A novel method-based reinforcement learning with deep temporal difference network for flexible double shop scheduling problem

24.07.04

임제원

I Problem Definition

❖ Flexible Double Shop Scheduling Problem(FDSSP)

- FDSSP를 해결하기 위해 DTDN 기반 강화학습 모델
- 유압 실린더 생산에 있어, 작업장과 조립장을 동시에 고려하는 FDSSP 문제
- 각 실린더는 여러 부품으로 조립되며, 제품은 주문에서 시작하여 조립으로 완성됨
- 작업장에는 세 대의 기계(Fine Turning, CNC Milling, Electric Spark)가 설치됨
- 조립장에는 두 대의 조립로봇(A1, A2)가 설치됨
- 조립 작업은 모든 작업이 완료된 후에 시작함
- 조립 작업은 짧고 고정적일 수 있으므로 조립 작업의 계획된 시작 시간은 주문의 배송날짜로부터 추정할 수 있음
- 논문에서는 각 작업의 완료 시간이 조립의 계획된 시작 시간에 최대한 근접하도록 함
- N개의 작업이 m개의 기계로 구성된 작업장에서 처리됨
- 각 작업 j는 O개의 세부 작업 k로 구성되어 있고, 지정된 경로에 따라 처리됨
- 기계마다 k의 처리시간에 큰 차이가 있음

Problem Definition

❖ Flexible Double Shop Scheduling Problem(FDSSP)

<조립 제약 수준 정의>

- 조립 후의 작업을 제약 작업(Constrained Job), 조립 전의 작업을 전면 작업(Front Job)이라 함
- 조립 제약 관계에 따라 Tight한 전면 제약이 없는 모든 작업의 제약 수준은 1로 설정함
- 제약 수준이 정의되지 않은 작업들로 구성된 집합을 U로 표현함
- U집합에서 Tight한 전면 작업 J_k를 꺼내어 J_{set} 을 만듦
- J_{set}의 모든 제약 수준이 결정되었는지 확인 후, J_k의 제약 수준을 $\max(L(J_{set})+1)$로 표현함
- J_{set}의 모든 제약 수준이 결정 되지 않을 경우, J_k를 다시 U집합에 넣어 제약 수준이 결정될 때까지 반복함
- 작업장에서 조립장으로 이동하는 시간은 생략함

