Paper Review:

Solving flexible job shop scheduling problems via deep reinforcement learning

인공지능 운영최적화 연구실
이태희
2025.03.07

Summary

- **State Representation 학습과 Policy 학습을 위한 새로운 DRL 프레임워크를 제안**
  - 복잡한 Graph Neural Network로 Node Message Aggregation을 사용하는 다른 연구와 달리,
    가벼운 다층 퍼셉트론 (MLP)를 State Embedding에 사용하여 알고리즘의 계산적 복잡도를 어느 정도$_{\text{to some extent}}$ 낮춤

- **새로운 State Representation 설계**
  - 후보 Actions의 Features를 직접적으로 투영하여
    Agent가 State 정보를 더욱 효율적으로 파악하고 더 나은 의사결정을 하도록 도움

- **새로운 Action Space 정의**
  - 두 개의 부분 문제를 풀듯 두 가지 Action Space(할당할 작업, 할당받을 설비)에서 의사결정하는 다른 연구와 달리,
    한 가지 Action Space에서 의사결정하여 하나의 Policy만 학습하면 됨

Flexible Job Shop Problem

목적함수: 총 소요 시간 $C_{\max}$ 최소화

(1) Parameters:

n: total number of jobs;
m: total number of machines;
i, h: index of jobs, $i, h = 1, \ldots, n$;
n_i: the number of operations of job i;
n_{max}: the maximum number of operations for all jobs;
j, g: index of operations, $j, g = 1, \ldots, n_i$;
k, e: index of machines, $k, e = 1, \ldots, m$;
O_{ij}: the jth operation of job i;
$\boldsymbol{\Omega_{ij}}$: the optional machine set of O_{ij};
O_{ijk}: the jth operation of job i is processed on machine M_k;
p_{ijk}: the processing time of O_{ij} on machine M_k;
$\bar{p}_{ij}$: the average processing time of O_{ij} on $\boldsymbol{\Omega_{ij}}$;
s_{ij}: the start time of O_{ij};
c_{ij}: the completion time of O_{ij};
L: a sufficiently large positive number;
C_i: the completion time of job i;
C_{max}: makespan.

(2) Decision variables:

$$x_{ijk} = \begin{cases} 1, & \text{if } O_{ij} \text{ selects processing machine } M_k; \\ 0, & \text{otherwise;} \end{cases}$$

$$y_{ijhgk} = \begin{cases} 1, & \text{if } O_{ijk} \text{ is a predecessor of } O_{hgk}; \\ 0, & \text{otherwise;} \end{cases}$$

Minimize:

$$C_{\max} = \max_{1 \le i \le n} \{C_i\} \tag{1}$$

Subject to:

$$s_{ij} + x_{ijk} \times p_{ijk} \le c_{ij} \tag{2}$$

$$c_{ij} \le s_{i(j+1)} \tag{3}$$

$$s_{ij} + p_{ijk} \le s_{hg} + L\left(1 - y_{ijhgk}\right) \tag{4}$$

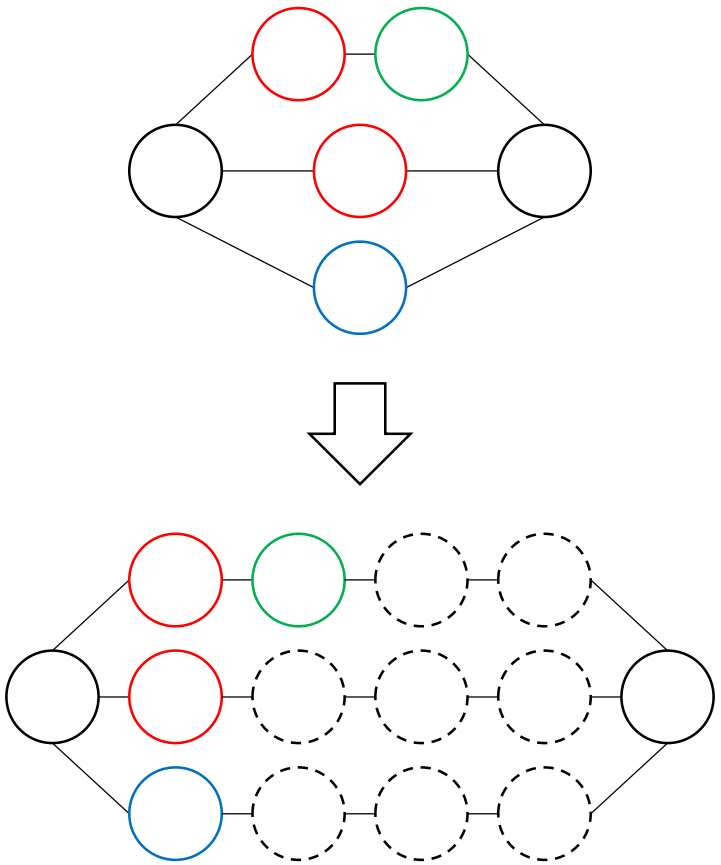
$$c_{ij} \le s_{i(j+1)} + L\left(1 - y_{hgi(j+1)k}\right) \tag{5}$$

$$\sum_{k=1}^{\left|\boldsymbol{\Omega_{ij}}\right|} x_{ijk} = 1 \tag{6}$$

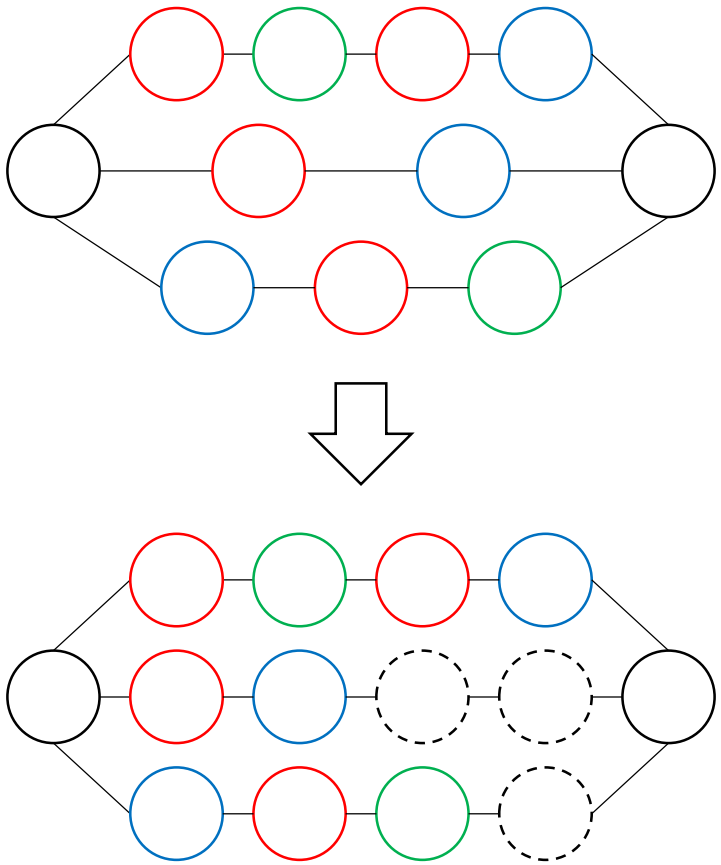
Flexible Job Shop Problem

m = 설비 수, n_{max} = $\max_j$(작업 j의 공정 수)

