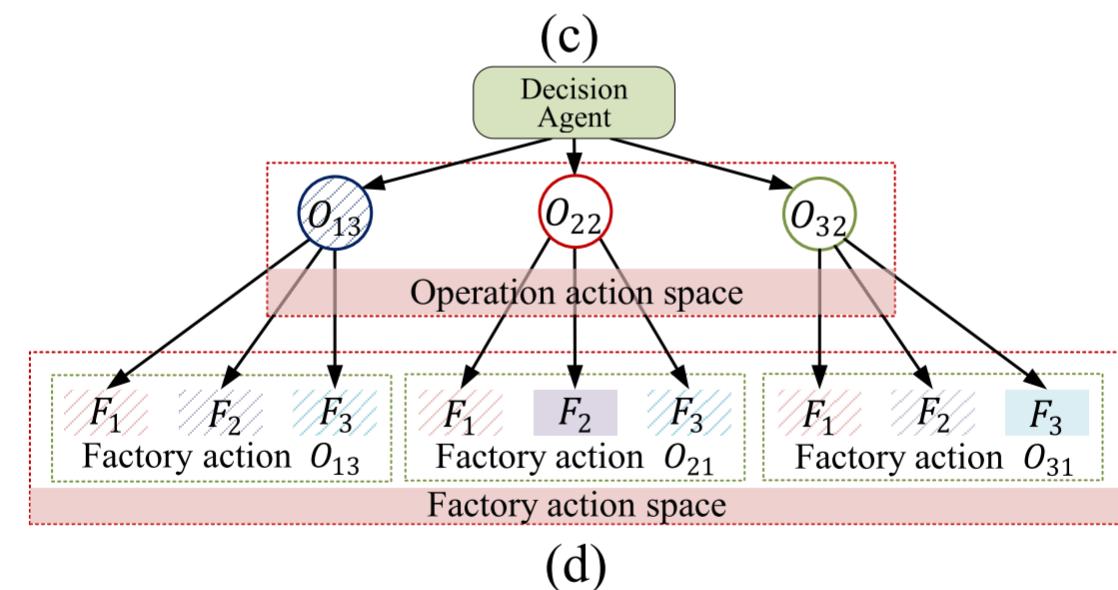
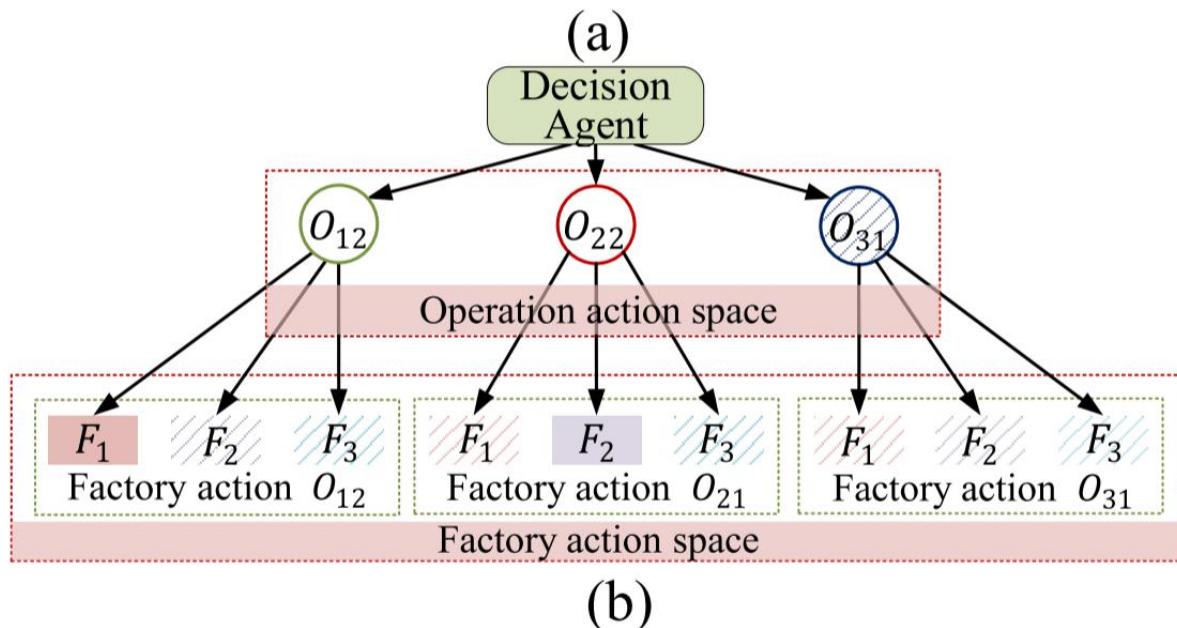
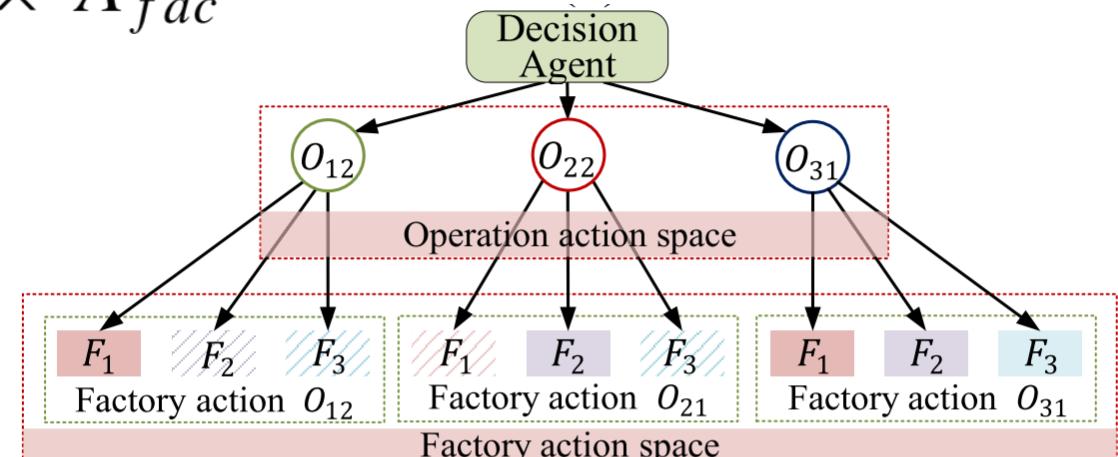
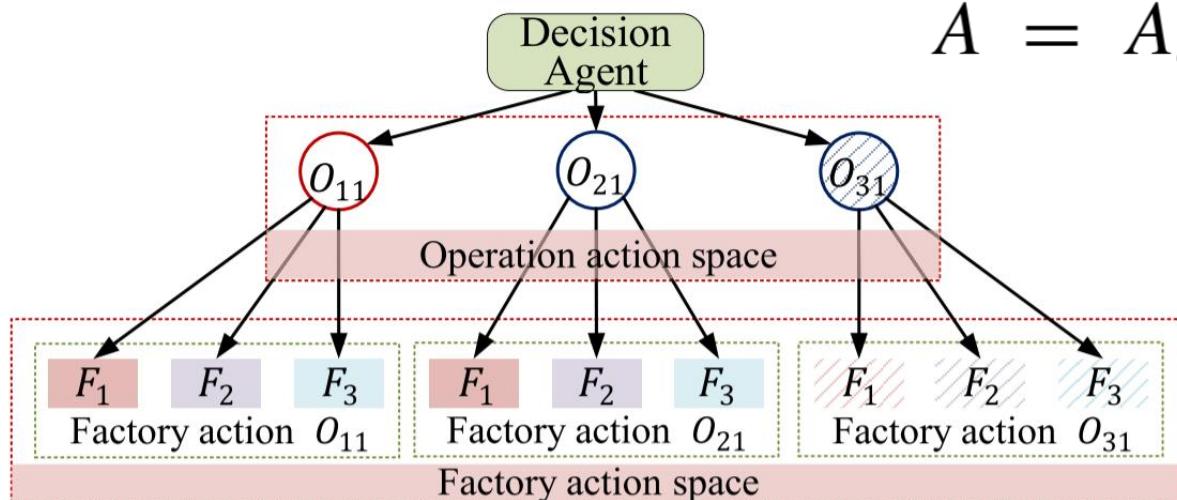


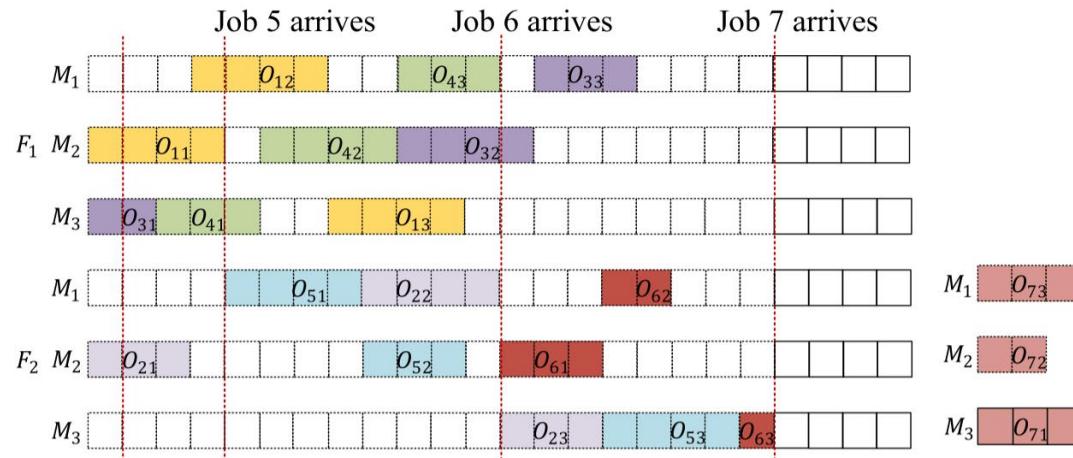
Fig. 1. Stitched disjunctive graph model.

# Method Design

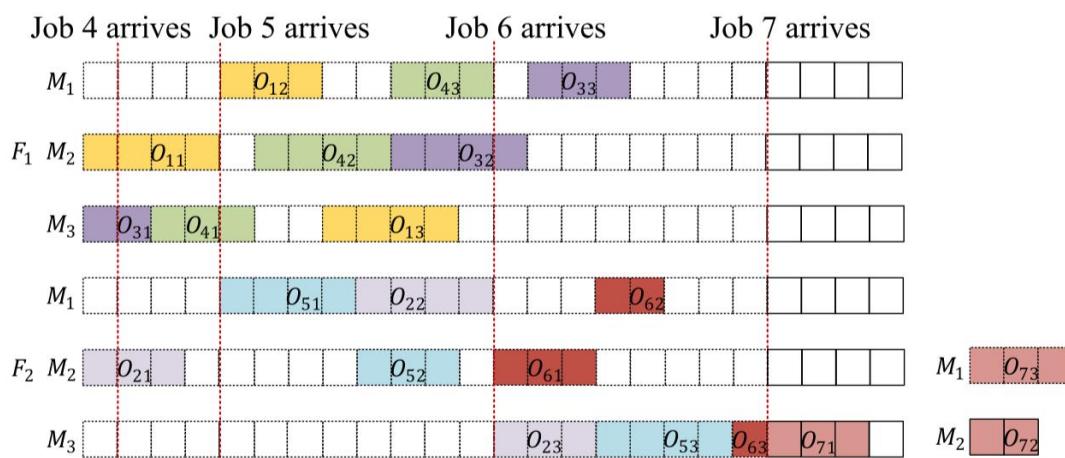
$$A = A_{op} \times A_{fac}$$



# Method Design



(a) Current state  $s_t$  and the information of the arrived job



(b) Next state  $s_{t+1}$  based on the selected action

Fig. 3. Illustration of the state transition in Case 2.

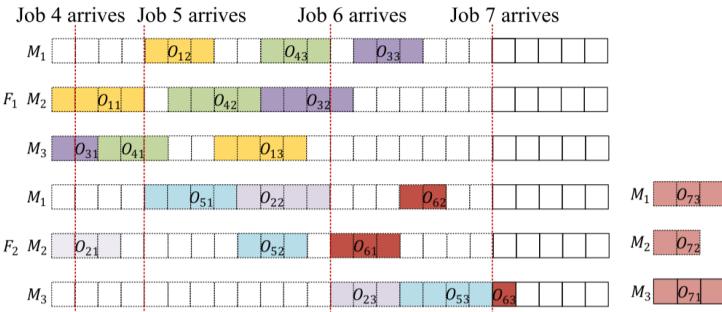
TABLE I  
SYMBOLS FOR CLASSIFYING THE STATE TRANSACTION

Symbol	Definition
$\mathcal{J}_{a\&u}$	Set of jobs that have arrived, but have not yet been scheduled
$\mathcal{A}_{a\&u}$	Set of the arrival times of the jobs in set $\mathcal{J}_{a\&u}$
$\mathcal{C}_s$	Set of the completion times of the operations that have been scheduled
$\mathcal{CF}_s$	Set of the completion times refers to the completion times of the first operations of each job in the scheduling scheme

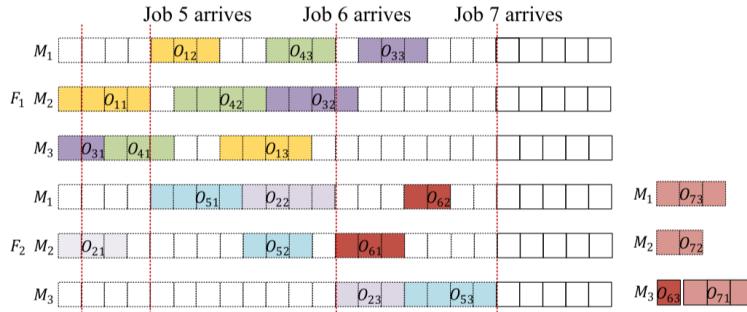
**Case 1:**  $\mathcal{J}_{a\&u} = \emptyset$ , i.e., no job that has arrived has not been scheduled. In this case, if there is at least an eligible operation available in the operation action space, the agent will select a suitable operation and determine the earliest start processing time. However, if there are no eligible operations available, the agent will wait until a new job arrives.

**Case 2:**  $\mathcal{J}_{a\&u} \neq \emptyset$  and  $\min(\mathcal{A}_{a\&u}) \geq \max(\mathcal{C}_s)$ , i.e., at least one job has arrived but not been scheduled. Additionally, the earliest arrival time in set  $\mathcal{A}_{a\&u}$  is not less than the maximum completion time of all the operations that have already been scheduled. An example is shown in Fig. 3(a), where  $\mathcal{J}_{a\&u} = \{7\}$ . The arrival time of job 7 equals the completion time of operation  $o_{63}$ .

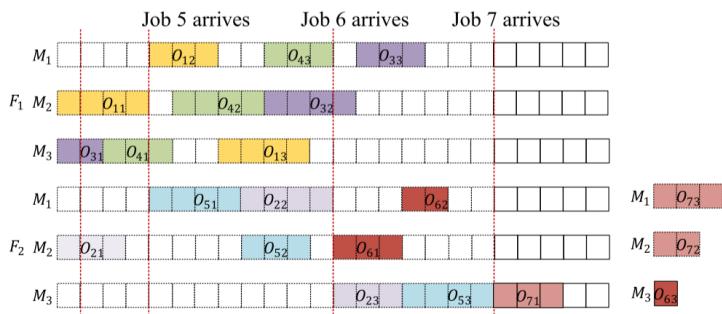
# Method Design



(a) Current state  $s_t$  and the information of the arrived job.



(b) Current state  $s_t$  updated by the new job arrival.

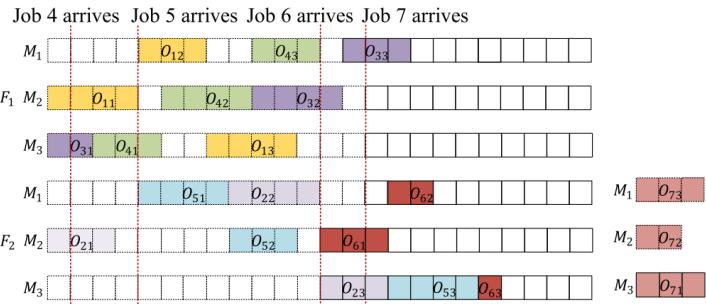


(c) Next state  $s_{t+1}$  based on the selected action.