I Problem Definition

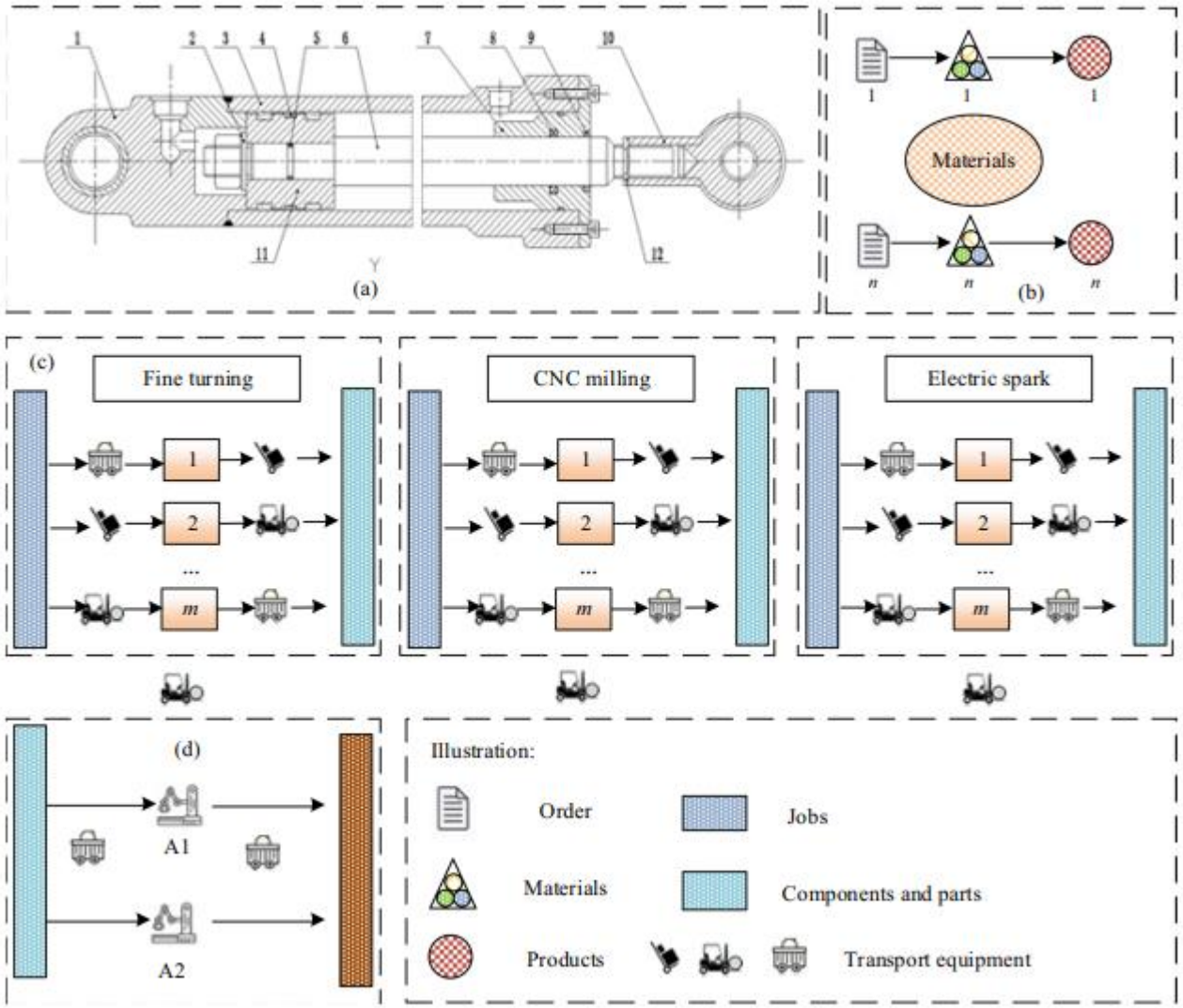


Figure 1. Integrated production of hydraulic cylinder: (**a**) structure charts; (**b**) production layout; (**c**) job shop; (**d**) assembly shop.

I Problem Definition

❖ Flexible Double Shop Scheduling Problem(FDSSP)

- 목적함수

$$\min(C_{\max}) = \min\left(\max_{1\le j\le n}(C_j)\right) = \min\left(\sum_{1}^{n} x_{jkm}t_{jkm}\right).$$

- 파라메터 설명

Sets	Description	Indices	Description
J	Set of the job	j	Indices of the job, $j \in J$
M	Set of the machine	m	Indices of the machine, $m \in M$
O	Set of the operation	k	Indices of the operation, $k \in O$
Job-related parameters	**Description**		
at	Arrival time of products		
d	Delivery time of products		
apt	Assembly time of the products		
o_{jf}	The first operation of the process path		
o_{jl}	The last operation of the process path		
t_{jkm}	Processing time of machine m used in the operation k_j		
S_{jkm}	Start time of machine m used in the operation k_j		
C_{jkm}	Processing time of machine m used in the operation k_j		
C_j	Processing time of the last operation for the job j		
$C_{\max}$	Makespan of the job j(maximum processing time of the last operation for the job j)		
L	Extreme value		

- 의사결정 변수

$$x_{jkm} = \begin{cases} 1, & if\ operation\ k\ of\ job\ j\ is\ processed\ on\ machine\ m \\ 0, & otherwise \end{cases}.$$

Problem Definition

❖ Flexible Double Shop Scheduling Problem(FDSSP)

<Definition of State-State>

- 상태 공간은 생산 환경의 주요 특징을 반영하여, 적절한 규칙을 선택하는 데 기반이 됨
- 생산 환경이 계속 변화하기 때문에 작업장에서 수십개의 상태 특성으로 설명됨
- 용이한 표현을 위해 프로세스의 처리 상태는 $P_{jk} = \{0,1\}$로 기록됨 처리되면 1, 아니면 0
- 기계에서 처리해야 할 작업은 처리시간의 내림차순으로 배열됨
- 프로세스 순서는 $list(m) = \{J_{m1}, J_{m2}, \ldots, J_{mv_m}\}$으로 표시되며 v_m은 기계 m에서 처리해야 할 작업 수

No	State features	Description
1	$x_{m,1} = \sum_{j=1,k=1,P_{j,k}=0}^{O_{jp}} T_{j,k}$	Total time of operations to be processed on machine m
2	$x_{m,2} = \sum_{j=1,k=1,P_{j,k}=0}^{O_{jp}} T_{j,k}$	Total time of operations processed on machine m
3	$x_{m,3} = J_{m,1}^{P_{j,k=0}}$	Time of the first operation in the sequence $List(m)$ to be processed on the machine m
4	$x_{m,4} = J_{m,2}^{P_{j,k=1}}$	Time of the second operation in the sequence $List(m)$ to be processed on machine m
5	$x_{m,5} = W_{j,1}^{P_{j,k=1}}$	Among all future operations, the time of the first operation in the sequence $List(m)$ to be processed on the machine m
6	$x_{m,6} = W_{j,2}^{P_{j,k=0}}$	Among all future operations, the time of the second operation in the sequence $List(m)$ to be processed on the machine m
7	$x_{m,7} = \sum_{i=1}^{n} (1 - P_{j,k})$	Total number of operations for all future processes on the machine m
8	$x_{m,8} = \sum_{i=1}^{n} (1 - P_{j,k}) T_{j,k}$	Total time for all future operations on machine m
9	$x_{m,9} = \begin{cases} 0, & if\ machine\ is\ idle \\ 1, & if\ machine\ is\ processing\ a\ job \end{cases}$	Machine states
10	$x_{m,10} = \sum_{J_k=1}^{n} (L(J_k))$	Total number of all jobs assembly constraint levels on the machine m

Problem Definition

❖ Flexible Double Shop Scheduling Problem(FDSSP)

<Definition of Action Space>

- 8가지의 Action이 후보 집합으로 선택되어 있음

No	SCH	Description
1	First come first served (FIFO)	Processing in sequence according to the arrival order of the job
2	Shortest processing time (SPT)	Sorted by the total processing time of the job on all machines from shortest to longest
3	Shortest remaining processing time (SRPT)	Sorted by the remaining processing time of the job on all machines from shortest to longest
4	Most operations remaining (MOR)	Sorted by the number of the remaining operations on all machines from shortest to longest
5	Earliest due date (EDD)	Sorted by the due date from shortest to longest
6	Apparent tardiness cost (ATC)	Sorted by the tardiness cost from shortest to longest
7	Total least operations remaining (TLOPR)	Sorted by assembly-related constraints of the job from shortest to longest
8	Select no job (SNJ)	Machines don't select processing each job