$m \geq n_{max}$

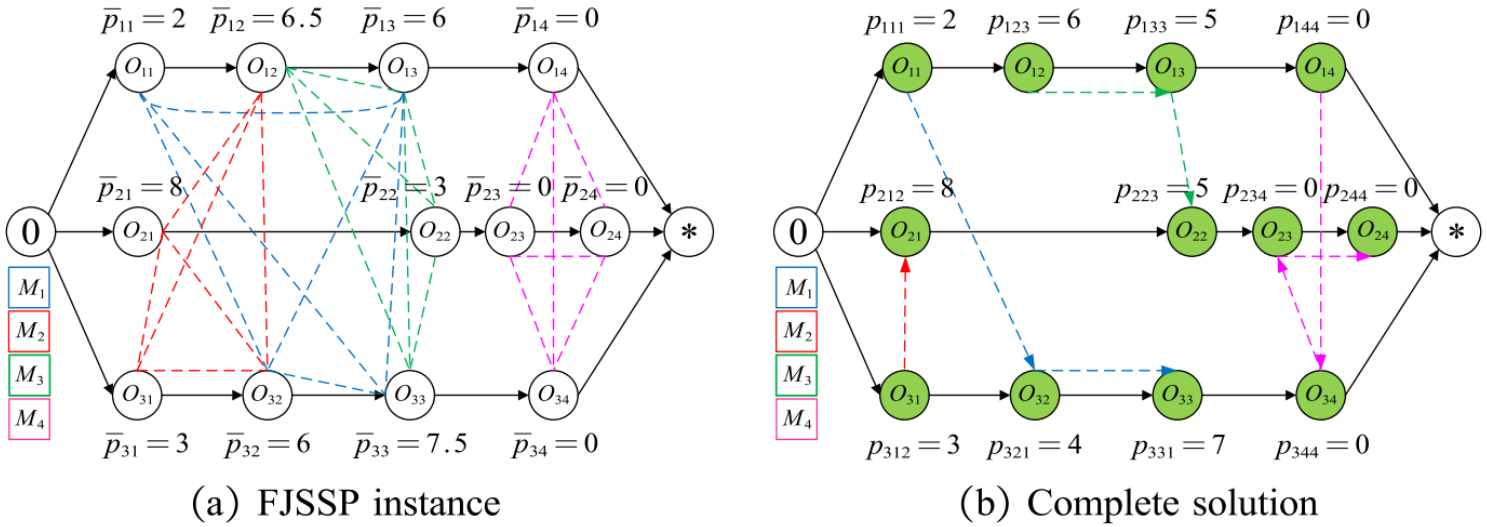


$m < n_{max}$



Flexible Job Shop Problem

목적함수: 총 소요 시간 C_{max} 최소화



(a) FJSSP instance (b) Complete solution

Fig. 1. Disjunctive graph representation of FJSSP after preprocessing.

Table 2
A 3 × 3 FJSSP instance.

Jobs	Operations	Optional machines and processing time		
		M_1	M_2	M_3
J_1	O_{11}	2	–	–
	O_{12}	–	7	6
	O_{13}	7	–	5
J_2	O_{21}	–	8	–
	O_{22}	–	–	3
J_3	O_{31}	–	3	–
	O_{32}	4	8	–
	O_{33}	7	–	8