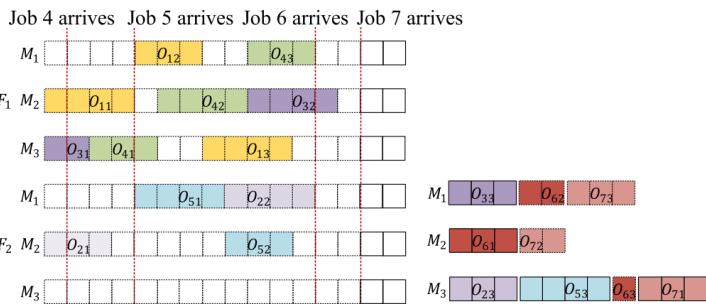
**Case 3:  $\mathcal{J}_{a\&u} \neq \emptyset$  and  $\min(\mathcal{A}_{a\&u}) \geq \max(\mathcal{CF}_s)$  and  $\min(\mathcal{A}_{a\&u}) \leq \max(\mathcal{C}_s)$ ,** i.e., at least one job has arrived but not been scheduled, the earliest arrival time in set  $\mathcal{A}_{a\&u}$  is not less than the maximum completion times in set  $\mathcal{CF}_s$ , and it is also not greater than the maximum completion time among all operations that have already been scheduled. For instance, in Fig. 4(a),  $\mathcal{J}_{a\&u} = \{7\}$ , and the arrival time of job 7 is between the completion times of operation  $o_{53}$  and operation  $o_{63}$ .

Fig. 4. Illustration of the state transition in Case 3.

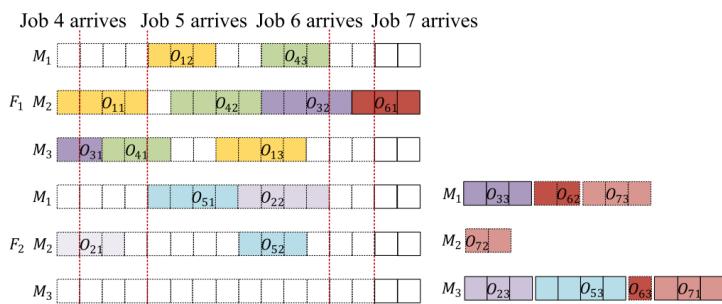
# Method Design



(a) Current state  $s_t$  and the information of the arrived job.



(b) Current state  $s_t$  updated by the new job arrival.



(c) Next state  $s_{t+1}$  based on the selected action

**Case 4:  $\mathcal{J}_{a\&u} \neq \emptyset$  and  $\min(\mathcal{A}_{a\&u}) < \max(\mathcal{C}\mathcal{F}_s)$ , i.e.,** at least one job has arrived but not been scheduled. Additionally, the smallest arrival time in set  $\mathcal{A}_{a\&u}$  is less than the maximum completion time in set  $\mathcal{C}\mathcal{F}_s$ . As in Fig. 5(a), the arrival time of job 7 is less than the completion time of operation  $o_{61}$ .

Fig. 5. Illustration of the state transition in Case 4.

# Method Design

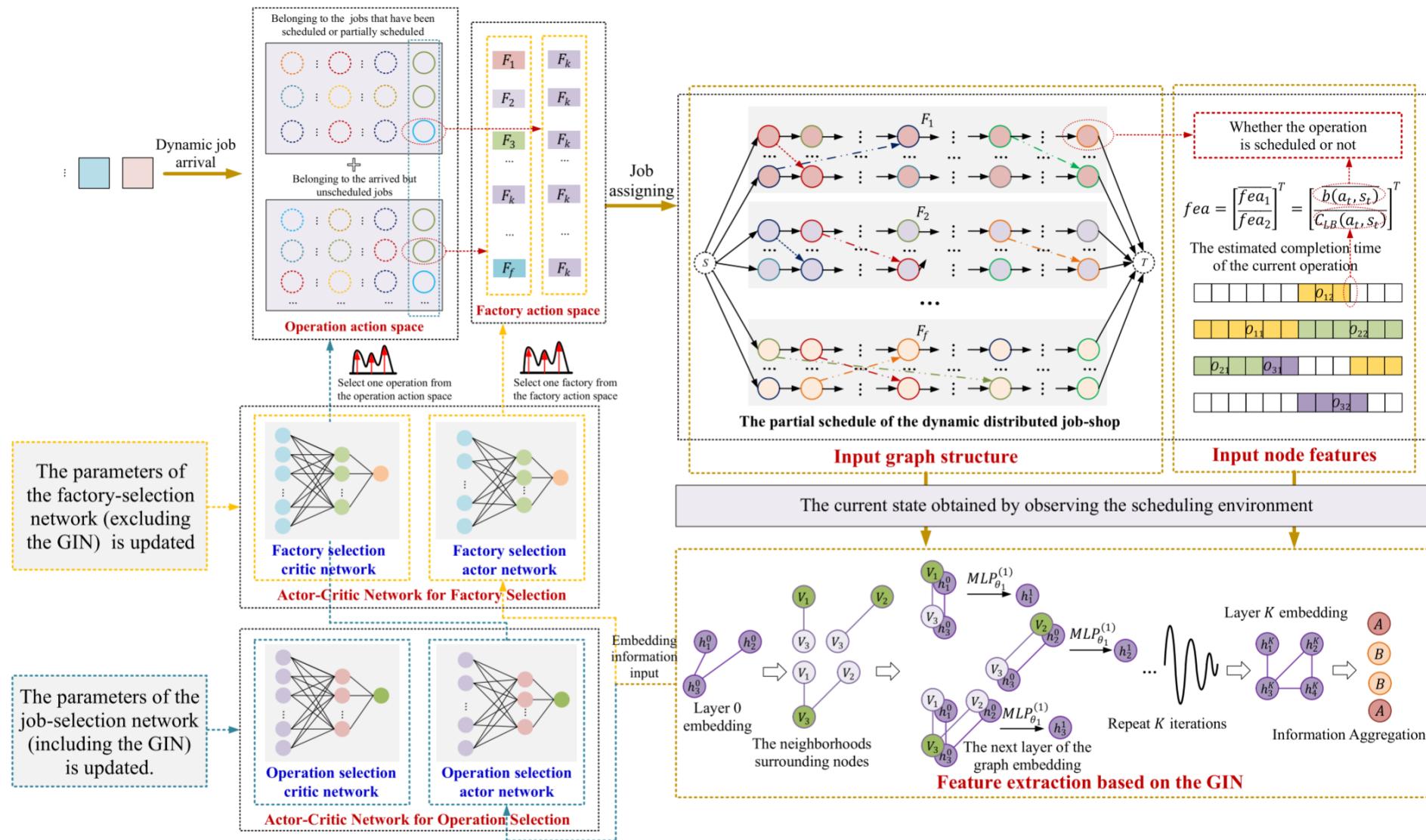


Fig. 6. The framework of GIN-based multi-action scheduling policy.



## Contents

- 01**  Problem Description
- 02**  Method Design
- 03**  Experiment Results
- 04**  Conclusion

# Method Design

$$RPI = \frac{Method_{sol} - Best_{sol}}{Best_{sol}} \times 100\% \quad (10)$$

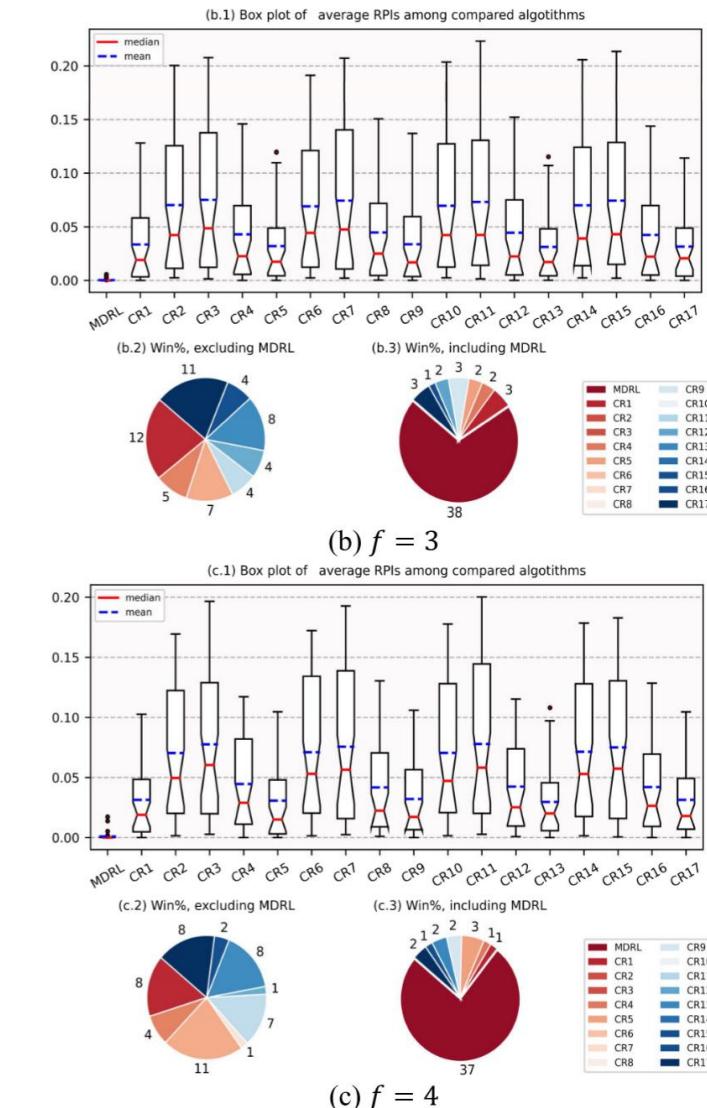
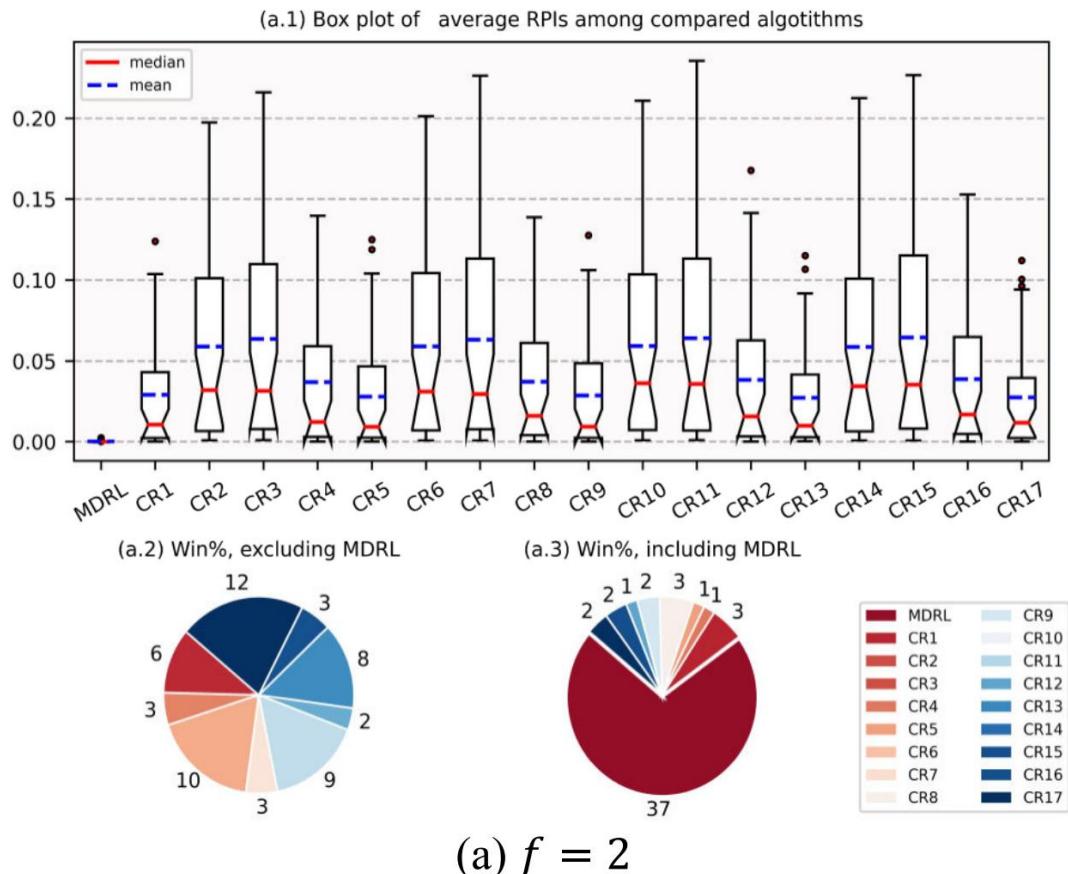


Fig. 7. Comparison among MDRL and PDRs.

# Method Design

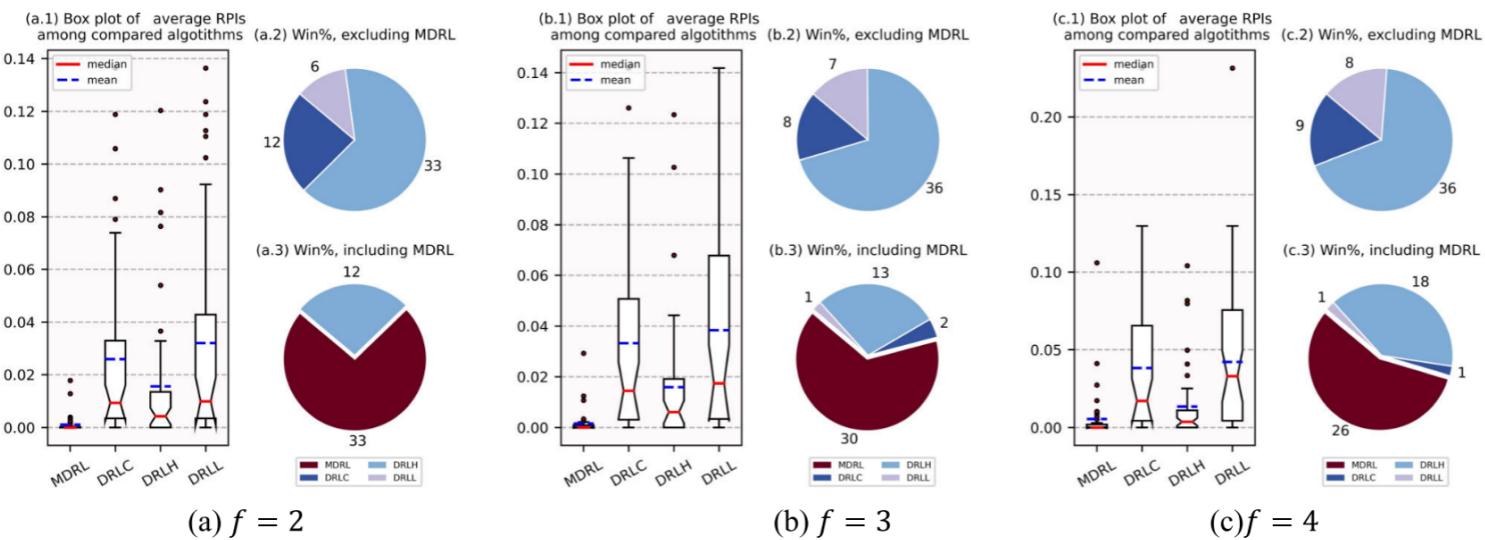


Fig. 8. Comparison among MDRL and DRLs.