Problem Definition

❖ Flexible Double Shop Scheduling Problem(FDSSP)

<Definition of Rewards>

- 각 상태 전환에 따라 즉각적으로 Reward 부여
- 누적 Rewards를 목표함수의 최적 값에 반영함
- 참고문헌에 의해 Makespan과 기계 사용률에는 직접적인 관계가 있음(Makespan 최소화=기계사용률 최대화)

$$U_{ave}(t) = \frac{1}{m}\sum_{k=1}^{m} U_k(t) = \frac{1}{m}\frac{\sum_{1}^{n}\sum_{k=1}^{O_i(t)} t_{jkm}x_{jkm}}{C_{\max}(t)}.$$

- $U_{ave}(t)$는 t 시점에서의 평균 기계 활용률을 의미함
- $C_{max}(t)$는t 시점에서 기계 m에 할당된 마지막 작업의 완료시간을 의미함
- O_t는 작업 i에 대한 현재 할당된 작업 수를 의미함
- $U_k(t)$는 t 시점에서 기계 m의 활용률을 의미함
- $r_k = U_k(t) - U_{k-1}(t)$

$$R = \sum_{k=1}^{K} r_k = \sum_{k=1}^{K} \left(U_k(t) - U_{k-1}(t)\right) = U_k(t).$$

❖ FDSSP를 위해 제안된 방법

1. Markov Decision Process(MDP)

- 다음 state는 현재 state와 만 관련된다는 속성
- E = {S, A(s), P, R, γ } - S: 상태공간 집합, A : 행동집합, P : 상태전이 함수, P : 보상 함수 γ : 할인율
- Agent가 전략을 따르며 얻을 수 있는 예상 누적 보상 함수를 최대화 하는 실험을 통해 최적을 전략을 찾는 것
- 보상 함수는 가치 함수를 추가하여 결정되는데, Bellman의 기대 방정식을 최적의 전략으로 업데이트 하면 다음과 같은 최적의 방정식이 산출됨 $V^*(s) = \max_\pi V_\pi(s) = \max_\pi R_s^a + \gamma \sum P_{ss'}^a V^*(s').$

2. Temporal Difference Algorithm(TD)

- TD는 Monte Carlo + Dynamic planning method로 고전적인 Bellman 공식을 사용함

1: **For** episode =1: M do
2: $\forall s \in S$, Initialize states value $V(s)$;
3: Set the initial state s_0 to the current state s_t;
4: Select actions according to s_t, state value $V(s)$ and strategy π;
5: Perform the actions at, determine the next decision moment state s_{t+1}, and calculate the reward r_{t+1};
6: Update $V(s_t)$ according to Eq. (8);
7: If s_{t+1} is not the terminated state: 8: $t = t + 1$, skip to step 3; 9: **End if** 10: **End for**

$$V(s_t) = V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)].$$

❖ FDSSP를 위해 제안된 방법

3. Deep Learning Model

(Deep Neural Network)

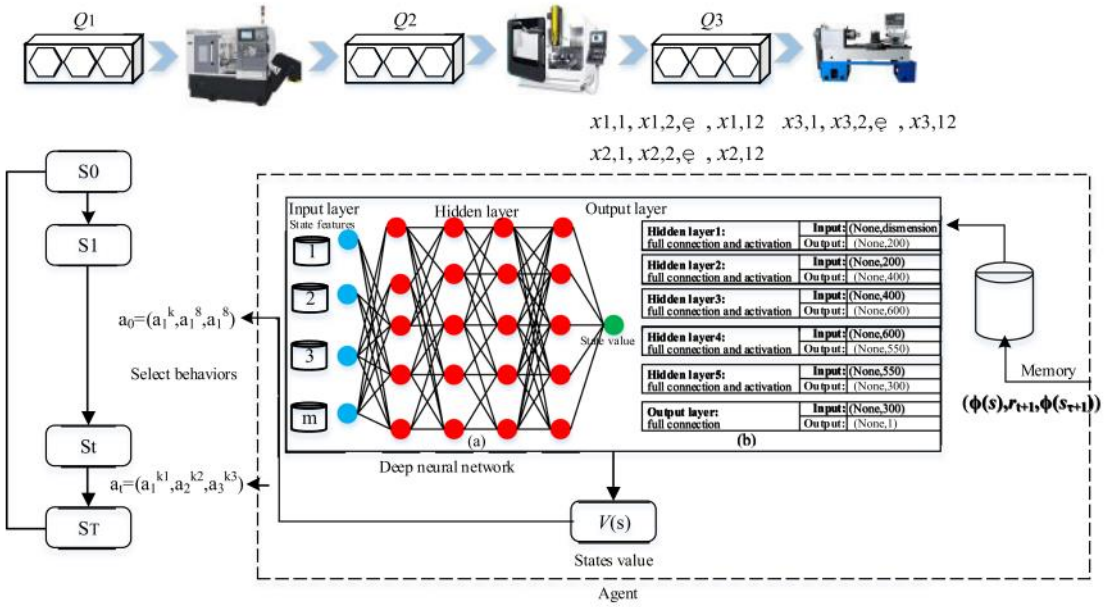
- 인공 신경망을 기반으로 하는 표현 학습의 한 유형, 심층 신경망 구조가 더 큰 용량과 공간을 가지고 있어 학습하기 쉬움

(Activation Function)