Proposing Framework

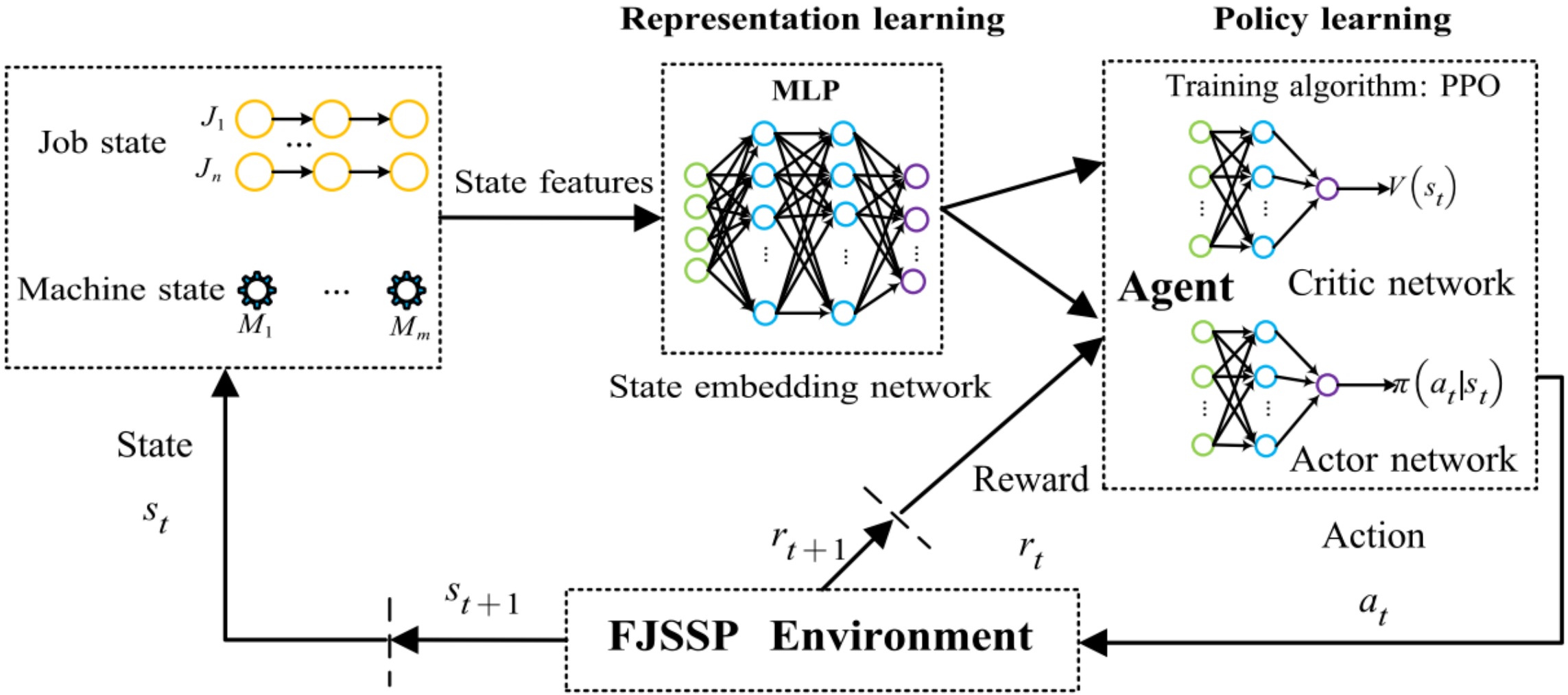


Fig. 2. The overall framework of the proposed method.

State

- $p_{ijk}(s_t)$: t 시점에서 작업 i의 j번째 공정(O_{ij})이 설비 k에서 소요되는 공정 시간, 설비 k에 할당될 수 없으면 -1
- $C_{LB}(O_{ij}, s_t)$: t 시점에서 O_{ij}의 기대 완료 시각
- $R_{LB}(O_{ij}, s_t)$: t 시점에서 O_{ij}이 할당될 경우 기대 잔여 공정 시간
- $S^E(O_{ij}, M_k, s_t)$: t 시점에서 O_{ij}이 설비 k에서 공정을 시작할 수 있는 가장 이른 시각

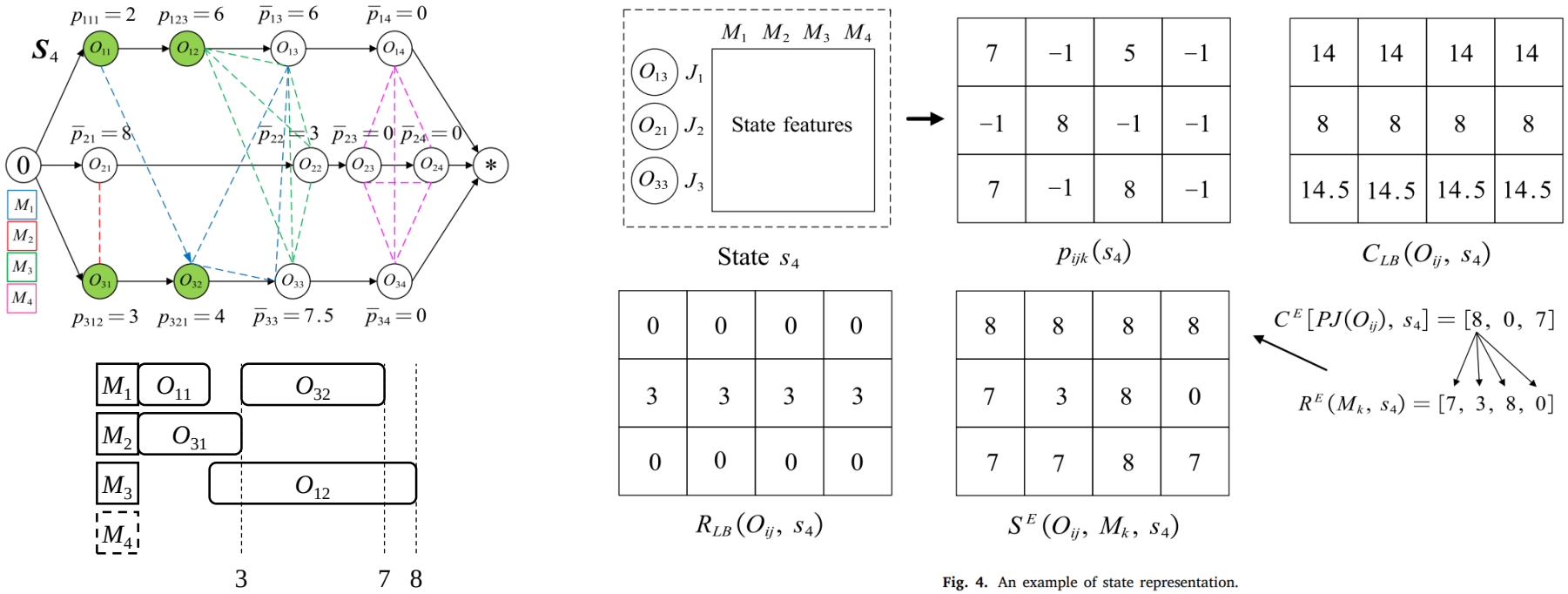


Fig. 4. An example of state representation.