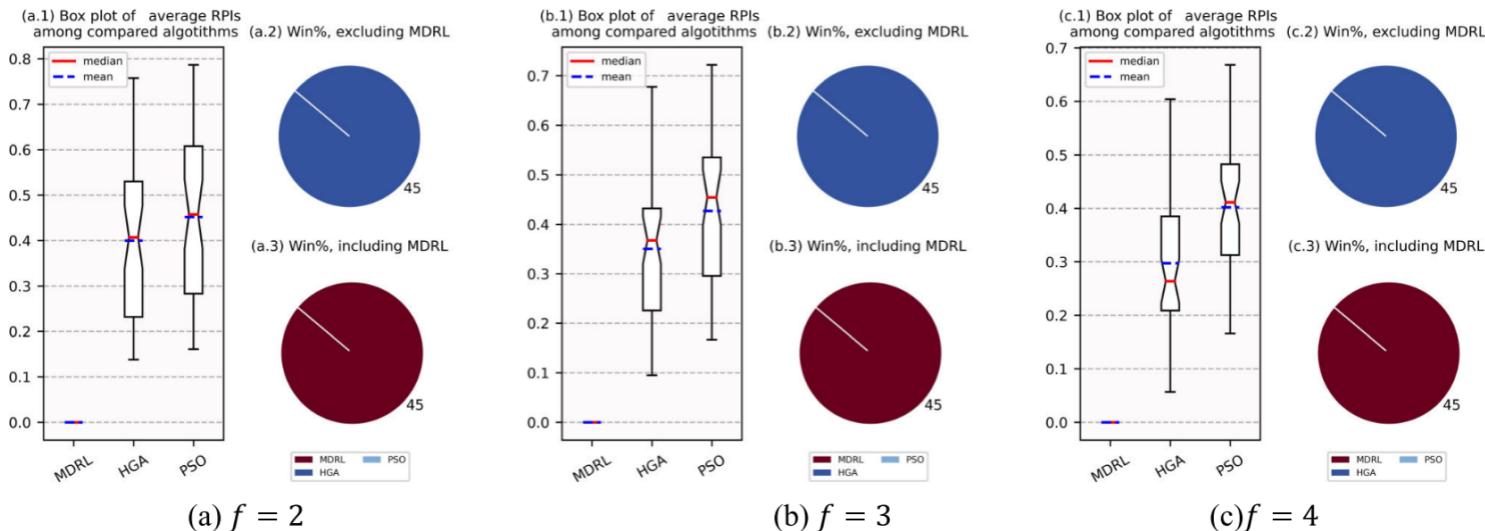


Fig. 9. Comparison among MDRL and metaheuristics.



## Contents

- 01**  Problem Description
- 02**  Method Design
- 03**  Experiment Results
- 04**  Conclusion

# Future Work

## Contributions

1. A hierarchical multi-action DRL method has been proposed for the DJSP with job arrivals, utilizing a multi-action MDP framework, GIN-based feature extraction, and dual decision policies to effectively handle dynamic job arrivals, demonstrating superior performance over PDRs, DRL methods, and metaheuristics in extensive experiments and a real-world case study.

## Future Work

Future research should incorporate transportation costs and machine breakdowns to further optimize scheduling in distributed manufacturing environments.

## Real-Time Scheduling for Flexible Job Shop With AGVs Using Multiagent Reinforcement Learning and Efficient Action Decoding

Yuxin Li , Qingzheng Wang, Xinyu Li , Member, IEEE,Liang Gao , Senior Member, IEEE, Ling Fu, Yanbin Yu, and Wei Zhou

**Publish Journal:** IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS (2025)



## Contents

- 01**  **Problem Description**
- 02**  **Method Design**
- 03**  **Experiment Results**
- 04**  **Conclusion**



## Contents

- 01**  Problem Description
- 02**  Method Design
- 03**  Experiment Results
- 04**  Conclusion

# Problem Description

- **Problem:** The dynamic flexible job shop scheduling problem (DFJSP) involves machine flexibility and limited logistics equipment, increasing the complexity of collaborative scheduling.
- **Challenge:** Frequent dynamic events introduce uncertainty, making it difficult to achieve efficient and reliable real-time scheduling.
- **Method:** A real-time scheduling approach based on multiagent reinforcement learning (MARL) is proposed to optimize task selection, machine allocation, and AGV assignment while enhancing robustness against disturbances.



## Contents

- 01**  Problem Description
- 02**  Method Design
- 03**  Experiment Results
- 04**  Conclusion

# Method Design

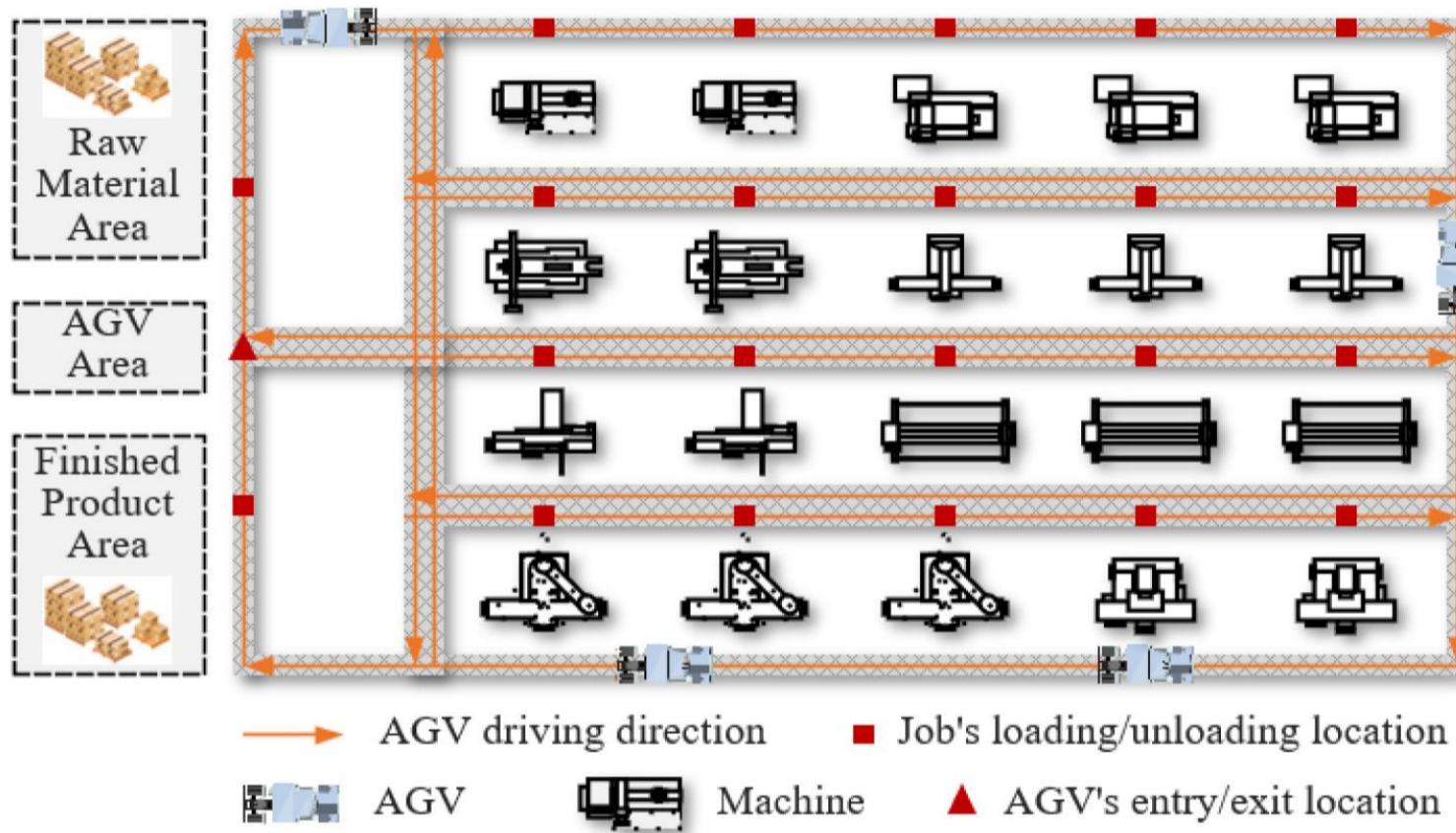


Fig. 1. Layout of flexible job shop with AGVs.

## 1) New Job Arrivals:

$$\min \text{TTC} = \min \sum_{i=1}^n (w_i \cdot \max (0, C_i - D_i)) \quad (1)$$

$$\text{MTBA} = \frac{\overline{N_{\text{ope}}} \cdot \overline{T^{\text{PT}}}}{U \cdot \sum_{x=1}^m N_x} + \frac{(\overline{N_{\text{ope}}} + 1) \cdot \overline{T^{\text{LT}}}}{U \cdot l} \quad (2)$$

$$D_i = A_i + \text{DDT}_i \cdot \sum_{j=1}^{N_i} \left( \sum_{k=1}^{N_x} T_{ijxk}^{\text{PT}} / N_x \right) + \text{DDT}_i \cdot (N_i + 1) \cdot \overline{T^{\text{LT}}} \quad (3)$$

## 2) Machine Breakdowns and Repairs:

$$A_g = \text{MTTR}/(\text{MTBF} + \text{MTTR}). \quad (4)$$

*3) Job Reworks:* In actual production, the material quality, improper worker operation and other reasons will cause the job quality to be unqualified, which leads to the requirement of job rework.

*4) Fuzzy Transportation Time:* Due to path congestion, AGV acceleration, AGV deceleration and other reasons, the actual transportation time of AGV will fluctuate based on the predetermined time.

# Method Design

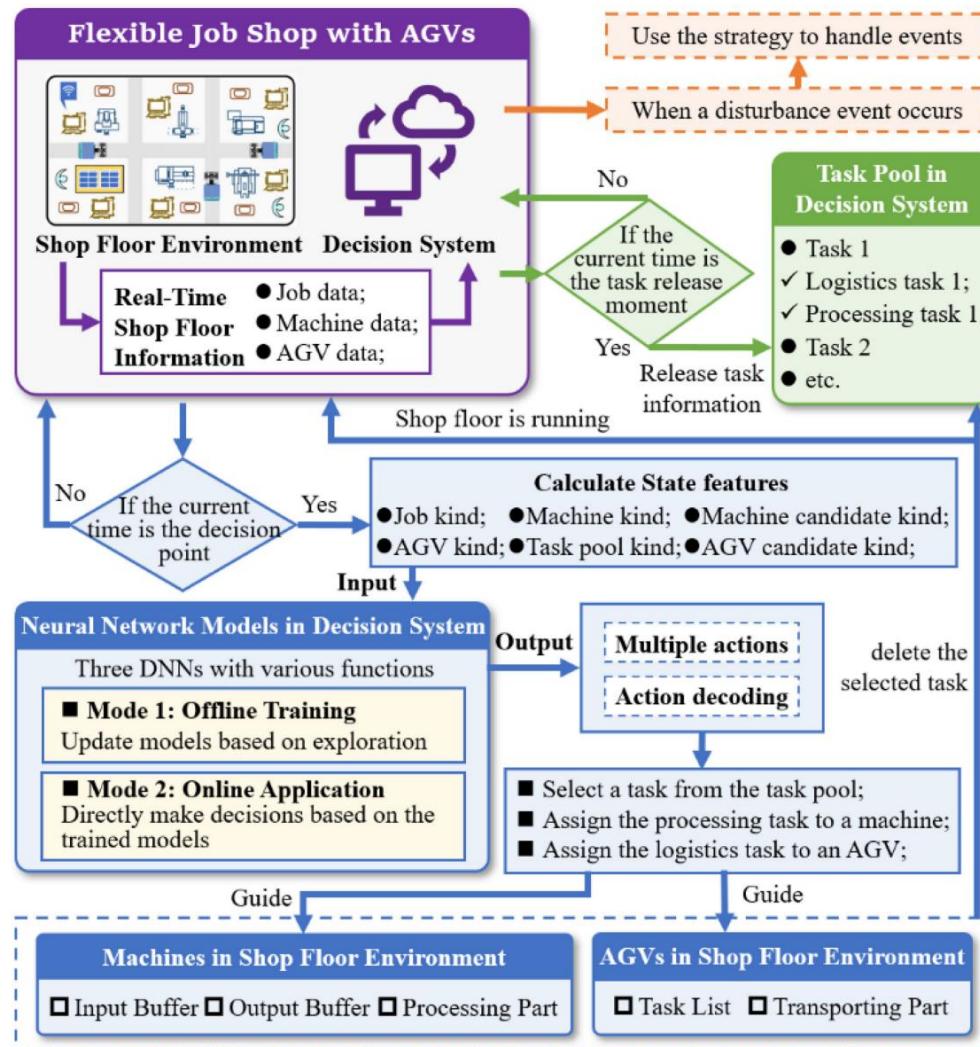


Fig. 2. Proposed real-time scheduling framework.

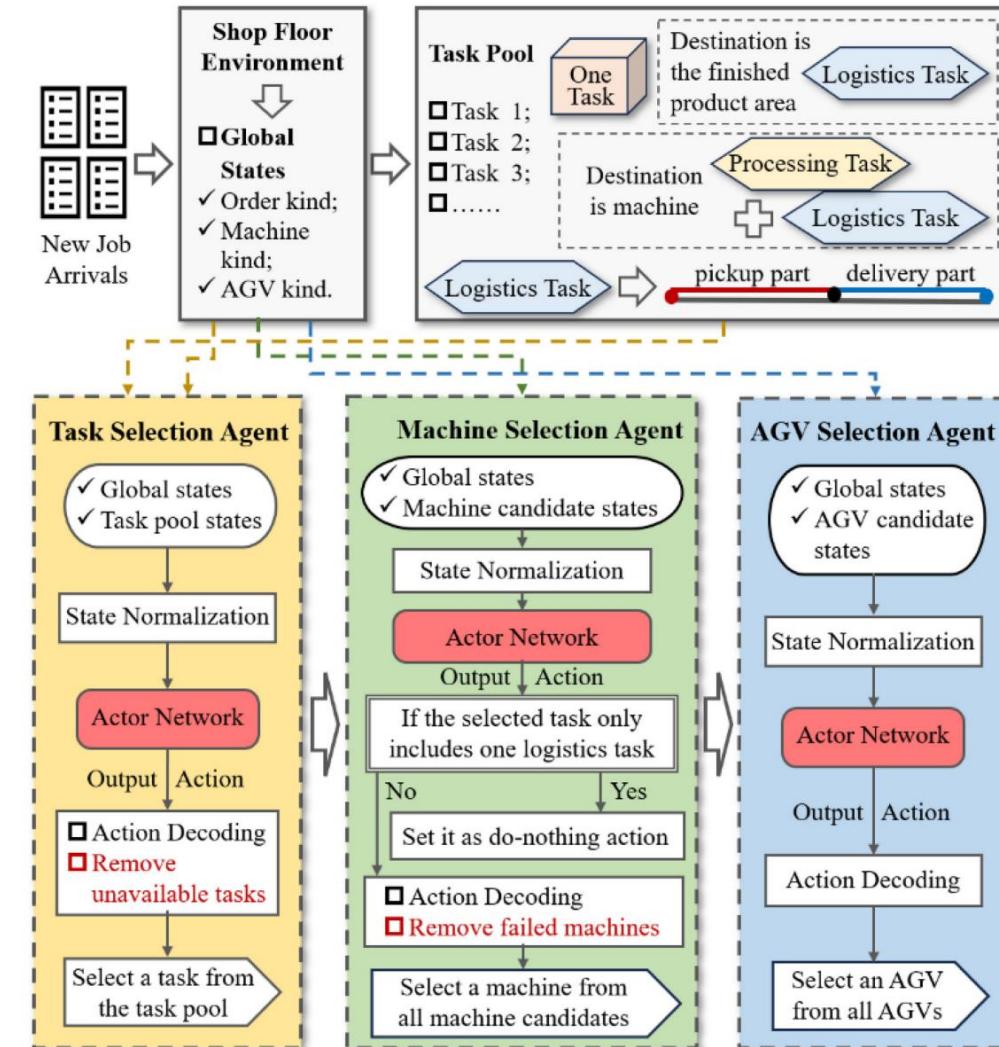


Fig. 3. Resource allocation process of agents.