- Activation Function은 뉴런이 학습하고 적응할 수 있는 기능을 제공함
- 선형 모델의 결함을 해결하기 위해 신경망에 비선형 요서를 통합함
- Activation Function을 사용하지 않으면 출력은 선형조합이 됨
- 이 논문에서는 Relu를 사용함

(Optimization Function)

- 신경망 훈련의 핵심 문제중 하나로 솔루션 프로세스의 속도를 높여줌
- 이 논문에서는 7개의 Connection Layer를 사용(1 input layer, 5 implicit layers, 1 output layer)



❖ FDSSP를 위해 제안된 방법

4. Exploration and exploitation

- 에이전트가 Greedy 전략을 사용함

$$P(s,a) = \begin{cases} 1-\varepsilon+\frac{\varepsilon}{|A(s)|}, & a = a^*(s) \\ \frac{\varepsilon}{|A(s)|}, & a \neq a^*(s) \end{cases} . \qquad a^*(s) = \arg\max \left[r^a_{ss'} + \gamma V(s') \right]$$

- A(s)는 상태의 후보 행동의 집합, 탐욕 전략 확률은 1- ε (0 < ε < 1)

5. Deep Temporal Difference Network Model(DTDN)

1: **Input**: Initialize playback memory D to capacity N
2: Initialize states value function V with weights θ
3: Initialize the target state value function $\hat{V}$ with weights $\theta^- = \theta$
4: **For** episode =1, M do
5: Initialize sequence $s_1 = \{x_1\}$ and pre-processed sequence $\phi_1 = \phi(S_1)$
6: **For** $t = 1, T$ do
7: with probability ε or eq. (7) Select a random action a_t
8: otherwise select $a_t = arg_a Q(\phi(s_t), a; \theta)$
9: Execute action a_t in the emulator and observe the reward r_t and image x_{t+1}
10: Set $s_{t+1} = s_t, a_t, x_{t+1}$ and pre-process $\phi_{t+1} = \phi(s_{t+1})$
11: Store transition $(\phi_t, a_j, r_j, \phi_{t+1})$ in D
12: Sample random minibatch of transitions $(\phi_t, a_j, r_j, \phi_{t+1})$ from D
13: Set: $y_i = \begin{cases} r_{j+1}, & if\ episode\ terminates\ at\ step\ j+1 \\ r_{j+1} + \gamma max_{a'} \hat{V}(\phi_{j+1}; \phi^-); & otherwise \end{cases}$
14: Perform a gradient descent step $[y_j - \gamma V(\phi(S_{t+1}); \theta)]^2$ Concerning the network parameters θ
15: Every C step reset $\hat{Q} = Q$
16: **End For**
17: **End For**

❖ Experiment Study

- 제안된 알고리즘의 유효성 평가를 위해 네가지 테스트 케이스를 활용하여 실험을 진행함
- Kacem에 설정된 표준 테스트에 따라 8개의 소규모 케이스를 사용하여 다른 알고리즘과 비교함
- 이후 제안된 디스패칭 규칙과의 비교 에서는 DTDN알고리즘을 Q-Learning, Deep Deterministic Policy Gradient (DDPG)와 다양한 퍼포먼스에 대해 비교함
- 대규모 인스턴스에서는 다양한 복잡성을 지닌 30개의 FDSSP 문제를 포함하는 테스트 사례를 설계함
- DTDN은 Python 3.6 JetBrains PyCharm Community Edition 2019.2.1 x 64 에서 코딩 진행
- CPU : Intel i9-10900x
- RAM : 16G

❖ Experiment Data

Instances	Benchmarks	Source
Case01-05	Kacem01-05	Kacem
Case06-15	Orb01-10	Hurink-data
Case16-20	Mt10c1-xxx	Barnes
Case21-35	01a-15a	ChambersBarnes
Case36-45	MK01-10	Brandimarte

❖ 성능 지표(상대 백분율 편차)

$$RPD = \frac{C_{\max} - LB}{LB}$$

- LB는 Branch and Bound Algorithm의 최적해

❖ Small Scale FDSSP

Studies	DACS	SM	PSO	HTSA	DTDN
CPU	2.7 GHz	2.4 GHz	NaN	1.7 GHz	3.7 GHz
CPU mark	7023	228	NaN	132	10,203
Relative ratio	0.6883	0.0223	NaN	0.0129	1.0000

				B&B		DACS		SM		PSO		HTSA		DTDN	
Prob	m	n	O	LB	UB	CM	T(s)	CM	T(s)	CM	T(s)	CM	T(s)	CM	T(s)
Case01	4	5	5	11	NaN	11	0.490	12	2.580	NaN	NaN	11	0.150	11	1.567
Case02	8	8	8	11	NaN	14	2.420	14	39.370	14	NaN	14	3.080	13	2.189
Case03	10	7	7	11	NaN	11	2.100	11	110.000	NaN	NaN	11	2.580	11	3.218
Case04	10	10	10	7	NaN	7	2.560	7	39.740	7	NaN	7	3.120	7	3.189
Case05	15	10	10	11	NaN	11	3.790	11	865.230	11	NaN	11	25.130	11	5.142

- CPU Time은 상대 비율로 계산함
- DTDN은 LB와 거의 동일하지만 CPU Time에서 차이가 크게 남

Case	1	2	3	4	5	6
1	5/19	4/42	3/85	7/59	3/87	–/42
2	4/46	5/65	2/56	3/68	8/66	–/41
3	5/65	4/12	2/78	6/25	3/53	–/51
4	3/–	5/–	3/–	3/–	7/–	–/–

Table 8. Design of the test case problems in sets: a ($O_{5,4}$); b ($O_{6,3}$).