State

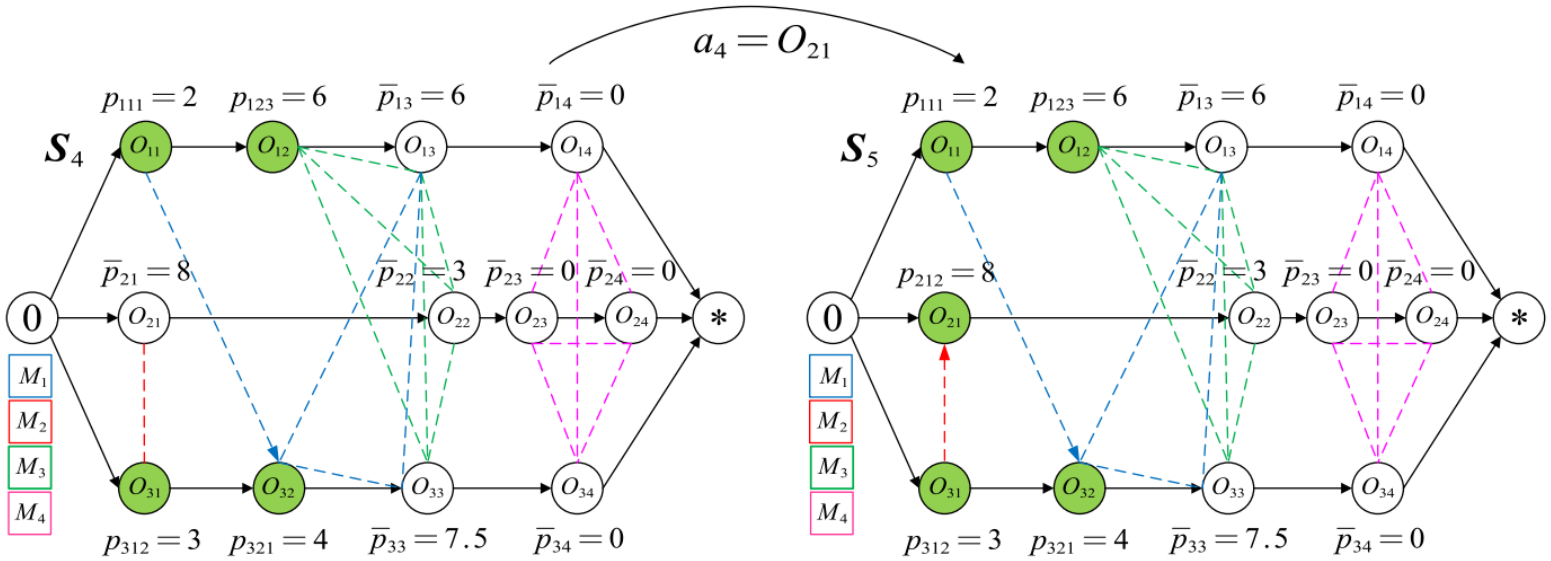
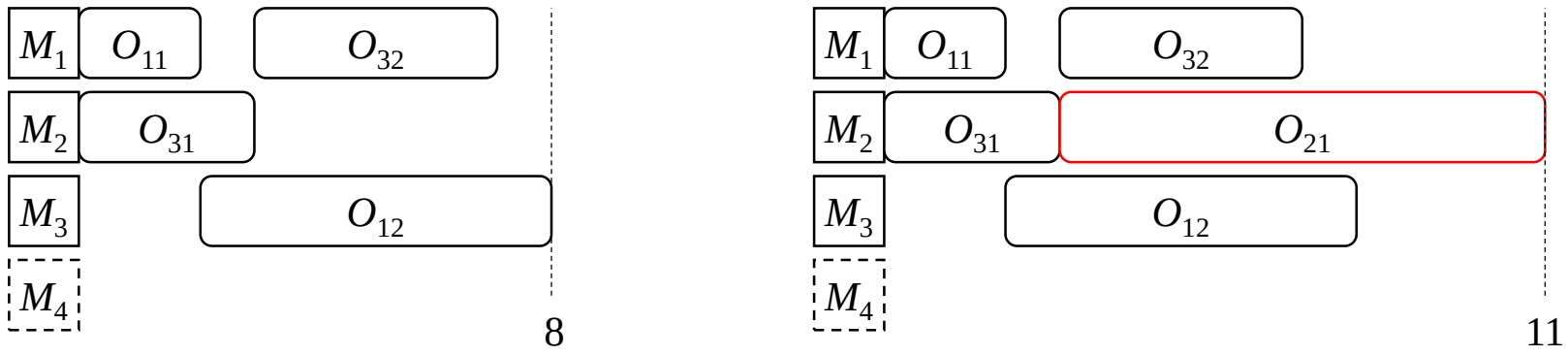
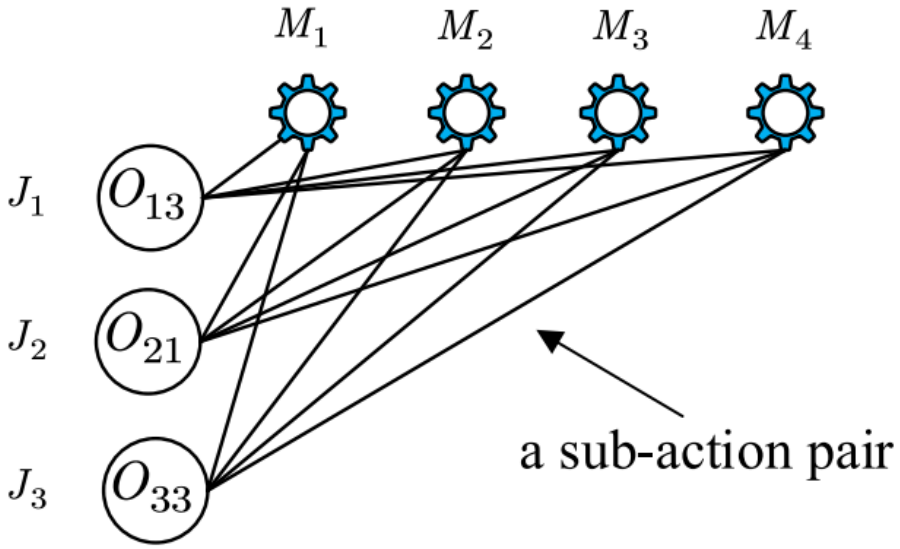


Fig. 3. An example of state transition.



Action & Reward

Invalid Action은 Masking 처리함



The set of candidate actions in state s_4:

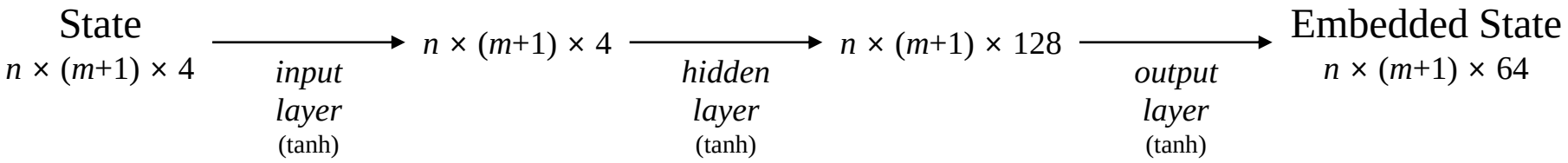
$$\begin{Bmatrix} O_{13}-M_1,\ O_{13}-M_2,\ O_{13}-M_3,\ O_{13}-M_4, \\ O_{21}-M_1,\ O_{21}-M_2,\ O_{21}-M_3,\ O_{21}-M_4, \\ O_{33}-M_1,\ O_{33}-M_2,\ O_{33}-M_3,\ O_{33}-M_4 \end{Bmatrix}$$

Fig. 5. An example of action space.