# Method Design

TABLE I  
REAL-TIME ATTRIBUTES OF WORKSHOP

Notations	Descriptions
<b>Variables for manufacturing system</b>	
$t$	Current time
<b>Variables for task pool</b>	
$T_g^P$	The $g$ -th task in the task pool, $g = 1, \dots, NTP$ , where $NTP$ is the number of tasks in the task pool
$J_g^P$	The job corresponding to task $T_g^P$
$J_{th}$	The sequence number of operation for task $T_g^P$ in job $J_g^P$
$T_{gj}^P$	Time of the $j$ -th processing task for job $J_g^P$
$FT_{gj}^P$	Time that the $j$ -th processing task for job $J_g^P$ has been finished
$T_{ga}^{DLT}$	Time of the delivery part of the $a$ -th logistics task for job $J_g^P$
$FT_{ga}^{DLT}$	Time that the delivery part of the $a$ -th logistics task for job $J_g^P$ has been finished
$RT_g^P$	Remaining time of job $J_g^P$ (consider processing and logistics)
$EAT_g^P$	Earliest available time of job $J_g^P$
$EWT_g^P$	Estimated weighted tardiness of job $J_g^P$
<b>Variables for all machines</b>	
$RTING_{xk}^M$	Remaining time of the in-process task for machine $M_{xk}$
$T_{xkw}^P$	Time of the $w$ -th processing task in the buffer for machine $M_{xk}$ , $w = 1, \dots, N_{xk}$ , where $N_{xk}$ is the number of processing tasks in the machine's buffer
$CT_{xk}^M$	Shortest time for completing the in-process task and all tasks in the buffer for machine $M_{xk}$
$DPT_{gxk}$	The sum of delivery time and processing time of task $T_g^P$ for machine $M_{xk}$
<b>Variables for all AGVs</b>	
$RTING_y^A$	Remaining time of the in-transport task for AGV $V_y$
$T_{ye}^{LT}$	Time of the $e$ -th logistics task in the list for AGV $V_y$ , $e = 1, \dots, N_y$ , where $N_y$ is the number of logistics tasks in the AGV's list
$CT_y^A$	Shortest time for completing the in-transport task and all tasks in the list for AGV $V_y$
$PUT_{yg}$	Pickup time of AGV $V_y$ for task $T_g^P$

TABLE II  
SIX KINDS OF STATE FEATURES

State	Content
<b>State features for all jobs</b>	
$SF_1$	the completion rate of the order; the mean/std of completion rates for all jobs.
<b>State features for all machines</b>	
$SF_2$	the mean/std of utilization rates for all machines; the mean/std of $CT_{xk}^M$ for all machines; the proportion of faulty machines among all machines.
<b>State features for all AGVs</b>	
$SF_3$	the mean/std of utilization rates for all AGVs; the mean/std of $CT_y^A$ for all AGVs.
<b>State features for task pool</b>	
$SF_4$	the number of tasks; the mean/std of processing times for all tasks; the mean/std of completion rates for all jobs; the mean/std of $RT_g^P$ for all jobs; the mean/std/min/max of $EAT_g^P$ for all jobs; the mean/std/min/max of $EWT_g^P$ for all jobs.
<b>State features for all machine candidates of the selected job</b>	
$SF_5$	the mean/std of utilization rates; the mean/std of $CT_{xk}^M$ ; the mean of $DPT_{gxk}$ ; the proportion of faulty machines among all machine candidates; the $EAT_g^P$ of the selected job; the $EWT_g^P$ of the selected job.
<b>State features for all AGV candidates of the selected job</b>	
$SF_6$	the $CT_y^A + PUT_{yg}$ of each AGV for the selected job; the $EAT_g^P$ of the selected job; the $EWT_g^P$ of the selected job; the $CT_{xk}^M$ of the assigned machine.

# Method Design

## Action Space

$$a^{TS}(t) = [w_{BEWT}, w_{LWKR}, w_{CR}, w_{EDD}, w_{S/RT}]. \quad (12)$$

$$a^{MS}(t) = [w_{SURM}, w_{SDPT}, w_{SCTM}]. \quad (13)$$

$$a^{AS}(t) = [w_{SPUT}, w_{SCTA}, w_{SCPT}]. \quad (14)$$

1) Define PDR  $\rho \in \mathcal{P}$ , where the PDR set  $\mathcal{P} = \{BEWT, LWKR, CR, EDD, S/RT\}$ .

2) Calculate the priority vector of PDR  $\rho$ :  $PV_\rho \in \mathbb{R}^{1 \times NTP}$ .

First, define the production attribute vector of PDR  $\rho$  as  $[SV_{\rho,g}, g = 1, \dots, NTP]$ . For example, the  $SV_{\rho,g}$  of BEWT or LWKR is  $EWT_g^P$  or  $-RT_g^P$ . Then,  $PV_\rho$  using Min-Max normalization is as follows:

$$PV_\rho = \left[ \frac{SV_{\rho,g} - SV_{\rho}^{\min}}{SV_{\rho}^{\max} - SV_{\rho}^{\min}}, g = 1, \dots, NTP \right]. \quad (15)$$

3) Calculate  $Sigmoid(a^{TS}(t)) \in \mathbb{R}^{1 \times 5}$  based on  $a^{TS}(t)$

$$Sigmoid(a^{TS}(t)) = \left[ \frac{1}{1 + e^{-w_\rho}}, w_\rho \in a^{TS}(t) \right]. \quad (16)$$

---

### Algorithm 1 Action Decoding

- 1: Task selection agent receives  $S_{TS}$ , and output  $a^{JS}(t)$
  - 2: Calculate  $PDR_{TS} = [PV_{BEWT}, PV_{LWKR}, PV_{CR}, PV_{EDD}, PV_{S/RT}]$
  - 3: Calculate  $PV_{TS} = Norm(Sigmoid(a^{TS}(t)) \cdot PDR_{TS}) + w^{TS} \cdot PV_{SEAT}$
  - 4: Consider machine breakdown and remove unavailable tasks
  - 5: Select the task with highest priority from the task pool
  - 6: Machine selection agent receives  $S_{MS}$ , and output  $a^{MS}(t)$
  - 7: **if** the selected task  $T_g^P \in T_1^{Pool}$  **then**
  - 8:   Calculate  $PDR_{MS} = [PV_{SURM}, PV_{SDPT}, PV_{SCTM}]$
  - 9:   Calculate  $PV_{MS} = Norm(a^{MS}(t) \cdot PDR_{MS}) + w^{MS} \cdot PV_{SCTM}$
  - 10:   Remove failed machines
  - 11:   select the machine with highest priority from all machine candidates
  - 12: **else**
  - 13:   Set  $a^{MS}(t) = \delta_0$  ( $\delta_0 = [0, 0, 0]$ ), and represents the do-nothing action
  - 14: AGV selection agent receives  $S_{AS}$  and output  $a^{AS}(t)$
  - 15: Calculate  $PDR_{AS} = [PV_{SPUT}, PV_{SCTA}, PV_{SCPT}]$
  - 16: Calculate  $PV_{AS} = Norm(a^{AS}(t) \cdot PDR_{AS}) + w^{AS} \cdot PV_{SCTA}$
  - 17: Select the AGV with highest priority from all AGV candidates
-

# Method Design

- 4) Define  $X = [x_g, g = 1, \dots, NTP] = \text{Sigmoid}(a^{TS}(t)) \cdot PDR_{TS}$ . Its matrix shape is  $\mathbb{R}^{1 \times 5} \cdot \mathbb{R}^{5 \times NTP} = \mathbb{R}^{1 \times NTP}$

$$x_g = \sum_{\rho \in \mathcal{P}} \left( \frac{1}{1 + e^{-w_\rho}} \cdot \frac{\text{SV}_{\rho,g} - \text{SV}_{\rho}^{\min}}{\text{SV}_{\rho}^{\max} - \text{SV}_{\rho}^{\min}} \right). \quad (17)$$

- 5) Based on Min-Max normalization, the half of  $PV_{TS}$ , i.e.,  $Y = \text{Norm}(\text{Sigmoid}(a^{TS}(t)) \cdot PDR_{TS}) = \text{Norm}(X) \in \mathbb{R}^{1 \times NTP}$ , is as follows:

$$Y = \left[ \frac{x_g - x_g^{\min}}{x_g^{\max} - x_g^{\min}}, g = 1, \dots, NTP \right]. \quad (18)$$

- 6) Another half of  $PV_{TS}$  is  $Z = w^{TS} \cdot PV_{SEAT} \in \mathbb{R}^{1 \times NTP}$ .  $w^{TS}$  is the adjustment weight. Rule SEAT represents that the task with smallest  $EAT_g^P$  is selected first, and the  $\text{SV}_{\rho,g}$  of SEAT is  $EAT_g^P$ . So  $Z$  is as follows:

$$Z = \left[ w^{TS} \cdot \frac{\text{SV}_{\rho,g} - \text{SV}_{\rho}^{\min}}{\text{SV}_{\rho}^{\max} - \text{SV}_{\rho}^{\min}}, g = 1, \dots, NTP, \rho = \text{SEAT} \right]. \quad (19)$$

- 7) Finally, the decision basis  $PV_{TS}$  of task selection agent is equal to  $Y+Z$ , and its shape is  $\mathbb{R}^{1 \times NTP}$ . Each element  $P_g$  in  $PV_{TS}$  is expressed by (20), and it represents the priority of task  $T_g^P$  in task pool

$$P_g = \frac{x_g - x_g^{\min}}{x_g^{\max} - x_g^{\min}} + w^{TS} \cdot \frac{\text{SV}_{\rho,g} - \text{SV}_{\rho}^{\min}}{\text{SV}_{\rho}^{\max} - \text{SV}_{\rho}^{\min}}. \quad (20)$$

---

## Algorithm 1 Action Decoding

---

- 1: Task selection agent receives  $S_{TS}$ , and output  $a^{JS}(t)$
  - 2: Calculate  $PDR_{TS} = [PV_{BEWT}, PV_{LWKR}, PV_{CR}, PV_{EDD}, PV_{S/RT}]$
  - 3: Calculate  $PV_{TS} = \text{Norm}(\text{Sigmoid}(a^{TS}(t)) \cdot PDR_{TS}) + w^{TS} \cdot PV_{SEAT}$
  - 4: Consider machine breakdown and remove unavailable tasks
  - 5: Select the task with highest priority from the task pool
  - 6: Machine selection agent receives  $S_{MS}$ , and output  $a^{MS}(t)$
  - 7: **if** the selected task  $T_g^P \in T_1^{Pool}$  **then**
  - 8:     Calculate  $PDR_{MS} = [PV_{SURM}, PV_{SDPT}, PV_{SCTM}]$
  - 9:     Calculate  $PV_{MS} = \text{Norm}(a^{MS}(t) \cdot PDR_{MS}) + w^{MS} \cdot PV_{SCTM}$
  - 10:    Remove failed machines
  - 11:    select the machine with highest priority from all machine candidates
  - 12: **else**
  - 13:    Set  $a^{MS}(t) = \delta_0$  ( $\delta_0 = [0, 0, 0]$ ), and represents the do-nothing action
  - 14: AGV selection agent receives  $S_{AS}$  and output  $a^{AS}(t)$
  - 15: Calculate  $PDR_{AS} = [PV_{SPUT}, PV_{SCTA}, PV_{SCPT}]$
  - 16: Calculate  $PV_{AS} = \text{Norm}(a^{AS}(t) \cdot PDR_{AS}) + w^{AS} \cdot PV_{SCTA}$
  - 17: Select the AGV with highest priority from all AGV candidates
-

# Method Design

## Reward

---

---

### Algorithm 2 Reward Calculation

---

```
1: for job  $i = 1$  to  $n$  do
2:   if job  $J_i$  has been finished then
3:     Calculate  $EWT_i = \max(0, C_i - D_i) \cdot w_i$ 
4:   else
5:     Calculate  $EWT_i = \max(0, t + RT_i - D_i) \cdot w_i$ 
6:   end if
7: next for
8: Calculate  $r_t = \sum_{i=1}^n EWT_i(t) - \sum_{i=1}^n EWT_i(t+1)$ 
9: Calculate  $r_t = r_t - \alpha \cdot \left( \sum_{x=1}^m \sum_{k=1}^{N_x} ITM_{xk} \right) / \sum_{x=1}^m N_x$ 
```

---

# Method Design

---

**Algorithm 3** MAPPO-Based Training Algorithm

---

1: Initialize the actor networks of three agents  $\pi_{TS}(\theta_{TS})$ ,  $\pi_{MS}(\theta_{MS})$ ,  $\pi_{AS}(\theta_{AS})$ ; the critic networks of three agents  $v_{TS}(\phi_{TS})$ ,  $v_{MS}(\phi_{MS})$ ,  $v_{AS}(\phi_{AS})$ ; three memory buffers  $MB_{TS}$ ,  $MB_{MS}$ ,  $MB_{AS}$

2: Initialize all hyperparameters of MAPPO, memory number  $MN = 0$

3: **for**  $episode = 1: L$  **do**

4:   Initialize the shop floor environment using problem  $PROB$

5:   **while** all jobs have not returned to the finished product area **do**

6:     Implement Algorithm 1, and assign resources to the selected task

7:     Execute three actions, and machines and AGVs keep running

8:     Update task pool in action execution process

9:     Observe the new state  $\{SF_1, SF_2, SF_3, SF_4\}_{t+1}$

10:    Calculate the reward  $r_t$  based on Algorithm 2

11:    **if** the current state is the terminal state **then**

12:      Capture the observations  $\{SF_5, SF_6\}_{t+1}$

13:    **end if**

14:    Put  $\{S_{TS}(t), a^{TS}(t), r_t, S_{TS}(t + 1)\}$  into  $MB_{TS}$

15:    Put  $\{S_{MS}(t), a^{MS}(t), r_t, S_{MS}(t + 1)\}$  into  $MB_{MS}$

16:    Put  $\{S_{AS}(t), a^{AS}(t), r_t, S_{AS}(t + 1)\}$  into  $MB_{AS}$

17:    Update the memory number  $MN = MN + 1$

18:    **if** one disturbance event occurs **then**

19:      Execute the corresponding strategy to handle it

20:    **end if**

21: **end while**

22: Form trajectory  $\tau$  using the transitions of current episode

23: Compute advantage estimate  $\widehat{A}$  via GAE using PopArt

24: Compute reward-to-go  $\widehat{R}$  on  $\tau$  and normalize with PopArt

25: **if**  $MN \geq$  memory update capacity  $MUC$  **then**

26:    Adam update  $\{\pi_{TS}(\theta_{TS}), \pi_{MS}(\theta_{MS}), \pi_{AS}(\theta_{AS})\}$  with (21)

27:    Adam update  $\{v_{TS}(\phi_{TS}), v_{MS}(\phi_{MS}), v_{AS}(\phi_{AS})\}$  with (23)

28:    Regenerate a new training problem  $PROB$

29:    Clear memory buffers  $\{MB_{TS}, MB_{MS}, MB_{AS}\}$  and set  $MN = 0$

30: **end if**

31: **next for**

---

# Method Design

## Strategy for Handling Disturbance Events

1) *New Job Arrivals*: When a new job arrives at the workshop, the decision system puts its first task into the task pool, so that the agents can assign machine and AGV to it.

2) *Machine Breakdowns*: When a machine fails, determine the corresponding disrupted tasks and put them into the task pool. These tasks include: 1) tasks being processed by the faulty machine; 2) tasks in the input buffer of the faulty machine; and 3) tasks of the above jobs that are assigned in advance. The agents will reallocate the disrupted tasks based on the workshop state and job urgency degree, which prevents these jobs from having a large tardiness due to machine failure.

When the machine has finished repairing, determine the corresponding disrupted tasks and put them into the task pool. These tasks include: 1) tasks in the input buffers of same-kind machines; 2) tasks of the above jobs that are assigned in advance; (note that exclude tasks that are being delivered). The agents will reallocate the disrupted tasks based on the workshop state and job urgency degree, which can prevent the repaired machine from being idle for a long time.

3) *Job Rework*: When a job rework event occurs, put the task corresponding to the rework operation into the task pool, and the agents can reassign machine and AGV to it.

4) *Fuzzy Transportation Time*: The DRL-based agents can learn to allocate manufacturing resources reasonably under fuzzy transportation time through the relevant training.