- 해당 표에서 알고리즘이 최적의 생산 일정을 점진적으로 생성하는 것을 볼 수 있음
- NEH Algorithm, NEH-KK Algorithm에 비해 각각 4.3%, 2.7% 개선된 것으로 나타남

❖ Comparisons with the proposed dispatching rules

	FIFO	SPT	LPT	SRPT	LRPT	MOR	EDD	QL		DDPG		DTDN	
Prob. (%)	Score	Score	Score	Score	Score	Score	Score	Score	RPD	Score	RPD	Score	RPD
Case06	7.74e2	7.17e2	7.51e2	7.03e2	7.41e2	7.56e2	6.66e2	9.18e2	1.98e−2	8.75e2	0.1435	8.75e2	1.44e−1
Case07	8.81e2	7.56e2	6.87e2	6.54e2	6.86e2	7.64e2	6.83e2	9.54e2	4.48e−2	8.86e2	0.1284	8.86e2	1.28e−1
Case08	7.15e2	8.52e2	7.03e2	6.96e2	7.01e2	7.15e2	6.75e2	9.18e2	8.96e−2	8.74e2	0.1443	8.51e2	1.75e−1
Case09	7.59e2	8.13e2	7.10e2	6.82e2	7.32e2	7.02e2	7.03e2	9.41e2	6.27e−2	8.88e2	0.1264	8.55e2	1.70e−1
Case10	7.68e2	7.70e2	8.07e2	7.83e2	6.92e2	7.31e2	7.13e2	9.09e2	1.00e−1	8.49e2	0.1701	8.48e2	1.79e−1
Case11	7.59e2	8.49e2	6.85e2	6.67e2	7.02e2	6.98e2	6.82e2	9.49e2	5.35e−2	9.13e2	0.0951	8.49e2	1.78e−1
Case12	8.36e2	7.88e2	8.45e2	8.02e2	6.99e2	7.42e2	7.05e2	9.36e2	6.80e−2	8.48e2	0.1788	8.63e2	1.59e−1
Case13	7.34e2	8.12e2	7.65e2	7.72e2	8.20e2	7.92e2	7.36e2	9.40e2	6.34e−2	8.80e2	0.1369	8.14e2	2.29e−1
Case14	7.86e2	7.40e2	7.26e2	6.41e2	6.53e2	6.87e2	6.93e2	9.38e2	6.63e−2	8.63e2	0.1585	8.49e2	1.78e−1
Case15	7.79e2	7.89e2	7.42e2	7.11e2	7.13e2	7.32e2	6.95e2	9.34e2	7.11e−2	9.75e2	0.1424	8.54e2	1.71e−1

Table 9. Results comparison of scheduling score and RPD in different methods on Orb data cases.

- 다른 7개의 Dispatching Rule과 Score, RPD를 비교한 결과 DTDN이 더 우수한 성능을 보임

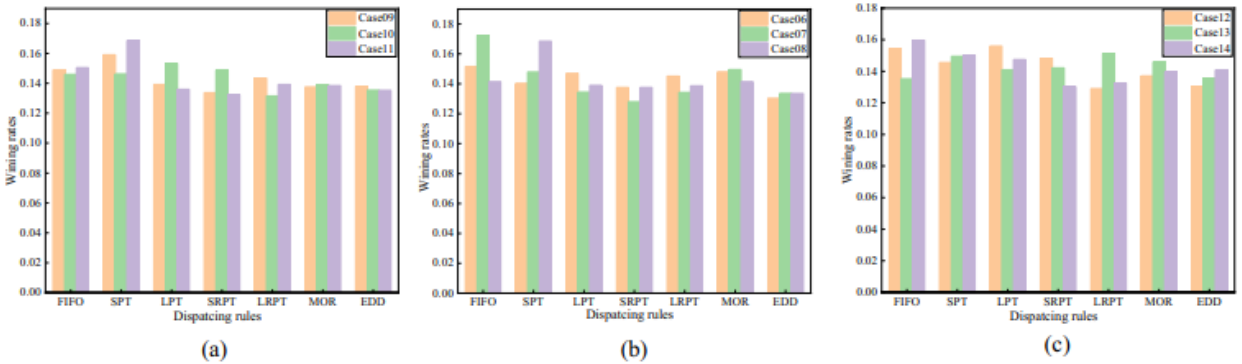


Figure 3. Performance comparison of action space (dispatching rules) under different data cases.

- 위 그림의 X축에 있는 규칙들은 action에서 10%이상 활용되었고, 이러한 액션들은 최적해를 구하는 데 기여도가 높아 활용도가 높은 것으로 알려짐. 다른 action의 빈도분포는 고르지만 성능이 뚜렷하지 않아, 후보 action에 다른 규칙을 추가하는 것을 고려할 수 있음