→ Reward $r_t = C_{\max}(s_t) - C_{\max}(s_{t+1})$
where $C_{\max}(s_t) = \max_i\{C_{LB}(O_{i(n\_i)}, s_t)\}$

Agent

State Embedding Network



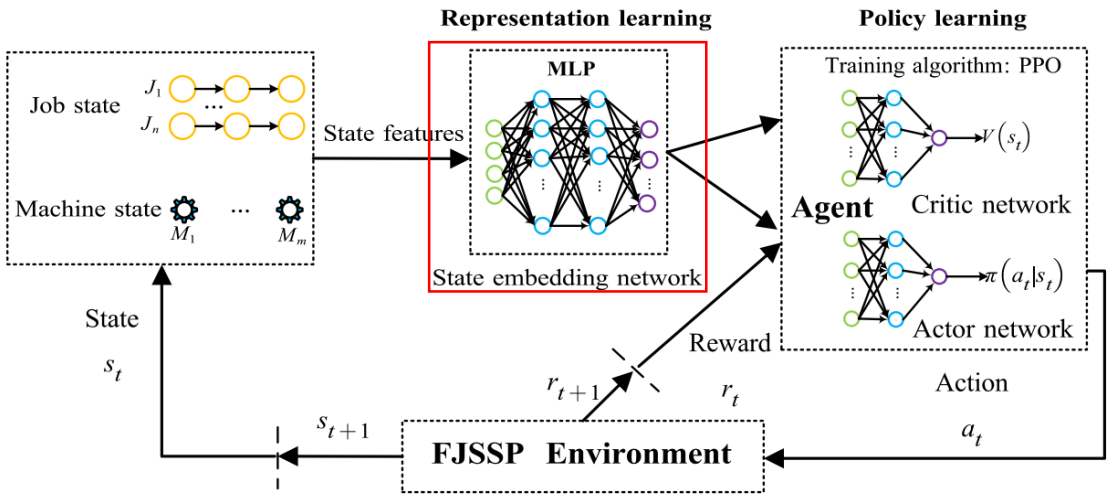
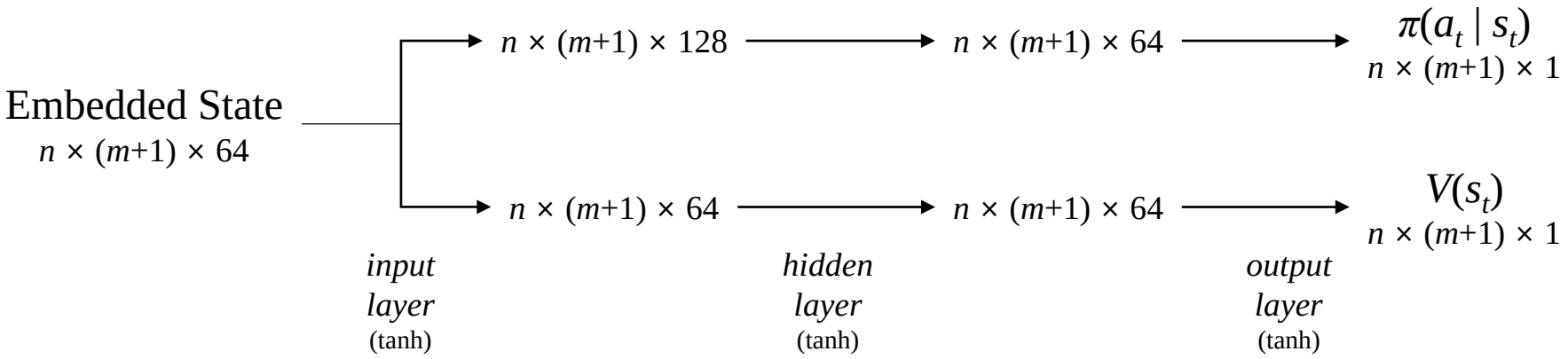


Fig. 2. The overall framework of the proposed method.

Agent

Actor & Critic Network



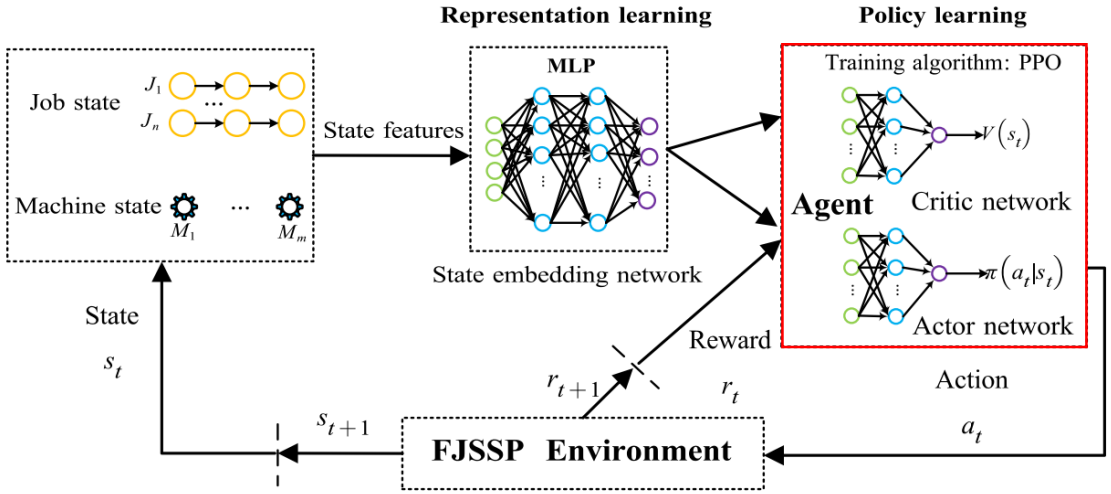


Fig. 2. The overall framework of the proposed method.

강화학습 방법론

PPO Algorithm (2017)

Algorithm 1: Pseudocode for updating the network model parameters via PPO algorithm.

Input: Actor network π_θ with parameter β, behavior actor network $\pi_{\theta_{old}}$ with $\theta_{old} = \theta$, and critic network v_ϕ with ϕ. N, K, V.