## Contents

- 01**  Problem Description
- 02**  Method Design
- 03**  Experiment Results
- 04**  Conclusion

# Experiment Results

## Performance Comparison Index

$$SP_Y^X = (\text{TTC}_Y - \text{TTC}_X) / \text{TTC}_X \cdot 100\% \quad (26)$$

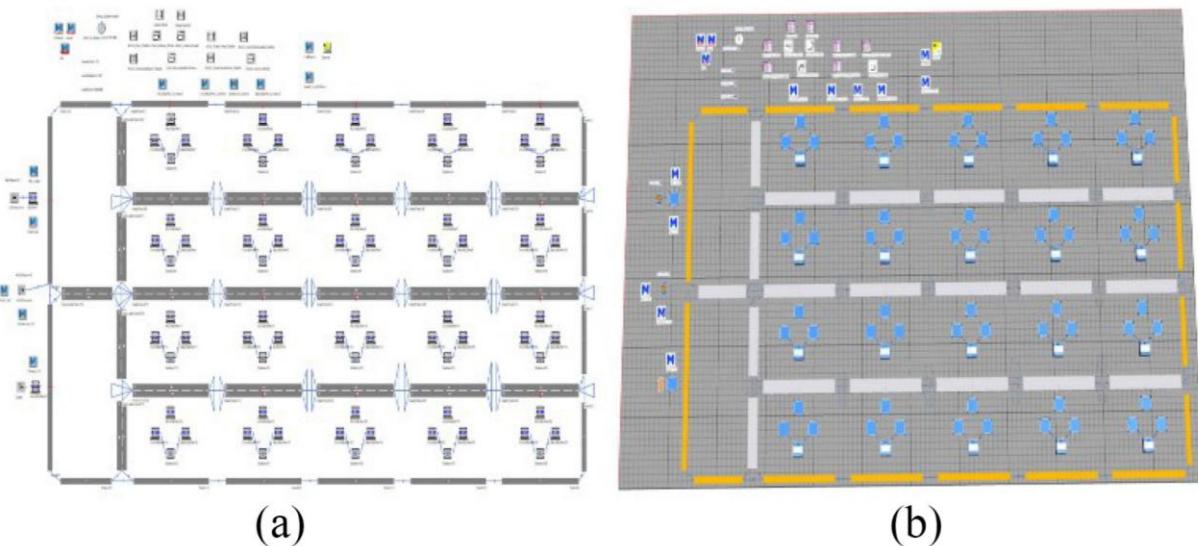


Fig. 4. Plant simulation model. (a) 2-D model. (b) 3-D model.

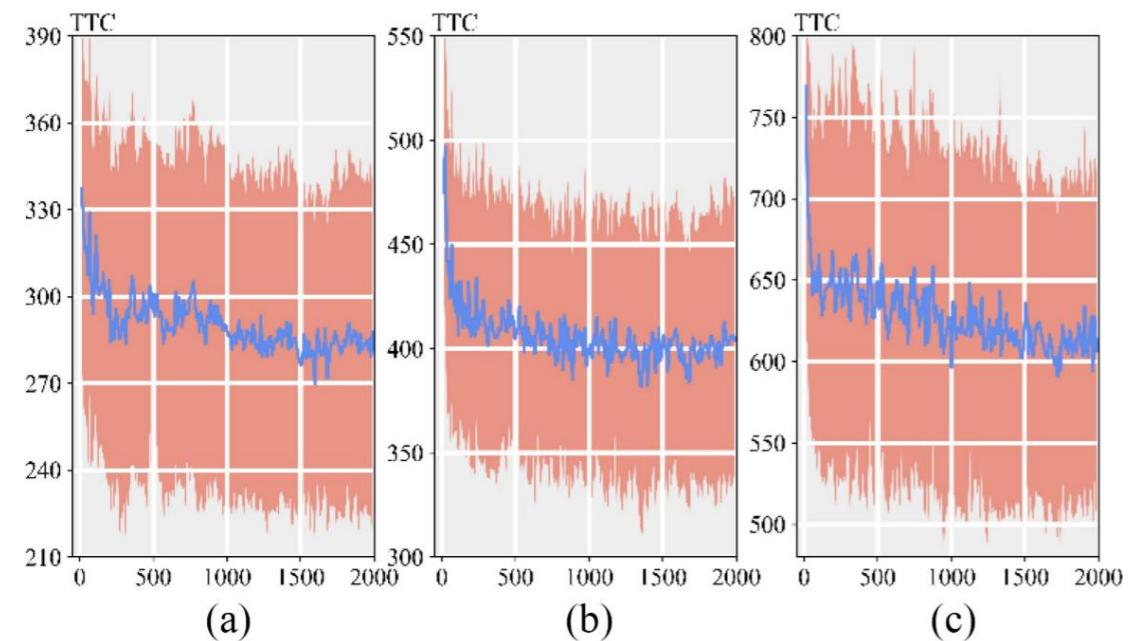


Fig. 5. Evaluation results under different problem settings. (a) Problem setting 1. (b) Problem setting 2. (c) Problem setting 3.

# Experiment Results

TABLE III  
ABLATION OF FOUR STRATEGIES IN ACTION DECODING

MAPPO Model	Sigmoid	TS adjustment	MS adjustment	AS adjustment
1	-0.4/7.5	1125.4/593.1	63.6/35.5	8468.6/5485.6
2	3.2/5.2	1164.3/589.8	51.2/36.8	7076/4353.8
3	4.5/9.9	1173.2/619.8	39.1/23.8	6067.6/3392.7
4	2.7/8.5	1245.9/686.6	30.1/15.6	6617.2/6130.3
5	3.8/9.4	1236.9/675.9	32.7/24.4	7362.8/5591.9
6	0.2/6.5	1256.4/660.9	31.4/22.4	9361.1/4663.3
7	4.5/10.6	1262.2/728.2	27.6/25.4	8379.5/5697.5
8	0.9/6.2	1338.1/728.9	23.7/14.8	7215.2/5819.3
9	1.6/6.8	1382.4/776.6	23.6/15.7	8949.3/6736.3
10	2.5/7.7	1498.7/714	14.9/12.3	7503.7/5408.3

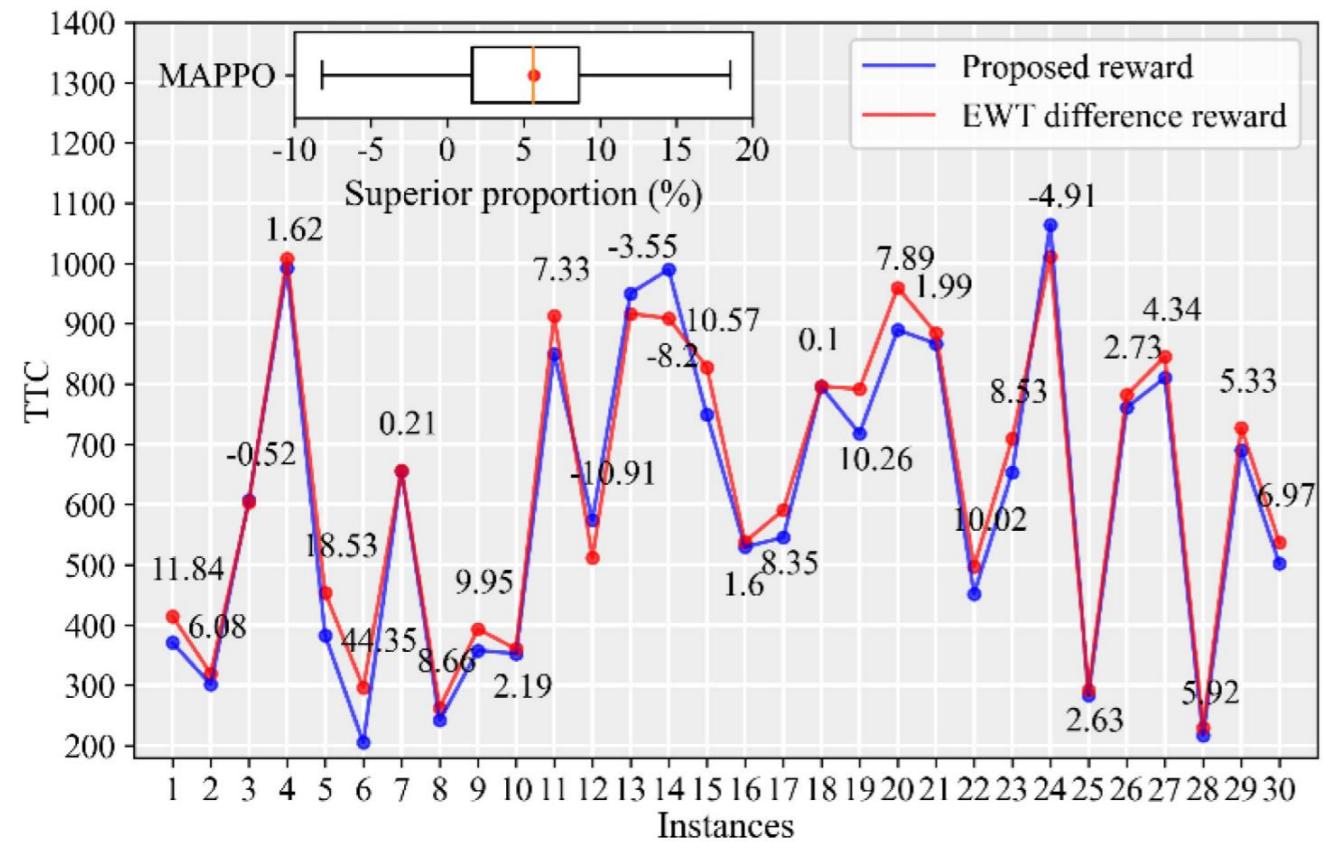


Fig. 6. Ablation of machine idle time in reward function.

# Experiment Results

TABLE IV  
COMPARISON WITH COMPOSITE PDRS CONSIDERING  
NEW JOB ARRIVALS

$J_{new}$	DDT	$U$	$R_1$	$R_2$	$R_3$	$R_4$	$R_5$	$R_6$	$R_7$	$R_8$	$R_9$	$R_{10}$
100	1.5	0.9	37	59	40	33	55	33	58	39	49	79
	0.95	15	28	25	16	29	17	27	26	30	30	55
	2	0.9	38	57	44	25	41	23	41	34	46	71
	0.95	31	58	47	19	45	25	48	34	41	41	61
150	1.5	0.9	36	63	46	29	51	27	66	40	52	83
	0.95	23	43	26	21	33	19	32	24	35	35	56
	2	0.9	50	96	78	43	74	41	76	59	68	118
	0.95	46	77	70	32	66	30	62	72	55	55	113
200	1.5	0.9	45	71	43	30	64	34	70	52	57	103
	0.95	33	50	39	29	48	29	50	39	46	46	69
	2	0.9	19	46	38	20	46	17	46	36	34	72
	0.95	32	63	40	28	55	25	49	38	49	49	79

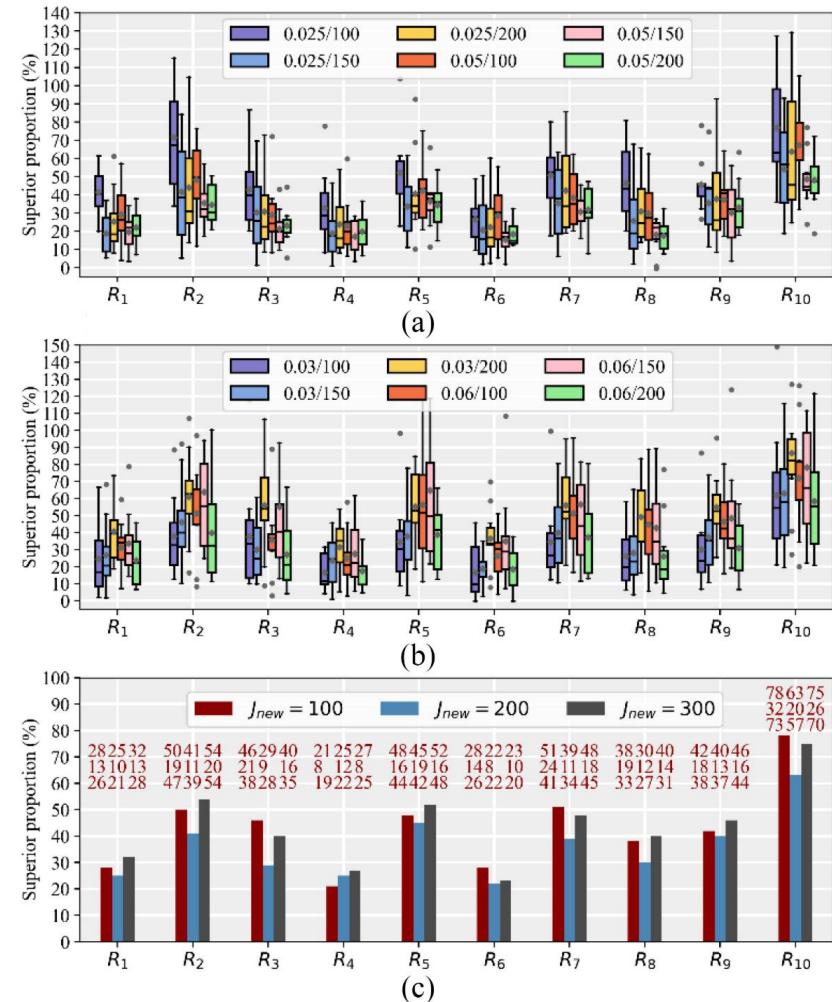


Fig. 7. Comparison with composite PDRs under various disturbance.  
(a) Robustness comparison under new job arrivals and machine breakdowns.  
(b) Robustness comparison under new job arrivals and job reworks.  
(c) Robustness comparison under new job arrivals and fuzzy transportation time.

# Experiment Results

TABLE V  
COMPARISON WITH GP AND DRL CONSIDERING NEW JOB ARRIVALS

$J_{new}$	$DDT_H$	$U$	$GR_1$	$GR_2$	$GR_3$	$GR_4$	Luo	Park	Li	Lei
100	1.5	0.9	21	28	21	24	41	6	18	-4
		0.95	12	9	10	15	19	7	14	4
	2	0.9	22	27	25	23	42	18	19	9
		0.95	20	26	20	19	27	14	14	3
150	1.5	0.9	26	27	21	21	39	14	19	4
		0.95	18	16	13	11	26	6	12	8
	2	0.9	36	48	36	39	66	20	34	6
		0.95	28	28	21	34	50	11	29	8
200	1.5	0.9	28	29	29	27	47	11	23	3
		0.95	23	21	18	20	33	14	25	4
	2	0.9	17	20	14	17	31	8	15	0
		0.95	22	33	20	16	28	13	23	0

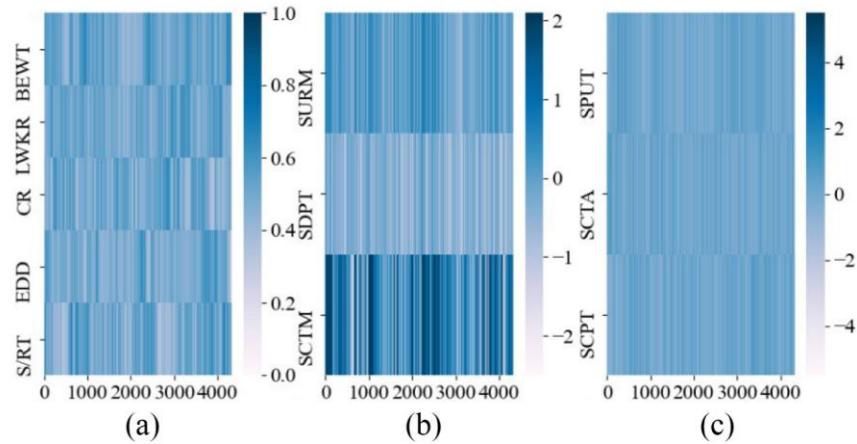


Fig. 9. Heatmap of actions on three instances. (a) Task selection agent.  
(b) Machine selection agent. (c) AGV selection agent.