❖ Large Scale FDSSP

Prob				B&B		HGA		MG		HMEA		DTDN	
	m	n	O	LB	UB	Score	RPD	Score	RPD	Score	RPD	Score	RPD
Case16	10	11	1	6.55e2	9.27e2	9.27e2	4.15e−1	9.28e2	4.17e−1	NaN	NaN	9.12e2	3.92e−1
Case17	10	12	1	6.55e2	9.14e2	9.10e2	3.89e−1	9.10e2	3.89e−1	NaN	NaN	9.10e2	3.89e−1
Case18	10	11	1	6.55e2	9.29e2	9.18e2	4.12e−1	9.18e2	4.02e−1	NaN	NaN	9.18e2	4.02e−1
Case19	10	12	1	6.55e2	9.29e2	9.18e2	4.02e−1	9.18e2	4.02e−1	NaN	NaN	9.18e2	4.02e−1
Case20	10	13	1	6.55e2	9.36e2	9.18e2	4.02e−1	9.06e2	3.83e−1	NaN	NaN	9.06e2	3.83e−1
Case21	10	5	1	2.51e3	2.53e3	2.52e3	5.19e−3	2.52e3	4.01e−4	3.83e3	9.22e−2	2.52e3	6.39e−3
Case22	10	5	1	2.23e3	2.24e3	2.23e3	1.36e−3	2.23e3	1.35e−3	3.40e3	5.26e−1	2.35e3	5.66e−2
Case23	10	5	1	2.23e3	2.24e3	2.23e3	4.49e−4	2.23e3	4.49e−4	3.01e3	3.52e−1	2.23e3	2.24e−3
Case24	10	5	1	2.50e3	2.57e3	2.52e3	4.79e−3	2.50e3	0.00e0	3.80e3	5.16e−1	2.52e3	7.59e−3
Case25	10	5	1	2.19e3	2.23e3	2.22e3	1.28e−2	2.22e3	1.23e−2	3.33e3	5.21e−1	2.26e3	1.64e−2
Case26	10	5	1	2.16e3	2.22e3	2.20e3	1.57e−2	2.20e3	1.90e−2	3.11e3	4.37e−1	2.20e3	1.94e−2
Case27	15	8	1	2.19e3	2.41e3	2.31e3	5.49e−2	2.28e3	4.39e−2	3.88e3	7.72e−1	2.32e3	6.13e−2
Case28	15	8	1	2.06e3	2.09e3	2.07e3	5.82e−3	2.07e3	3.88e−3	3.33e3	6.18e−1	2.08e3	9.70e−3
Case29	15	8	1	2.06e3	2.07e3	2.07e3	2.43e−3	2.07e3	2.43e−3	2.98e3	4.47e−1	2.07e3	6.31e−3
Case30	15	8	1	2.18e3	2.36e3	2.32e3	6.29e−2	2.29e3	5.19e−2	4.00e3	8.36e−1	2.35e3	7.94e−2
Case31	15	8	1	2.02e3	2.08e3	2.07e3	0.00e0	2.06e3	2.28e−2	3.17e3	5.70e−1	2.08e3	3.32e−2
Case32	15	8	1	1.97e3	2.05e3	2.03e3	3.10e−2	2.03e3	3.30e−2	3.24e3	6.46e−1	2.04e3	3.35e−2
Case33	20	10	1	2.16e3	2.30e3	2.26e3	4.44e−2	2.26e3	4.59e−2	3.92e3	8.14e−1	2.28e3	5.51e−2
Case34	20	10	1	2.16e3	2.18e3	2.17e3	2.78e−3	2.17e3	2.78e−3	3.45e3	5.96e−1	2.16e3	9.25e−4
Case35	20	10	1	2.16e3	2.17e3	2.17e3	1.85e−3	2.17e3	2.78e−3	3.34e3	5.44e−1	2.17e3	5.09e−3
				LB	UB	HGA		SLGA		TLBO		DTDN	
Case36	10	6	2	3.60e1	4.20e1	4.00e1	1.11e−1	4.00e1	1.11e−1	6.20e1	7.22e−1	4.20e1	1.67e−1
Case37	10	6	3.5	2.40e1	3.20e2	2.60e1	8.33e−2	2.70e1	1.25e−1	4.80e1	1.00e0	3.00e1	2.50e−1
Case38	15	8	3	2.04e2	2.11e2	2.04e2	0.00e0	2.04e2	0.00e0	3.74e2	8.33e−1	2.04e2	0.00e0
Case39	15	8	2	4.80e1	8.10e1	6.00e1	2.50e−1	6.00e1	2.50e−1	1.36e2	2.44e0	6.20e1	2.92e−1
Case40	15	4	1.5	1.68e2	1.86e2	1.72e2	2.38e−2	1.72e2	2.38e−2	2.65e2	5.77e−1	1.72e2	2.38e−2
Case41	10	15	3	3.30e1	8.60e2	5.80e1	7.58e−1	6.90e1	1.10e01	9.40e1	1.85e0	9.00e1	1.73e1
Case42	20	5	3	1.33e2	1.57e2	1.39e2	4.51e−2	1.44e2	8.27e−2	2.46e2	8.50e−1	1.58e2	1.88e−1
Case43	20	10	1.5	5.23e2	NaN	5.23e2	0.00e0	5.23e2	0.00e0	6.23e2	1.91e−1	5.23e2	0.00e0
Case44	20	10	3	2.99e2	3.69e2	3.07e2	2.68e−2	3.20e2	7.02e−2	3.92e2	3.11e−1	3.20e2	7.02e−2
Case45	20	15	3	1.65e2	2.96e2	1.97e2	1.94e−1	2.54e2	5.39e−1	2.75e2	6.67e−1	2.41e2	4.61e−1

Table 10. Results comparison of scheduling score and RPD in different methods on Orb data cases.