Output: θ,ϕ

```
for n = 1, 2, ..., N do
    /* n means an episode.                                   */
    for t = 1, 2, ..., T do
        while s_t is not terminal do
            Collect the experience ⟨s_t, a_t, r_t, s_{t+1}⟩.
        end
        if s_t is terminal then
            break
        end
    endfor
    for k = 1, 2, ..., K do
        The experience of an episode is collected to calculate
         the loss L_t(θ, φ), and the parameters θ and φ are
         updated.
    endfor
    if n%V = 0 then
        A performance verification of the policy model.
    end
endfor
```

Table 3
The hyperparameter settings of the PPO algorithm.

Hyperparameters	Values
Number of episodes N	300, 400, 500, 600
Number of policy updates K	1
Number of interval episodes per validation V	3, 4, 5, 6
Learning rate lr	1×10^{-4}
Policy loss coefficient c_p	2
Entropy loss coefficient c_e	0.01
Critic loss coefficient c_v	1
Discount factor γ	1
Clipping coefficient ϵ	0.2

Experiment

- 실험 환경
  - Intel® CORE™ i7-9750H, NVIDIA GeForce GTX 1660 Ti (VRAM 6144MB)
  - Python 3.7.0, PyTorch 1.6.0
- 훈련 데이터: 문제 크기는 작업 10, 설비 5 고정(n_{max}에 대한 정보 없음)Size = Batch Size = 한 Episode에 풀이하는 FJSP 문제 수 인듯?

Table 4
Details of the dataset.

Datasets	Source	Seed	Range	Size
Training dataset	Synthetic	200	$1 \leqslant p \leqslant 99$	300, 400, 500, 600
Validation dataset	Synthetic	100	$1 \leqslant p \leqslant 99$	100

- 시험 데이터
  - Barnes(Barnes and Chambers, 1996), Brandimarte(Brandimarte, 1993), Dauzere(Dauzère-Pérès and Paulli, 1997), Hurink(Hurink, Jurisch and Thole, 1994)

Table 4
Details of the dataset.

Datasets	Source	Seed	Range	Size
Test dataset 1	Barnes	–	$1 \leqslant p \leqslant 99$	21
Test dataset 2	Brandimarte	–	$1 \leqslant p \leqslant 20$	10
Test dataset 3	Dauzere	–	$1 \leqslant p \leqslant 100$	18
Test dataset 4	Hurink(rdata)	–	$1 \leqslant p \leqslant 99$	40
	Hurink(edata)		$1 \leqslant p \leqslant 99$	40
	Hurink(vdata)		$1 \leqslant p \leqslant 99$	40

Experiment

- 비교군
  - 우선순위 규칙 3가지 : FIFO + EET, MWKR + EET, MOPNR + EET
    - FIFO = 가장 빨리 할당될 수 있는 O_{ij} 선택
    - MWKR = 본 공정 포함 평균 잔여 공정 시간이 가장 긴 O_{ij} 선택
    - MOPNR = 본 공정 포함 잔여 공정 수가 가장 많은 O_{ij} 선택
    - EET = O_{ij}를 할당받을 수 있는 설비 중 가장 이른 시각에 시작할 수 있는 M_k 선택
  - 메타휴리스틱 기반 방법론 4가지
    - Improved Jaya Algorithm;IJA (Caldeira and Gnanavelbabu, 2019)
    - Regular GA:RegGA (Rooyani and Defersha, 2019)
    - Two-Stage Genetic Algorithm;2SGA (Rooyani and Defersha, 2019)
    - Self-learning Genetic Algorithm;SLGA (Chen et al., 2020)

Experiment

- 비교군
  - SOTA 방법론 2가지 :
    - Lei et al., 2022

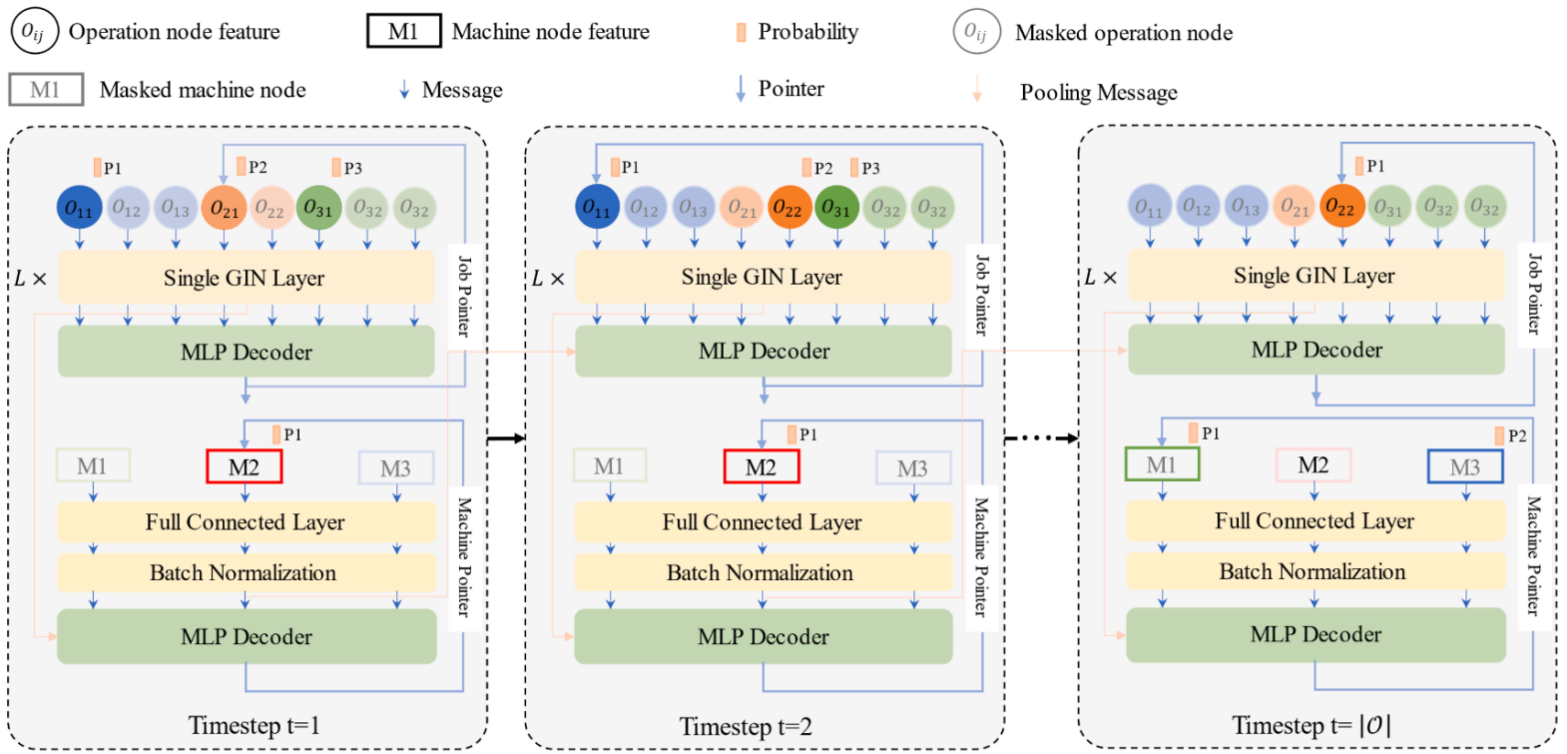


Fig. 3. The MPGN architecture for the FJSP.