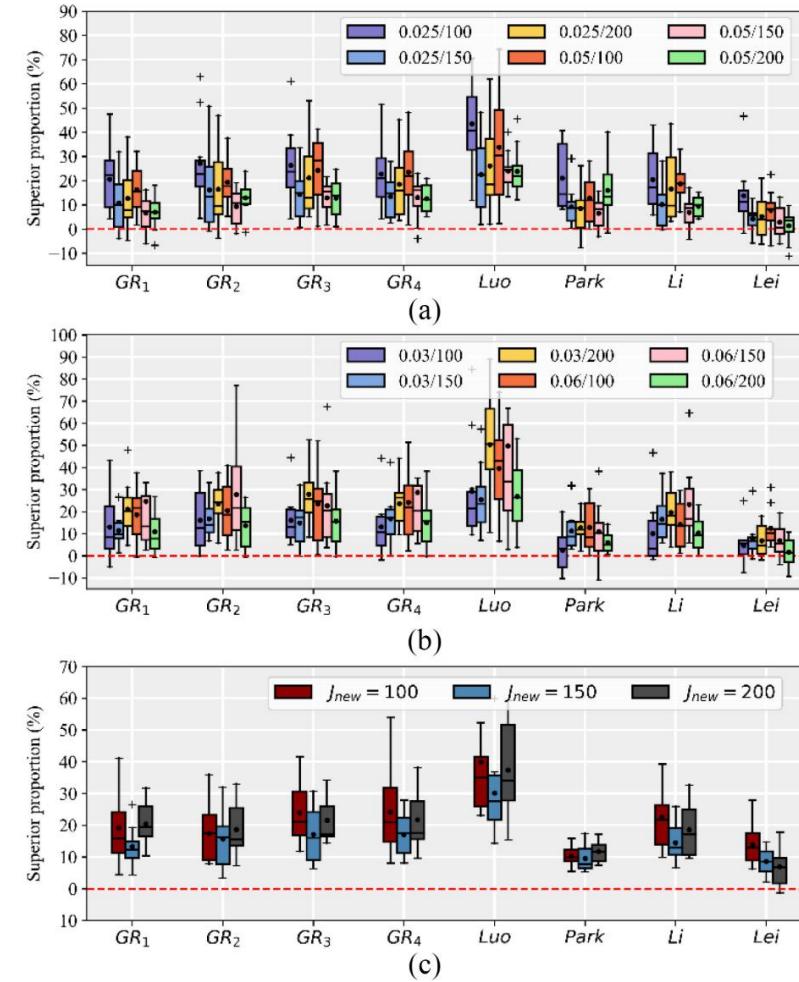


Fig. 8. Comparison with GP and DRL under various disturbance events.  
(a) Robustness comparison under new job arrivals and machine breakdowns.  
(b) Robustness comparison under new job arrivals and job reworks. (c) Robustness comparison under new job arrivals and fuzzy transportation time.



## Contents

- 01**  Problem Description
- 02**  Method Design
- 03**  Experiment Results
- 04**  Conclusion

# Future Work

## Contributions

This paper proposes a real-time scheduling method for DFJSP AGVs based on MARL, which optimizes action decoding through PDR weights and adjustment mechanisms, improves learning efficiency, and designs a generalized state space, improved reward function, and disturbance handling strategy to enhance performance and robustness. The experimental results show that this method is superior to composite PDRs, GP, and four DRL methods, and can achieve efficient and stable workshop scheduling under various disturbance conditions.

## Future Work

Further research can combine domain knowledge to optimize action decoding, introduce advanced deep learning models, and improve the reward function based on inverse reinforcement learning to enhance the stability and optimality of the algorithm.

## A deep multi-agent reinforcement learning approach to solve dynamic job shop scheduling problem

Renke Liu, Rajesh Piplani, Carlos Toro

**Publish Journal:** Computers and Operations Research (2023)



## Contents

- 01**  **Problem Description**
- 02**  **Method Design**
- 03**  **Experiment Results**
- 04**  **Conclusion**



## Contents

- 01**  Problem Description
- 02**  Method Design
- 03**  Experiment Results
- 04**  Conclusion

# Problem Description

- **Background:** The manufacturing industry is undergoing a data-driven transformation, necessitating real-time production scheduling techniques to manage the increasing complexity and volatility of modern shop floors.
- **Challenge:** Existing dynamic scheduling methods struggle with invariable problem sizes and non-stationary environments, limiting their effectiveness in real-world applications.
- **Method:** A deep multi-agent reinforcement learning-based approach with double deep Q-networks, centralized training with decentralized execution, and enhanced state-action representations is proposed to achieve efficient and scalable dynamic job shop scheduling.

# Problem Description

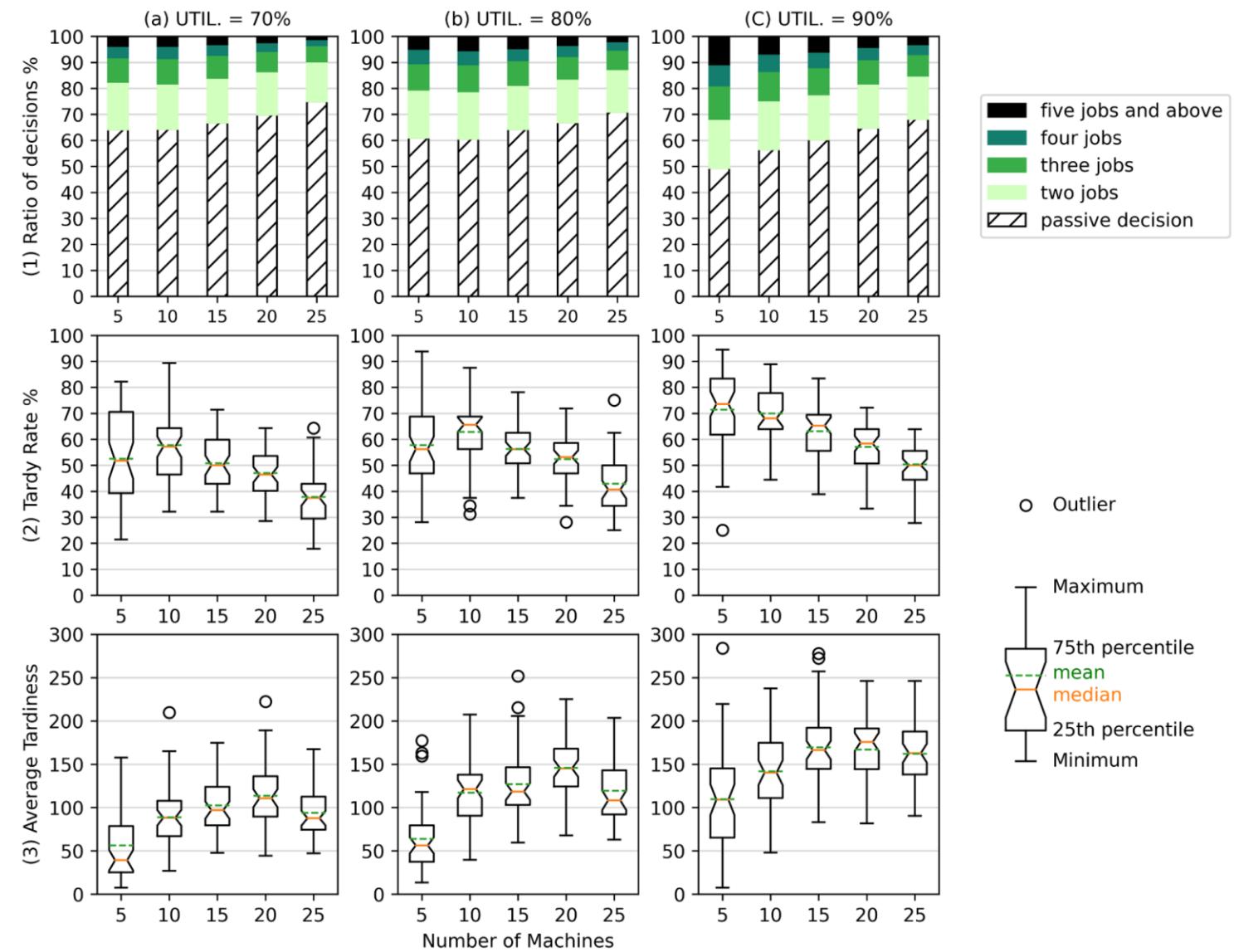


Fig. 1. Patterns of production under three levels of utilization rate.



## Contents

- 01**  Problem Description
- 02**  Method Design
- 03**  Experiment Results
- 04**  Conclusion

# Method Design

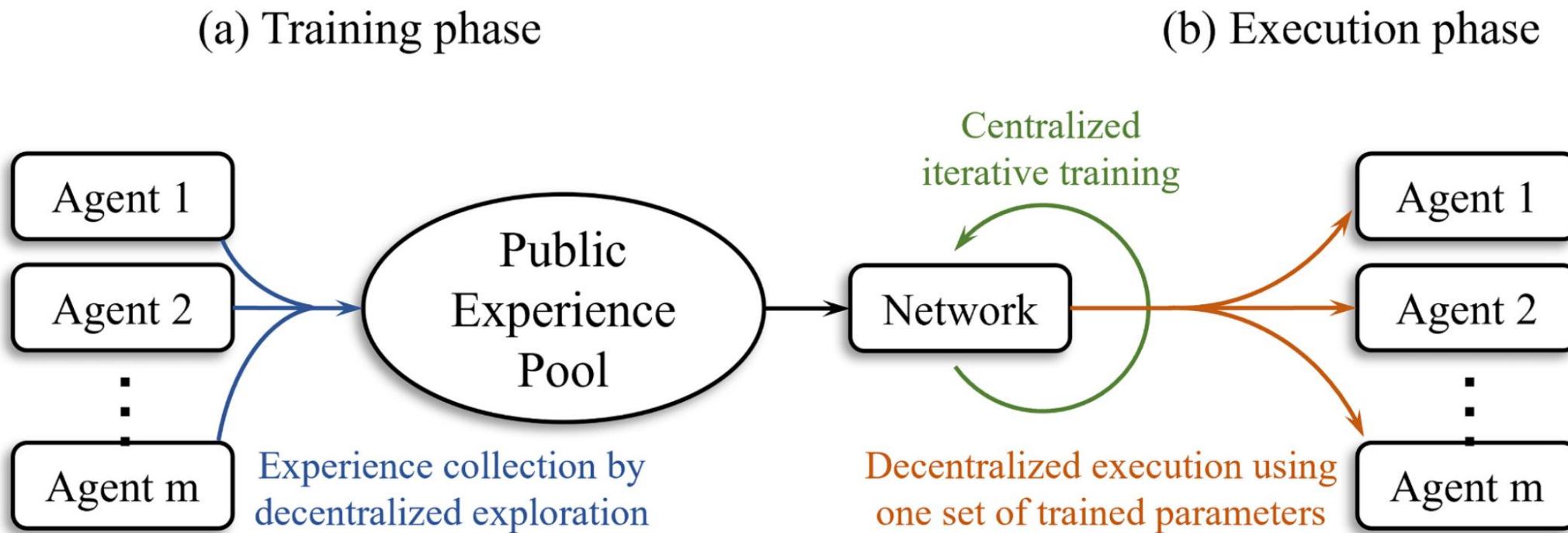


Fig. 2. Centralized training and decentralized execution scheme.

# Method Design

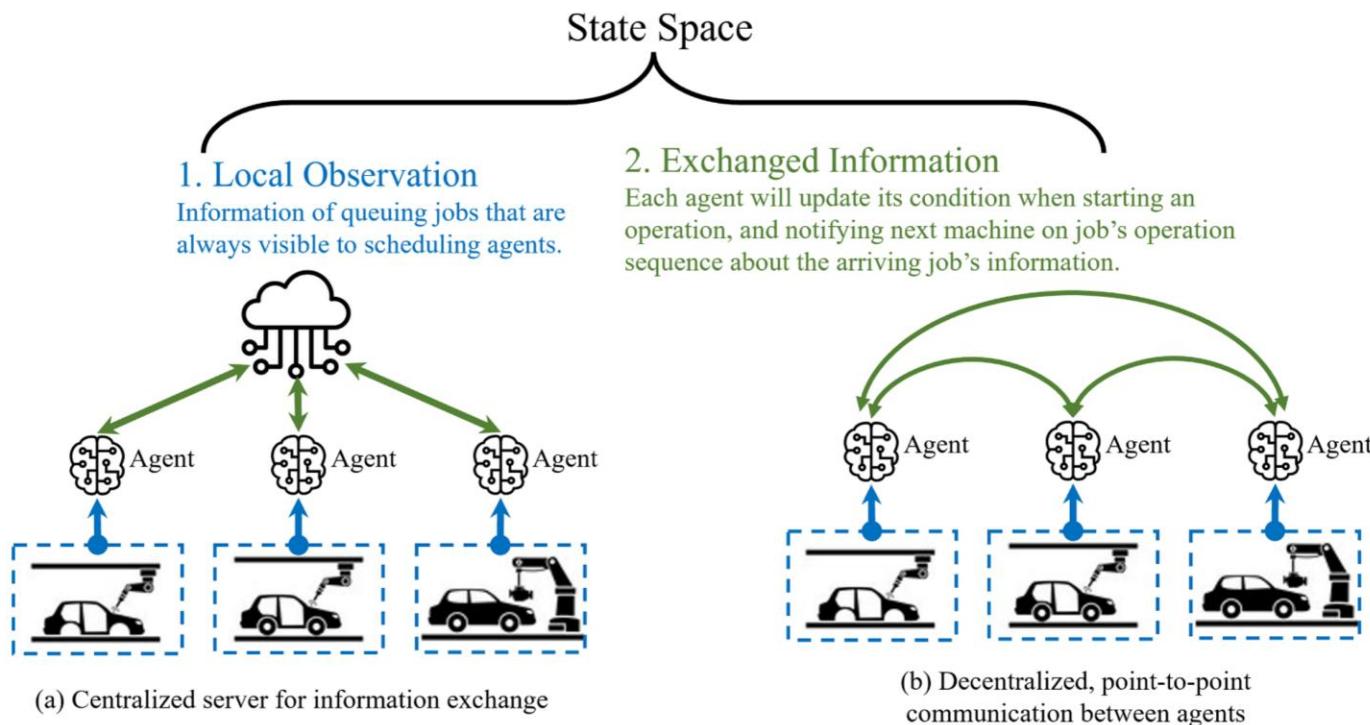


Fig. 3. Hypothetical information flow on the shop floor.

- (1) **Shortest processing time (SPT)**: selects the job that can be processed in the shortest time, to minimize the slack time consumption for other queuing jobs.
- (2) **Least work remaining (LWKR)**: selects the job that could exit system in the shortest time (estimated), to reduce the overall congestion level.
- (3) **Minimal slack (MS)**: selects the most urgent job to protect it from being tardy or incurring too much tardiness.
- (4) **Work in queue (WINQ)**: selects the job that exposes its succeeding operation to least congestion, to balance the workload of machines and avoid wasting machine idle time.

Some extra variables are introduced to facilitate the information exchange in system: (1) for job  $J_i$ , let the available time of its succeeding machine be denoted as  $SAM_i$ ; (2) the time  $J_i$  have waited in the current queue, be denoted as  $CQ_i$ ; and (3) the time till next arrival of job to machine  $M_k$ , be denoted as  $NA_k$ .

Five features are extracted for each candidate job: (1)  $t_{i,k}$ : the time expense of processing the candidate job, also the prolonged queuing time for other jobs (if selected); (2)  $WR_i$ : the minimal time before the completion of a job, showing how likely the congestion in the system can be reduced; (3)  $S_i$ : slack time, measurement of the urgency of a job; (4)  $SAM_i$ : a surrogate measurement of the queuing time of job's next operation; (5)  $CQ_i$ : the time that the agent has detained the job, indicating the accountability of agent in job's tardiness (if any).