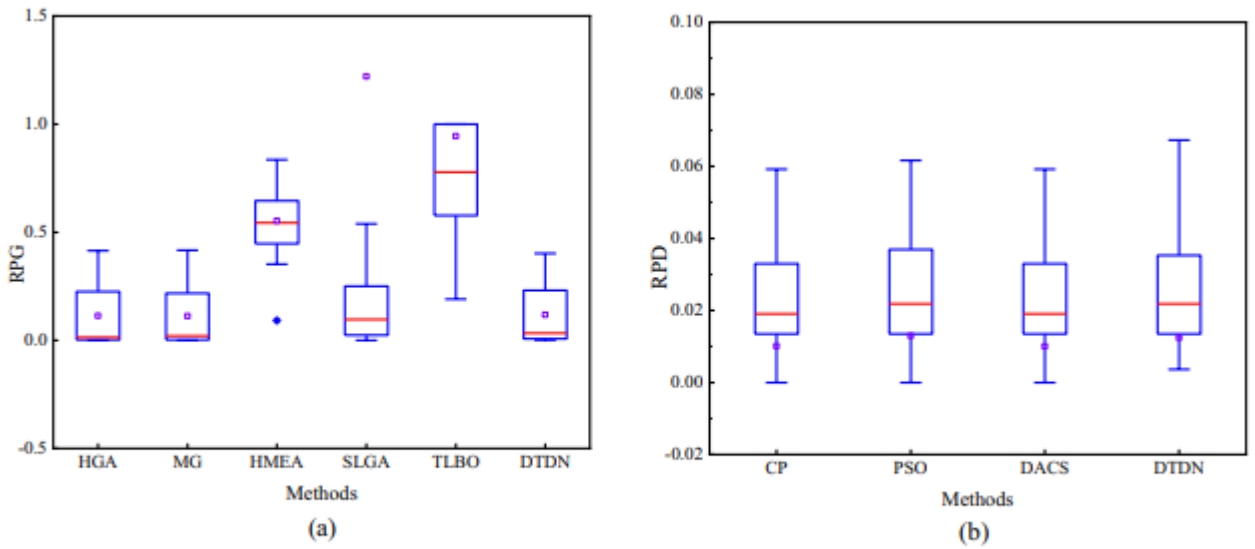


Figure 4. Box plots based on the results in Tables 10 and 11.

- Large 상황에서도 다양한 문제 크기에서 비교한 결과, DTDN이 더 좋은 결과를 보임
- 얻어진 결과는 기존의 최적화 방법보다 우수하여 DTDN의 유효성 입증