Experiment

- 비교군
  - SOTA 방법론 2가지 :
    - Song et al., 2022

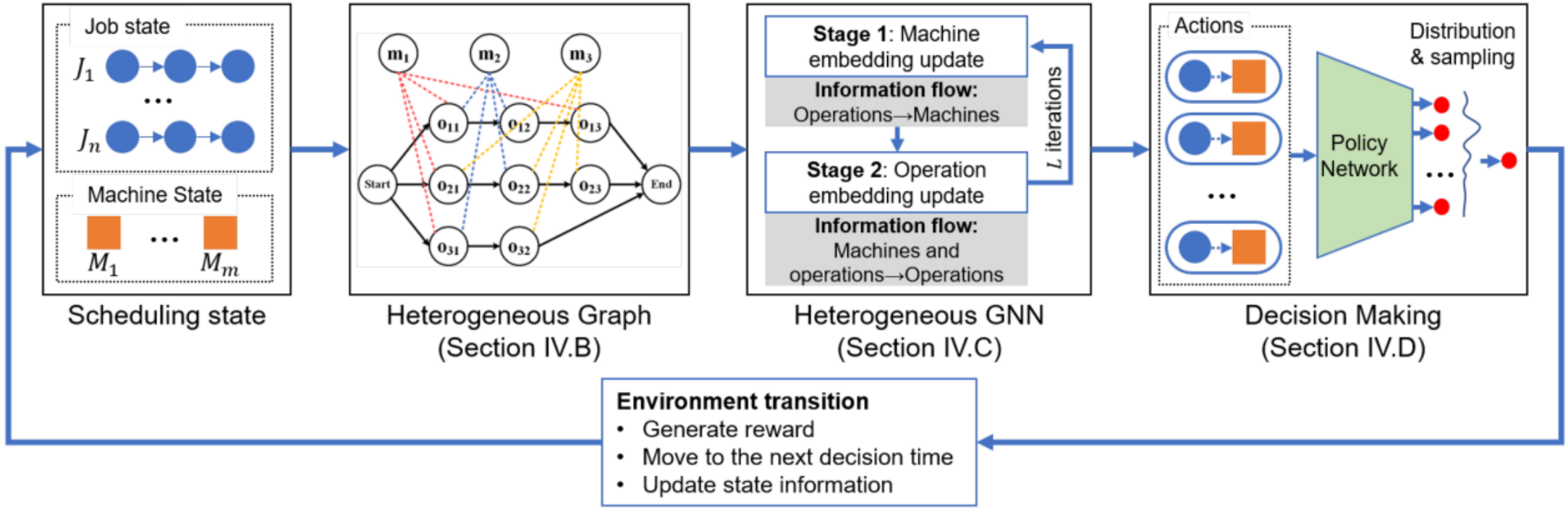
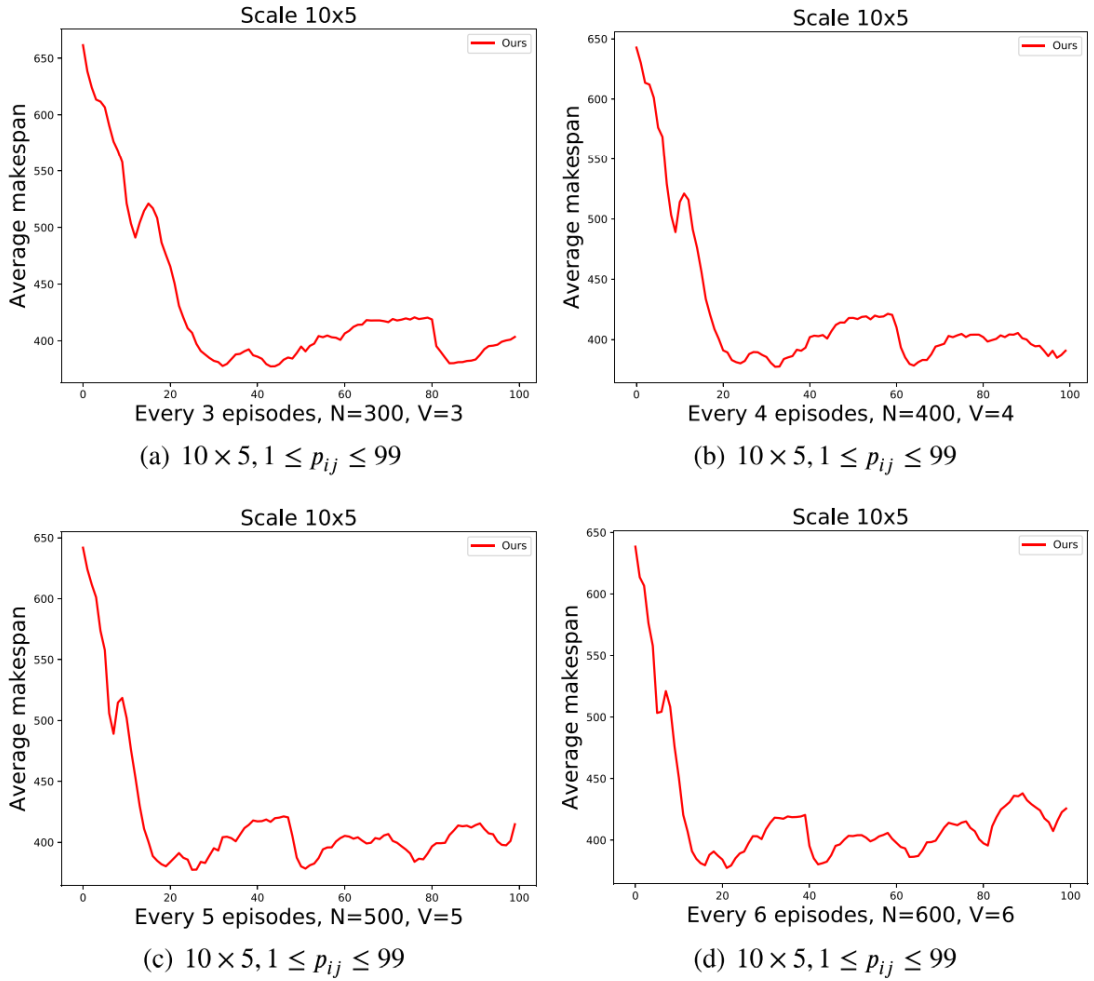


Fig. 3. Workflow of the proposed method.