# Method Design

---

**Algorithm 1** Minimal-repetition approach to build the state space for  $M_k$

---

**Require:**  $J^k$

- 1: Initialize empty state tensor:  $s_t \leftarrow []$
- 2: Initialize the buffer job set:  $J^{buffer} \leftarrow J^k$
- 3: Initialize the rule set:  $Rules \leftarrow [SPT, LWKR, MS, WINQ]$
- 4: Initialize the rule count:  $x \leftarrow 1, y \leftarrow 1$
- 5: **while**  $J^{buffer} \neq empty$  and  $x \leq 4$  **do**
- 6:     Select  $J_a$  from  $J^{buffer}$  by  $x^{th}$  rule in  $Rules$
- 7:     Append feature vector  $[t_{a,k}, WR_a, S_a, SAM_a, CQ_a]$  to  $s_t$
- 8:      $x \leftarrow x + 1$
- 9:     Delete  $J_a$  from  $J^{buffer}$
- 10: **end while**
- 11: **while**  $x \leq 4$  **do**

---

- 12:     Select  $J_b$  from  $J^k$  by  $y^{th}$  rule in  $Rules$
- 13:     Append feature vector  $[t_{b,k}, WR_b, S_b, SAM_b, CQ_b]$  to  $s_t$
- 14:      $x \leftarrow x + 1, y \leftarrow y + 1$
- 15: **end while**
- 16: **if** a job that will arrive at  $M_k$  is being processed elsewhere **then**
- 17:     Identify the job  $J_c$  that will arrive
- 18:     Append the feature vector  $[t_{c,k}, WR_c, S_c, AM_k, NA_k]$  to  $s_t$
- 19: **else**
- 20:     Append the dummy feature vector  $[0, 0, 0, AM_k, 0]$  to  $s_t$
- 21: **end if**
- 22: **return**  $s_t$

---

# Method Design

## Chronology joint action and knowledge-based reward shaping

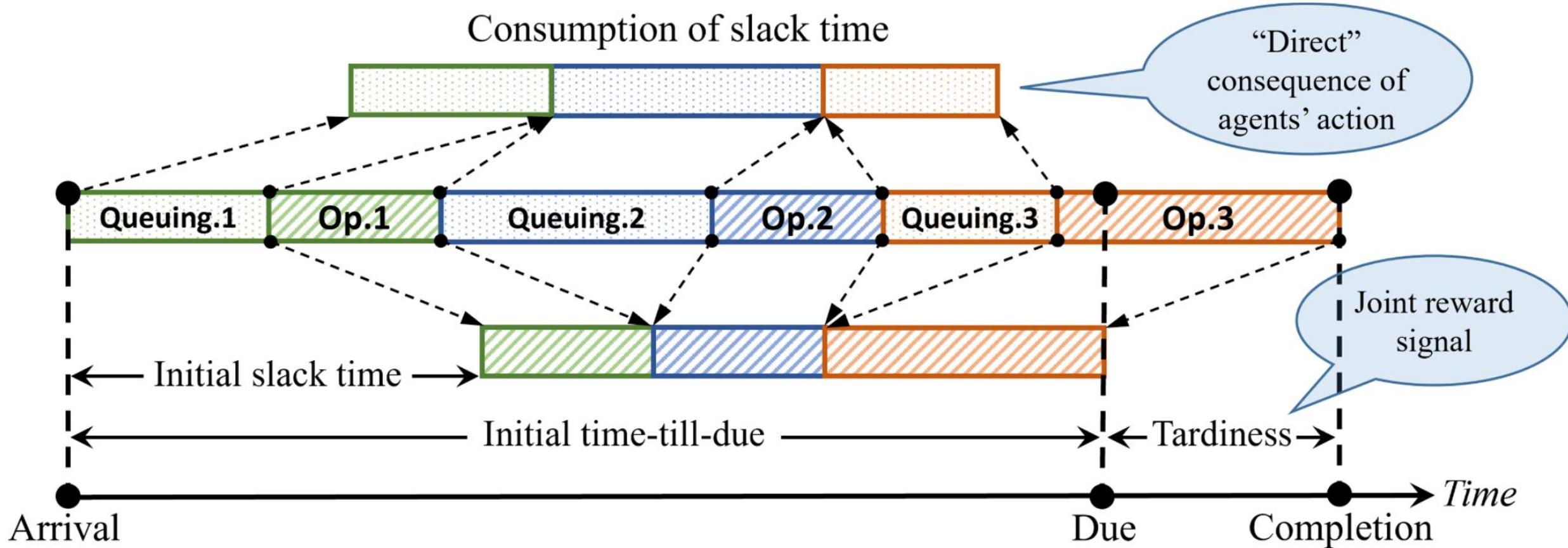
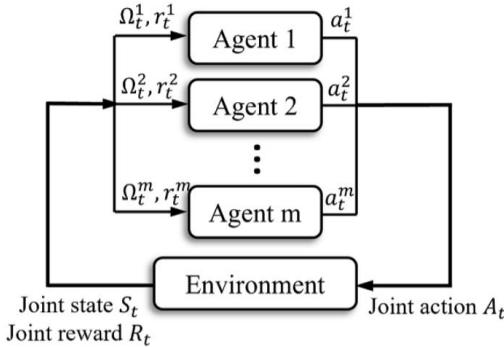


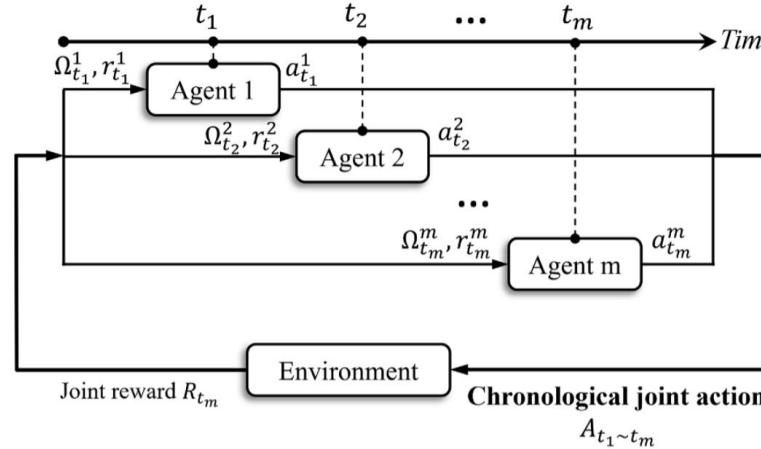
Fig. 4. Example production history.

# Method Design

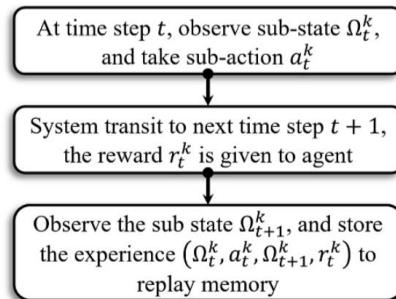
(a.1) Agent-environment interaction in common MARL tasks



(a.2) Agent-environment interaction in scheduling problem



(b.1) Transition and experience building in classic deep MARL algorithm



(b.2) Transition and experience building in proposed deep MARL algorithm

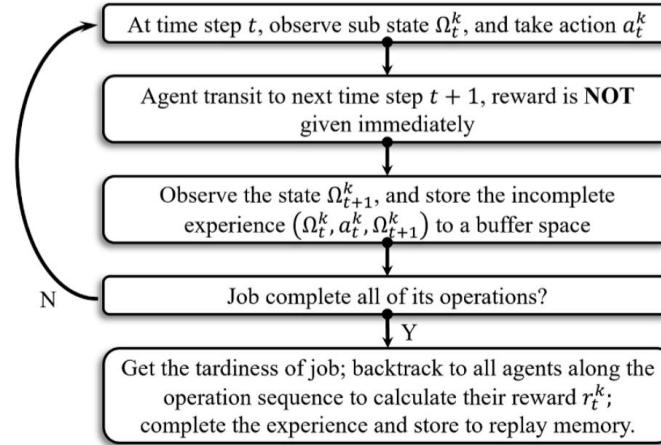


Fig. 5. Comparison between common MARL tasks and scheduling problem.

- Queue time before each operation

$$Q_i = [Q_i^1, \dots, Q_i^{O_i}]$$

- Slack time upon the job's arrival at each machine

$$SA_i = [S_i^1, \dots, S_i^{O_i}]$$

- The reward vector for each agent in the trajectory

$$R_i = [R_i^1, \dots, R_i^{O_i}]$$

**Algorithm 2** Calculation of reward upon the completion of job  $J_i$

**Require:**  $T_i, Q_i$  and  $SA_i$

```

1: Initialize empty reward vector  $R_i \leftarrow []$ 
2: if  $T_i == 0$  then
3:   Fill reward vector with 0s:  $R_i \leftarrow [0, \dots, 0], |R_i| = O_i$ 
4: else
5:   for  $j \leftarrow 1$  to  $O_i$  do
6:     Re-construct the queue time:  $RQ_i^j = (1 - \alpha) \times Q_i^j + \alpha \times Q_i^{j+1}$ 
7:     Calculate the criticality factor:  $\beta = 1 - (S_i^j / (|S_i^j| + \delta))$ 
8:     Get the reward:  $R_i^t = -(\beta \times RQ_i^j / \varphi)^2$ 
9:     Clip the  $R_i^t$  to  $[-1, 0]$ 
10:    Append  $R_i^t$  to  $R_i$ 
11:   end for
12: end if
13: return  $R_i$ 

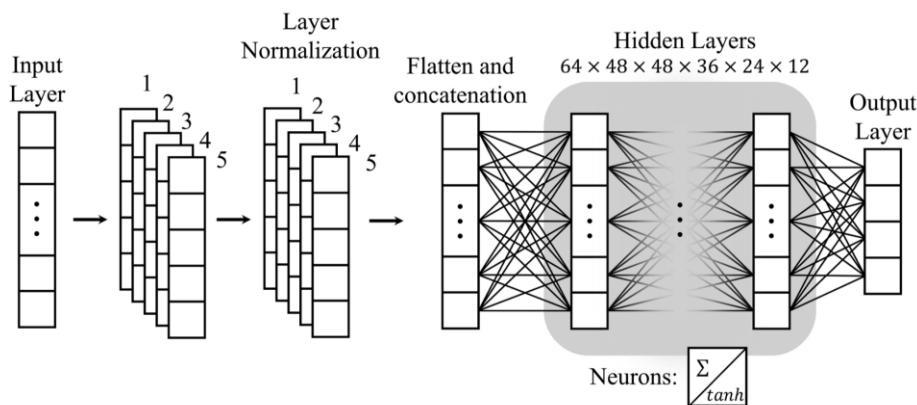
```



## Contents

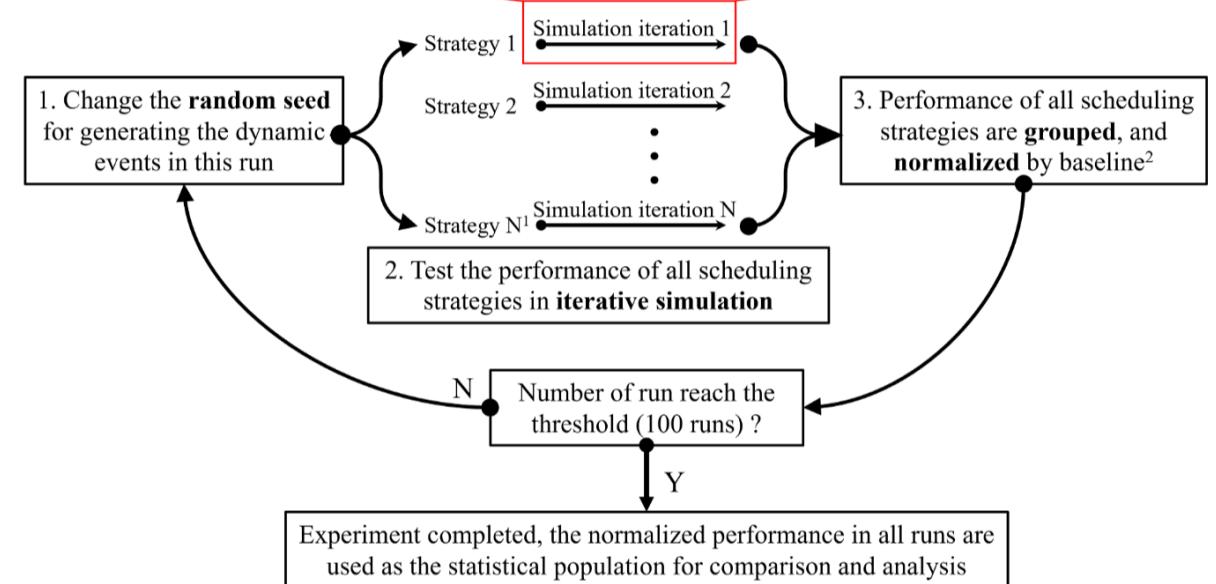
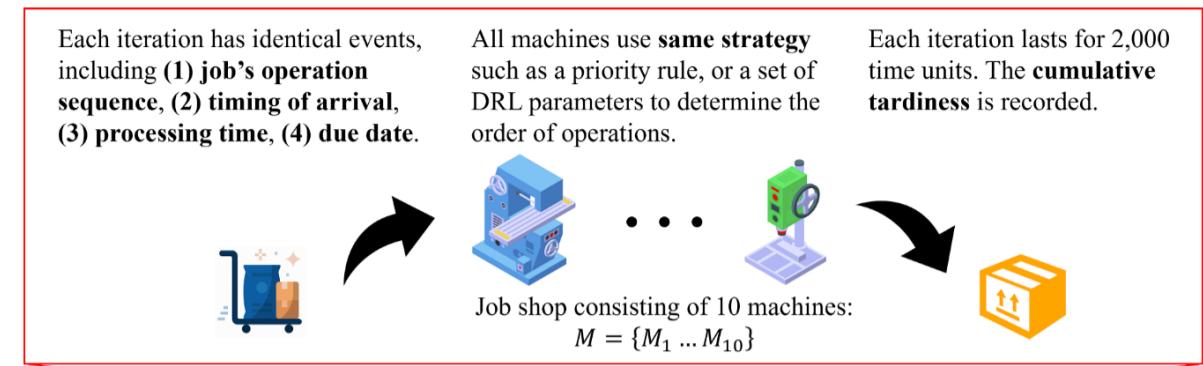
- 01**  Problem Description
- 02**  Method Design
- 03**  Experiment Results
- 04**  Conclusion

# Experiment Results



**Table 1**  
Parameters of training and hyperparameters of MLP.

Category	Parameter	Value
deep MARL	Discount factor ( $\gamma$ )	0.95
	Exploration rate ( $\epsilon$ )	40% decay to 10%
	Minibatch size	64
	Replay memory size	1024
MLP	Activation function	tanh
	Loss function	Huber loss
	Optimizer	SGD, momentum = 0.9
	Learning rate	$5 \times 10^{-3}$ decay to $10^{-3}$
	Hidden layers	$64 \times 48 \times 48 \times 36 \times 24 \times 12$