❖ Case Study : production scheduling problem

Prob				B&B		CP		PSO		DACS		DTDN	
	n	m	O	LB	UB	C_{max}	RPD	C_{max}	RPD	C_{max}	RPD	C_{max}	RPD
Ins01	6	2	2	4.16e2	5.23e2	4.23e2	1.68e−2	4.23e2	1.68e−2	4.23e2	1.68e−2	4.23e2	1.68e−2
Ins02	6	2	2	4.56e2	5.12e2	4.64e2	1.75e−2	4.64e2	1.75e−2	4.64e2	1.75e−2	4.64e2	1.75e−2
Ins03	6	2	2	5.10e2	6.17e2	5.17e2	1.37e−2	5.17e2	1.37e−2	5.17e2	1.37e−2	5.17e2	1.37e−2
Ins04	6	2	2	3.78e2	4.42e2	3.89e2	2.91e−2	389e2	2.91e−2	3.89e2	2.91e−2	3.89e2	2.91e−2
Ins05	6	2	2	4.31e2	5.36e2	4.31e2	0.00e0	4.31e2	0.00e0	4.31e2	0.00e0	4.60e2	6.73e−2
Ins06	6	2	3	3.80e2	4.27e2	3.84e2	1.05e−2	3.84e2	1.05e−2	3.84e2	1.05e−2	3.82e2	5.26e−3
Ins07	6	2	3	4.07e2	4.97e2	4.12e2	1.23e−2	4.12e2	1.23e−2	4.12e2	1.23e−2	4.12e2	1.23e−2
Ins08	6	2	3	3.89e2	4.76e2	3.97e2	2.06e−2	3.97e2	2.06e−2	3.97e2	2.06e−2	3.97e2	2.06e−2
Ins09	6	2	3	4.57e2	5.39e2	4.68e2	2.41e−2	4.68e2	2.41e−2	4.68e2	2.41e−2	4.68e2	2.41e−2
Ins10	6	2	3	3.02e2	3.99e2	3.06e2	1.32e−2	3.06e2	1.32e−2	3.06e2	1.32e−2	3.06e2	1.32e−2
Ins11	6	3	2	3.14e2	4.37e2	3.26e2	3.92e−2	3.26e2	3.82e−2	3.26e2	3.82e−2	3.26e2	3.82e−2
Ins12	6	3	2	3.47e2	4.26e2	3.53e2	1.73e−2	3.53e2	1.73e−2	3.53e2	1.73e−2	3.53e2	1.73e−2
Ins13	6	3	2	4.36e2	5.33e2	4.56e2	4.59e−2	4.58e2	4.95e−2	4.56e2	4.59e−2	4.56e2	4.59e−2
Ins14	6	3	2	3.94e2	4.93e2	4.05e2	2.79e−2	4.05e2	2.79e−2	4.05e2	2.79e−2	4.05e2	2.79e−2
Ins15	6	3	2	2.87e2	3.75e2	3.04e2	5.92e−2	3.04e2	5.92e−2	3.04e2	5.92e−2	3.04e2	5.92e−2
Ins16	6	3	3	2.76e2	3.63e2	2.80e2	1.45e−2	2.93e2	6.16e−2	2.80e2	1.45e−2	2.77e2	3.62e−3
Ins17	6	3	3	3.25e2	3.97e2	3.37e2	3.69e−2	3.37e2	3.69e−2	3.37e2	3.69e−2	3.37e2	3.69e−2
Ins18	6	3	3	3.65e2	3.23e2	2.71e2	−2.57e−1	2.74e2	−2.49e−1	2.71e2	−2.58e−1	2.71e2	−2.58e−1
Ins19	6	3	3	3.46e2	4.33e2	3.54e2	2.31e−2	3.54e2	2.31e−2	3.54e2	2.31e−2	3.54e2	2.31e−2
Ins20	6	3	3	2.98e2	3.77e2	3.09e2	3.69e−2	3.09e2	3.69e−2	3.09e2	3.69e−2	3.08e2	3.36e−2
Ins21	10	2	2	4.22e2	5.87e2	4.38e2	3.79e−2	4.47e2	5.92e−2	4.38e2	3.79e−2	4.36e2	3.32e−2
Ins22	10	2	2	4.76e2	5.98e2	4.97e2	4.41e−2	4.97e2	4.41e−2	4.97e2	4.41e−2	4.97e2	4.41e−2
Ins23	10	2	2	3.43e2	3.95e2	3.54e2	3.21e−2	3.54e2	3.21e−2	3.54e2	3.21e−2	3.54e2	3.21e−2
Ins24	10	2	2	4.32e2	5.33e2	4.46e2	3.24e−2	4.46e2	3.24e−2	4.46e2	3.24e−2	4.46e2	3.24e−2
Ins25	10	2	2	4.87e2	6.23e2	5.07e2	4.11e−2	5.21e2	6.98e−2	5.07e2	4.11e−2	5.07e2	4.11e−2
Ins26	10	2	3	4.21e2	5.47e2	4.48e2	6.41e−2	4.69e2	1.14e−1	4.54e2	7.84e−2	4.68e2	1.12e−2
Ins27	10	2	3	4.67e2	5.98e2	4.86e2	4.07e−2	5.03e2	7.71e−2	4.86e2	4.07e−2	4.86e2	4.07e−2
Ins28	10	2	3	4.02e2	5.88e2	4.22e2	4.98e−2	4.39e2	9.20e−2	4.24e2	5.47e−2	4.22e2	4.98e−2
Ins29	10	2	3	5.03e2	6.98e2	5.25e2	4.37e−2	5.34e2	6.16e−2	5.34e2	6.16e−2	5.25e2	4.37e−2
Ins30	10	2	3	4.75e2	6.21e2	4.79e2	8.42e−3	4.79e2	8.42e−3	4.79e2	8.42e−3	4.79e2	8.42e−3
Ins31	10	3	2	4.25e2	5.22e2	4.25e2	0.00e0	4.27e2	4.71e−3	4.25e2	0.00e0	4.25e2	0.00e0
Ins32	10	3	2	4.24e2	5.17e2	4.34e2	2.36e−2	4.34e2	2.36e−2	4.34e2	2.36e−2	4.34e2	2.36e−2
Ins33	10	3	2	3.35e2	4.73e2	3.68e2	9.85e−2	3.76e2	1.22e−1	3.70e2	1.05e−1	3.70e2	1.05e−2
Ins34	10	3	2	3.74e2	5.27e2	4.01e2	7.22e−2	4.03e2	7.75e−2	4.01e2	7.22e−2	4.22e2	1.28e−2
Ins35	10	3	2	3.86e2	4.87e2	4.09e2	6.00e−2	4.12e2	6.74e−2	4.09e2	6.00e−2	4.10e2	6.22e−2
Ins36	10	3	3	3.93e2	5.07e2	4.15e2	5.60e−2	4.15e2	5.60e−2	4.15e2	5.60e−2	4.15e2	5.60e−2
Ins37	10	3	3	4.13e2	5.37e2	4.52e2	9.44e−2	4.63e2	1.21e−2	4.52e2	9.44e−2	4.52e2	9.44e−2
Ins38	10	3	3	3.87e2	5.89e2	4.19e2	8.27e−2	4.21e2	8.85e−2	4.19e2	8.27e−2	4.19e2	8.27e−2
Ins39	10	3	3	4.97e2	6.27e2	5.21e2	4.83e−2	5.34e2	7.44e−2	5.21e2	4.83e−2	5.21e2	4.83e−2
Ins40	10	3	3	3.69e2	4.79e2	3.86e2	4.61e−2	3.91e2	5.96e−2	3.88e2	5.15e−2	3.84e2	4.07e−2

Table 11. Results Comparison of makespan and RPD in extended Nourali's test cases.

- 40가지 데이터 사례를 포함한 FDSSP의 유용한 벤치마크 제안
- DTDN의 솔루션을 비교한 결과 이 알고리즘의 최적해는 모두 LB,UB 사이에 있으며 유효함
- DTDN의 성능은 나머지 세 알고리즘의 성능과 비슷하지만 대규모 문제의 경우 CPU Time에서 훨씬 효율적인 모습을 보임

IV Conclusion

- 해당 논문은 FDSSP를 위한 효율적 DTDN 방법을 제안하여 Makespan을 최소화함
- 이 알고리즘은 단순히 구성된 휴리스틱, 집단 지능 알고리즘에 비해 더 적은 수의 반복으로 나은 결과를 얻어냄