Experiment

학습 곡선



(a) $10 \times 5, 1 \leq p_{ij} \leq 99$ (b) $10 \times 5, 1 \leq p_{ij} \leq 99$

(c) $10 \times 5, 1 \leq p_{ij} \leq 99$ (d) $10 \times 5, 1 \leq p_{ij} \leq 99$

Fig. 6. The training curves of the policy models under different sizes of training datasets.

Experiment

학습한 Size 별 시험 데이터에서의 성능

Table 5
Average gap and average time of policy models on public datasets under different sizes of training datasets.

10 × 5	N = 300, V = 3		N = 400, V = 4		N = 500, V = 5		N = 600, V = 6	
	Gap	Time (s)	Gap	Time (s)	Gap	Time (s)	Gap	Time (s)
Barnes	**13.83%**	0.391	17.88%	0.392	15.44%	0.402	16.39%	0.397
Brandimarte	17.00%	0.406	**13.24%**	0.406	17.84%	0.404	16.47%	0.403
Dauzere	11.60%	0.804	14.91%	0.798	11.16%	0.809	**11.02%**	0.808
Hurink(rdata)	13.41%	0.274	13.98%	0.273	**12.09%**	0.275	12.66%	0.275
Hurink(edata)	15.69%	0.274	15.95%	0.274	**15.54%**	0.271	16.57%	0.273
Hurink(vdata)	6.50%	0.278	6.30%	0.272	**5.37%**	0.272	6.51%	0.276

Experiment

비교군과의 성능 비교

Table 6
Comparison of average gap and average time on public datasets with various baselines.

Methods	Barnes		Brandimarte		Dauzere		Hurink(rdata)		Hurink(edata)		Hurink(vdata)	
	Gap	Time (s)	Gap	Time (s)	Gap	Time (s)	Gap	Time (s)	Gap	Time (s)	Gap	Time (s)
IJA	2.40%	21.03	8.50%	15.43	6.10%	180.8	3.90%	19.72	4.60%	15.24	2.70%	17.85
RegGA	–	–	8.39%	280.10	–	–	–	–	–	–	3.20%	191.40
2SGA	–	–	3.17%	57.60	–	–	–	–	–	–	0.39%	51.43
SLGA	–	–	6.21%	283.28	–	–	–	–	–	–	–	–
FIFO+EET	27.91%	0.019	28.98%	0.017	15.00%	0.036	17.38%	0.018	19.89%	0.016	7.14%	0.019
MWKR+EET	54.71%	0.018	39.50%	0.019	24.69%	0.036	26.60%	0.018	44.68%	0.018	8.96%	0.020
MOPNR+EET	50.66%	0.017	43.39%	0.018	32.17%	0.038	26.47%	0.019	43.67%	0.018	13.14%	0.019
Song et al. (2022)	17.88%	1.516	30.04%	1.305	**8.88%**	2.716	**11.02%**	1.421	16.66%	1.424	**4.41%**	1.405
Lei et al. (2022)	28.96%	2.048	13.59%	2.054	15.68%	3.917	16.51%	1.591	23.01%	1.558	6.96%	1.544
Ours	**13.83%**	**0.391**	**13.24%**	**0.406**	11.02%	**0.808**	12.09%	**0.275**	**15.54%**	**0.271**	5.37%	**0.272**

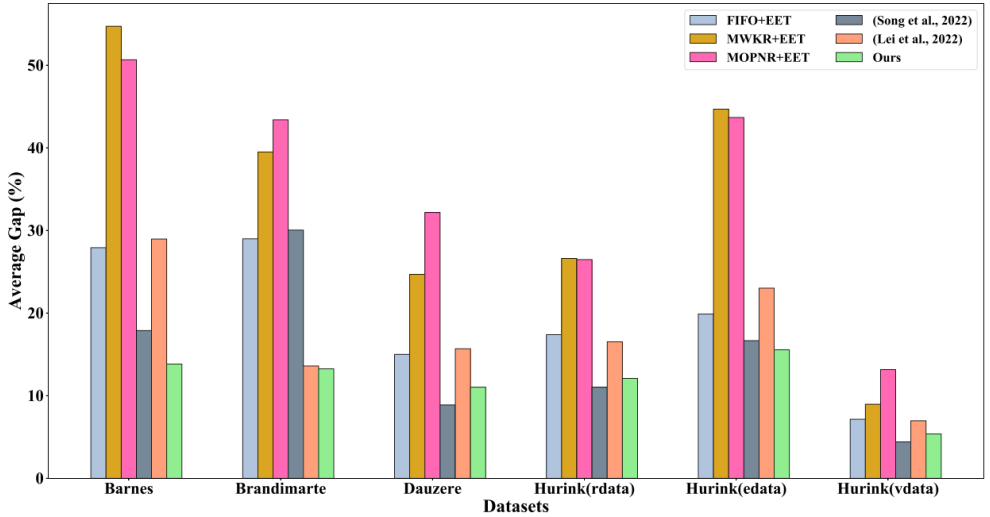


Fig. 7. Comparison chart with other learning-based methods and composite PDRs.

Conclusions of Review

- Graph 기반 Network, Attention 기반 Network 등 복잡한 Network를 쓰지 않고도 좋은 성능을 보인 것이 흥미로움
- 코드는 요청 시 제공한다고 하여 직접 확인하지 못함
- 메타휴리스틱 방법론의 결과가 나오지 않은 기준을 명시하지 않고,
  반복 탐색으로 해를 개선하는 방식이라고만 소개한 것이 아쉬움

- 논문 외적인 내용) 최근에는 State를 가공하지 않고 **있는 그대로 Agent에게 입력**시키는 방안이 연구되고 있다고 함
- 위 내용과 본 논문의 접근법을 함께 접목하는 연구 방향을 고려하게 됨