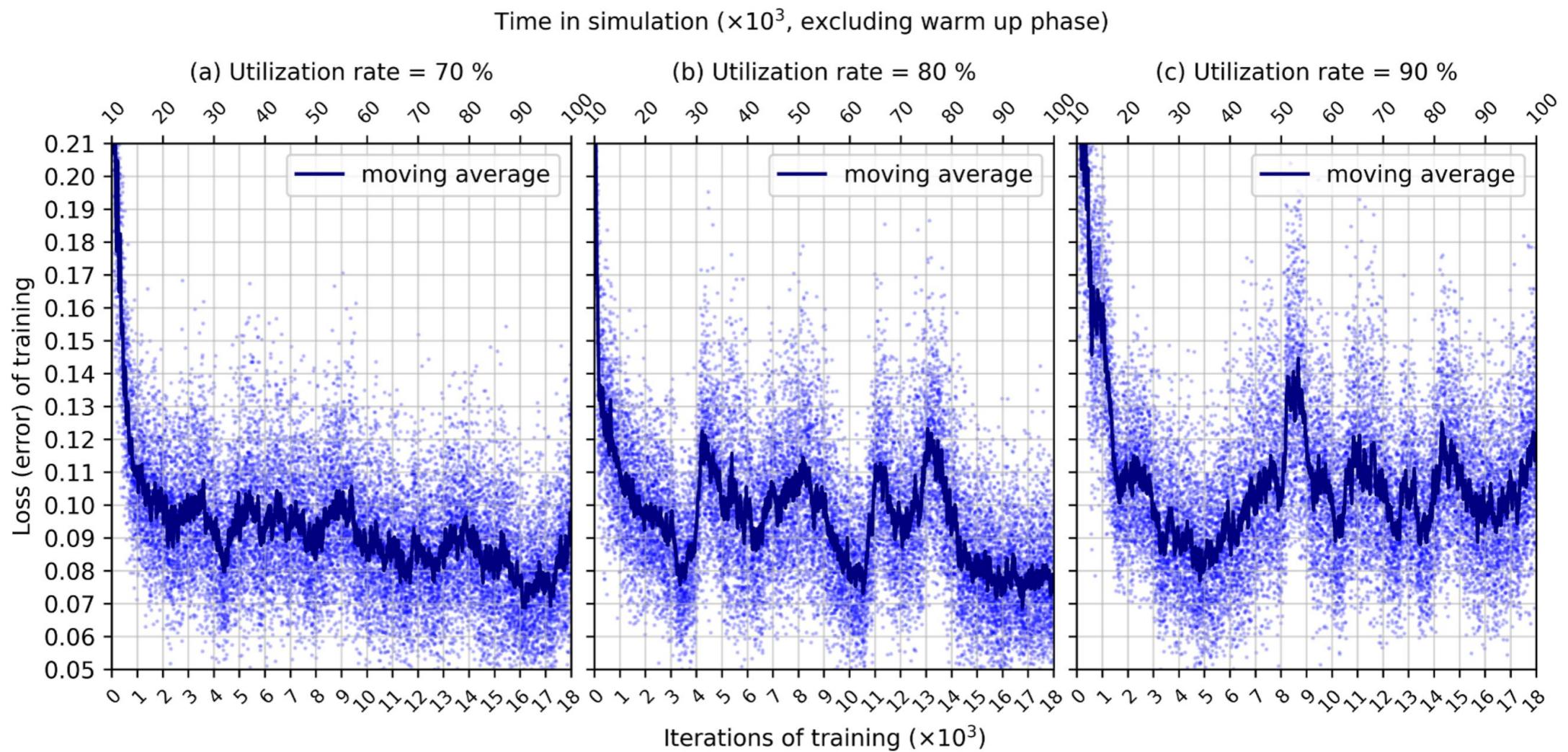


<sup>1</sup> The number of strategies and respective number of iterations in a simulation run subject to the size of benchmark set. In Section 5.1, N = 8; while in Section 5.2, N = 21.

<sup>2</sup> Baseline is the performance of algorithm with minimal enhancements (I-DDQN algorithm in Section 5.1), or the priority rule that manages the job sequence in a most passive way (FIFO rule in Section 5.2).

Fig. 8. Experiment in dynamic environments (in one scenario).

# Experiment Results



**Fig. 7.** Training loss.

# Experiment Results

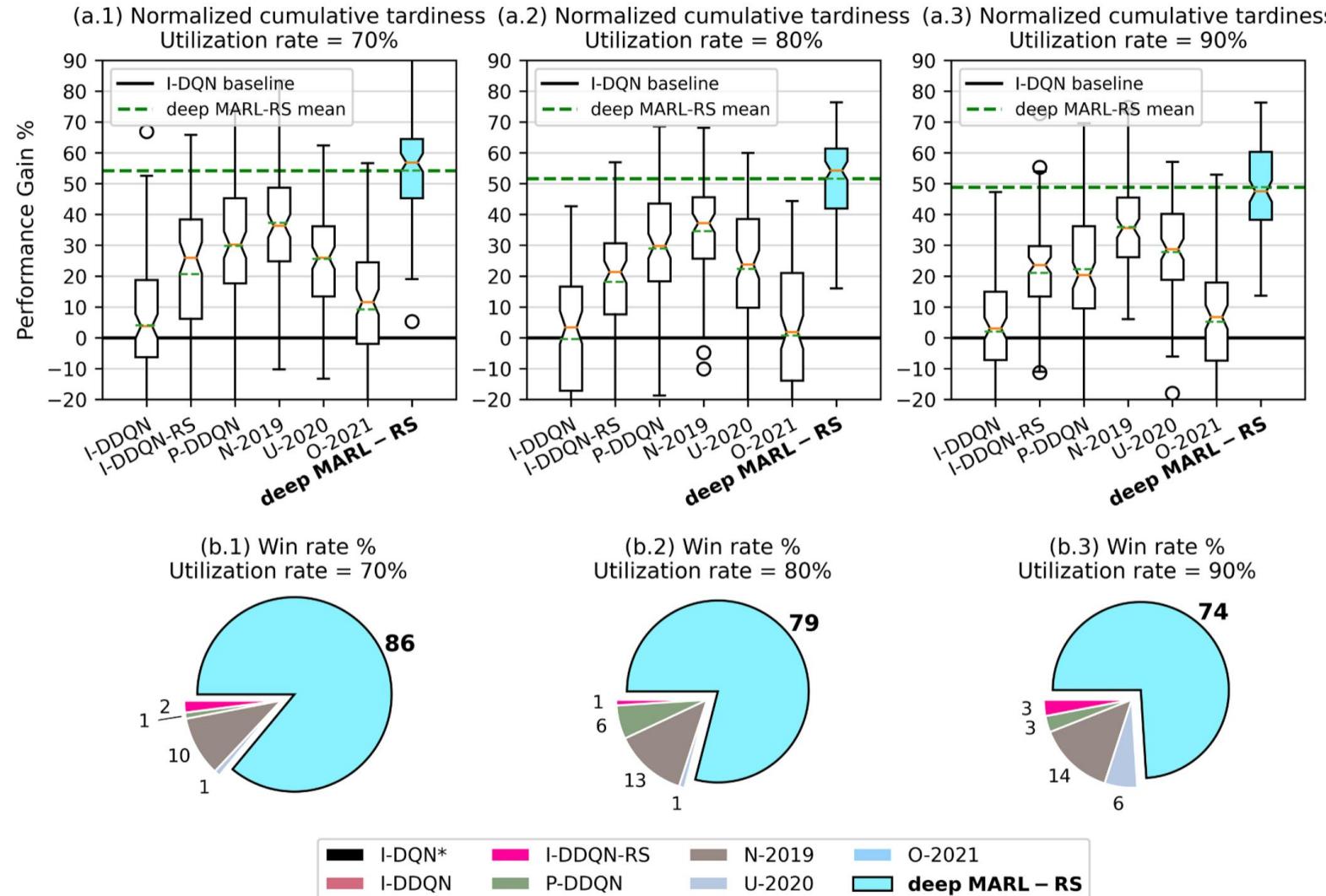


Fig. 9. Result of ablation study and peer comparison in dynamic environment.

**Table 4**  
Quantified contribution of components (by percentage).

Category	Calculation	Contribution (by percentage) under three utilization levels		
		70%	80%	90%
Training scheme + Parameter sharing	$\frac{T_{sum}^{I-DDQN} - T_{sum}^{P-DDQN}}{T_{sum}^{I-DDQN}}$	24.44	27.73	19.00
	$\frac{T_{sum}^{I-DDQN-RS} - T_{sum}^{deepMARL-RS}}{T_{sum}^{I-DDQN-RS}}$	40.46	38.96	33.83
Reward-shaping	$\frac{T_{sum}^{I-DDQN} - T_{sum}^{I-DDQN-RS}}{T_{sum}^{I-DDQN}}$	15.00	16.75	17.97
	$\frac{T_{sum}^{P-DDQN} - T_{sum}^{deepMARL-RS}}{T_{sum}^{P-DDQN}}$	32.86	29.81	32.10

# Experiment Results

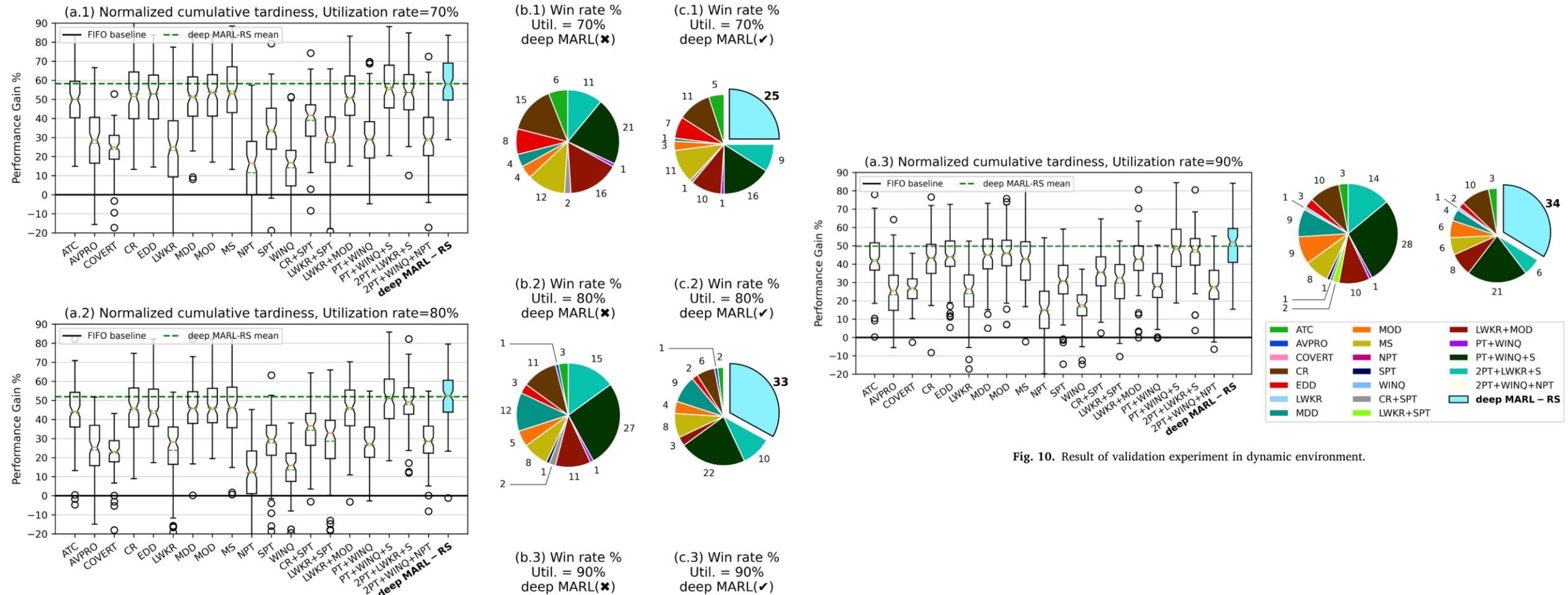


Fig. 10. Result of validation experiment in dynamic environment.

# Experiment Results

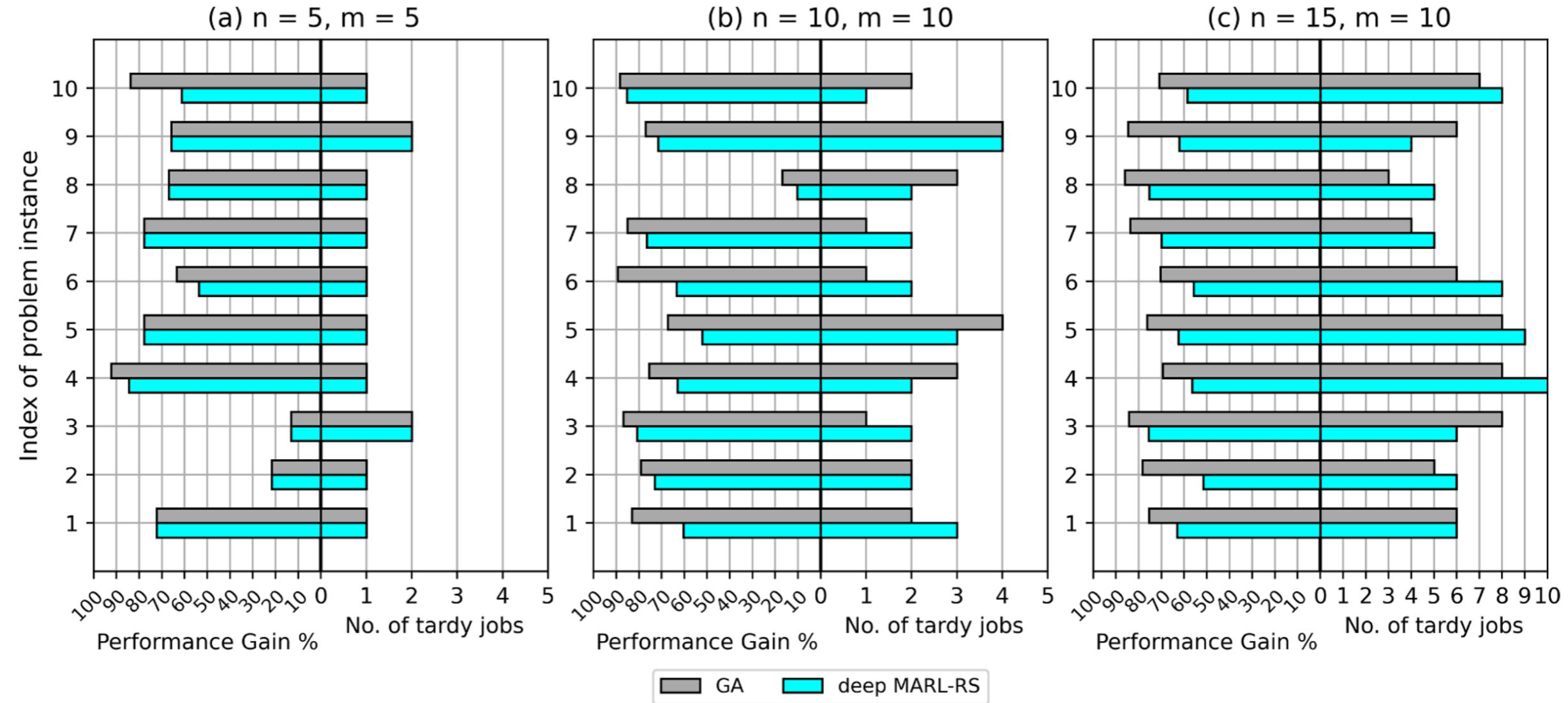


Fig. 11. Performance on static problem instances.

# Experiment Results

**Table 7**

Average CPU time for scheduling decision.

Problem Size ( $n \times m$ )	Category	Priority rule (FIFO)	Deep MARL-RS	Genetic Algorithm
$5 \times 5$	Single decision	$9.46 \times 10^{-7}s$	$2.11 \times 10^{-4}s$	$3.26s$
	All decisions	$2.37 \times 10^{-5}s$	$5.28 \times 10^{-3}s$	
$10 \times 10$	Single decision	$9.87 \times 10^{-7}s$	$2.12 \times 10^{-4}s$	$31.43s$
	All decisions	$9.87 \times 10^{-5}s$	$2.12 \times 10^{-2}s$	
$15 \times 10$	Single decision	$9.96 \times 10^{-7}s$	$2.14 \times 10^{-4}s$	$62.80s$
	All decisions	$1.49 \times 10^{-4}s$	$3.21 \times 10^{-2}s$	



## Contents

- 01**  Problem Description
- 02**  Method Design
- 03**  Experiment Results
- 04**  Conclusion

# Future Work

## Contributions

1. Proposed a deep MARL approach for dynamic job scheduling, enabling low-latency and high-quality decision-making.
2. Introduced novel state and action representations to overcome fixed-size constraints in traditional scheduling.
3. Developed knowledge-based reward shaping to enhance agent learning.
4. Validated the approach through ablation studies, outperforming existing methods and priority rules.

## Future Work

1. Establish benchmark problems for dynamic scheduling.
2. Improve model performance with deeper networks.
3. Develop automated industrial data processing.
4. Optimize RL training problem design.

# Thank you for listening